

Federated Search for Heterogeneous Environments

Jaime Arguello

CMU-LTI-11-008

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Thesis Committee:

Jamie Callan (chair)
Jaime Carbonell
Yiming Yang
Fernando Diaz (Yahoo! Research)

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
In Language and Information Technologies*

© 2011, Jaime Arguello

Para Sophia, con todo mi amor.

Acknowledgments

This dissertation would have not been possible without the persistent guidance and encouragement of my mentors. I owe a big ‘thank you’ to my advisor, Jamie Callan. It was during the Fall of 2004, when I took two half-semester courses taught by Jamie (Digital Libraries and Text Data Mining), that I discovered the field of Information Retrieval. Seven years later, I am writing my dissertation acknowledgments. Jamie’s advice on both research and life in general has been a great source of insight and support. His feedback on research papers, presentations, lecture slides, proposals, and this dissertation has taught me a great deal about communicating effectively. From Jamie’s example, I have learned valuable lessons on research, teaching, and mentoring.

I would like to thank Jaime Carbonell, Yiming Yang, and Fernando Diaz for agreeing to be in my thesis committee. Their feedback was critical in making this dissertation stronger. Fernando Diaz helped shape many of the ideas presented in this dissertation. It was during an internship with Fernando at Yahoo! where I began working on vertical selection. I enjoyed it so much I returned for a second internship a year later. I have been fortunate to have had Fernando as a mentor and collaborator ever since.

I wish to thank the rest of the group at Yahoo! Labs Montreal for making my two internships so rewarding. I was lucky to have had Jean-François Crespo as a manager, who provided everything necessary to run experiments quickly. Jean-François Paiement was a helpful collaborator on the work on domain adaptation for vertical selection. Hughes Buchard, Alex Rochette, Remi Kwan, and Daniel Boies provided much needed feedback, help with tools, and a model work environment. Merci pour tout!

I would like to thank Carolyn Rosé for taking me in as a masters student and for guiding me through my first two years doing research. More generally, I would like to thank all the LTI faculty for putting together such an engaging curriculum and the LTI staff for keeping the place running and always being patient with my questions.

Grad school would have not been the same without my friendship with Jon Elsas. It is quite possible I ran every research idea through Jon. Those hundreds of lunch and coffee-break discussions are much appreciated. Thank you to all my other friends at CMU, who provided such valuable feedback on every practice talk I subjected them to. And, thank you to my officemates, Grace Hui-Yang and Anagha Kulkarni, for providing such a nice environment to work in.

I would also like to thank all my friends outside of CMU for taking me away from work (admittedly, without much convincing) and for making those times so worthwhile. I am grateful to Tori Ames for her friendship and support during my last year as a PhD student.

Finally and most importantly, I would like to thank my family for their love and encouragement. Leaping into the unknown is easier when you have someone there to pick you up and cheer you on. None of this would have been possible without my mother, who taught me the value of loving what you do.

Abstract

In information retrieval, federated search is the problem of automatically searching across multiple distributed collections or resources. It is typically decomposed into two subsequent steps: deciding which resources to search (*resource selection*) and deciding how to combine results from multiple resources into a single presentation (*results merging*). Federated search occurs in different environments. This dissertation focuses on an environment that has not been deeply investigated in prior work.

The growing heterogeneity of digital media and the broad range of user information needs that occur in today's world have given rise to a multitude of systems that specialize on a specific type of search task. Examples include search for news, images, video, local businesses, items for sale, and even social-media interactions. In the Web search domain, these specialized systems are called *verticals* and one important task for the Web search engine is the prediction and integration of relevant vertical content into the Web search results. This is known as *aggregated web search* and is the main focus on this dissertation.

Providing a single-point of access to all these diverse systems requires federated search solutions that can support *result-type* and *retrieval-algorithm heterogeneity*. This type of heterogeneity violates major assumptions made by state-of-the-art resource selection and results merging methods.

While existing resource selection methods derive predictive evidence exclusively from sampled resource content, the approaches proposed in this dissertation draw on machine learning as a means to easily integrate various different types of evidence. These include, for example, evidence derived from (sampled) vertical content, vertical query-traffic, click-through information, and properties of the query string. In order to operate in a heterogeneous environment, we focus on methods that can learn a vertical-specific relationship between features and relevance. We also present methods that reduce the need for human-produced training data.

Existing results merging methods formulate the task as score normalization. In a more heterogeneous environment, however, combining results into a single presentation requires satisfying a number of layout constraints. The dissertation proposes a novel formulation of the task: *block ranking*. During block-ranking, the objective is to rank sequences of results that must appear grouped together (vertically or horizontally) in the final presentation. Based on this formulation, the dissertation proposes and empirically validates a cost-effective methodology for evaluating aggregated web search results. Finally, it proposes the use of machine learning methods for the task of block-ranking.

Contents

1	Introduction	1
1.1	Unique Properties of Aggregated Web Search	3
1.2	The Goal and Contribution of this Dissertation	6
1.2.1	Summary of Contributions	10
1.3	Impact on Other Applications	11
1.4	Dissertation Outline	12
2	Prior Research in Federated Search	13
2.1	Cooperative vs. Uncooperative Federated Search	13
2.2	Resource Representation	14
2.2.1	How much to sample	14
2.2.2	What to sample	15
2.2.3	When to re-sample	16
2.2.4	Combining collection representations.	16
2.3	Resource Selection	16
2.3.1	Problem Formulation and Evaluation	16
2.3.2	Modeling Query-Collection Similarity	17
2.3.3	Modeling the Relevant Document Distribution	19
2.3.4	Modeling the Document Score Distribution	20
2.3.5	Combining Collection Representations	22
2.3.6	Modeling Search Engine Effectiveness	23
2.3.7	Query Classification	25
2.3.8	Machine Learned Resource Selection	26
2.4	Results Merging	27
2.5	Summary	28
3	Vertical Selection	30
3.1	Formal Task Definition	32
3.2	Classification Framework	32
3.3	Features	32
3.3.1	Corpus Features	33
3.3.2	Query-Log Features	36
3.3.3	Query Features	38
3.3.4	Summary of Features	39
3.4	Methods and Materials	39

3.4.1	Verticals	40
3.4.2	Queries	40
3.4.3	Evaluation Metric	41
3.4.4	Implementation Details	42
3.4.5	Single-evidence Baselines	42
3.5	Experimental Results	42
3.6	Discussion	44
3.6.1	Feature Ablation Study	44
3.6.2	Per Vertical Performance	45
3.7	Related Work in Vertical Selection	46
3.8	Summary	48
4	Domain Adaptation for Vertical Selection	50
4.1	Formal Task Definition	51
4.2	Related Work	51
4.3	Vertical Adaptation Approaches	54
4.3.1	Gradient Boosted Decision Trees	54
4.3.2	Learning a Portable Model	55
4.3.3	Adapting a Model to the Target Vertical	58
4.4	Features	59
4.4.1	Query Features	59
4.4.2	Query-Vertical Features	59
4.5	Methods and Materials	60
4.5.1	Verticals	60
4.5.2	Queries	61
4.5.3	Evaluation Metrics	61
4.5.4	Unsupervised Baselines	62
4.6	Experimental Results	63
4.7	Discussion	63
4.8	Summary	68
5	Vertical Results Presentation: Evaluation Methodology	69
5.1	Related Work in Aggregated Web Search Evaluation	71
5.1.1	Understanding User Behavior	71
5.1.2	Aggregated Web Search Evaluation	72
5.2	Layout Assumptions	74
5.3	Task Formulation: Block Ranking	75
5.4	Metric-Based Evaluation Methodology	75
5.4.1	Collecting Block-Pair Preference Judgements	76
5.4.2	Deriving the Reference Presentation	77
5.4.3	Measuring Distance from the Reference	79
5.5	Methods and Materials	80
5.5.1	Block-Pair Preference Assessment	81

5.5.2	Verticals	81
5.5.3	Queries	82
5.5.4	Empirical Metric Validation	83
5.6	Experimental Results	83
5.6.1	Assessor Agreement on Block-Pair Judgements	84
5.6.2	Assessor Agreement on Presentation-Pair Judgements	85
5.6.3	Metric Validation Results	86
5.7	Discussion	88
5.8	Summary	89
6	Vertical Results Presentation: Approaches	92
6.1	Formal Task Definition	93
6.2	Related Work	94
6.3	Features	95
6.3.1	Pre-retrieval Features	95
6.3.2	Post-retrieval Features	97
6.3.3	Summary of Features	100
6.4	Block-Ranking Approaches	101
6.4.1	Classification Approach	101
6.4.2	Voting Approach	103
6.4.3	Learning to Rank Approaches	104
6.5	Methods and Materials	106
6.5.1	Verticals and Queries	107
6.5.2	Block-Pair Preference Assessment	108
6.5.3	Modeling Bias Towards Vertical Results	110
6.5.4	Evaluation Metric	110
6.5.5	Implementation Details	112
6.6	Experimental Results	113
6.7	Discussion	115
6.7.1	Effect of Parameter α on LTR variants	116
6.7.2	Feature Contribution to Overall Performance	117
6.7.3	Feature Contribution to Per-Vertical Performance	118
6.7.4	LTR Model with Cross-Product Features	121
6.7.5	Binning Analysis	122
6.8	Summary	123
7	Resource Selection in a Homogeneous Environment	126
7.1	Formal Task Definition	127
7.2	Classification-based Approach	128
7.3	Features	128
7.3.1	Corpus Features	128
7.3.2	Query Category Features	130
7.3.3	Click-through Features	130

7.3.4	Summary of Features	131
7.4	Methods and Materials	131
7.4.1	Evaluation Testbeds	131
7.4.2	Queries	132
7.4.3	Evaluation Metrics	132
7.4.4	Implementation Details	133
7.4.5	Single-Evidence Baselines	134
7.5	Experimental Results	134
7.6	Discussion	138
7.6.1	Collection Representation Quality	138
7.6.2	Feature Ablation Studies	139
7.7	Summary	140
8	Conclusions	142
8.1	Summary	142
8.1.1	Resource Selection	142
8.1.2	Results Presentation	144
8.2	Thesis Contributions	146
8.3	New Directions	148
8.3.1	Aggregated Search with Dynamic Content and User Interests	148
8.3.2	Improving the Coherence between Cross-Vertical Results	149
8.3.3	Aggregated Mobile Search	150
8.3.4	Search Tool Recommendation	151
A	Vertical Blocks	164
B	User Study Queries and Descriptions	167

List of Figures

1.1	Given the query “first man in space”, an aggregated web search system determines that the <i>video</i> , <i>news</i> , and <i>images</i> verticals are relevant and incorporates their top results into the Web results as shown.	2
4.1	Feature portability values (π^{hmap}) for all features across sets of source verticals. Features ordered randomly along x-axis. The features that were consistently the most portable were query-vertical features. As shown, many of these correspond to existing unsupervised resource selection methods used in prior work.	66
5.1	An example presentation of vertical results for “pittsburgh”, where the <i>maps</i> vertical is presented above, the <i>news</i> vertical within, and the <i>images</i> vertical below the top Web results. Consistent with the major commercial search engines (i.e., Bing, Google, and Yahoo!), we assume that same-vertical results must appear grouped together—horizontally (e.g., <i>images</i>) or vertically (e.g. <i>news</i>)—within a in the output presentation.	70
5.2	The vertical results presentation task can be formulated as <i>block ranking</i> . . .	76
5.3	Approach Overview.	77
5.4	Block-pair assessment interface. Two blocks are displayed randomly side-by-side (in this case <i>books</i> block and w_3) and the assessor is asked which would best satisfy a user for the given query and description: the left block, the right block, or neither.	78
5.5	Example vertical blocks.	82
5.6	Presentation-pair assessment interface. Two presentations are displayed randomly side-by-side and the assessor is asked which they prefer: the left presentation, the right presentation, or neither.	84
5.7	Context-aware block-pair assessment interface.	88
5.8	Example of a multi-dimensional vertical display. Given the query “michelle obama”, Yahoo! presents <i>news</i> , <i>images</i> , <i>video</i> , and <i>twitter</i> within a horizontal tabbed display. Other verticals can appear embedded within the Web results below.	90

6.1	Vertical slotting positions. The classification approach predicts an output presentation by assigning verticals to slots s_{1-4} using threshold parameters τ_{1-4} . Ties (i.e., verticals assigned to the same slot) are broken using the prediction probability.	103
6.2	Given the query “pittsburgh”, a search engine recommends a set of related queries (red, dashed box). The related query “pittsburgh photos” (blue, solid box) suggests that users who issue the query “pittsburgh” often want to see photos (i.e., it suggests that the <i>images</i> vertical is relevant).	108
7.1	Collection size distribution of our three experimental testbeds.	132
7.2	Number of instances in which a collection of a given size (bin) contributes at least 10 relevant documents to a test query.	139
8.1	A commercial search engine presents <i>news</i> and <i>images</i> results in response the query “joplin”.	150
A.1	Example vertical blocks.	164
A.2	Example vertical blocks.	165
A.3	Example vertical blocks.	166

List of Tables

3.1	Vertical descriptions.	40
3.2	Percentage of queries for which the vertical (or no vertical) was labeled relevant. Percentages do not sum to one because some queries were assigned multiple relevant verticals.	41
3.3	Single Vertical Precision (\mathcal{P}). Approaches are listed in ascending order of \mathcal{P} . A significant improvement over all worse-performing approaches is indicated with a Δ at the $p < 0.05$ level and a \blacktriangle at the $p < 0.005$ level.	43
3.4	Feature type ablation study. A \blacktriangledown denotes a significant improvement ($p < 0.005$) over our classifier using all features	44
3.5	Per vertical precision (\mathcal{P}). Verticals without a query-log are marked with \star . Verticals without a Wikipedia-sampled surrogate corpus are marked with $*$	46
4.1	Vertical descriptions.	61
4.2	Target-trained (“cheating”) results: AP and NDCG.	62
4.3	Portability results: normalized AP and NDCG. A $\blacktriangle(\blacktriangledown)$ denotes significantly better(worse) performance compared to Soft.ReDDE. A $\Delta(\nabla)$ denotes significantly better(worse) performance compared to the unbalanced basic model with all features. Significance was tested at the $p < 0.05$ level.	64
4.4	Trada results: normalized AP and NDCG. A $\blacktriangle(\blacktriangledown)$ denotes significantly better (worse) performance compared to Soft.Redde. A $\Delta(\nabla)$ denotes a significantly better(worse) performance compared to the base model. Significance was tested at the $p < 0.05$ level.	65
4.5	Target-trained results. Given access to target-vertical training data, a model that uses all features (portable + non-portable) performs significantly better than one that is given access to only the non-portable ones. A \blacktriangledown denotes a significant degradation in performance at the $P < 0.05$ level.	67
5.1	The Fleiss’ Kappa agreement on presentation-pairs.	86
5.2	Metric agreement with majority preference. Significance is denoted by Δ and \blacktriangle at the $p < 0.05$ and $p < 0.005$ level, respectively.	87
6.1	Vertical-to-URL mapping. URLs correspond to either the actual vertical (e.g., the <i>shopping</i> vertical was constructed using the eBay search API) or content related to the vertical (e.g., <code>http://www.cooks.com</code> contains recipes and is thus related to the <i>recipe</i> vertical). A $*$ denotes a “wildcard” operation.	98

6.2	Vertical-to-keyword mapping.	99
6.3	Block-types along with their textual representations.	100
6.4	Summary of Features. Pre-retrieval features are independent of the block under consideration. Thus, every block-type is associated with the same set of 80 pre-retrieval features. Post-retrieval features, on the other hand, are derived from the Web or vertical search engine or directly from those results presented in the block. Therefore, different types of blocks are associated with a different set of post-retrieval features. Some features are common to multiple block-types, and others are unique to a particular type. Text-similarity features are marked in gray.	102
6.5	Number of queries (out of 1070) in which each block-type was presented within the top 3 ranks of σ_q^* as a function of pseudo-votes.	111
6.6	Block-ranking results in terms of average $K^*(\sigma_q^*, \sigma_q)$. Statistical significance was tested using a one-tailed paired t-test, comparing performance across pairs of queries. A $\Delta(\nabla)$ denotes significantly better(worse) performance compared to the classification approach, a $\blacktriangle(\blacktriangledown)$ denotes significantly better(worse) performance compared to the voting (all pairs) approach, and a $\wedge(\vee)$ denotes significantly better(worse) performance compared to the LTR-G approach.	114
6.7	Block-ranking results in terms of $K(\sigma_q^*, \sigma_q)$. A $\blacktriangle(\blacktriangledown)$ denotes a significant improvement(drop) in performance for an LTR variant with α tuned compared to its counterpart for which α is forced to zero. A $\Delta(\nabla)$ denotes significantly better(worse) performance compared to the classification approach.	116
6.8	Feature ablation results for the classification approach and the LTR-S approach. A $\blacktriangle(\blacktriangledown)$ denotes a significant improvement(drop) in performance compared to the model with access to all features.	117
6.9	Feature contribution to per-vertical performance, based on AP. Statistical significance is tested using a one-tailed paired t-test, comparing across cross-validation test-folds. A $\blacktriangle(\blacktriangledown)$ denotes a significant improvement(drop) in performance compared to the model with access to all features.	119
6.10	Block-ranking performance in terms of average $K^*(\sigma_q^*, \sigma_q)$. Statistical significance was tested using a one-tailed paired t-test, comparing performance across pairs of queries. A $\Delta(\nabla)$ denotes significantly better(worse) for LTR-C vs. LTR-S. A $\blacktriangle(\blacktriangledown)$ denotes significantly better(worse) performance for LTR-C vs. LTR-G.	122
6.11	Binning evaluation results.	124
7.1	Results for experimental condition <i>gov2.1000.1000</i>	135
7.2	Results for experimental condition <i>gov2.250.1000</i>	136
7.3	Results for experimental condition <i>gov2.30.1000</i>	136
7.4	Results for experimental condition <i>gov2.1000.300</i>	137
7.5	Results for experimental condition <i>gov2.250.300</i>	137

7.6	Results for experimental condition <i>gov2.30.300</i>	138
7.7	Feature type ablation study. A significant drop in performance, using a paired t-test on queries, is denoted with a \triangle at the $p < 0.05$ level and a \blacktriangle at the $p < 0.005$ level.	140

Introduction

In the broadest sense, the goal of information retrieval (IR) is to rank items by their relevance to a user's information need, expressed using a query. The earliest advancements in the field were in the area of ad-hoc retrieval. Ad-hoc retrieval assumes that the retrievable items consist of textual documents contained within a *single* collection and assumes relevance to be *topical* relevance, which can be typically modeled using text-similarity. In more recent years, however, the scope of the field has broadened to address more specific and nuanced information-seeking tasks. On the Web, these include, for example, search for news, images, videos, local businesses, blogs, weather information, on-line discussions, items for sale, digitized books, scientific publications, movie times, and, more recently, social-media interactions like Twitter or Facebook status updates.

One common finding in empirical IR research is that different information-seeking tasks require different solutions. In other words, a single monolithic search engine cannot effectively support the wide range of search tasks afforded by today's search systems. Different search tasks are associated with different types of media (e.g., images, video, news, blog feeds), which may require different representations to facilitate search. Different search tasks are associated with different definitions of relevance and may therefore require different retrieval algorithms. For example, recency is important for news search [28], geographical proximity is important for local business search [1], and authority is important in micro-blog search [109]. Finally, different search objectives may benefit from different ways of presenting results. For example, news results may need to present their publication date, local results may need to be displayed geographically in a map, and shopping results may need to display their selling price. As a result, search systems today are more specialized and diverse than ever before. This gives rise to a new challenge: how do we provide users with integrated access to all these diverse search services within a single search interface?

In the Web search domain, specialized search services are referred to as vertical search services or *verticals*. Verticals are typically developed and maintained by the same search company, but address a specific information-seeking task. Verticals common to the major commercial search providers (i.e., Bing¹, Google², Yahoo!³), include, for example, *news*, *images*, *video*, *local*, *blogs*, *finance*, *shopping*, *weather*, *movies*, and *sports*. There are currently two ways that users can access vertical content. If the vertical has direct search capabilities and the user is aware of them, the query can be issued directly to the vertical.

¹<http://www.bing.com>

²<http://www.google.com>

³<http://www.yahoo.com>

In other cases, however, the user may not know that a vertical is relevant or may want results from multiple verticals at once. For these reasons, one important step for commercial search providers has become the prediction and integration of relevant vertical content into the Web search results, as shown in Figure 1.1. In the research community, this is referred to as *aggregated web search* [74]. The goal of aggregated web search is to provide integrated access to multiple search services (including Web search) within a *single* search interface.

Search results for "first man in space":

- web**
 - [Yuri Gagarin - Wikipedia, the free encyclopedia](#)

Jump to [Space flight](#): In his post-flight report, Gagarin recalled his experience of spaceflight, having been the **first human in space**: ...

MiG-15 - Vostok 1 - Lost Cosmonauts

[en.wikipedia.org/wiki/Yuri_Gagarin](#) - Cached - Similar
 - [NASA - Yuri Gagarin: First Man in Space](#)

Yuri Gagarin becomes the **first man in space** on April 12, 1961 April 12 was already a huge day in space history twenty years before the launch of the first ...

[www.nasa.gov/mission_pages/.../gagarin_anniversary.html](#) - Cached - Similar
- video**
 - [Videos for first man in space - Report videos](#)
 - [First Man in Space - Skydiving From The Edge ...](#)

7 min - Mar 27, 2007

Uploaded by Baskiskg

[youtube.com](#)
 - [First man In Space - Skydiving from the edge ...](#)

2 min - Dec 31, 2004

Uploaded by --

[video.google.com](#)
- web**
 - [Yuri Gagarin: 50th Anniversary of First Man in Space; 7 Top Things ...](#)

Apr 12, 2011 ... It is 50 years now since Yuri Gagarin became the **first man in space**. Here, a look at seven facts -- or myths -- about the Soviet ...

[abcnews.go.com](#) > Technology
- news**
 - [News for first man in space](#)
 - [Fifty years since the first man in space](#)

48 minutes ago

FIFTY YEARS AGO today, Yuri Gagarin became the **first man in space** when the Russians packed him in a pod and sent him skywards, the US followed and within Inquirer (blog) - 1682 related articles

[Joe Kittinger's 102800-Foot Jump Predates The First Man In Space ...](#)

Huffington Post - 2 related articles - Shared by 10+

[50th Anniversary of First Man in Space: Best Space Movie Scene of ...](#)

Fox News - 3 related articles
- images**
 - [Images for first man in space - Report images](#)
- web**
 - [Yuri Gagarin: 50th anniversary of the first man in space - Telegraph](#)

Apr 12, 2011 ... In the West, memories of the **Space Race** are dominated by Neil Armstrong and Apollo. But in Russia, it is the cult of Gagarin that rules, ...

[www.telegraph.co.uk/.../Yuri-Gagarin-50th-anniversary-of-the-first-man-in-space.html](#) -

Figure 1.1: Given the query “first man in space”, an aggregated web search system determines that the *video*, *news*, and *images* verticals are relevant and incorporates their top results into the Web results as shown.

In information retrieval, *federated search* (also known as *distributed information retrieval*) is the task of automatically searching across multiple distributed collections. It has been an active area of IR research for over 20 years. In the research literature, federated search is often formulated as three separate, but interrelated subtasks: (1) gathering information

about each collection's contents (*resource representation*), (2) deciding which collections to search given a query (*resource selection*), and (3) merging results from the selected collections into a single ranking (*results merging*) [15]. Resource representation happens off-line and its main objective is to inform resource selection (i.e., to help predict which resources have relevant content without issuing the query to the resource). Resource selection and results merging happen every time a query is issued to the federated search system.

Aggregated web search can be viewed as a federated search problem, with similar subtasks. It is often impractical, if not impossible, to issue the query to *every* available vertical search service. Therefore, the first sub-task, *vertical selection*, is to predict *which* few verticals (if any) are relevant to the query. Vertical selection is analogous to resource selection. Once a set of verticals have been selected, the second sub-task, *vertical results presentation*, is to predict *where* in the Web results to present the vertical results. Vertical results presentation is analogous to results merging. As we discuss in more detail later, there are some important differences. For example, in aggregated web search, results from the same vertical must be presented together in the final results (e.g., *video*, *news*, and *images* in Figure 1.1). However, similar to results merging, the objective in vertical results presentation is to combine results from potentially multiple search services into a single presentation of results.

The main focus of the dissertation is on new methods for vertical selection and vertical results presentation. While aggregated web search can be viewed as a type of federated search task, it is associated with unique challenges and opportunities that have not been deeply investigated in prior federated search research. To motivate our research agenda, we describe these at a high level in the next section.

1.1 Unique Properties of Aggregated Web Search

A vertical search service is designed to satisfy a specific type of information need (e.g., image search, product search, job search). For this reason, different verticals often retrieve different types of results (e.g., images, product descriptions, job listings) and use significantly different retrieval algorithms. We refer to these two properties of aggregated web search as *result-type* and *retrieval-algorithm heterogeneity*. Prior federated search research investigated environments where collections focus on different topics [50, 94, 113], have a skewed size distribution [90, 94, 95, 96, 104], have a skewed relevant document distribution [90, 94, 95, 96, 104], and vary in terms of search engine effectiveness [96]. However, result-type and retrieval-algorithm heterogeneity have been ignored. Assumptions about document type and retrieval algorithm heterogeneity affect resource selection and results merging.

Prior federated search research often distinguishes between a *cooperative* and an *uncooperative* environment. In an uncooperative environment, resources are assumed to provide the system no more than the same functionality they provide their human users: a search interface. Partly for this reason, state-of-the-art resource selection methods,

designed for an uncooperative environment, derive evidence *exclusively* from sampled collection documents, obtained (off-line) by issuing queries to the resource and downloading results. In contrast, aggregated web search occurs in at least a *semi-cooperative* environment. We assume that the system and its target verticals are operated by the same search company or its business partners. In this environment, the system may have access to sources of evidence beyond collection documents. For example, for verticals with direct search capabilities, alternative sources of evidence include vertical-specific query-traffic data, click-through data, and query-reformulation data. This type of evidence conveys how users interact directly with the vertical search engine and may be helpful in predicting vertical relevance. A vertical selection system should be capable of incorporating these various sources of evidence into selection decisions.

In addition to deriving evidence exclusively from (sampled) resource content, existing methods apply the *same* scoring function to *every* resource in order to decide which to select. In other words, existing methods assume that predictive evidence is similarly predictive across resources. The ReDDE algorithm [94], for example, uses the predicted relevance of samples from different resources in order to estimate the relevant document distribution across resources. Implicitly, ReDDE assumes a consistent relationship between the number of relevant documents in the resource and its relevance: if collection \mathcal{A} has twice as many relevant documents than \mathcal{B} , then \mathcal{A} is twice as relevant as \mathcal{B} . This is not necessarily true in aggregated web search. A *weather* vertical that retrieves a single relevant result may be more relevant than a *news* vertical that retrieves ten relevant results. This suggests that aggregated web search requires methods that can learn a *vertical-specific* relationship between predictive evidence and vertical relevance, even if we focus exclusively on content-based evidence.

Some verticals focus on specific topics (e.g., *health, auto, travel*) or types of media (e.g., *images, video*). For this reason, it is sometimes possible to detect vertical intent from the query string alone, for example, based on the query topic (e.g., “swine flu” \rightarrow *health*), based on named entity types appearing in the query (e.g., “bmw reviews” \rightarrow *auto*), or based on query keywords (e.g., “obama inauguration pics” \rightarrow *images*). This type of evidence is not inherently associated with a vertical. We refer to this type of evidence as *resource agnostic evidence*. Useful correlations between resource-agnostic evidence and resource relevance must be learned using some form of supervision (e.g., by learning a model using training data). Existing resource selection methods do not provide mechanisms for easily learning predictive relationships between resource-agnostic evidence and resource relevance. This point also resonates with the previous one. We may benefit from methods that are capable of learning a *vertical-specific* predictive relationship between agnostic evidence and vertical relevance. We can imagine, for example, that different verticals will focus on different topics and be associated with different query keywords.

In some federated search environments, resource selection can be formulated as resource *ranking*. During resource ranking, the objective is not to make a binary selection decision for every available resource, but rather to *prioritize* resources for selection. Re-

source ranking is an appropriate formulation of the task when two assumptions hold true: when at least one resource must be selected in order to retrieve documents for the user and when an effective merged retrieval can be produced by always selecting some fixed number $k < n$ of resources from the top ranks, where n is the number of available resources. In aggregated web search, these assumptions do not hold true. Given n candidate verticals, it is possible for the user to not want to see *any* vertical results (i.e., they may just want to see Web results). This may happen, for example, when the query is a navigational query. Therefore, in aggregated web search, the task cannot be formulated as *vertical ranking*. It may be that the best choice for the system is to entirely suppress even the most confident vertical.

The aggregated web search environment is highly dynamic. Changes to the environment can originate from a change in the set of candidate verticals, a change in the contents of a particular vertical, and a change in the interests of the user population. A *news* vertical is an example of a dynamic vertical search service. Solutions that assume a static environment may not be appropriate for a dynamic environment. A federated search solution well-suited for a dynamic environment should be robust to changes in the environment and, in cases when necessary, it should be possible to learn a new model without extensive human effort.

So far, we have focused on vertical selection. Aggregated web search is also unique in terms of merging, the task of combining results from those resources (or verticals) selected into a single presentation of results. In some federated search environments, results merging is free from layout constraints. That is, documents from different resources can be interleaved freely in the merged ranking. In the absence of constraints, results merging is often formulated as *score normalization*—converting scores from different resources so that they are directly comparable and can be used to infer a merged ranking. In aggregated web search, however, results merging must satisfy a number of layout constraints. For example, results from the same vertical must appear grouped together (vertically or horizontally) within the Web search results page (see Figure 1.1). This constraint is partly due to the fact that results from different verticals are presented differently. For example, *news* results display their publication date, *images* and *video* results are displayed as thumbnail images, and *local* results are displayed geographically in a map. Given such layout constraints, the merging task cannot be formulated solely as score normalization. Vertical results presentation requires new solutions.

To summarize, aggregated web search exemplifies a federated search environment that has not been deeply investigated in prior work. In terms of *vertical selection*, the task of predicting which verticals are relevant, aggregated web search requires approaches that:

- do not assume result-type and retrieval-algorithm homogeneity;
- can integrate various types of evidence;
- can learn a vertical-*specific* relationship between predictive evidence and vertical relevance;

- can predict when *no vertical is relevant*; and
- can be (re-)trained without extensive human effort.

In terms of *vertical results presentation*, the task of predicting where to present the verticals results, aggregated web search requires approaches that can aggregate cross-vertical content while satisfying a number of layout constraints.

1.2 The Goal and Contribution of this Dissertation

Federated search has been an active area of information retrieval research for close to two decades. Most of this work, however, was conducted under the assumption that resources return similar types of (text-rich) documents (*result-type homogeneity*) and use similar retrieval algorithms (*retrieval-algorithm homogeneity*). These assumptions are deeply embedded in state-of-the-art methods for resource selection and results merging.

Resource Selection In terms of resource selection, existing approaches can be divided into two general classes.⁴ So-called *large document models* treat each resource as a single, monolithic “large document” and select resources based on a retrieval from a large document index [14, 98, 113]. These methods adapt well-studied text-based document ranking functions to ranking resources. On the other hand, so-called *small document models* predict resource relevance as a function of sample relevance [90, 94, 95, 96, 104]. In the most general sense, these small document methods proceed as follows. First, samples from every resource are combined within a centralize sample index. Then, given a query, a retrieval from this index is used to produce a scoring of samples. Finally, each collection is scored based on some function of the predicted relevance of its samples. Small document models differ in how they go from a scoring of samples to a scoring of resources. However, at a high level, they all follow this sequence of steps.

From this high-level description, we can see that both large and small document models assume *result-type homogeneity*. That is, they assume that content from different resources can be combined in a single index—either a large document index or a centralized sample index. Furthermore, both large and small document models assume *retrieval-algorithm homogeneity*. That is, they assume that a single (text-based) retrieval algorithm (internal to the federated search system) can be used to predict relevance across resources.

Neither of these assumptions are true in an aggregated web search environment. While some vertical search services are associated with text-rich documents (e.g., *blogs, news*), others are associated with text-impooverished documents (e.g., *images, video*), and others with no documents at all (e.g., *calculator, language translation*). Thus, it may not be possible to combine cross-vertical samples within a single index. Furthermore, even if cross-resource content could be combined within a single index, because they address

⁴These methods are reviewed in more detail in Section 2.3.

different types of information needs, different verticals predict relevance differently. For example, as previously mentioned, recency is important when searching for news [28], geographical proximity is important when searching for local businesses [1], and the author’s connectivity in the network is important when searching for Twitter updates [109]. These various definitions of relevance cannot be modeled by a single retrieval algorithm.

Additionally, content-based methods have another potential limitation: they do not provide ways to easily incorporate other types of evidence. Partly because the environment is not completely uncooperative, in aggregated web search we have access to multiple sources of non-content-based evidence that might be useful for selection. This includes information derived from vertical query-traffic, from historical click-through data, and from properties of the query string (e.g., whether the query is about a particular topic or contains a particular keyword).

To address these limitations, this dissertation takes a new approach to resource selection. We cast resource selection as a supervised machine-learned classification problem. By design, our approach differs from existing resource selection methods in three respects. First, it incorporates *multiple* sources of evidence as input features. Second, it uses *training data* (e.g., a set of queries with vertical-relevance judgements) to learn a predictive relationship between features and vertical relevance. Third, we learn a different model for each vertical. Therefore, we allow the system the freedom to learn a *vertical-specific* relationship between features and vertical relevance. We demonstrate in Chapter 3 that this approach outperforms existing methods and that feature-integration is a major contributor to its success. No single type of feature is solely responsible for its performance.

While a supervised machine learning approach to resource selection has many advantages, one limitation is that it requires training data, for example, in the form of a set of queries with relevance judgements for a set of verticals. In practice, the aggregated web search environment is dynamic. A new vertical can be introduced to the set of candidate verticals, content within a vertical can change, and the interests of the user population can shift. Prior work showed that even a change in resource content degrades performance for an already-tuned resource selector [93]. Human annotation is costly in terms of time and resources. An annotation effort may make sense as a one-time investment. However, it may not be feasible to annotate a new set of queries every time the environment undergoes a significant change. Therefore, in Chapter 4, we propose models that attempt to make maximal use of already-available training data. In particular, we address the scenario where we have a set of existing verticals (associated with training data) and we are given a new vertical (associated with no training data). We call the existing verticals the *source* verticals and the new vertical the *target* vertical. Our goal is to learn a predictive model for the target vertical using only source-vertical training data.

In machine learning, the field of *domain adaptation* is concerned with algorithms that generalize from one domain to another using little or no training data in the new domain. This dissertation presents the first exploration of domain adaptation applied to resource selection. We investigate vertical selection models that exhibit two properties:

portability—the ability of a model to be trained once and then applied effectively to any *arbitrary* vertical, including one with no training data—and *adaptability*—the ability of a model to adjust its parameters to a *specific* vertical. We present portable models that can be trained on one set of verticals and applied to another. Also, we present an approach that adapts a portable model to a specific target vertical using no target-vertical training data.

We show that features can be divided into two classes: *portable features*, those that are correlated with relevance (in the same direction) across verticals and *non-portable features*, those that are correlated with relevance for a *specific* vertical, but not across verticals. We show that learning an effective portable model (one that make predictions with respect to any vertical) requires focusing on the most portable features. We present a way of identifying these portable features automatically. Furthermore, we show that, given target-vertical training data, the most predictive features are the non-portable ones. In other words, the features that help the most are precisely those that do not generalize across verticals. We demonstrate, however, that non-portable features can be harnessed using target-vertical predictions from a portable model.

As previously mentioned, most federated search research was conducted under the assumption that resources contain similar types of documents and use similar algorithms to retrieve content. To test the generality of a machine-learning approach to resource selection, in Chapter 7 we focus precisely on this type of federated search environment. While existing resource selection methods are expected to perform well in this environment, we demonstrate that a machine learning approach is more robust. It performs either at the same level or significantly better across a number of experimental conditions.

Among one of the research questions investigated in Chapter 7 is the effect of resource representation quality on resource selection performance across methods. Existing content-based methods make use of resource samples to predict resource relevance. Therefore, we might expect these methods to perform comparatively well when given access to high-quality representations. That is, when sample sets are large (relative to the resource) for those resources with many relevant documents. We investigate the importance of evidence-integration across various experimental conditions. For example, when given access to high-quality representations, does a machine learning approach learn to focus on content-based evidence? Likewise, when given access to impoverished representations, does it learn to focus on other sources of evidence (e.g., evidence derived from historical click-through data)?

Finally, we investigate whether a machine learning approach can be trained without human judgements of any kind. Our *zero judgement training* approach automatically generates training data using retrievals that merge content from *all* available resources. Our basic assumption is that, on average, a retrieval that merges content from every resource will be superior to one that merges content from only a few. Thus, we train a machine learning model to select those few resources whose merged ranking most closely approximates one that merges content from all.

Results Merging In terms of results merging, existing methods were developed under the assumption that an appropriate presentation of results is an unconstrained interleaving of results from different resources. For this reason, existing methods cast the task as score normalization—normalizing retrieval scores from different resources so that they are directly comparable and can be used to produce a single ranking.

In aggregated web search, results merging or *vertical results presentation* is the task of deciding where in the Web results to present the vertical results and it is subject to a number of layout constraints. Most importantly, results from the same vertical, if presented, must appear grouped together (either horizontally or vertically) in the final presentation. Therefore, the task cannot be merely formulated as score normalization.

We propose a novel formulation of the problem. Given a set of layout constraints, which we define explicitly in Chapter 5, we cast vertical results presentation as *block ranking*. A result *block* is defined as a sequence of Web or vertical results that must appear grouped together in the final presentation.

Based on this formulation, we propose a methodology for evaluating a particular presentation of results (i.e., a particular ranking of blocks). At a high-level, our approach proceeds as follows. First, given a query and a set of blocks, we collect (redundant) human preference judgements on block-pairs. Then, we use these judgements to construct a “gold standard” or *reference* presentation. This reference presentation is essentially a ranking of blocks that attempts to be consistent with preferences expressed on block-pairs. Finally, we propose that any alternative presentation for the query can be evaluated using a rank-based metric to measure its distance or similarity to the reference presentation.

Compared to evaluation methodologies adopted in prior work, our approach has several advantages. First, given a query, it requires a relatively few number of human judgements in order to evaluate *any* possible presentation of results. This means that it can be used to construct an evaluation testbed (a set of queries with block-pair preference judgements). In information retrieval, evaluation testbeds are important because they facilitate the direct comparison between alternative approaches. Second, our approach does not require an operational system with users.⁵ Finally, our approach is grounded on user preference behavior. Chapter 5 presents a user study that empirically validates the metric. We show that when assessors agree that one presentation is better than another, the metric agrees with the stated preference.

An evaluation metric that can automatically score alternative presentations for a query is not only essential for comparing systems, it is also essential for model building—for training a system to produce high-quality output. In Chapter 6, we propose machine learning methods that rank *blocks* in response to a query. In information retrieval, learning-to-rank methods are the state-of-the-art in document retrieval. These methods learn to rank individual documents based on features derived from the query (e.g., whether it contains a domain name [56]), the document (e.g., its PageRank [12]), and

⁵In an operational environment, algorithms can be tested based on implicit user feedback (clicks and skips) [28, 78].

the query-document pair (e.g., its BM25 score [80]). This dissertation presents the first investigation of learning-to-rank methods for the purpose of ranking blocks of results from different search services.

There are three novel components to this work. First, in terms of features, how do we represent a sequence of documents as a single unit of retrieval? In other words, how do we aggregate document evidence and query-document evidence across a *multiple* documents (i.e., those that compose a particular block)? Second, different verticals retrieve different types of results, each associated with a unique set of meta-data. For example, *news* results have publication date, *local* results have a location, *shopping* results have a product name. How do we rank blocks that are associated with different feature representations? Finally, even if a feature is common to multiple verticals, it may have a *vertical-specific* predictive relationship with relevance. For example, the number of retrieved results may be predictive evidence for *news*, but not for *weather*, which retrieves at most a *single* result. The query-term “weather” may be positive evidence for *weather*, but negative evidence for *local*. The presence of a city name in the query may be positive evidence for *local*, but negative evidence for *finance*. Thus, block-ranking may require methods that can exploit a *vertical-specific* relationship between features and relevance. Learning-to-rank methods rank documents by adopting a consistent feature representation and learning a feature-to-relevance relationship that is uniform across documents. Chapter 6 presents approaches that learn to rank different types of elements (i.e., blocks from different verticals), where each type is associated with a unique set of features and, possibly, a unique feature-to-relevance relationship.

1.2.1 Summary of Contributions

The dissertation makes the the following contributions to the field of information retrieval.

1. **An evaluation of existing resource selection methods on the task of vertical selection.** We show that existing methods are not well-suited for an aggregated web search environment.
2. **A machine-learning approach to resource selection capable of integrating multiple sources of evidence.** We show that a machine-learning approach outperforms existing methods in two different environment: in an aggregated web search environment and in a more traditional federated search environment.
3. **An understanding of the contribution of feature-integration in resource selection.** We investigate the effectiveness of various different types of evidence: content-based evidence (derived from resource content), query-log evidence (derived from resource-specific query-traffic), click-through evidence (derived from previously seen clicks on resource content), and query-string evidence (derived from properties of the query). We demonstrate that no single feature type is exclusively responsible for performance. Furthermore, we demonstrate the importance of differ-

ent types of features under different experimental conditions (e.g., given different levels of resource-representation quality).

4. **The first investigation of domain adaptation applied to resource selection.** We contribute a set of machine learning models that can generalize across verticals, as well as adapt their parameters to a *specific* vertical that is associated with no training data.
5. **A new formulation of the vertical results presentation task.** We formulate the task as *block-ranking*.
6. **A new methodology for evaluating aggregated web search results.** Our methodology includes an interface for collecting block relevance judgements in a quick and inexpensive manner and a way of using these judgements to conduct off-line metric-based evaluation.
7. **An evaluation of machine-learning approaches to vertical results presentation.** These methods, viewed broadly, address the task of ordering different types of items, which have an inconsistent feature representation and a non-uniform (item-type-specific) relationship between features and the item's target rank.

1.3 Impact on Other Applications

Most of the dissertation focuses on aggregated web search, which, as previously mentioned, is characterized with two important properties: different resources retrieve different types of results (*result-type heterogeneity*) and different resources use different retrieval algorithms (*retrieval-algorithm heterogeneity*). While we focus on aggregated web search, these two types of heterogeneity occur in other federated search environments. Thus, the solutions presented in the thesis may be applicable to other federated search tasks.

Library search, for example, must support single-point access to various types of media, such as books, encyclopedic entries, archived news and magazine articles, film, and conference proceedings. Likewise, desktop search must support single-point access to various file-types, such as documents, spreadsheets, presentations, images, email, schedule information, and applications. Kim and Croft [60] approached the task of target file-type prediction in desktop search as a type of resource selection problem.

Mobile search is another environment where aggregation takes place. Currently, commercial mobile search providers support many of the same vertical search services familiar to Web search users.⁶ One important difference with mobile search, however, is the availability of rich information about the user's current context. Contextual information includes, for example, the time of day, the user's location, and whether the user is stationary or moving. Additionally, historical data can provide clues about the user's activity. For example, are they likely to be at home, at work, or in an unfamiliar

⁶For a demonstration, see <http://mobile.yahoo.com/search>.

location? Aggregated mobile search may benefit from methods that can easily integrate a wide range of contextual evidence.

The search scenarios mentioned above are all situations where the user issues a query. Aggregation also occurs in situations where there is no explicitly stated information need. On-line portals such as Yahoo!⁷ and AOL⁸ as well as on-line newspapers such as the New York Times⁹ provide visitors with access to a wide-range of content, for example, videos, images, blogs, international news, local news, movie news, real-state news, financial news, job listings, classifieds, personals, and even advertisements. Each type of content can be viewed as a type of vertical. An important task for the system, then, is deciding *which* type of content to display and *where* to display it. In the absence of a user query, we can imagine harnessing various types of user-interest data, for example, from bursts in the production of a particular type of content, from the user's preferences (if the user is a returning user), from queries issued to the portal's search engine (if one exists), from previous user interactions (clicks and skips), or from trending topics on blogs or on Twitter¹⁰.

1.4 Dissertation Outline

The remainder of the dissertation is organized as follows. Chapter 2 surveys prior work in federated search, including resource representation (Section 2.2), resource selection (Section 2.3), and results merging (Section 2.4). Chapters 3-6 focus on aggregated web search, where the goal is to incorporate relevant vertical results into Web search results. Chapter 3 evaluates a machine learning approach to vertical selection, the task of predicting *which* verticals to search for a given query. To address the limitation of requiring training data, Chapter 4 focuses on domain adaptation for vertical selection, where the task is to learn a predictive model for a new vertical using only existing-vertical training data. Chapters 5 and 6 focus on vertical results presentation, the task of predicting *where* to present results from different verticals. Chapter 5 describes an evaluation methodology for vertical results presentation and Chapter 6 focuses on machine learning methods that predict how to present results. To test the generality of our vertical selection approach (Chapter 3), in Chapter 7 we focus on resource selection in a more traditional federated search environment, where resources retrieve similar types of documents and use similar retrieval algorithms. Finally, Chapter 8 concludes the dissertation by summarizing the major results and contributions and highlighting potential areas for future work.

⁷<http://www.yahoo.com>

⁸<http://www.aol.com>

⁹<http://www.nytimes.com>

¹⁰<http://www.twitter.com>

Prior Research in Federated Search

Federated search, or *distributed information retrieval*, is the problem of automatically searching across multiple distributed collections or resources. Aggregated web search—the detection and integration of relevant vertical content into the Web search results—can be viewed as one type of federated search task. The goal of this chapter is to survey prior research in federated search. Throughout the rest of this dissertation, we use algorithms described in this chapter in two ways: to generate features for our models and to serve as baselines for comparison.

Most prior federated search research partitions the problem into three separate, but interrelated, tasks. *Resource representation* is the task of gathering information about the contents of each resource. *Resource selection* is the task of deciding which resources to search for a given query. *Results merging* is the task of combining results from different resources—those selected—into a single document ranking. It is often impractical (if not impossible) to issue the query to every available resource. Therefore, the goal of resource selection is to determine which *few* resources are most likely to have relevant content. Resource representation and selection go hand-in-hand. Ultimately, the goal of resource representation is to inform resource selection. Resource representation happens off-line, while resource selection and results merging happen every time a query is issued to the system.

In the following sections, we review prior work in representation (Section 2.2), selection (Section 2.3), and merging (Section 2.4). We focus on approaches intended for an uncooperative environment.

2.1 Cooperative vs. Uncooperative Federated Search

Prior research in federated search often distinguishes between a *cooperative* and an *uncooperative* environment. In a cooperative environment, resources provide the system with all the information needed to perform federated search accurately and efficiently. A cooperative environment may occur, for example, when the system and its target resources are operated by the same search company. In an uncooperative environment, resources provide the system no more than the functionality they provide their human users: a search interface. An uncooperative environment may occur, for example, when the system searches across external digital libraries that cannot be crawled and centrally indexed

In a cooperative environment, federated search can be done using a cooperative fed-

erated search protocol such as STARTS [41]. The STARTS protocol standardizes how resources should publish content descriptions, defines a unified query language to retrieve documents from resources, and specifies result set statistics to be provided alongside search results to facilitate results merging. Cooperative protocols such as STARTS have the advantage that resource descriptions are complete and accurate. However, they have a few drawbacks. First, they require coordination between resource providers. Second, they assume that there exists a single description template that can effectively summarize the contents of every resource. Such a template may become overly complex as resources specialize on a wider range of content and media (e.g., news, blogs, images, video, items for sale). Third, because representation is not handled locally by the system, every resource is responsible for providing the system with complete and accurate content descriptions. In some cases, we may prefer a configuration in which the federated search system is exclusively responsible for representation. Perhaps for these reasons, most federated search research after Gravano *et al.*'s work on the STARTS protocol assumes an uncooperative environment.

2.2 Resource Representation

Resource representation has a dual objective: to gather information about the contents of each resource and to represent resource content in a way that informs selection. The manner in which resource content is represented (e.g., term-frequency information [98]) depends on the resource selection algorithm (e.g., language-model-based selection [98]).

Regardless of how the content is represented, in an uncooperative environment, resource statistics (e.g., term-frequency counts) must be collected by issuing queries to the resource and analyzing their results. This process is referred to as *resource sampling*. The end goal of resource sampling is to produce a sample set that is highly representative of unsampled, unseen documents within the collection. In general, resource sampling iterates over the following steps: a query is issued to the collection, a few documents are downloaded, the resource description is updated, and the process iterates. Callan and Connell's *query-based sampling* (QBS) approach derives single-term queries for sampling from the emerging resource description, which takes the form of term-frequencies for the set of documents downloaded so far [16]. Callan and Connell compared federated search retrievals using complete vs. partial descriptions (acquired using QBS) and concluded that they are probably indistinguishable to a human.

The research questions associated with resource sampling include: (1) how much to sample, (2) what to sample, (3) when to re-sample, and (4) how to make optimal use of sampled documents for representation (an ultimately selection).

2.2.1 How much to sample

Callan and Connell [16] show that effective resource descriptions, suitable for resource selection, can be produced by sampling as few as 300 documents per collection. Although

their results were obtained on testbeds with relatively small collections (with fewer than 50,000 documents), sampling about 300 documents per collection has become standard practice, even on testbeds with larger collections [24, 76, 91, 92, 94, 95, 96, 104]. To some extent, this was done to make results comparable with prior work. In fact, Shokouhi *et al.* show that federated search results, based on the final merged retrieval, improve when uniformly sampling about 1,000 documents per resource [92].

An alternative to uniformly sampling the same number of documents per resource is *adaptive sampling* [20, 92]. Shokouhi *et al.* [92] sample from each resource until the rate of previously unseen terms drops below threshold. They show that adaptive sampling outperforms uniformly sampling 300 documents per resource. However, during adapting sampling, every sample set consisted of more than 300 samples. Therefore, while adaptive sampling outperformed 300 documents per resource, it is unclear from this study whether it outperforms larger samples of *equal* size.

Caverlee *et al.* [20] propose adaptive sampling in two phases. In the first phase, an equal number of documents is sampled from each collection. In the second phase, the remaining sampling budget is reallocated by sampling more from collections predicted to have a weaker intermediate representation. Two budget-reallocation approaches outperformed uniformly sampling 300 documents: sampling proportional to the estimated number of documents in the collection and sampling proportional to the estimated vocabulary size of the collection. As opposed to Shokouhi *et al.* [92], Caverlee’s experiments on adaptive sampling were conducted with a fixed budget of $n \times 300$ total samples, where n is the number of collections.

To summarize, given a relatively small sampling budget (e.g., $n \times 300$ total samples), adaptive sampling outperforms uniform sampling. It is unclear, however, whether its performance improvement diminishes with a larger sampling budget.

2.2.2 What to sample

Besides deciding how much to sample, another question is what to sample, which in an uncooperative setting is instantiated as how to select queries for sampling. The traditional QBS approach is to derive sampling queries from the ongoing representation. The risk, however, is that this does not guarantee that the sampled documents will be typical of those *requested by users* at test time. Shokouhi *et al.* [92] show that probing collections using high-frequency query-log queries can produce more effective representations. Their hypothesis was that query-log queries can help bias the sample towards documents likely to be requested at test time and that this leads to more effective representations. Their results suggest this to be true. However, the same set of query-log queries was used to sample from every collection. This makes their results slightly inconclusive. It may be that the *single* set of query-log probe queries used in the experiment was particularly effective. A better evaluation would have used multiple sets of query-log queries and reported variance in performance.

2.2.3 When to re-sample

Ensuring that sampled documents are representative of unsampled documents in the resource is difficult when resources are dynamic—when documents are periodically added to and removed from the resource. This may occur for example in a news collection, where new content is produced daily. Shokouhi *et al.* [93] investigate re-sampling strategies in environments where the system has no prior knowledge of a resource’s update schedule. Several heuristics were explored. Re-sampling according to collection size (i.e., re-sampling larger collections more frequently) outperformed re-sampling according to level of query traffic (i.e., re-sampling higher query-traffic collections more frequently). Re-sampling outperformed sampling just once, showing that representations can become stale if collections change.

2.2.4 Combining collection representations.

The final research question is how to make optimal use of sampled documents. A risk of deriving content summaries from document samples is not representing meaningful content in the collection (e.g., missing important vocabulary terms). Ipeirotis and Gravano [50] address this problem by combining content summaries (in their case, term frequency information) from semantically-related collections. Their approach proceeds in two steps. First, each collection is assigned to a node in a topic hierarchy. A collection’s topical affinity is determined by issuing topically-focused queries to the collection and observing their hit counts. Second, each collection’s language model (used in Ipeirotis and Gravano’s resource selection approach) is smoothed with those assigned to related nodes in the hierarchy. We revisit this approach in Section 2.3.

2.3 Resource Selection

As previously mentioned, it is often impractical (if not impossible) to send the query to every resource. Therefore, the goal of resource selection is to decide which *few* resources are most likely to have relevant content. More formally, the goal is to decide which $k < n$ collections to select for a given query.

2.3.1 Problem Formulation and Evaluation

Most prior research assumes that k —the number of collections to select—is given. For this reason, resource selection is often cast as a resource *ranking* problem. If k is given, the system is only required to produce a ranking of n collections, from which the top k are selected and evaluated.

Given this formulation of the problem, resource selection is often evaluated based on the system’s ranking of resources. The most common metric is R_k , which evaluates a ranking of n collections up to some rank k . The assumption behind R_k is that the best a

system can do is to rank resources based on their number of *true* relevant documents (in descending order).

For a given query q and value k , let,

$$R_{k,q} = \frac{\sum_{i=1}^k |E_{i,q} \cap \mathcal{D}_q^+|}{\sum_{i=1}^k |B_{i,q} \cap \mathcal{D}_q^+|}.$$

Here, $E_{i,q}$ denotes the i^{th} resource in the algorithmic ranking, $B_{i,q}$ denotes the i^{th} resource in a relevant-document-based ranking, and \mathcal{D}_q^+ denotes the set of all document relevant to q . R_k is the average of $R_{k,q}$ across queries. R_k ranges between 0 and 1 and is easy to interpret—it corresponds to the (average) relevant-document-recall from the top k collections, normalized by the best possible recall given k .

Resource selection is only the first step in a two-step process: selection and merging. An evaluation based on R_k has the advantage that it isolates results merging performance from resource selection evaluation. However, precisely because of this, it is difficult to determine how values in R_k correlate with the quality of the end-to-end system output: the merged results. Selecting resources based on their number of relevant documents may not always be the best strategy. For instance, even if a resource has many relevant documents, its search engine may not *retrieve* them. And, even if it retrieves them, the merging algorithm may not rank them high. That is, the merging algorithm may handle documents from some collections better than others. Previous experiments show that maximizing R_k does not necessarily maximize the quality of the final merged results [90]. For this reason, the more recent trend is to hold the merging algorithm constant and to evaluate resource selection based on the final merged results, borrowing evaluation metrics from document retrieval (e.g., $P@10$ of the merged results for a given value $k < n$).

2.3.2 Modeling Query-Collection Similarity

Some methods rank collections by comparing the text in the query with the text in the *entire* collection, using metrics adapted from document retrieval. These methods model the collection as a single unit of retrieval and make no distinction between documents in the collection. For this reason, they are sometimes referred to as “large document models”. CORI [14] adapts INQUERY’s inference net document ranking approach to rank collections. Collection C_i is scored according to the query likelihood, $P(q|C_i)$, where the likelihood of query term $w \in q$ is given by,

$$\text{CORI}_w(C_i) = b + (1 - b) \times \frac{df_{w,i}}{df_{w,i} + 50 + 150 \times \frac{cw_i}{\text{avg}_{cw}}} \times \frac{\log\left(\frac{|\mathcal{C}|+0.5}{cf_w}\right)}{\log(|\mathcal{C}| + 1.0)},$$

where $df_{w,i}$ is the document frequency of query term w in C_i , cf_w is the number of collections containing query term w , $|\mathcal{C}|$ is the number of target collections, cw_i is the number of terms in C_i , avg_{cw} is the average number of words per collection, and b is the default belief.

Xu and Croft [113] and Si *et al.* [98] adapt approaches from language model based retrieval. Xu and Croft [113] rank collections based on the Kullback-Leibler divergence between the query and collection language model,

$$KL_q(C_i) = \sum_{w \in q} P(w|q) \log \left(\frac{P(w|q)}{P(w|C_i)} \right),$$

where $P(w|q)$ and $P(w|C_i)$ are the probabilities of query term w in the query and collection language models, respectively. Si *et al.* [98] rank collections based on the query generation probability given a collection’s language model,

$$P(q|C_i) = \prod_{w \in q} \lambda P(w|C_i) + (1 - \lambda)P(w|G),$$

where $P(w|C_i)$ is linearly smoothed using the probability of query term w in a global language model, $P(w|G)$.

Large document models have the advantage of being straight-forward adaptations of well-studied document ranking techniques. However, because they do not distinguish between documents in the collection, the collection language model is dominated by the larger documents. This is a potential disadvantage if we care about relevance at the *document* level. Also, they compare the text in the query with the text in the *entire* collection, making no distinction between documents that are related and unrelated to the query. Because of this, large document models favor collections with a high proportion of relevant-to-non-relevant documents. They may favor a small topically focused collection (related to the query), when a larger more topically diverse collection contains more relevant documents.

An alternative to modeling a collection as a single, query-independent language model is to focus only on those documents that are most similar to the query. Seo and Croft [86] score collections using the similarity between the query and the collection’s most similar m documents. More specifically, their approach scores collection C_i using the geometric average query likelihood from C_i ’s top m documents,

$$\text{GAVG}_q(C_i) = \left(\prod_{d \in \text{top } m \text{ from } C_i} P(q|d) \right)^{\frac{1}{m}}.$$

Seo and Croft applied this approach to the task of blog ranking, where they treat a blog as a collection of documents: its blog posts. Although they assume a complete view of the blog/collection, in an uncooperative federated search environment we can imagine applying this method using sampled documents. In the blog ranking task, the GAVG method outperformed a method that uses a global (query-independent) representation of the collection, fundamentally similar to the large document models presented above.

In blog ranking, users are primarily interested in blogs that are *consistently* relevant to the query across its blog posts. In this sense, topically-diverse blogs should be penalized irrespective of the query. In their results, Seo and Croft show that a large document

model, which uses a global representation of the blog, can be effectively used to penalize diversity in a blog. Above we claimed that large document approaches model the proportion of relevant-to-non-relevant documents in the collection. This result from Seo and Croft [86] supports our claim that large document models may disfavor a topically diverse collection, even if it has many relevant documents.

In the GAVG scoring function, parameter m is query- and collection-independent. However, the number of relevant documents in a collection may be greater than or less than m . In the next section, we introduce methods that address this limitation by directly modeling the relevant document distribution across collections.

2.3.3 Modeling the Relevant Document Distribution

GLOSS [40] scores a collection based on its expected number of relevant documents, as follows,

$$\text{GLOSS}_q(C_i) = |C_i| \times \phi(q, C_i)$$

where $\phi(q, C_i)$ denotes the query-collection similarity.

While GLOSS estimates the number of relevant documents in a collection, it does so using the *entire* collection’s language model. Essentially, it assumes that the collection’s language is evenly distributed across documents. vGLOSS [42], a vector-space extension of GLOSS, more explicitly addresses the fact that different documents cover different topics. vGLOSS scores collections proportional to the number of documents exceeding a query-document similarity threshold,

$$\text{vGLOSS}_q(C_i) = \sum_{d \in C_i} \mathcal{I}(\phi(q, d) > \tau) \times \phi(q, d),$$

where \mathcal{I} is the indicator function, $\phi(q, d)$ is the query-document similarity, and τ is a parameter. The *ideal* vGLOSS algorithm requires collections to publish their document term vectors in order to compute $\phi(q, d)$.

ReDDE [94] follows the same principle as vGLOSS. It prioritizes collections by their expected number of relevant documents. However, it differs from vGLOSS in two respects. First, it treats all predicted relevant documents equally, that is,

$$\text{ReDDE}_q(C_i) \approx \sum_{d \in C_i} \mathcal{I}(\phi(q, d) > \tau).$$

Second, it does not require collections to publish their document term vectors. It accomplishes this by using a *centralized sample index*, an index that combines samples from every collection.

ReDDE proceeds as follows. First, it conducts a retrieval from the centralized sample index. Given this retrieval, it uses a rank-based cut-off to predict a set of relevant sampled documents, all considered equally relevant. Finally, it assumes that each (predicted)

relevant sample represents some number of *unseen* relevant documents in the collection from which it originates.

ReDDE is described in detail because it is used in methods proposed in this thesis. ReDDE scores collection C_i according to,

$$\text{ReDDE}_q(C_i) = \mathcal{SF}_i \times \sum_{d \in S_i} P(\text{rel}|d, q) \quad (2.1)$$

where $P(\text{rel}|d, q)$ is the probability that document d is relevant to q and \mathcal{SF}_i is the *scale factor* of collection C_i , defined by,

$$\text{scale factor}(C_i) = \mathcal{SF}_i = \frac{|C_i|}{|S_i|}. \quad (2.2)$$

The scale factor quantifies the difference between the size of the original collection, $|C_i|$, and its sample set size, $|S_i|$. ReDDE models $P(\text{rel}|d, q)$ as a binary relevant/non-relevant variable based on document d 's *projected* rank in an *unobserved* full-dataset retrieval, $\hat{\mathcal{R}}_{q, \text{full}}(d)$. Document d 's projected full-dataset rank is estimated as the sum of scale factors associated with samples ranked above d in the retrieval from the centralized sample index,

$$\hat{\mathcal{R}}_{q, \text{full}}(d) = \sum_{j=1}^{\mathcal{R}_{q, S}(d)-1} \sum_{i=1}^{|\mathcal{C}|} \mathcal{I}(d_j \in C_i) \times \mathcal{SF}_i.$$

$P(\text{rel}|d, q)$ is modeled according to,

$$P(\text{rel}|d, q) = \begin{cases} 1 & \text{if } \hat{\mathcal{R}}_{q, \text{full}}(d) < (\tau \times |\mathcal{C}_{\text{full}}|) \\ 0 & \text{otherwise,} \end{cases}$$

where $|\mathcal{C}_{\text{full}}| = \sum_{C_i \in \mathcal{C}} |C_i|$ and τ is a parameter.

Between ReDDE and CORI, prior work shows that ReDDE is more robust. It performs either at the same level or significantly better than CORI across testbeds.

2.3.4 Modeling the Document Score Distribution

A *federated* retrieval (one that selects and merges results from $k < n$ collections) can be viewed as a substitute for a *full-dataset* retrieval (one that searches a single-index which combines all n collections). In fact, if we assume full-dataset retrieval to be effective on average, then we can formulate the resource selection task as that of selecting the collections whose merged ranking more closely approximates a full-dataset ranking. For example, we may want to score each resource based on its contribution to the top ranks of a full-dataset ranking. This is the underlying assumption made by the next set of algorithms: CRCS [90], SUSHI [104], and UUM [95].

Shokouhi's Central Rank-based Collection Selection (CRCS) [90] method is similar to ReDDE. The only difference is that while ReDDE models sample relevance using a binary

relevant/non-relevant function, CRCS uses a real value. Two functions are proposed: one decays relevance linearly, CRCS(l), and one decays relevance exponentially, CRCS(e), as a function of rank. One disadvantage of ReDDE is the importance of parameter τ —all sampled documents ranked above this threshold are considered *equally* relevant. CRCS makes more fine-grained judgments about document relevance. Given a retrieval from the centralized sample index, each sample’s contribution to its collection’s is proportional to its rank.

To model the full-dataset ranking, CRCS assumes that for each sample, with a particular relevance score, there are *scale factor* number of document in its collection with *exactly* the same score. Essentially, this imposes a limit on how well a full-dataset retrieval can be approximated. This is because the (unseen) documents represented by a sampled document must appear adjacent in the unobserved full-dataset retrieval. More fine grained approximations of a full-dataset retrieval can be made by estimating each collection’s full-dataset score curve. A collection’s *full-dataset score curve* is one that expresses (estimated) full-dataset score as a function of (estimated) full-dataset rank.

Si and Callan’s Unified Utility Maximization (UUM) method [95] proceeds in two phases: a training phase, which happens off-line, and a test phrase, which happens at query time. During the training phrase, the objective is to learn a logistic model that expresses a document’s probability of relevance as a function of its centralized sample index score. This is accomplished using a small set of training queries (e.g., 50) with relevance judgements on documents, which are scattered across collections. Each training query is issued to every collection and each collection’s top documents are downloaded and scored according to the centralized sample index statistics. After doing this for every training query, a logistic model is fit to predict relevance as a function of centralized sample index score.

During the test phase, the objective is to estimate, for each collection, a probability of relevance curve using all of its (mostly unseen) documents. Let $R_i(r)$ denote collection C_i ’s probability of relevance curve, where $r \in [1, |C_i|]$. $R_i(r)$ returns the probability of relevance of the r^{th} most relevant document from C_i . Given a probability of relevance curve from every collection, collections can be prioritized according to different criteria. For example, they can be prioritize by their total document relevance,

$$\sum_{r=1}^{|C_i|} R_i(r),$$

or the relevance of their top N documents, to maximize $\mathcal{P}@N$,

$$\sum_{r=1}^N R_i(r).$$

UUM approximates each collection’s probability of relevance curve R_i as follows. First, the query is issued to the centralized sample index. From this ranking, the algorithm extracts a sub-ranking corresponding to those documents sampled from C_i . Then, UUM assumes that between every two sampled documents adjacent in this sub-ranking

there are \mathcal{SF}_i documents in C_i whose retrieval score degrades linearly. Given this assumption, a full-dataset score curve can be constructed by linearly interpolating the scores from each of C_i 's samples, assuming \mathcal{SF}_i unseen documents between each point. Finally, UUM translates this full-dataset score curve into a probability of relevance curve using the logistic model learned in the training phase.

Like UUM, Thomas and Shokouhi's SUSHI approach [104] prioritizes collections based on their contribution to an approximated full-dataset retrieval. SUSHI differs from UUM in two respects. First, it does not translate centralized sample index scores to probabilities of relevance. Instead, it uses the centralized sample index scores directly. Second, it does not construct a collection's full-dataset score curve by linearly interpolating points. Instead, it uses curve fitting. Like UUM, SUSHI first issues the query to the centralized sample index. Next, it extracts the sub-ranking corresponding to C_i 's samples and scales this sub-ranking by assuming \mathcal{SF}_i unseen documents between adjacent points. Next, it approximates the score curve of all (mostly unseen) documents from C_i by fitting a curve to the observed points. SUSHI fits a linear, exponential, and logarithmic function and chooses the one which fits best according to the R^2 metric. SUSHI then interleaves the points from the different collection-specific score curves to approximate a full-dataset retrieval. To maximize $P@N$, resource selection is done by prioritizing collections by their contribution to the top N full-dataset retrieval results.

In terms of performance, both SUSHI and UUM outperform ReDDE on testbeds where relevant documents are evenly distributed across collections. The gap narrows on testbeds where the collection size distribution is skewed and the larger collections contain more relevant documents. SUSHI and UUM have not been directly compared in prior work.

2.3.5 Combining Collection Representations

Many of the approaches discussed so far derive evidence from *sampled* documents rather than the full collection. A significant risk with sampling is misrepresenting the collection content, particularly if the collection is large relative to the sample set. We discuss two methods that aim to minimize the negative effect of incomplete content descriptions by combining content descriptions from semantically-related collections.

Ipeirotis and Gravano [50] propose the following approach. First, collections are assigned to nodes in a topic hierarchy, similar to the Open Directory Project (ODP) hierarchy.¹ This is accomplished by issuing topically-focused queries to each collection and observing their hit counts. Then, each category node is treated as a large *pseudo-collection*, represented by combining the representations of those collections assigned to it directly (assigned to the same node) or indirectly (assigned to a more general node). In general, the manner in which the representations of related collections are combined will depend on the resource selection method. Specifically, Ipeirotis and Gravano use a large document model. Therefore, pseudo-collections were represented by summing the term

¹<http://www.dmoz.org>

counts from those collections assigned to it. At test time, the selection process traverses down the hierarchy, at each step selecting a pseudo-collection using a non-hierarchical resource selection method (e.g., CORI [14]). The search trickles down the hierarchy until selecting a pseudo-collection with k collections or less.

In later work, Ipeirotis and Gravano [51] combine collection descriptions using a smoothing technique known as *shrinkage*. Similar to the method above, collections are assigned to nodes in a topic hierarchy and the term statistics from collections assigned to the same node are combined to form node-specific language models. Then, a collection’s language model is smoothed by linearly interpolating its language model with that of its related categories, from its parent up to the root node (a language model of the full vocabulary). The mixing coefficients for a given node in the tree are found using Expectation-Maximization. At test time, collections are selected using a non-hierarchical “large document” resource selection algorithm, using each collection’s “shrunk” content description.

2.3.6 Modeling Search Engine Effectiveness

So far, we have discussed methods that rank collections based on their content. Content-based methods, however, ignore the fact that even if a resource has relevant content, its search engine may not *retrieve* it. The following methods address this limitation by considering the search engine’s expected effectiveness in resource selection decisions.

To our knowledge, the first method to consider retrieval effectiveness in federated search was proposed by Voorhees *et al.* [106]. This method has a training and test phase. During training, the system issues every training query to every resource and observes the number of relevant documents retrieved by each. Then, given a previously unseen (test) query q , the system ranks resources based on their average number of *retrieved* relevant documents for the set of training queries most similar to q . The test query’s nearest-neighbor (training) queries are obtained using the cosine similarity in the query-term space. Computing the query-query similarity based on query-term overlap risks not finding enough training queries similar to the test query. A second approach uses query clustering. For each search engine, all training queries are clustered using as query similarity metric the overlap in the top documents retrieved. Notice that query-clusters could have also been collection independent. In this case, however, they were collection *dependent*.² Given a collection-dependent clustering, each cluster is assigned an effectiveness score based on the average number of relevant documents retrieved by queries assigned to it. Finally, at test time, given a previously unseen query, resources are prioritized by the effectiveness of the query cluster most similar to it. Clusters are represented by their term centroid. Both approaches implicitly model search engine effectiveness by focusing on documents *retrieved* from the collection.

Si and Callan propose the Returned Utility Maximization (RUM) model [96], an extension of UUM that models search engine effectiveness. RUM estimates a search en-

²This choice was not evaluated empirically.

engine’s effectiveness by comparing its retrievals with those produced with a presumably effective ranking algorithm on a subset of its documents. RUM proceeds as follows. First, a set of J training queries is used to build mapping functions specific to collection C_i of the form,

$$\phi_{ij} : \mathcal{R}_{C_i, q_j}(d) \rightarrow \mathcal{R}_{S, q_j}(d),$$

Mapping function ϕ_{ij} is constructed by issuing training query q_j to collection C_i (producing ranking \mathcal{R}_{C_i, q_j}), downloading the top documents, and scoring these using a supposedly effective retrieval algorithm according to the centralized sample index statistics (producing ranking \mathcal{R}_{S, q_j}). From these two rankings, mapping function, ϕ_{ij} , maps ranks in C_i ’s (possibly-ineffective) retrieval to ranks in the (assumed to be) effective retrieval.

Recall that UUM constructs, for every collection C_i , a probability of relevance curve, $R_i(r)$, where $r \in [1, |C_i|]$. Function $R_i(r)$ specifies a document’s estimated probability of relevance as a function of its rank in a retrieval of collection C_i . RUM models retrieval effectiveness by injecting the mapping functions, ϕ_{ij} , learned in the training phrase, into the collection selection criterion. For example, if selecting collections by their total number of relevant documents, instead of maximizing,

$$\sum_{r=1}^{|C_i|} = R_i(r),$$

RUM maximizes,

$$\sum_{r=1}^{|C_i|} = \frac{1}{J} \sum_{j=1}^J R_i(\phi_{ij}(r)).$$

If selecting collections by their expected number of top N relevant documents, instead of maximizing,

$$\sum_{r=1}^N = R_i(r),$$

RUM maximizes,

$$\sum_{r=1}^N = \frac{1}{J} \sum_{j=1}^J R_i(\phi_{ij}(r)).$$

Injecting the mapping functions into the collection selection criterion has the effect that documents from collections with an ineffective search engine will tend to have lower probabilities of relevance and therefore lower ranks in the approximated full-dataset retrieval.

At test time, RUM combines a collection’s J mapping functions with equal weight. In other words, while each mapping function is specific to a query, RUM’s application of these J mapping functions is query-independent. In this respect, Voorhees *et al.* [106] goes one step further by considering the similarity between the test query and the training queries.

2.3.7 Query Classification

Query classification is the task of assigning queries to one or more categories, and is done, for example, when processing different query classes using customized retrieval algorithms or representations [57]. If we consider target collections as query categories, we may view resource selection in federated search as a type of query classification problem. The major difference is that in resource selection each target “category” is associated with a document collection that can be used to inform selection.

Because queries are terse, query classification methods often generate evidence from sources other than the query string. Among these resources are query-logs [6], query click-through data [108], and documents associated with target categories [57, 88, 89].³ In spite of the similarity between query classification and resource selection, the most successful query classification approaches differ from most resource selection methods in one respect—they tend to combine different types of evidence. As we have seen, most resource selection methods score collections using a *single* ranking function and derive evidence *exclusively* from collection documents.

Bietzel *et al.* [6] classify queries into semantic categories using a large (unlabeled) query-log and a technique known as *selectional preference* (SP). According to SP, the query “interest rates” belongs to target category *finance* because the terms “interest” and “rates” co-occur with terms known to be finance-related. Bietzel *et al.* combine a SP-based classifier with two supervised classifiers: a rote classifier and a bag-of-words text classifier. A combination approach outperforms all three base classifiers.

Wen *et al.* [108] introduce several query-similarity metrics. Although they focus on query clustering, their similarity metrics could be used in any memory-based approach to classification, such as K-Nearest Neighbor (KNN). They explored query similarity based on the query string, degree of click overlap, and the average pair-wise text similarity between clicked documents. Their most effective similarity metric, evaluated on clustering, was a combination of these three sources of evidence.

Kang and Kim [57] categorize queries by type of information need: *informational* vs. *known-item*. To inform classification, they harvested documents likely to be relevant to each query type. Web documents were assigned to either a “known-item query” collection or an “informational query” collection based on URL length—short URLs tend to be website entry pages which tend to satisfy known-item queries. Kang and Kim focus on two types of evidence: hit counts derived from these two collections and features derived from the query string. A linear combination of all features outperformed all single-evidence baselines.

Shen *et al.* [89] also use a meta-classification approach, and, like Kang and Kim [57], use corpus-based evidence. Each target category was automatically mapped (based on title overlap) to a set of “intermediate” categories (e.g., ODP categories), each associated with a set of documents. The end result is a corpus of documents associated with each target category. These category-specific collections were used to produce three base clas-

³Approaches in this last category use techniques similar to those used in resource selection.

sifiers. The first classifier resembles ReDDE [94]. The query is issued to an index of all category-specific collections combined and the classifier predicts the category most frequently represented in the top-ranked documents. The second method uses a category’s related documents as training data for a multi-class bag-of-words text classifier that is applied to the query string to make a category prediction. The third classifier makes predictions based on the query likelihood given the category language model, derived from its associated documents. In this respect, it resembles the large document model presented in Si *et al.* [98], with one major difference: intermediate categories are not mapped to a single target category. Instead, this method derives a “soft” intermediate-to-target category mapping using the similarity between associated documents. Again, their best-performing approach was a combination of these three classifiers.

In contrast to these approaches, Li *et al.* classify queries into query intent categories *product* and *jobs* and propose a technique that derives evidence exclusively from the query string [67]. Instead of enriching the query representation using external resources, they increase the amount of training data using a query click graph. Query labels are propagated to unlabeled queries with clicks on similar documents. Their semi-supervised approach outperformed a baseline trained only on the labeled “seed” queries. Focusing on query string features is efficient. Feature enrichment requires, for example, issuing the query to a collection of category-representative documents. However, query string features have a potential shortcoming. If the language associated with a particular category shifts significantly, the model may need to be retrained. This may not be the case when using external evidence such as the number of times the query appears in a set of category-related documents. A trained model may be able to maintain its performance as long as external resources change in order to reflect the concept drift.

2.3.8 Machine Learned Resource Selection

Prior to the work presented in this dissertation, Xu and Li [112] are the only ones to cast resource ranking as a machine learning problem. Collections are represented as feature vectors, a combination of query-independent *static features* (i.e., the size of the collection, level of query traffic) and query-dependent *dynamic features* (i.e., the query hit count in the collection, the query hit count in anchor text linking to a collection document). Two machine learning approaches were evaluated: combining collection-specific binary classifiers (i.e., ranking collections by classifier’s confidence value of a positive prediction) and a rank-learning approach. Rank-learning outperformed classification and both machine learning methods outperformed a CORI baseline.

Although this is a result in favor of machine learned resource selection, the work has some limitations. First, their results cannot be easily compared with previous work. Algorithms were evaluated by their ability to prioritize collections using a five-point relevance scale. Most prior resource selection work is evaluated either based on a strict ordering of resources using R_k or based on the quality of the merged document ranking. Second, both machine learned methods and CORI used different types of evidence. Therefore, it is unclear whether the machine learned methods are inherently superior

to CORI or used evidence better-suited for this particular testbed. A better comparison would have used CORI as one type of dynamic feature.

The work conducted for this dissertation (particularly that presented in Chapter 7) influenced other researchers to consider casting resource selection as a supervised classification task. Hong *et al.* [48] also train logistic regression models, as we do. Their work differs from ours in several respects. First, a set of binary resource-relevance labels were generated by observing the number of relevant documents (judged manually) retrieved from each resource for a set of training queries. Secondly, they limited their features only to existing content-based resource selection methods (e.g., CORI [14], GAVG [86]). Finally, and more importantly, they propose methods that exploit the relatedness between different resources. The idea is to favor a resource for selection if an “independent” model assigns a high prediction probability to those resources that are the most similar. Several (query-independent and query-dependent) resource-similarity measures were evaluated. The best resource-similarity measure was found to be query-dependent.

2.4 Results Merging

Results merging is the task of integrating results from different resources—those selected—into a single merged ranking. Even when resources adopt a similar retrieval algorithm, they often use different representations (e.g., stemming, stopword removal) and have different corpus statistics (e.g., *idf* values). For these reasons, resource-specific retrieval scores (for the same query) may not be directly comparable across resources. To address this problem, the goal of results merging is to perform score normalization. That is, to transform each retrieved document’s resource-specific score (not comparable across resources) into a resource-general score (comparable across resources). Results from different resources can then be ranked based on their normalized scores. We review two widely-used score-normalization techniques.

CORI-merge [14] assigns each retrieved document a collection-general score that is a function of its collection-specific score and its collection’s resource selection score. CORI-merge scores document d , originating from C_i , according to,

$$\mathcal{S}_{CORI}(d) = \frac{\hat{\mathcal{S}}_i(d) + 0.4\hat{\mathcal{S}}_i(d)\hat{\mathcal{S}}_{\text{coll}}(C_i)}{1.4},$$

where $\hat{\mathcal{S}}_i(d)$ is document d ’s collection specific score (scaled to zero min and unit max) and $\hat{\mathcal{S}}_{\text{coll}}(C_i)$ is collection C_i ’s resource selection score (scaled to zero min and unit max).

CORI-merge applies the same normalization function to all collections and all queries. In contrast, Si and Callan’s *semi-supervised learning* (SSL) approach [97] learns a different transformation model for each query-collection combination. Supposed we had access to a centralized index of all collection content. A retrieval from this centralized index would give us a resource-general score for every document in every resource. While we may not have access to a centralized index of all content, we may have access to a centralize *sample* index that combines samples from every resource. SSL proceeds as follows.

First, it issues the query to every selected resource and to the centralized sample index. Then, for each selected resource, it uses linear regression to learn a transformation between a result’s resource-specific score and it’s centralized sample index score. To learn this transformation, it uses as training data pairs of scores associated with documents that are both retrieved from the resource and present in the centralized sample index. In the absence of retrieved documents that are also in the centralized sample index, it downloads a few results and scores them using statistics from the centralized sample index.

2.5 Summary

Several trends are noticeable from surveying prior research in federated search. In terms of prior resource selection work, we can draw the following conclusions.

- Most resource selection methods score collections using a single function with relatively few parameters that are tuned manually on a few training queries (e.g., 50).
- In most prior work, resource selection is formulated as a resource ranking problem. That is, given a set of n collections and a query q , the algorithm selects $k \leq n$ collections from which to retrieve documents. The objective is to produce the best retrieval possible for a given value k . In theory, the quality of the merged results may not always improve with greater values of k . No prior work addresses the problem of selecting, for a given query, the optimal k value.
- No single resource selection method outperforms the rest across all testbeds. This may be in part because some methods can be tuned particularly well to a testbed with certain properties. For example, if the evaluation is based on R_k , ReDDE can be tuned particularly well to an environment where the collection size is skewed and most of the relevant documents are in the larger collections.
- Most resource selection methods derive evidence exclusively from collection documents, for example, by comparing the text in the query with all the text in a collection (e.g., CORI [14]) or by estimating the number of relevant documents in a collection using samples (e.g., ReDDE [94]). Other types non-content-based evidence, derived, for example, from queries previous issued to a collection or from the topic of the query, have been ignored.
- There has been a divide between resource selection and query classification work. While most resource selection methods focus on a single source of evidence (mostly collection samples), the most successful query classification approaches combine sources of evidence. Evidence-integration approaches have not been applied to the task of resource selection.

In terms of prior results merging work, we can draw the following conclusions.

- Existing approaches to results merging assume that results from different collections can be interleaved freely in an unconstrained fashion. For this reason, the task is often formulated as score normalization.
- To make scores from different resources comparable, the state of the art approach, SSL [97], assumes that samples from different resources can be combined in a centralized sample index and that each resource's retrieval algorithm can be modeled internally within the system using a single scoring function.

Vertical Selection

In addition to web search, commercial search providers maintain many different search services customized for a particular type of information need. These specialized search services (referred to as *verticals*) include, for example, search for news, images, video, local businesses, movie times, items for sale, and weather forecast information, to name a few. There are two ways that users can access vertical content. In some cases, if the user wants results from a particular vertical, the query can be issued directly to the vertical search engine. In other cases, however, a user may not know that a vertical is relevant or may want results from multiple verticals at once. For these reasons, an important task for commercial search provider is the detection and integration of relevant vertical content in response to a web search query. This task is referred to as *aggregated web search*.

Aggregated web search can be viewed as a type of federated search, reviewed extensively in Chapter 2. Like federated search, it can also be decomposed into two separate tasks. It is often impractical, if not impossible, to issue the query to every vertical in order to decide which (if any) to present to the user. Therefore, the first task, *vertical selection*, is to predict (using only pre-retrieval evidence) *which* verticals are likely to be relevant. The second task, *vertical results presentation*, is to predict *where* in the Web results to present or embed the vertical results. In this chapter, we focus on the first task: vertical selection.

Vertical selection is related to the task to *resource selection* in federated search (Section 2.3). Existing solutions to resource selection divide into two classes. So-called large document models (Section 2.3.2) treat resources as large documents and adapt well-studied, text-based document retrieval algorithms to predict resource relevance [14, 98, 113]. On the other hand, small document models (Sections 2.3.3 and 2.3.4) first conduct a retrieval from an index that combines sampled documents from every resource, and then predict resource relevance based on sample relevance [90, 94, 95, 96, 104].

While vertical selection can be viewed as a type of resource selection, the aggregated web search environment violates two of the major assumptions made by state-of-the-art resource selection methods. Both large- and small-document models were designed for environments where resources retrieve similar types of (text-rich) documents and use a similar retrieval algorithm, which can be approximated internally by the federated search system. In other words, existing resource selection methods assume *result-type* and *retrieval-algorithm homogeneity* across resources. Neither of these assumptions are

¹This work was conducted during an internship in Yahoo! Montreal and was published in SIGIR 2009 with co-authors Fernando Diaz, Jamie Callan, and Jean-François Crespo [3].

true in an aggregated web search environment. First, different verticals retrieve items with different levels of text-richness (e.g., *news*, *images*). Second, because they are carefully tuned to satisfy a unique type of information need, different verticals use different retrieval algorithms. This suggests that need for a different approach to vertical selection.

Furthermore, existing resource selection methods have a second potential limitation: they derive evidence *exclusively* from collection content (usually sampled) and do not provide a means to easily incorporate other types of non-content-based evidence. Partly because the environment is not completely uncooperative, in an aggregated web search environment the system may have access to other sources of evidence. Some verticals, for example, are associated with a vertical-specific search interface through which users can issue queries directly to the vertical. Therefore, vertical query-traffic is another potentially useful source of evidence. Some verticals focus on a specific topic (e.g., *health*, *auto*, *travel*) or type of media (e.g., *images*, *video*). Thus, it may be possible to detect vertical relevance from the query string alone, for example, based on the query topic (e.g., “swine flu” → *health*) or based on the presence of a particular named entity type (e.g., “bmw reviews” → *auto*) or query-term (e.g., “obama pictures” → *images*).

To address these potential limitations, we cast vertical selection as a machine learning problem. That is, we draw on machine learning as a way to combine *multiple* sources of evidence as input features and learn a predictive relationship between features and vertical relevance using training data. In particular, we exploit three different types of features. Vertical *corpus features* derive evidence from (sampled) vertical content. Vertical *query-log features* derive evidence from vertical-specific query traffic. Finally, *query features* derive evidence from the query string. Vertical corpus and query-log features enrich the query representation beyond the query string and focus on two potentially complementary sources of evidence: corpus features relate to content production (i.e., content in the vertical) and query-log features relate to content demand (i.e., content sought by users).

Another assumption of existing resource selection methods is that evidence is similarly predictive of relevance across *all* resources. In this chapter, as in the rest of the dissertation, we relax this assumption. We propose a classification framework that combines vertical-specific binary classifiers, each trained independently on its own training data (its own set of queries with human relevance judgements with respect to one vertical). Thus, the classification-framework is allowed the freedom to learn a vertical-*specific* relationship between features and vertical relevance. Consider our query features, mentioned above. Some of these features exploit, for example, the topical category of the query (determined automatically using a query classifier)—if the query is health-related, the *health* vertical is relevant. Because verticals focus on different topics, exploiting this type of evidence requires learning a vertical-specific relationship between query topic and vertical relevance.

Our objectives in this chapter are to investigate the importance of evidence-integration in vertical selection and to compare our machine-learning, evidence-integration approach with state-of-the-art resource selection methods, which, by design focus on a single type of evidence.

3.1 Formal Task Definition

Let \mathcal{V} denote the set of all candidate verticals and \mathcal{Q} denote the set of all queries. We assume that each query $q \in \mathcal{Q}$ is associated with a set of true relevant verticals $\mathcal{V}_q \in \mathcal{V}$. In aggregated web search, it is possible for no vertical to be relevant to query q . In this case, the optimal choice for the vertical selector is to predict *no relevant vertical* and simply show the web results. This may occur, for example, when the query is a navigational query. We denote the absence of any true relevant vertical by $\mathcal{V}_q = \emptyset$.

In this chapter, we address the task of *single vertical selection*. Given q , the objective is to predict a *single* relevant vertical, $\tilde{v}_q \in \mathcal{V}_q$, if one exists, and to predict *no relevant vertical*, $\tilde{v}_q = \emptyset$, otherwise.

In general, it is possible for query q to be associated with *multiple* true relevant verticals. In other words, it is possible for $|\mathcal{V}_q| > 1$. In this sense, we have simplified the task. However, as described in Section 3.4, only about 30% of our evaluation queries were associated with more than one relevant vertical. The remaining 70% were associated with either a single relevant vertical or none.

3.2 Classification Framework

Our classification-based framework consists of $|\mathcal{V}| + 1$ independent binary classifiers: one for each vertical $v \in \mathcal{V}$ and one to predict that no vertical is relevant. Each binary classifier is trained independently (i.e., using potentially different training data) to make a binary prediction with respect to its class (i.e., its vertical or the *no relevant vertical* class). We train logistic regression models using the Liblinear Toolkit [35].¹ We chose logistic regression based on its accuracy and short training time on other large-scale classification tasks [68] and because logistic regression can be used to output prediction probabilities, which we use as describe below.

In order to make a single prediction per query (either a *single* relevant vertical or that no vertical is relevant) we combine our independent binary classifiers as follows. Let $P_v(y = 1|q)$ denote the probability of a positive prediction from vertical v 's classifier and let $P_\emptyset(y = 1|q)$ denote the probability of a positive prediction from the *no relevant vertical* classifier. If the most confident positive prediction is from the *no relevant vertical* classifier, then we predict *no relevant vertical*. Otherwise, we predict the most confident vertical prediction if it exceeds a predefined threshold τ . In other words, in the absence of a confident vertical relevance prediction, we default to predicting *no relevant vertical*. Parameter τ is tuned on validation data.

3.3 Features

We exploit three different types of features. Corpus features derive evidence from sampled vertical documents or from external documents associated heuristically with each

¹Available at <http://www.csie.ntu.edu.tw/~cjlin/liblinear/>

vertical. The idea is to use the predicted relevance of vertical documents to help predict the relevance of the vertical. Query-log features derive evidence from vertical-specific query-traffic. These features exploit the similarity between the query and those issued to the vertical directly by users, which describe the types of information needs satisfied by the vertical. Query features derive evidence from the query-string. As opposed to corpus features and query-log features, the value of a query feature is independent of the vertical under consideration. These features, exploit, for example, the query topic or the presence of a particular named entity type.

3.3.1 Corpus Features

Corpus features help predict vertical relevance based on the (predicted) relevance of documents associated with the vertical. We investigate two methods for associating documents with a vertical. One option is to sample results directly from the vertical. In federated search, resource sampling is often used for resource-representation [16]—a resource is represented by a sample of documents intended to be typical of unseen documents within the resource.

Generating corpus features from vertical-sampled documents has one potential limitation. At the core, corpus features derive evidence of vertical relevance from sample relevance. However, most methods for predicting sample relevance are text-based (i.e., they focus on the query-sample text-similarity). This is a problem for verticals that retrieve text-impooverished items (e.g. *images, video, maps*). For this reason, in addition to deriving corpus-features from documents sampled directly from the vertical, we also derive corpus features from text-rich documents that are external to the vertical, but associated with the vertical using manual heuristics.

In the next two sections, we describe our two methods for producing corpora intended to represent the contents of each vertical. First, we describe our method for sampling vertical documents. Then, we describe our method for associating external documents with a vertical. Each method produces a set of vertical-representative corpora, which are then used to produce our corpus features.

Direct Vertical Sampling

As discussed in Section 2.2, *query-based sampling* is a method for gathering a representative set of samples from every resource, which are then used to inform resource selection. The idea is to iteratively issue queries to the resource and download documents. In the original query-based sampling algorithm, each query used for sampling is derived from those documents downloaded in previous iterations [16]. Shokouhi *et al.* [93] showed that sampling using popular query-log queries biases the sample towards documents more likely to be requested by users and can improve resource selection accuracy. We follow a similar approach, with one major difference. While Shokouhi *et al.* used the same set of web search queries to sample from every collection, we use different sets of queries, each produced from the vertical’s own query-log.

We use the following sampling method. First, we issue each of the 1,000 most frequent (non-stopword) query-log unigrams as a query and downloaded the top 100 documents for each. Then, we uniformly sample at most 25,000 documents from the union of these sets. Sample sets of 25,000 documents are much larger than those used in prior federated research [24, 76, 91, 92, 94, 95, 96, 104]. However, some of our verticals are also much larger than resources investigated in prior work. For example, the *news* vertical contained approximately 15.4M documents. Furthermore, one reason prior work used sample sets of about 300 documents was because in an uncooperative environment the system may have a limit on the number of queries it can issue to the resource. In an aggregated web search environment, however, we do not have this restriction.

Sampling using vertical query-log terms has two potential advantages. First, it decouples sampling queries from sampled documents. This may be beneficial when samples are text-impooverished. Second, it biases the sample towards documents most likely to be requested by users. This may be beneficial when the vertical is large and most of its content is rarely requested, either because it is outdated, of low quality, or not of general interest. We denote the set of documents sampled directly from vertical v by $S_v^{vertical}$.

Associating External Documents with Verticals

To associate external documents with each vertical, we adopt an approach similar to that of Shen *et al.* [88, 89]. That is, we make use of a body of documents that has been annotated with category information: Wikipedia.² Each Wikipedia article is associated with a set of Wikipedia categories. Using the terminology from Shen *et al.* [88, 89], our objective is to associate each vertical with a set of intermediate (Wikipedia) categories. By doing this, each document associated with a vertical’s set of intermediate (Wikipedia) categories is associated with the vertical. We associated intermediate (Wikipedia) categories with verticals using hand-crafted regular expressions. For example, a collection of documents associated with the *autos* vertical was harvested from all articles assigned a Wikipedia category containing any of the terms “automobile”, “car”, and “vehicle”.

We cannot claim that our regular expressions produce an optimal Wikipedia-article-to-vertical mapping. However, deriving evidence from vertical-related Wikipedia articles may have some advantages. Wikipedia articles are rich in text, have a consistent format, and are usually semantically coherent and on topic. We denote the set of Wikipedia articles mapped to vertical v by S_v^{wiki} .

Above, we describe our two methods for constructing vertical-representative corpora. Next, we describe our corpus features. We generated two types of corpus features: ReDDE.top features and retrieval effectiveness features.

ReDDE.top Features

The ReDDE algorithm, described in detail in Section 2.3, is a well-studied resource selection method. Given a query, ReDDE scores each resource based on its predicted number

²<http://www.wikipedia.org>

of relevant documents. To do this, it first conducts a retrieval from a centralized sample index, which contains a mix of documents sampled from every resource. Given this retrieval, it then uses a rank-based threshold τ to predict a set of relevant samples. Finally, it assumes that each predicted relevant sample represents some number of unseen relevant documents in its originating resource.

ReDDE can be viewed as a voting algorithm—each predicted relevant sample contributes a fixed number of votes in favor of its resource. The exact number of votes is based on the resource’s scale factor, which measures the difference between the resource size and the resource’s sample set size. A potential limitation of ReDDE is the importance of parameter τ . Every sample ranked above τ is considered *equally* relevant and, therefore, contributes an *equal* number of votes. For this reason, we used a slight variation of ReDDE, which we refer to as ReDDE.top.

Like ReDDE, ReDDE.top first conducts a retrieval from a centralized sample index. Then, it scores vertical v according to,

$$\text{ReDDE.top}_q^*(v) = \frac{|v|}{|\mathcal{S}_v^*|} \times \sum_{d \in \mathcal{R}_{100}} \mathcal{I}(d \in \mathcal{S}_v^*) P(q|d),$$

where $P(q|d)$ is document d ’s retrieval score given query q , $|v|$ is the number of documents in vertical v and \mathcal{S}_v^* denotes either $\mathcal{S}_v^{\text{vertical}}$ (the set of documents sampled directly from v) or $\mathcal{S}_v^{\text{wiki}}$ (the set of Wikipedia documents mapped to v). The major difference between ReDDE.top and ReDDE is that in ReDDE.top the number of votes that sample d contributes to its resource score is proportional to $P(q|d)$.

We used two distinct sets of ReDDE.top features: one set derived from vertical-sampled documents and one set derived from Wikipedia articles (mapped to each vertical heuristically). Each set was normalized separately, such that $\sum_{v \in \mathcal{V}} \text{ReDDE.top}_q^{\text{vertical}}(v) = 1$ and $\sum_{v \in \mathcal{V}} \text{ReDDE.top}_q^{\text{wiki}}(v) = 1$.

Retrieval Effectiveness Features

ReDDE.top has one potential disadvantage: it uses a single retrieval to score samples from different verticals. In this environment, however, verticals focus on documents with varying degrees of text-richness (e.g., *news*, *images*, *video*). Therefore, retrieval scores may not be directly comparable across vertical samples. For example, the retrieval algorithm may be biased towards samples from text-rich verticals. To minimize this type of bias, it is necessary to estimate the relevance of sampled content without directly comparing retrieval scores across collections.

Our solution is to index collection samples separately and then to use a retrieval effectiveness measure as proxy for the number of relevant samples in the sample set. Retrieval effectiveness prediction is the task of automatically detecting when a retrieval returns relevant content, for example, based on observable properties of the top-ranked documents. Our assumption is that a high retrieval effectiveness score means that the collection (in our case, the set of vertical-representative documents) has relevant content.

We used a retrieval effectiveness measure known as Clarity [25], which uses the Kullback-Leibler divergence to compare the language of the top ranked documents with that of the collection,

$$\text{Clarity}_q(C) = \sum_w P(w|q) \log \left(\frac{P(w|q)}{P(w|C)} \right),$$

where $P(w|q)$ and $P(w|C)$ are the query and collection language models, respectively. The query language model was estimated using the top 100 documents, \mathcal{R}_{100} , according to,

$$P(w|q) = \frac{1}{\mathcal{Z}} \sum_{d \in \mathcal{R}_{100}} P(w|d)P(q|d),$$

where $P(q|d)$ is the query likelihood score given document d , and $\mathcal{Z} = \sum_{d \in \mathcal{R}_{100}} P(q|d)$. Clarity’s assumption is that in an effective retrieval the top ranked documents will use language that is distinguished from topic-general language derived from the entire index. As with ReDDE.top features, we generate two sets of Clarity features: one set derived from vertical-sampled documents and one set derived from Wikipedia articles associated with a vertical. Each set was normalized separately, such that $\sum_{v \in \mathcal{V}} \text{Clarity}_q^{\text{vertical}}(v) = 1$ and $\sum_{v \in \mathcal{V}} \text{Clarity}_q^{\text{wiki}}(v) = 1$.

3.3.2 Query-Log Features

Query-log features derive evidence from vertical-specific query traffic. That is, they consider the similarity between the query and those previously issued directly to the vertical by users. The assumption is that queries issued directly to the vertical describe the types of information needs the vertical satisfies.

Query Likelihood

Query-likelihood features consider the query-generation probability given a language model derived from the vertical’s query-log. Let θ_v^{qlog} denote a language-model derived from vertical v ’s query-log. We generate one query-likelihood features per vertical, as follows,

$$\text{QL}_q(v) = \frac{1}{\mathcal{Z}} P(q|\theta_v^{qlog}),$$

where $P(q|\theta_v^{qlog}) = \prod_{w \in q} P(w|\theta_v^{qlog})$ and $\mathcal{Z} = \sum_{v \in \mathcal{V}} P(q|\theta_v^{qlog})$.

Vertical query-log language models were constructed using a year’s worth of vertical query-traffic. Additionally, to inform classification in the *no relevant vertical* class, we collected a month’s worth of queries issued to the Web search engine. We assume that most Web search queries do not seek vertical content. We used a month’s worth of Web queries (rather than year’s worth) because Web search has more query traffic than vertical search. The CMU-Cambridge Language Modeling Toolkit was used to build a

unigram language model from each query-log.³ Each language model’s vocabulary was defined by its most frequent 20,000 unigrams and we used Witten-Bell smoothing [110].

Query likelihood features were evaluated under two conditions: allowing and disallowing zero probabilities from out of vocabulary (OOV) terms. In the first condition, a single OOV query term results in a zero probability from the vertical. In the second condition, $P(\text{OOV}|\theta_v^{\text{qlog}})$ was estimated proportional to the frequency of terms not in vertical v ’s top 20,000 query-log terms.

Soft.ReDDE Features

Soft.ReDDE features are related to ReDDE.top features (the set of ReDDE.top features that uses external documents mapped heuristically to each vertical). Like ReDDE.top features, Soft.ReDDE features also use external (Wikipedia) documents. However, different from ReDDE.top features, which use a binary Wikipedia-article-to-vertical assignment, Soft.ReDDE uses a *soft* assignment. In other words, every Wikipedia article has some degree of membership to every vertical. The degree of membership is based on the similarity between the Wikipedia article’s language model and the vertical’s language model, derived from its query-traffic. Compared to ReDDE.top features, Soft.ReDDE features have two potential benefits. First, every Wikipedia article contributes, more or less, depending on its correlation, to a vertical’s score. Second, Soft.ReDDE features do not use a manual mapping between an article and a vertical, which may misrepresent the vertical.

Soft.ReDDE features were generated as follows. Let θ_d denote the language model of Wikipedia article d and θ_v^{qlog} denote vertical v ’s query-log language model. We computed the degree of membership ϕ between article d and vertical v using the Bhattacharyya correlation [7],

$$\mathcal{B}(\theta_d, \theta_v^{\text{qlog}}) = \sum_w \sqrt{P(w|\theta_d)P(w|\theta_v^{\text{qlog}})},$$

normalizing across verticals,

$$\phi(d, v) = \frac{\mathcal{B}(\theta_d, \theta_v^{\text{qlog}})}{\sum_{v' \in \mathcal{V}} \mathcal{B}(\theta_d, \theta_{v'}^{\text{qlog}})}.$$

Given query q , we generate one Soft.ReDDE feature per vertical using a retrieval from an index of the entire English Wikipedia. The Soft.ReDDE feature value for vertical v is given by,

$$\text{Soft.ReDDE}_q(v) = \frac{1}{\mathcal{Z}} \sum_{d \in \mathcal{R}_{100}} \phi(d, v) P(q|d),$$

where $\mathcal{Z} = \sum_{d \in \mathcal{R}_{100}} P(q|d)$. We normalize Soft.ReDDE features across verticals such that $\sum_{v \in \mathcal{V}} \text{Soft.ReDDE}_q(v) = 1$.

³<http://svr-www.eng.cam.ac.uk/prc14/toolkit.html>

3.3.3 Query Features

Query features are generated from the query and are independent of the vertical under consideration. We generated three types of query features. Rule-based vertical intent features exploit a correlation between certain key words and the relevance of particular vertical (e.g. “obama inauguration pics”). Geographic features correspond to various different geographic entity types possibly appearing in the query. Category features are motivated by the fact that some verticals (e.g., *health*, *travel*) are topically focused. Thus, knowing the topic of the query may help in vertical selection.

Rule-based Intent Features

Rule-based intent features are based on a set of 45 classes aimed to characterize query intent (e.g., *local phone*, *product*, *person*, *weather*, *movies*, *driving direction*, *music artist*). Some of these 45 features map conceptually one-to-one to a target vertical (e.g., *movies* → *movies*, *autos* → *autos*). Others map many-to-one (e.g., {*sports players*, *sports*} → *sports*, {*product review*, *product*} → *shopping*). Others do not map directly to a vertical, but may provide (positive or negative) evidence in favor of a vertical (e.g., *patent*, *events*, *weather*). Each of these boolean features is associated with a set of manual rules based on regular expressions and dictionary lookups. A query may be associated with multiple classes, each triggered if at least one rule in its inventory matches the query.

Geographic Features

Geographic features were produced using a rule-based geographic annotation tool that outputs a probability vector for a set of geographic entities appearing in the query. We focused on the following 17 geographic features: *airport*, *colloquial* (i.e., location information associated with a named entity, such as “North Shore Bank”), *continent*, *country*, *county*, *estate*, *historical county*, *historical state*, *historical town*, *island*, *land feature*, *point of interest* (e.g. Eiffel Tower), *sports team*, *suburb*, *supername* (i.e., a region name, such as Middle East), *town*, and *zip code*. We used the probability of each entity being present in the query as a feature. Geographic features are intended to inform classification into verticals whose queries often mention a location name, such as *local*, *travel*, and *maps*.

Query Category Features

As previously mentioned, some verticals, such as *health*, *autos*, and *travel*, are topically focused. For such verticals, a potentially useful source of evidence is the topic of the query (e.g., if the query is about fitness, then the *health* vertical is relevant).

One way to classify queries into topics is to directly apply a trained topic classifier to the query string. However, queries are terse. Therefore, instead of classifying the query directly, we predict query categories based on the query’s association with topically-classified documents. This approach is common in prior query classification work [88, 89]. We used a large document collection where each document is classified into a set

of categories. Documents in this collection were classified using a proprietary maximum entropy classifier from Yahoo!.

Given a retrieval from this collection, the query is classified based on the categories assigned to the top 100 documents. We divide categories into two disjoint sets: general categories (e.g., *recreation*, *science*, *health*) and specific categories (e.g., *recreation/sports*, *recreation/travel*, *health/nutrition*). Every document in the collection is associated with a vector of confidence values $P(y_j|d)$. We set the value of category feature y_j to be the sum of its confidence values in the top 100 documents,

$$\text{CAT}_q(y_j) = \sum_{d \in \mathcal{R}_{100}} P(y_j|d),$$

We focused on 14 general category features and 42 specific category features, for a total of 56 query category features.

3.3.4 Summary of Features

In summary, the total number of features is as follows. Let n denote the number of candidate verticals. In the experiments below, $n = 18$. We focused on two types of vertical corpus features: ReDDE.top and Clarity. Each contributes $2n$ features: n using vertical samples and n using Wikipedia articles that were heuristically mapped to each vertical. We focused on two types of vertical query-log features: Soft.REDDDE and the query-likelihood given the vertical’s query-log language model. Each contributes n features (one per vertical). Finally, we focused on three types of query features: rule-based intent, geographic, and category features. Because they are vertical-independent, these are not a function of n . We focused on 45 rule-based intent features, 17 geographic features, and 56 query-category features.

In the *ideal* case, the total number of features would be 226 (i.e., $6n + 45 + 17 + 56 = 226$). For practical reasons, however, we used fewer features than in the ideal case. The *autos*, *maps*, *sports*, and *tv* verticals did not have query-logs available. Query-logs were used directly and indirectly to derive different types of features. Query likelihood features use vertical query-logs directly. ReDDE.top features (using vertical-sampled documents) rely on vertical query-logs indirectly because vertical query-log queries were used for sampling. Soft.ReDDE features use vertical query-logs to estimate the vertical language model. These feature types did not have a feature associated with the *autos*, *maps*, *sports*, and *tv* verticals. Also, for practical reasons, some verticals were not associated with Wikipedia articles. The *directory* vertical and the “no relevant vertical” class are too broad to be characterized by a set of Wikipedia categories while the *maps* vertical intersects semantically with *local* and *travel*. The actual total number of features was 190.

3.4 Methods and Materials

Our goal is to test the effectiveness of our supervised classification-based framework on the task of *single vertical selection*—predicting a single relevant vertical if one exists,

vertical	retrievable items
autos	car reviews, product descriptions
directory	web page directory nodes
finance	financial data and corporate information
games	hosted online games
health	health-related articles
images	online images
jobs	job listings
local	business listings
maps	maps and directions
movies	movie show times
music	musician profiles
news	news articles
reference	encyclopedic entries
shopping	product reviews and listings
sports	sports articles, scores, and statistics
travel	travel and accommodation reviews and listings
tv	television listings
video	online videos

Table 3.1: Vertical descriptions.

or predicting that no vertical is relevant. In this section, we describe our experimental methodology: our verticals, evaluation data, evaluation metric, and baseline methods used for comparison.

3.4.1 Verticals

We focused on 18 verticals, described in Table 4.1. Each vertical corresponds to a Yahoo! property.⁴ Some of these verticals, for example, *news*, *images*, and *local*, are currently integrated into Yahoo! Web search results. Others, for example, *health*, *tv*, and *games*, exist only as Yahoo! properties. Users can access the vertical directly, and in most cases, search within the vertical. However, content from these vertical, is not currently surfaced in response to Web search queries.

These verticals differ across several meaningful dimensions: number of documents, level of text-richness (e.g., *news*, *images*), topic (e.g., health, finance, sports), and level of vertical query-traffic.

3.4.2 Queries

Our evaluation data consisted of 25,195 unique, randomly sampled queries issued to the Yahoo! Web search engine. Given a query, human editors were instructed to assign verticals to one of four relevance grades ('most relevant', 'highly relevant', 'relevant',

⁴As previously noted, these experiments were conducted during an internship at Yahoo! Labs Montreal. We are thankful to Yahoo! for providing these resources.

‘not relevant’) based on their best guess of the user’s vertical intent. For the purpose of this study, we collapsed ‘most relevant’, ‘highly relevant’, and ‘relevant’ into one class: ‘relevant’.

The vertical distribution across queries is described in Table 3.2. As previously mentioned, only 30% of all queries were assigned more than one true relevant vertical. The remaining 70% were assigned a single relevant vertical or none. 26% were assigned no relevant vertical. Of those assigned multiple true relevant verticals (30%), many were ambiguous to an assessor attempting to guess the user’s intent. For example, the query “hairspray” was assigned verticals *movies*, *video*, and *shopping* (“hairspray” is a movie, a Broadway play, and a hair product).

maps	1.1%
jobs	1.5%
movies	2.3%
games	2.6%
finance	2.6%
tv	2.7%
autos	3.0%
video	3.1%
sports	3.3%
health	4.3%
directory	4.4%
music	4.6%
news	5.1%
images	6.0%
travel	8.7%
reference	15.4%
local	19.1%
shopping	20.3%
none	26.3%

Table 3.2: Percentage of queries for which the vertical (or no vertical) was labeled relevant. Percentages do not sum to one because some queries were assigned multiple relevant verticals.

3.4.3 Evaluation Metric

We evaluated single vertical selection in terms of *single vertical precision*, given by,

$$\mathcal{P} = \frac{1}{|\mathcal{Q}|} \left(\sum_{q \in \mathcal{Q} | \mathcal{V}_q \neq \emptyset} \mathcal{I}(\tilde{v}_q \in \mathcal{V}_q) + \sum_{q \in \mathcal{Q} | \mathcal{V}_q = \emptyset} \mathcal{I}(\tilde{v}_q = \emptyset) \right), \quad (3.1)$$

where \mathcal{I} is the indicator function. Single vertical precision is the percentage of queries for which the model makes a correct prediction. The first summation determines the number of queries for which a single vertical is correctly predicted. The second summation determines the number of queries for which *no relevant vertical* is correctly predicted.

In addition to single vertical precision, we also report *coverage* (% cov), defined as the percentage of queries for which a (single) vertical is predicted (correctly or incorrectly). Statistical significance was tested using a 2-tailed paired t-test (paired on queries).

3.4.4 Implementation Details

All features were scaled to zero minimum and unit maximum. Features associated one-to-one with a vertical (Clarity, ReDDE.top, query-log likelihood and Soft.ReDDE) were normalized across verticals before scaling. Supervised training/testing was done via 10-fold cross validation. Threshold parameter τ was tuned for each training fold on a held-out set of 500 queries. Given a query, the classification framework predicts a single vertical if the most confident vertical prediction exceeds parameter τ and the confidence from the *no relevant vertical* classifier.

3.4.5 Single-evidence Baselines

To evaluate the effectiveness of our classification framework, we compare its performance with *eight* different baselines. We refer to these as *single-evidence* baseline because, like most resource selection methods, each focuses on a *single* type of evidence, for example, the predicted relevance of sampled vertical content or the similarity between the query and the vertical’s query-traffic.

Our eight single-evidence baselines were the following: all four combinations of Clarity and ReDDE.top using vertical-sampled and Wikipedia-sampled documents, the query likelihood given the vertical’s query-log language model (both, allowing and disallowing zero probabilities), Soft.ReDDE, and an approach that always predicts “no relevant vertical”. One of these baselines, ReDDE.top, is a slight variation of ReDDE, which has produced strong results in various different federated search evaluations [90, 94, 104].

With the exception of the *no relevant vertical* predictor, given a query, all these baselines produce a score for each vertical. The higher the score the more relevant the vertical. Each was adapted for single-vertical selection as follows. First, scores were mass-normalized across verticals. Then, we predict the *single* vertical with the highest score if its score exceeds threshold τ . Threshold parameter τ was tuned on validation data.

3.5 Experimental Results

In this section, we compare the performance of our classification framework with our eight single-evidence baselines. Results, in terms of single vertical precision \mathcal{P} , are presented in Table 3.3.

Several results are worth noting. First, the *no.rel* approach obtained $\mathcal{P} = 0.263$ because 26.3% of queries had no true relevant vertical.

Both Clarity using vertical- and Wikipedia-sampled documents performed significantly worse than *no.rel*. This suggests that Clarity scores for a given query may not be directly comparable across collections with different corpus statistics. In prior work,

	\mathcal{P}	% cov
clarity.vertical	0.254	3.4%
clarity.wiki	0.256 Δ	2.7%
no.rel	0.263 \blacktriangle	0.0%
redde.top.wiki	0.293 \blacktriangle	54.4%
QL	0.312 \blacktriangle	61.9%
soft.redde	0.324 \blacktriangle	43.6%
redde.top.vertical	0.336 \blacktriangle	45.7%
QL (zero probs)	0.368 \blacktriangle	51.0%
classification	0.583 \blacktriangle	64.3%

Table 3.3: Single Vertical Precision (\mathcal{P}). Approaches are listed in ascending order of \mathcal{P} . A significant improvement over all worse-performing approaches is indicated with a Δ at the $p < 0.05$ level and a \blacktriangle at the $p < 0.005$ level.

Clarity was used to compare retrievals from different queries on the same collection, but not retrievals from the same query on different collections. Further experiments are needed to determine whether Clarity can be adapted for purpose of vertical selection, or resource selection in general.

ReDDE.top with vertical samples outperformed ReDDE.top with Wikipedia samples, in spite of more verticals having a Wikipedia-sampled collection than a vertical-sampled collection. We examined the types of classification errors made by both methods. Both approaches performed comparably with respect to the “no relevant vertical” class. However, *redde.top.wiki* more often predicted a wrong vertical, that is, a vertical not in the true relevant set \mathcal{V}_q . Precision on queries for which a vertical was predicted was 0.382 for *redde.top.vertical* and 0.284 for *redde.top.wiki*. This suggests that our heuristic mapping of Wikipedia categories to verticals may have misrepresented one or more vertical.

The query likelihood given the vertical’s query-log language model was the best single-evidence predictor. This method performed better when allowing than when disallowing zero probabilities. This may have been due to the non-uniformity of $P(\text{OOV})$ estimates across vertical language models. Each vertical’s $P(\text{OOV})$ estimate was based on the frequency of terms not in its top 20,000, which is expected to be greater for verticals with a more open vocabulary. A vertical’s $P(\text{OOV})$ estimate affects the probability estimates of within vocabulary terms through discounting. Different $P(\text{OOV})$ estimates across verticals may have made the query likelihood given by different vertical language models less comparable.

Finally, The classification approach outperformed all single-feature baselines by a large margin—a 58% improvement over the best single-evidence predictor, QL. This confirms our expectation that vertical selection can be cast as a machine learning problem.

3.6 Discussion

3.6.1 Feature Ablation Study

Compared to our single-evidence baselines, the classification approach has two main advantages: access to training data and the ability to integrate multiple types of features. In this section, we present a feature-type ablation study with a dual purpose: to explore the contribution of different feature types and to confirm that the classification approach benefits from integrating multiple sources of evidence. In other words, we wish to confirm that the classifier is not just benefiting from a single source of evidence and lots of training data.

Additionally, we conducted a feature ablation study to compare the contribution of corpus features derived from vertical-sampled documents and corpus features derived from Wikipedia articles associated with a vertical.

Table 3.4 shows the percent change in precision associated with each feature type. Keep in mind that features were not evaluated in isolation. Therefore, a non-significant performance drop in \mathcal{P} does not necessarily mean that the feature captures no useful evidence, as features may be correlated.

feature type analysis			
feature variation	\mathcal{P}	% diff	% cov
all features	0.583		64.30%
no query-likelihood	0.583	0.03%	64.36%
no rule-based intent	0.583	-0.03%	64.30%
no clarity	0.582	-0.10%	63.68%
no geographical	0.577▼	-1.01%	65.30%
no (general) category	0.572▼	-1.84%	63.91%
no redde.top	0.568▼	-2.60%	60.27%
no soft.redde	0.567▼	-2.67%	62.47%
no (specific) category	0.552▼	-5.33%	64.19%

vertical/wikipedia-based feature type analysis			
feature variation	\mathcal{P}	% diff	% cov
no vertical	0.577▼	-3.1%	62.06%
no wikipedia	0.574▼	-3.5%	62.67%

Table 3.4: Feature type ablation study. A ▼ denotes a significant improvement ($p < 0.005$) over our classifier using all features

In terms of feature types, omitting query-likelihood, rule-based intent, and Clarity features did not produce a significant drop in performance. It is possible that query-likelihood features, the best single-evidence predictor, did not contribute significantly because they are correlated with Soft.ReDDE features, which did contribute significantly and, like query-likelihood features, also derive evidence from the vertical query-log.

Recall that rule-based features are binary and each feature attempts to match a set of regular expressions and dictionary terms to the query. It turned out that there were only 4,367 (18%) queries with at least one non-zero rule-based intent feature. This may have been why they did not contribute significantly to performance. As previously mentioned, Clarity scores for the same query and different collections may not be directly comparable.

The largest contribution to performance came from query-category features (i.e., the specific categories), which characterize the topic of the query. Interestingly, query-category features are not derived from the vertical (i.e., they are not derived from vertical samples or from the vertical query-log). Our classifier learns to associate these features with each vertical from training data. The contribution of the specific query-category features was significantly greater than that of general query-category features. This was because our general categories were too coarse to discriminate between our verticals. For example, the general category *recreation* conflates *recreation/sports*, *recreation/auto*, and *recreation/travel*, which map conceptually to different verticals. The second and third most helpful features were Soft.ReDDE and ReDDE.top features, respectively. This analysis confirms that different sources of evidence contribute significantly to performance.

To evaluate the usefulness of evidence derived directly from the vertical, we omitted ReDDE.top and Clarity features using vertical-sampled documents. Likewise, to evaluate the usefulness of evidence derived from surrogate corpora (in our case, Wikipedia), we omitted ReDDE.top and Clarity features using Wikipedia-sampled documents. Removing either set of features produced a significant drop in \mathcal{P} . Vertical- and Wikipedia-sampled documents were sampled using different techniques and have a different sample set size distribution. Thus, we cannot (and did not intend to) directly compare one against the other. However, this result confirms that external documents (associated with a vertical heuristically) can provide evidence complementary to evidence derived directly from the vertical.

3.6.2 Per Vertical Performance

In this section, we look at per-vertical performance. The objective is to determine which verticals are easier than others. Table 3.5 shows precision for each vertical (using all features). Here, given vertical v , per-vertical precision is defined by,

$$\mathcal{P}_v = \frac{1}{|\mathcal{Q}_v|} \sum_{q \in \mathcal{Q}} \mathcal{I}(\tilde{v}_q = v) \wedge (q \in \mathcal{Q}_v),$$

where \mathcal{Q}_v is the subset of queries in \mathcal{Q} with $v \in \mathcal{V}_q$.

As previously noted, some verticals lacked query-logs (\star) and/or a Wikipedia-sampled surrogate corpus (\ast). Verticals *autos*, *sports*, and *tv* performed well in spite of lacking features derived from query-logs. Verticals *video*, *news*, and *reference* performed poorly in spite of having all resources. Therefore, the difference in performance across verticals cannot be attributed only to missing features.

vertical	\mathcal{P}
travel	0.842
health	0.788
music	0.772
games	0.771
autos*	0.730
sports*	0.726
tv*	0.716
movies	0.688
finance	0.655
local	0.619
jobs	0.570
shopping	0.563
images	0.483
video	0.459
news	0.456
reference	0.348
maps**	0.000
directory*	0.000

Table 3.5: Per vertical precision (\mathcal{P}). Verticals without a query-log are marked with *. Verticals without a Wikipedia-sampled surrogate corpus are marked with *.

The system performed best on verticals that focus on a coherent topic with identifiable vocabulary (i.e., *travel*, *health*, *games*, *music*). The vocabulary associated with these verticals may have been the least confusable with that of other verticals. Performance for these topically-coherent verticals was higher than for *shopping*, *reference*, and *no relevant vertical*, which had *more* positive examples for training. This result is consistent with the fact that query-categories features, which characterize the topic of the query, were found to be the most predictive, as previously shown in Table 3.4.

Performance was the worst on the verticals *images*, *video*, *news*, *reference*, *maps*, and *directory*, possibly for several different reasons. The *maps* vertical had the fewest positive instances for training, was feature-impoverished, and probably confusable with *local* and *travel*. Verticals *images* and *video* focus on a type of media rather than a specific genre. Queries related to *reference* and *directory* characterize broad encyclopedic information needs. The *news* vertical tends to be highly dynamic and may require features that characterize whether the query occurs as part of a burst in content demand.

3.7 Related Work in Vertical Selection

We cast vertical selection as a supervised machine learning problem and integrate diverse types of evidence as input features. Other researchers proposed similar solutions to similar problems. Some of this work precedes the work presented in this chapter (e.g., news-vertical prediction [28, 63]). Other work was done after (e.g., vertical selection in the presence of user feedback [29])

The SIGIR 2008 Workshop on Aggregated Search [74] was one of the first forums to discuss aggregated web search as a new and interesting research area. The work presented in the workshop focused a wide-range of problems, including search results summarization and visualization, especially in environments with rich meta-data. Of those papers that focused on aggregating content from *different* search engines [75, 107], none of them addressed the task of vertical selection, but rather assume that every vertical is presented for every query (in the same position).

Both König *et al.* [63] and Diaz [28] focused on approaches to vertical selection for the *news* vertical, a particularly challenging vertical due its dynamic nature—a query’s newsworthiness is likely to change with time. Both formulate the task as predicting whether a user will click on news results and both assume a fixed presentation strategy: to present news results above the Web results or not at all. König *et al.* [63] made use of query-features (e.g., whether the query contains a URL) and corpus features (e.g., the hit counts from various news- and non-news-related corpora). Additionally, Diaz [28] made use of query-traffic features (e.g., the number of times the query was recently issued to the Web and news search engines).

The probability that a user will click on news results can be estimated in two ways: (1) using a model that considers various contextual features (e.g., corpus and query-log statistics) and (2) using user-interaction data from previous impressions of the query (e.g., observed clicks and skips from previous system decisions). In addition to an off-line model, Diaz [28] introduced an approach that combines both estimation methods. The model automatically relies more heavily on user-interaction data as it becomes available for the query. Moreover, it can pro-actively display news results more frequently for queries with little user-interaction data. In subsequent work, Diaz extended the multiple-vertical framework presented in this chapter to consider user-interaction data [29].

Song *et al.* [99] focused on the task of *searchable website recommendation*, where the goal is to display (alongside the Web search results) a short ranking of third-party searchable websites (e.g., ebay, flickr, amazon, imdb). Searchable websites can be viewed as verticals. Song *et al.* [99] combined various features derived from user-interaction data, for example, the similarity between the query and those associated with clicks on searchable website results and the average dwell time on searches from a particular website. Interestingly, searchable website recommendation occurs in a type of uncooperative environment; third party search engines do not typically divulge user-interaction data. Song *et al.* [99] collected user-interaction data using an opt-in plug-in distributed with a major commercial browser. This work shows that a feature-integration approach capable of combining various types of user-interaction features can be useful in an uncooperative setting, where interactions can be logged at the client rather than the server side.

Similar to our approach, the work discussed so far enriches the query representation using various types of contextual features (e.g., different user interactions and different statistics derived from collection content and query-traffic). Li *et al.* [67] take the opposite approach, using only the terms in the query as features. Their method, however, is to vastly expand the training set by propagating labels to unlabeled queries using a

click-graph. The assumption is that queries with a similar click pattern are relevant to the same verticals. Evaluation was conducted separately for two verticals: *jobs* and *shopping*. Deriving evidence exclusively from the query-string has the advantage that feature generation is fast. The main issue, however, is that performance may degrade as the environment changes in terms of content and user interest. One advantage of focusing on evidence beyond the query-string is that a model may be able to maintain performance as long as the resources used for feature generation (e.g., query-logs and corpora) are updated frequently to reflect changes in the environment.

The field of *sponsored search* is concerned with the placement of advertisements (ads) in response to a Web search query and typically assumes a fixed location for presenting the set of most relevant ads (e.g., above the Web results). As advocated in Broder *et al.* [13], part of the task is to decide when to completely suppress ads. The assumption is that displaying marginally relevant or non-relevant ads “trains” users to ignore ads in the future. The task of predicting whether to display or suppress the set of most relevant ads can be viewed as a type of vertical selection with one vertical: ads.

Similar to our approach, Broder *et al.* [13] cast the problem as a supervised classification task. Different types of features were investigated and, consistent with our results, no single type of feature was exclusively responsible for performance. As might be expected, many of the features that were effective consider the text-similarity between the query and the set of candidate ads. More interestingly, however, other features that were effective consider the text-similarity *between* the set of ads themselves. This shows that “vertical” relevance can be a function of the coherence between the (top) vertical results. Our Clarity feature, which in our case was ineffective, attempts to harness this type of evidence, though we applied it to sampled and surrogate content rather than the *actual* vertical results. This suggests that directly comparing Clarity scores from different verticals might have been the problem. Broder’s classification, evidence-integration approach outperformed a single-evidence baseline that predicts whether to display ads based on their retrieval score.

In more recent work, Guo *et al.* [44, 45] considered the task of predicting whether the user will click on ads during the remainder of the search session. This work is the first to consider features from previous retrievals within the session, which include low-level user interactions such as mouse movements, temporal delays, and scrolls. Within-session low-level interactions were found to improve performance. We do not make use of session-level features in this dissertation. Investigating their effectiveness for vertical selection and vertical results presentation is an important direction for future work.

3.8 Summary

We evaluated a classification-based approach to vertical selection. The proposed classification approach outperformed all single-evidence baselines, which included adaptations of existing methods for resource selection [94] and retrieval effectiveness prediction [25].

More importantly, we demonstrated the importance of feature-integration for the task

of vertical selection. A set of feature ablation studies confirmed that removing different types of features results in a significant drop in performance. Therefore, one important advantage of the classification-based approach is its ability to easily integrate evidence as input features. In other words, the classifier did not merely capitalize on a single source of evidence and lots of training data.

In terms of corpus evidence, features derived from vertical samples as well as external surrogate corpora contributed significantly to performance. In terms of query-traffic evidence, access to vertical query-logs helped. Not only was the query-likelihood given the vertical's query-log language model the best single-evidence predictor, but query-logs were used (directly and indirectly) in generating various other features. For Soft.ReDDE, vertical query-logs were used to associate external documents to verticals. For ReDDE.top, they were used to sample from the vertical. Both of these features significantly improved performance.

In terms of cross-vertical results, performance was the best for topically coherent verticals (e.g., *travel*, *health*, *games*, *music*, *autos*, etc.) Good performance across these verticals was at least partly attributed to our query-category features, which characterize the topical distribution of the query. Interestingly, this is precisely the type of evidence that requires some form of human supervision in order to learn a predictive relation between the query category and the relevance of a particular vertical. In our case, our models learned to associate categories to verticals from training queries. Performance was poor on verticals that are not topically focused (e.g., *reference*), are text-impooverished (e.g., *images*) and dynamic (e.g., *news*).

Domain Adaptation for Vertical Selection

While a supervised approach to vertical selection outperforms state-of-the-art resource selection methods, one of its main drawbacks is that it requires extensive training data (e.g., a set of queries with relevance labels with respect to a vertical). Human annotation is resource intensive in terms of time and money. An annotation effort may be sensible as a one-time investment. However, in practice, the aggregated search environment is dynamic. Verticals can be added to or removed from the set of candidate verticals. Content can be added to or removed from an existing vertical. The interests of the user population may drift, effectively changing the vertical content most likely to be requested by users. It may not be feasible to annotate a fresh new set of queries every time the environment undergoes a significant change. Advancing a machine learning approach to vertical selection, therefore, requires methods that maximize the system's return on editorial investment. As the environment changes, how can we make maximal use of annotations we already have in order to maintain performance? In this chapter, we focus on the case where a new vertical is introduced to the set of candidate verticals (the first type of change suggested above). How can we use training data collected for a set of *existing* verticals to learn a predictive model for a *new* vertical (associated with no training data)? Although we focus on the introduction of a new vertical, the methods described in this chapter may also be effective in learning new models to account for changes in vertical content and/or user interest.

We explore the following scenario. We assume a set of existing verticals (referred to as the *source* verticals) for which we have collected training data (a set of queries with vertical-relevance judgements). Then, we assume that we are given a new vertical (referred to as the *target* vertical), associated with no training data. Our objective is to learn a predictive model for the target vertical using *only* source-vertical training data.

Our approach to the problem focuses on two model properties: *portability* and *adaptability*. A *portable* model is one that can be trained once and then effectively applied to *any* vertical, including the new vertical (associated with no training data). An *adaptable* model is one that can be tailored to a specific vertical, including the new vertical. If a model is adaptable, its parameters can be automatically adjusted to suit the new vertical at no additional editorial cost. We present models that exhibit these two properties and evaluate their performance across a set of 11 target verticals. Our end goal is to show

¹This work was conducted during a second internship in Yahoo! Montreal and was published in SIGIR 2010 with co-authors Jean-François Paiement and Fernando Diaz [4].

that a machine learning approach to vertical selection can capitalize on human labels collected for one set of existing verticals to learn a model for a new one.

4.1 Formal Task Definition

In this set of experiments, we consider the relevance of vertical v with respect to query q independent of any other vertical. Let $y_v(q)$ denote a function that outputs the true relevance of v with respect to q . In the general vertical selection setting, the goal is to learn a function f_v that approximates y_v . Here, we focus on the following scenario. Suppose we have a set of existing *source* verticals \mathcal{S} with a set of labeled queries $\mathcal{Q}_{\mathcal{S}} = \cup_{s \in \mathcal{S}} \mathcal{Q}_s$, where \mathcal{Q}_s denotes the set of queries with relevance labels with respect to source vertical s . Then, suppose we are given a new *target* vertical t with no labeled data. Our objective is to learn a function f_t that approximates y_t using *only* source-vertical training data $\mathcal{Q}_{\mathcal{S}}$. The quality of an approximation will be measured using some metric that compares the predicted and true query labels. We use notation,

$$\mu(f_t, y_t, \mathcal{Q}_t)$$

to refer to the evaluation of function f_t on query set \mathcal{Q}_t , which has relevance labels with respect to t . This metric μ can be classification-based (e.g. accuracy) or rank-based (e.g. average precision).

4.2 Related Work

The task of using existing-vertical training data to learn a predictive model for a new vertical can be viewed as a type of *domain adaptation*. In machine learning, domain adaptation is the task of using training data from one or more *source* domains to learn a predictive model for a *target* domain. The domain adaptation problem arises when the source and target data originate from different distributions. Suppose, for example, we want to learn a classifier to distinguish between positive and negative *restaurant* reviews (i.e., the target domain), but only have training data in the *movie* domain (i.e., the source domain). A model trained on movies may not generalize to restaurants, for various reasons. It may be, for example, that the prior class distribution (i.e., positive and negative) is different across domains. It may also be that the predictive relationship between features and the target class is different. The term “unpredictable” may signal a positive review in the movie domain, but a negative review in the restaurant domain. Domain adaptation algorithms solve this problem in various ways and have been applied to classification tasks such as sentiment analysis [10], named-entity tagging [43], and document ranking [39]. No prior work, however, has applied domain adaptation to vertical selection.

Different approaches to domain adaptation assume different amounts of training data in the target domain. In our case, we assume none. However, while we focus on this

extreme case, one could imagine collecting a small amount of (possibly noisy) target-vertical training data using implicit feedback or active learning. Thus, we review domain adaptation approaches that consider various amounts of target-domain training data.

As mentioned above, a potential challenge in domain adaptation is that the class distribution may be different across domains. An easy solution is to re-weight (or over/under-sample) source-domain instances such that the class distribution resembles the target-domain class distribution [69]. This, of course, assumes that the target-domain class distribution can be estimated reliably. Also, it assumes that the predictive relationship between features and the target class is consistent across domains. It is *only* the class distribution that is different.

Domain adaptation is more difficult when features have an inconsistent predictive relationship with the target class. One line of work performs instance-weighting (or instance filtering) to down-weight (or remove) “misleading” source-domain training instances. Jiang and Zhai [55] do this by training a model on whatever little target-domain data is available and removing those source-domain instances that are misclassified by this model. Gao *et al.* [39] propose a similar solution that does not require any target-domain training data. Their approach is to train a classifier to predict whether an instance originates from the source or target domain. Then, they keep only those source-domain training instances that are misclassified (or nearly misclassified) as being from the target domain. The idea is to favor source-domain training instances that are confusable with the target-domain instances.

An alternative to instance-weighting is feature weighting (or feature selection). The objective is to focus only on those features that are similarly predictive across the source and target domains. Saptal and Sarawagi [83] cast this as a constrained optimization problem. A model is trained to, not only maximize the likelihood of the source-domain training data, but also to select those features which minimize the distance between the source and target distributions. The distance function is the difference between the (source and target) mean values across features.

The work discussed so far assumes a single source domain. Jiang [54] proposes an approach that uses training data from *multiple* source domains. First, a generalizable subset of features is identified based on their predictiveness across source domains. Then, a model that focuses heavily on these features is used to produce predictions on the target domain. Finally, a target-domain classifier with access to all features is trained using these predictions as pseudo-training examples. This work is very similar to the work presented in this chapter. The major difference is that we use a learning algorithm that can model complex feature interactions. This learning algorithm is used both to produce the target-vertical pseudo-labels and the *final* target-vertical predictions.

Feature selection can be viewed as a type of feature representation change. An alternative to feature selection is feature augmentation (increasing the feature space). The goal, however, is the same: to find a representation that behaves similarly across domains. Daumé III [26] augments the feature space by making three copies of each feature: a source-domain, target-domain, and domain-agnostic copy. Then, a model is learned by

pooling together the source and target training data. Effectively, this allows the model to weight each copy differently, depending on the feature’s correlation with the target class across domains. For example, if a feature is only predictive in the target domain, the model can assign a high (positive/negative) weight to its target-specific copy and a near-zero weight to its other two copies. If a feature behaves similarly across domains, the model can assign a high (positive/negative) weight to its domain-agnostic copy.

Blitzer *et al.* [9] propose a feature augmentation method that does not require target-domain training data. Their *structural correspondence learning* approach is based on the distinction between *pivot* and *non-pivot* features. A pivot feature is one that occurs frequently across domains and has a similar predictive relationship with the target class (e.g., the term “bad” in movie and restaurant reviews). A non-pivot feature is one that occurs more frequently in one domain and may have a different predictive relationship with the target class (e.g., the term “unpredictable” in movie and restaurant reviews). The aim of structural correspondence learning is to identify the correspondence between non-pivot features across domains. The assumption is that correspondences between non-pivot features are encoded in the weights assigned to each when training linear classifiers to predict the value of each pivot feature using non-pivot features.

A different approach to domain adaptation is to train a model on the source domain and then to only fine-tune its parameters using the available target-domain training data. Chen *et al.* [21] propose a simple approach that uses the Gradient Boosted Decision Trees (GBDT) learning algorithm. GBDT uses a boosting framework to combine weak learners (i.e., decision trees) to produce a more complex model. During each GBDT training iteration, a new decision tree is trained on the residuals of the current model’s predictions. Their approach to domain adaptation is to first train a model on the source domain and then to simply continue the training process on target-domain training data. One advantage of this approach is that the feature representation across domains can be different (even completely disjoint). Our approach to domain adaptation uses this algorithm, but during adaptation, we continue the GBDT training process using target-domain *pseudo*-training data.

If we assume a small amount of target-vertical training data, our problem is also related to *multi-task learning* or *transfer learning*. The multi-task learning problem arises when the data is associated with *multiple* output variables, as in the case of multi-class classification. Rather than learn each class independently, the aim of multi-task learning is to share information across predictive tasks. The assumption is that overall performance for *any* particular class can be improved by exploiting correlations between classes. In our case, it may be possible to improve performance for the target vertical by exploiting correlations with other verticals associated more training data.

There is a large body of prior work on multi-task learning. Thus, our goal is to provide only a brief overview. Early work used neural networks with multiple inputs (i.e., features) and multiple outputs (i.e., predictions for each task) [19, 105]. Using a single neural network, predictive relationships between tasks can be modeled within the network’s *hidden* layer(s) and learned using back-propagation. Shrinkage methods

combine predictions for different classes that are learned independently [11]. Combination coefficients can be learned using cross-validation. Regularization-based methods exploit associations between target classes during training. The assumption is that models learned for similar tasks should have similar model parameters [33]. Hierarchical Bayesian approaches, like neural net approaches, model relations between tasks using shared latent variables [117].

While multi-task learning approaches may be useful for our task, there are two caveats. First, they require at least some amount of target-vertical training data. The methods proposed in this chapter require none. Second, at least some multi-task learning approaches (e.g., neural-network methods [19]) assume a *common* dataset with labels for every class. In aggregated web search, verticals labeled during different time periods will likely be associated with different queries. The methods proposed in this chapter can handle source verticals with different (even completely disjoint) training sets.

4.3 Vertical Adaptation Approaches

In this chapter, as in Chapter 3, we cast vertical selection as a classification task. In other words, we train models (using only source-vertical training data) to predict the relevance of the new target vertical as a function of a set of features, described later in Section 4.4.

Several different models are proposed. In Section 4.3.2, we propose several *portable* models. A model is portable if, once trained, can be effectively applied to *any* vertical, including one without training data. Then, in Section 4.3.3, we propose models that are *adaptable*. A model is adaptable if it can be fine-tuned to a specific vertical, including one without training data. All proposed methods use the same base learning algorithm: Gradient Boosted Decision Trees (GBDT) [38]. We adopted GBDT because it is able to model complex feature interactions and has been effective in other tasks such as text categorization [61] and rank-learning [119]. Next, we provide a general description of GBDT and then describe our portable and adaptable models.

4.3.1 Gradient Boosted Decision Trees

The main component of a GBDT model is a regression tree. A regression tree is a simple binary tree. Each internal node corresponds to a feature and a splitting condition which partitions the data. Each terminal node corresponds to a response value, the predicted output value. GBDT combines regression trees in a boosting framework to form a more complex model. During training, each additional regression tree is trained on the residuals of the current prediction. In our case, we chose to minimize logistic loss, which has demonstrated effective performance for vertical selection in prior work [2, 3, 28, 29]. That is, the model maximizes,

$$\mu_{\log}(f_v, y_v, \mathcal{Q}) = - \sum_{q \in \mathcal{Q}} \ell_{\log}(f_v(\phi_v(q)) \times y_v(q)) \quad (4.1)$$

where $\phi_v(q)$ denotes a feature generating function and ℓ_{\log} is the logistic loss function,

$$\ell_{\log}(z) = \log(1 + \exp(-z)) \quad (4.2)$$

Note that feature generator $\phi_v(q)$ is a function of v because at least some features are derived from the vertical under consideration. That is, their value depends on v .

4.3.2 Learning a Portable Model

A *portable* model is one that can make accurate relevance predictions with respect to *any* vertical. Once trained, a portable model can be used as a black-box predictor—given a query and an arbitrary vertical, it can predict whether the vertical is relevant to the query. On the other hand, a model is *non-portable* if it can make relevance predictions with respect to a *specific* vertical, but not a different one.

Let us examine the distinction between a portable and non-portable vertical selection model with an example. Consider a single-evidence model that predicts a vertical relevant based on the number of times the query was previously issued to the vertical by users. This model would likely be portable because this type of evidence—the frequency of the query in the vertical’s query-traffic—is likely to be positively correlated with relevance across verticals. The higher the value the more relevant the vertical *irrespective of the vertical*. On the other hand, consider a single-evidence model that predicts a vertical relevant if the query is classified as related to the *travel* domain. This would be a non-portable model because this type of evidence is likely to be predictive for a *travel* vertical, but not one that focuses on a different domain.

Our goal is to use only source-vertical training data to learn a portable model and then to apply the portable model to the new target vertical.

The Basic Portable Model

The objective of learning a portable model (denoted by f_*) is to learn a generic (vertical-agnostic) relationship between features and vertical relevance. One way to do this is to train a model to maximize the average performance across source verticals. The assumption is that if f_* performs consistently well across \mathcal{S} , then f_* will perform well on the target t , even if not trained on any target-vertical training examples.

More formally, we will define the portability of f_* using a metric that quantifies performance for a vertical $s \in \mathcal{S}$ and a function that averages performance across verticals in \mathcal{S} . For example, the portability, π , which uses the arithmetic mean of the logistic loss metric is defined by,

$$\pi_{\log}^{\text{avg}}(f_*, y_S, \mathcal{Q}_S) = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \mu_{\log}(f_*, y_s, \mathcal{Q}_s). \quad (4.3)$$

\mathcal{Q}_s is the set of training queries for source s and \mathcal{Q}_S is the set of those sets. Similarly, y_s provides labels for vertical s and y_S is the set of these functions. We refer to the model

which optimizes π_{\log}^{avg} as the *basic model*. Notice that,

$$\pi_{\log}^{\text{avg}}(f_{\star}, y_S, \mathcal{Q}_S) = -\frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \sum_{q \in \mathcal{Q}_s} \ell_{\log}(f_{\star}(\phi_s(q)) \times y_s(q))$$

As a result, the solution which maximizes π_{\log}^{avg} is equivalent to the solution which minimizes the logistic loss across all feature-vector/relevance-label pairs from all source verticals. In other words, to train a basic portable model, we can simply perform standard GBDT training on a pooling of each source vertical’s training set.

As previously mentioned, feature generator ϕ_v is vertical-specific because at least some features are generated from the vertical under consideration. During portable-model training, we pool together each source vertical’s training set. It should be noted that while the value of a particular feature may depend on the vertical, the semantics of each feature are consistent across verticals. In other words, all feature vectors are identically indexed and have the same length. So, for example, if feature $\phi_v(q)_i$ corresponds to the number of times the query was issued by users directly to v , then $\phi_{v'}(q)_i$ refers to the number of times the query was issued to v' .

In the next two sections, we describe two possible limitations of this basic portable model and suggest potential improvements.

Vertical Balancing

The set of positive instances within the basic model’s training set will correspond to the union of relevant query-vertical pairs from *all* source verticals. For this reason, we expect the portable model to focus on vertical-agnostic evidence (consistently correlated of the positive class) and ignore vertical-specific evidence (inconsistently correlated of the positive class). The assumption is that by focusing on evidence that is *not* conflicting with the positive class, the basic model will be effective on the target (even in the absence of target training data).

One challenge, however, is that the positive instances within the basic model’s training pool will be skewed towards the most popular verticals (those that tend to be relevant often). This may be problematic because the vertical that contributes the greatest number of positive examples may be reliably predicted relevant based on vertical-specific evidence, unlikely to be predictive of the target. To compensate for this, we consider a *weighted* average of metrics across verticals. Specifically,

$$\pi_{\log}^{\text{wavg}}(f_{\star}, y_S, \mathcal{Q}_S) = \frac{1}{\mathcal{Z}} \sum_{s \in \mathcal{S}} w_s \mu_{\log}(f_{\star}, y_s, \mathcal{Q}_s) \quad (4.4)$$

where $\mathcal{Z} = \sum_{s \in \mathcal{S}} w_s$. We use the simple heuristic of weighting a vertical with the inverse of its prior,

$$w_s = \frac{1}{p_s}$$

where p_s is the prior probability of observing a query with relevant vertical s . This value is approximated with the training data,

$$p_s \approx \frac{\sum_{q \in \mathcal{Q}_s} y_s(q)}{|\mathcal{Q}_s|}$$

The goal is to make training instances from minority verticals more influential and those from majority verticals less.

It is easy to see that Equation 4.4 is a generalization of Equation 4.3. Because we use logistic loss, this technique reduces to training with an instance weighted logistic loss where the instances are weighted by w_s , the weight of the vertical,

$$\pi_{\log}^{\text{wavg}}(f_*, y_{\mathcal{S}}, \mathcal{Q}_{\mathcal{S}}) = -\frac{1}{Z} \sum_{s \in \mathcal{S}} \sum_{q \in \mathcal{Q}_s} w_s \ell_{\log}(f_*(\phi_s(q)) \times y_s(q))$$

As with the basic model, we can use standard GBDT training to optimize for this metric.

Feature Weighting

An alternative to optimizing for a portable model is to find portable features and to train a model using only those. A portable feature is defined as a feature which is highly correlated with relevance (in the same direction) across all verticals. Recall that, across verticals, all features are identically indexed. Let ϕ^i be a predictor based only on the value of feature i . In previous work, the effectiveness of features across verticals was shown to be very dependent on the vertical being considered. In order to address the expected instability of feature predictiveness across verticals, we adopt a harmonic average for our aggregation method.

$$\pi^{\text{havg}}(\phi^i, y_{\mathcal{S}}, \mathcal{Q}_{\mathcal{S}}) = \frac{|\mathcal{S}|}{\sum_{s \in \mathcal{S}} \frac{1}{\mu(\phi^i, y_s, \mathcal{Q}_s)}} \quad (4.5)$$

Additionally, features, on their own, are not scaled to the label range, making the use of logistic loss difficult. Instead of constructing a mapping from a feature value to the appropriate range, we adopt a rank-based metric. Let $\rho_f(\mathcal{Q})$ be the ranking of \mathcal{Q} by f . We use average precision as our rank-based metric,

$$\mu_{\text{AP}}(f, y, \mathcal{Q}) = \sum_{r=1}^{|\mathcal{Q}|} y(\rho_f(\mathcal{Q})_r) \times \mathcal{P}_r(y, \rho_f(\mathcal{Q})) \quad (4.6)$$

where $\rho_f(\mathcal{Q})_k$ denotes the query at rank k and \mathcal{P}_k is the precision at rank k ,

$$\mathcal{P}_k(y, \rho) = \frac{1}{k} \sum_{r=1}^k y(\rho_r)$$

In other words, for each feature, we rank queries by feature value and compute the harmonic mean average precision across verticals.¹ Having computed the portability of

¹Because we do not know whether the feature value has a positive or negative relationship with the label, we compute $\pi_{\text{AP}}^{\text{havg}}(f, y_{\mathcal{S}}, \mathcal{Q}_{\mathcal{S}})$ using ρ induced in both directions and use the max.

each feature, we build a portable model by restricting our training to the most portable features.

The most portable features were selected by inspecting the distribution of portability values. Because portability values are in the unit range, we model our data with the Beta distribution. We fit the Beta distribution using the method of moments and then select features whose portability is in the top quartile of this distribution.

4.3.3 Adapting a Model to the Target Vertical

Above, we focus on ways of improving the portability of a model by influencing the model to ignore evidence that is vertical-specific. The argument is that a model that focuses heavily on vertical-specific evidence will not generalize well to a new target vertical.

Given access to target-vertical training data, Chapter 3 reveals two meaningful trends. First, given a wide-range of input features, most features contribute significantly to performance. The analysis presented in Section 3.6.1 demonstrates no single feature type is exclusively responsible for effective vertical prediction. Second, the features that contributed the most to performance, which characterize the topic of the query, are vertical-specific (assuming that verticals focus on different topics). Based on these observations, while ignoring vertical-specific evidence seems necessary to improve a model’s portability, a model customized to a particular vertical is likely to benefit from it.

In the context of adaptation for web search, Chen *et al.* [21] propose several ways to adapt an already-tuned GBDT model given data in a new domain. Their approach, Tree-based Domain Adaptation (TRADA), essentially consists of continuing the GBDT training process on labeled data from the target domain. More specifically, a set of new regression trees are appended to the existing model while minimizing a loss function (logistic loss, in our case) on the target data.

In our case, the challenge of using TRADA to adapt a model to a specific target is that we lack target-vertical training data. In the context of semi-supervised learning, self-training or bootstrapping [115] is the process of re-training a model using previous model predictions on unlabeled data. We combine self-training and model adaptation in the following way. First, we use a portable model to label a set of queries with respect to the target vertical. Then, we use TRADA to adapt the portable model to its own target-vertical predictions.

Tree adaptation provides a method for adjusting the modeling of all features. Just as we can select portable features for a portable model, we can select vertical-specific, non-portable features while adapting a model to a specific target vertical. That is, the base model may focus on portable features while the additional trees—added in the context of pseudo-training data—may focus on non-portable features. In this case, we use the same feature portability measure (Equation 4.5) but select the *least portable* features for tree augmentation. Pseudo-labels for the target vertical were produced using the portable model’s prediction confidence value with respect to the target vertical. We used the simple heuristic of considering the top $N\%$ most confident positive predictions as

positive examples and the bottom $(100 - N)\%$ predictions as negative examples.

4.4 Features

We generated a number of features which we believe are correlated with vertical relevance and are generalizable across queries. Our features can be viewed as belonging to the following two classes.

1. *Query features* are specific to the query and are independent of the vertical under consideration. These include, for example, whether the query relates to the *travel* domain or whether the query contains the term “news”.
2. *Query-vertical features* are specific to the query-vertical pair (i.e., they are generated from the vertical). These include, for example, the similarity between the query and those previously issued directly to the vertical by users.²

4.4.1 Query Features

Query features are generated from the query and are independent of the vertical. These features are described more completely in Section 3.3.3. Rule-based intent features include regular expressions and dictionary look-ups likely to correlate with vertical intent (e.g., does query contain the term “news”?). Geographic features correspond to the output of a geographic named-entity tagger (e.g., does the query contain a city name?). Categorical features correspond to the output of a query-domain categorization algorithm (e.g., is the query related to the travel domain?). The total number of query features was 118.

4.4.2 Query-Vertical Features

Query-vertical features are query- and vertical-dependent. In other words, their value depends on the query *and* the vertical under consideration. We generated five query-vertical features.

ReDDE.top, described in Section 3.3.1, scores the vertical based on its predicted number of relevant documents for a given query. To do this, it uses a retrieval from a centralized sample index, which combines documents sampled from each vertical. Soft.ReDDE, also described in Section 3.3.1, is a variant of ReDDE.top, but uses a soft document-to-vertical assignment (possibly using an external collection) to compute the vertical’s score. We used an external collection: the English Wikipedia. We included two different Soft.ReDDE features. These differ in the computation of the Wikipedia-article-to-vertical assignment, which is based on the similarity between the Wikipedia article and vertical language model. One version of Soft.ReDDE derived the vertical language model

²One could also imagine having query-independent *vertical features*, for example, whether the vertical is experiencing a sudden increase in query traffic. We did not make use of vertical features in this work.

from vertical samples and a second version derived the vertical language model from the vertical’s query-log. In addition to ReDDE.top and Soft.ReDDE, we used a third resource selection algorithm: GAVG (for geometric average), described in Section 2.3. Like ReDDE.top, GAVG also uses a centralized sample index, but scores the vertical based on the geometric average score from its top m samples. Finally, we use the query likelihood given a language model constructed from the vertical’s query-log. In total, this corresponds to 5 query-vertical features: ReDDE.top, two Soft.ReDDE features, one GAVG feature, and one query-log feature. Each feature type was mass normalized across verticals.

4.5 Methods and Materials

The objective is to predict the relevance of a new target vertical for a given query. For this reason, we evaluate on a per-vertical basis. Given a set of verticals \mathcal{V} with query-vertical relevance labels, each vertical was artificially treated as the new target vertical and all remaining verticals as the source verticals. That is, for each $t \in \mathcal{V}$, we set $\mathcal{S} = \mathcal{V} - t$.

Given a set of queries with vertical-relevance judgements, evaluation numbers were averaged across 10 cross-validation test folds, using the remaining 90% of data for training. GBDTs require tuning several parameters: the number of trees, the maximum number of nodes per tree, and the shrinkage factor (see Friedman [38] for details). These were tuned using a grid search and 10-fold cross-validation on each training fold. In all cases, cross-validation folds were split randomly. Significance is tested using a 2-tailed unpaired t-test.

TRADA was self-trained using predictions made on the test set. More specifically, at each cross-validation step, a basic model was tuned on the training fold (90% of all queries) and applied to the test fold (10% of all queries). Then, a TRADA model was pseudo-trained using predictions on the test fold. In other words, for a given vertical, we trained 10 basic and 10 TRADA models. Rather than tune pseudo-training parameter N , we present results for $N = 2.5, 5, 10\%$.

4.5.1 Verticals

We focused on 11 verticals, described in Table 4.1. These 11 verticals are a subset of the 18 verticals used in Chapter 3. For practical reasons, some verticals in Chapter 3 had missing features (e.g., not every vertical had a vertical-specific search interface from which to derive query-log features). To facilitate a more fair comparison of performance across (target) verticals, in this chapter we focus on those 11 verticals which did not have missing features. These verticals differ across different dimensions: size, domain, document type, and level of query traffic.

Table 4.1: Vertical descriptions.

vertical	retrievable items
finance	financial data and corporate information
games	hosted online games
health	health-related articles
images	online images
jobs	job listings
local	business listings
movies	movie show times
music	musician profiles
news	news articles
travel	travel and accommodation reviews and listings
video	online videos

4.5.2 Queries

Our evaluation data consisted of the same set of 25,195 randomly sampled Web queries used in Chapter 3. Given a query, human editors were instructed to assign verticals to one of four relevance grades (‘most relevant’, ‘highly relevant’, ‘relevant’, ‘not relevant’) based on their best guess of the user’s vertical intent. It is possible for a query to have multiple verticals tied for a particular relevance grade. For about 25% of queries *all* verticals were labeled ‘not relevant’. These are queries for which a user would prefer to see only Web search results.

4.5.3 Evaluation Metrics

We are interested in the accuracy of a model’s target-vertical predictions. Given a set of predictions for a target vertical, precision and recall can be computed by setting a threshold on the prediction confidence value. Rather than report a single precision-recall operating point, we evaluate using ranking metrics. These metrics are computed from a ranking of test queries by descending order of prediction confidence value, the probability that the target vertical is relevant to the query.

We adopt two rank-based metrics: average precision (AP) and normalized discounted cumulative gain (NDCG). In computing AP, the labels ‘most relevant’, ‘highly relevant’, and ‘relevant’ were collapsed into a single grade: ‘relevant’. We then compute average precision according to Equation 4.6.

NDCG differs from AP in two respects. First, it differentiates between relevance grades. Second, given a target vertical, t , it favors a model that is more confident on queries for which t is more relevant. Put differently, compared to AP, it punishes high-confidence errors more severely than low-confidence errors. Following Järvelin and Kekäläinen [52], NDCG for a target vertical t , evaluated over queries \mathcal{Q}_t , is computed as,

$$\mu_{\text{NDCG}}(f, y_t, \mathcal{Q}_t) = \frac{1}{Z} \sum_{r=1}^{|\mathcal{Q}_t|} \frac{2^{y_t(\rho_f(\mathcal{Q}_t)_r)} - 1}{\log(\max(r, 2))} \quad (4.7)$$

Table 4.2: Target-trained (“cheating”) results: AP and NDCG.

vertical	AP	NDCG
finance	0.556	0.861
games	0.741	0.919
health	0.800	0.945
hotjobs	0.532	0.814
images	0.513	0.855
local	0.684	0.926
movies	0.575	0.851
music	0.791	0.934
news	0.339	0.748
travel	0.797	0.947
video	0.290	0.701

where y maps the relevance grade to a scalar (‘most relevant’: 3, ‘highly relevant’: 2, ‘relevant’: 1, ‘not relevant’: 0). The normalizer \mathcal{Z} is the DCG of an optimal ranking of queries with respect to the relevance.

Recall that our objective is to achieve the best performance possible without target-vertical training data. A major motivation is to alleviate the need for a model trained on target vertical data. Therefore, it is useful to measure our performance as a fraction of the performance achievable by a model with access to target training data. We show AP and NDCG results given *human-annotated* target-vertical training data in Table 4.2. A target-specific model (using all available features) was trained and tested for each vertical using 10-fold cross-validation. As in all results, we present performance averaged across test folds. Given our objective, this can be considered a “cheating” experiment. However, these numbers present a kind of upper bound for our methods. Our goal is to approximate these numbers using no target-vertical training data. For this reason, all results beyond this section are normalized by these numbers (i.e., results are reported as percentage of target-trained performance). Also, this normalization facilitates an understanding for the cost-benefit of labeling the new vertical given source-vertical labels and our proposed methods.

4.5.4 Unsupervised Baselines

Section 4.4.2 describes several query-vertical features that are used as input signals to our models. Prior work shows that each of these can be used as an unsupervised single-evidence vertical predictor [2, 3]. In other words, these methods can make target vertical predictions without any (source- or target-vertical) training data. Therefore, to justify the added complexity of our models, we compared against these unsupervised single-evidence approaches. We present results for the unsupervised baseline that performed best in this evaluation: Soft.ReDDE [3]. Soft.ReDDE (the version for which the vertical language model was estimated using vertical samples) performed equal to or better than

the next best single-evidence method for all but 3/11 verticals based on both AP and NDCG.

4.6 Experimental Results

We present portability results in Table 4.3. We report performance for our three portable models, each trained using data from every vertical except the target. Each basic model uses all features and weights all instances equally. Each basic+VB model uses all features and performs instance weighting to balance the positive instances from different verticals. Finally each basic+FW model uses only the most portable features and weights all instances equally.

Across both metrics, both vertical balancing (basic+VB) and feature weighting (basic+FW) improve the performance of the basic model (basic). Performance across verticals was either statistically indistinguishable or better. Compared to each other, feature weighting (basic+FW) significantly improved the basic model across more verticals (8/11 for both metrics). Compared to Soft.ReDDE, the only method that did noticeably better was the basic model with feature weighting (basic+FW). Performance was significantly better for 4 verticals based on AP and 5 based on NDCG. Performance was significantly worse for one vertical based on AP and none based on NDCG.

We present adaptability results in Table 4.4. TRADA adapts a basic model using its own target-vertical predictions as pseudo-training data. Given its superior performance, we used the unbalanced basic model with only the most portable features (basic+FW). We refer to this as the *base* model. TRADA was tested under two conditions. In the first condition, TRADA is given access to all features for adaptation. In the second condition, it is given access to *only* the non-portable features (precisely those purposely ignored in order to improve the portability of the base model).

Table 4.4 presents several meaningful results. First, TRADA does poorly when the adapted model is given access to all features (columns 4-6). In contrast, when given access to only the non-portable features (TRADA+FW), results improve (columns 7-9).

TRADA+FW performs either equal to or better than the base model in all but one case for AP and in *all* cases for NDCG. Similarly, across metrics, TRADA+FW significantly outperforms Soft.ReDDE across most verticals for all values of N . It performs significantly worse than Soft.ReDDE in three cases in terms of AP and none in terms of NDCG. Overall, we view this as a positive result in favor of adaptability. TRADA succeeds at adapting a portable model to a specific target vertical at no additional editorial cost.

4.7 Discussion

In the previous section, we demonstrated that the performance improvement for both portable and adaptive models requires measuring individual feature portability. In order to investigate precisely which features were being selected, we plot the value of π_{AP}^{avg}

vertical	Soft.ReDDE	basic (all feats.)	basic+VB (all feats.)	basic+FW (only portable feats.)
finance	0.446	0.209▼	0.199▼	0.392 [△]
games	0.720	0.636	0.724	0.683
health	0.823	0.797	0.793	0.839
jobs	0.155	0.193	0.226▲	0.321▲ [△]
images	0.283	0.365▲	0.404▲	0.390▲
local	0.696	0.543▼	0.614▼ [△]	0.628▼ [△]
movies	0.477	0.294▼	0.388▼ [△]	0.478 [△]
music	0.757	0.673▼	0.700▼	0.780 [△]
news	0.559	0.293▼	0.434▼ [△]	0.548 [△]
travel	0.487	0.571▲	0.618▲ [△]	0.639▲ [△]
video	0.525	0.449	0.539	0.691▲ [△]
avg	0.539	0.457	0.513	0.581

(a) AP

vertical	Soft.ReDDE	basic (all feats.)	basic+VB (all feats.)	basic+FW (only portable feats.)
finance	0.776	0.663▼	0.651▼	0.775 [△]
games	0.910	0.884	0.918	0.903
health	0.953	0.950	0.946	0.960
jobs	0.563	0.583	0.607	0.671▲ [△]
images	0.712	0.745	0.776▲	0.768▲
local	0.905	0.875▼	0.897 [△]	0.910 [△]
movies	0.775	0.685▼	0.745	0.798 [△]
music	0.937	0.922	0.922	0.957▲ [△]
news	0.852	0.703▼	0.781▼ [△]	0.875 [△]
travel	0.846	0.881▲	0.908▲ [△]	0.911▲ [△]
video	0.817	0.816	0.828	0.902▲ [△]
avg	0.822	0.792	0.816	0.857

(b) NDCG

Table 4.3: Portability results: normalized AP and NDCG. A ▲(▼) denotes significantly better(worse) performance compared to Soft.ReDDE. A [△]([▽]) denotes significantly better(worse) performance compared to the unbalanced basic model with all features. Significance was tested at the $p < 0.05$ level.

in Figure 4.1. As it turns out, the same five features were consistently chosen as the most portable for all target verticals (i.e., for all sets of source verticals). Interestingly, these correspond to our five query-vertical features (Section 4.4.2). Conversely, those

vertical	basic+FW		trada (all features)			trada+FW (only non-portable)		
	Soft.ReDDE	(only portable)	(N=2.5%)	(N=5%)	(N=10%)	(N=2.5%)	(N=5%)	(N=10%)
finance	0.446	0.392	0.364	0.328▼	0.226▼▽	0.476	0.407	0.339▼
games	0.720	0.683	0.735	0.660	0.491▼▽	0.819▲△	0.817▲△	0.787△
health	0.823	0.839	0.814	0.813	0.592▼▽	0.907▲△	0.868▲	0.818
jobs	0.155	0.321▲	0.360▲	0.384▲	0.345▲	0.390▲	0.348▲	0.323▲
images	0.283	0.390▲	0.320▽	0.370▲	0.405▲	0.410▲	0.499▲△	0.523▲△
local	0.696	0.628▼	0.523▼▽	0.601▼	0.609▼	0.562▼▽	0.614▼	0.663
movies	0.477	0.478	0.493	0.462	0.411▽	0.640▲△	0.587▲△	0.578▲△
music	0.757	0.780	0.751	0.778	0.760	0.868▲△	0.866▲△	0.838▲△
news	0.559	0.548	0.509	0.556	0.523	0.607	0.665▲△	0.615
travel	0.487	0.639▲	0.531▽	0.573▲▽	0.597▲	0.744▲△	0.709▲△	0.710▲△
video	0.525	0.691▲	0.633	0.648	0.586	0.735▲	0.722▲	0.688▲
avg	0.539	0.581	0.548	0.561	0.504	0.651	0.646	0.626

(a) AP

vertical	basic+FW		trada (all features)			trada+FW (only non-portable)		
	Soft.ReDDE	(only portable)	(N=2.5%)	(N=5%)	(N=10%)	(N=2.5%)	(N=5%)	(N=10%)
finance	0.776	0.775	0.760	0.718	0.632▼▽	0.826	0.765	0.734
games	0.910	0.903	0.920	0.885	0.778▼▽	0.947▲△	0.951▲△	0.938△
health	0.953	0.960	0.947	0.939	0.827▼▽	0.974▲	0.960	0.953
jobs	0.563	0.671▲	0.720▲	0.736▲	0.710▲	0.747▲	0.698▲	0.676▲
images	0.712	0.768▲	0.745	0.775▲	0.790▲	0.801▲	0.850▲△	0.869▲△
local	0.905	0.910	0.885▼▽	0.907	0.898	0.901	0.911	0.930▲△
movies	0.775	0.798	0.808	0.790	0.732▽	0.856▲	0.842▲	0.840▲
music	0.937	0.957▲	0.939	0.939	0.931	0.977▲△	0.974▲△	0.965▲
news	0.852	0.875	0.850	0.868	0.828	0.898	0.908	0.879
travel	0.846	0.911▲	0.875▽	0.890▲	0.889▲	0.944▲△	0.933▲△	0.926▲
video	0.817	0.902▲	0.869	0.865	0.827	0.930▲	0.896▲	0.880
avg	0.822	0.857	0.847	0.847	0.804	0.891	0.881	0.872

(b) NDCG

Table 4.4: Trada results: normalized AP and NDCG. A ▲(▼) denotes significantly better (worse) performance compared to Soft.Redde. A △(▽) denotes a significantly better(worse) performance compared to the base model. Significance was tested at the $p < 0.05$ level.

features that were the least portable were consistently our remaining query features (Section 4.4.1). In hindsight, this observation makes sense. Our query-vertical features included several existing resource selection methods: GAVG [86], ReDDE.top [3], and Soft.ReDDE [4]. These methods, by design, score resources using a *single* metric that is expected to be positively correlated with resource relevance *irrespective of the resource*. This result places two-decade’s worth of unsupervised resource selection approaches under a new light. Existing resource selection methods are portable and can therefore be

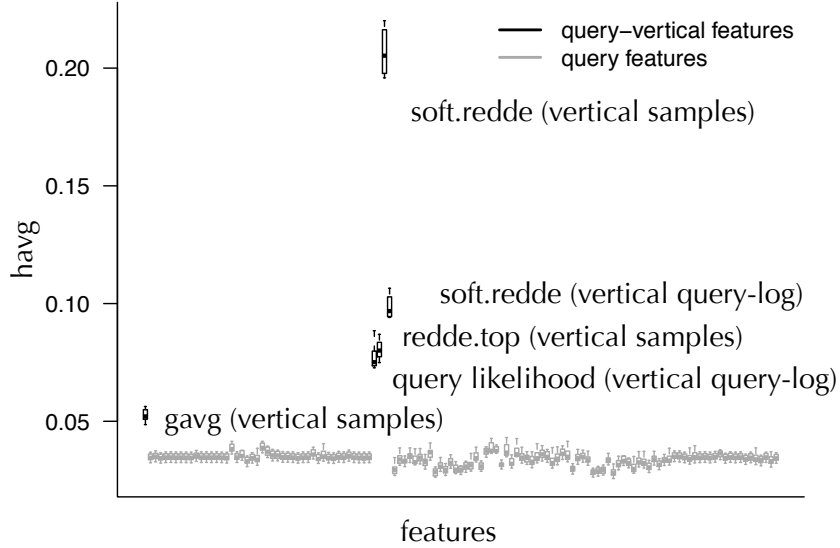


Figure 4.1: Feature portability values (π^{hmap}) for all features across sets of source verticals. Features ordered randomly along x -axis. The features that were consistently the most portable were query-vertical features. As shown, many of these correspond to existing unsupervised resource selection methods used in prior work.

used to harness non-portable (vertical-specific) evidence (e.g., the topic of the query). In other words, existing methods play an important role in adapting a portable model to a new vertical at not additional editorial cost.

Vertical balancing significantly improved the basic model only across 4/11 verticals based on AP and 3/11 based on NDCG. Recall that we introduced balancing in order to discourage examples from source verticals with high priors dominating the training set. However, this does not address cases where several sources have the same non-portable features correlated with relevance. For example, *video* and *images* tend to be relevant to queries that mention a celebrity name; *travel* and *local* tend to be relevant to queries that mention a geographic entity; and *finance* and *jobs* tend to be relevant to queries that contain a company name. Even though vertical balancing addresses a single vertical’s dominance in the training set, it does not address a small coalition of related verticals causing the model to focus on non-portable features. We believe that a more robust averaging technique—for example, the harmonic average used for feature portability—may result in superior performance in the presence of similarly predictive source verticals.

In the previous section, TRADA improved considerably when given access to *only* the non-portable features for adaptation. We believe this is due to the following. TRADA was pseudo-trained using predictions from the base model (basic+FW). This was our best portable model and was given access to only the most portable features. When TRADA is

given access to all features (including the most portable ones), the adapted model tends to focus on the portable features (and ignore the non-portable ones) in order to better fit the original base-model predictions. On the other hand, when TRADA is given access to only most portable features for adaptation, then it *must* focus on the non-portable, vertical-specific ones. As it turns out, when given access to target-vertical training data, the non-portable, vertical-specific features are highly effective for predicting the target vertical. In Table 4.5, we compare the performance of two target-trained models: one that is given access to all features (portable and non-portable) and one that is given access to only the most portable ones.³ Performance across all verticals deteriorates substantially when a target-trained model is not given access to the most non-portable features. When given access to only the non-portable features for adaption, TRADA is forced to focus on these highly effective vertical-specific features. This mechanism can be seen as a sort of regularization ensuring that both portable and non-portable features are used for prediction.

vertical	all features: portable and		vertical	all features: portable and	
	non-portable	only portable		non-portable	only portable
finance	0.556	0.360▼	finance	0.861	0.758▼
games	0.741	0.623▼	games	0.919	0.877▼
health	0.800	0.690▼	health	0.945	0.917▼
hotjobs	0.532	0.271▼	hotjobs	0.814	0.673▼
images	0.513	0.282▼	images	0.855	0.711▼
local	0.684	0.605▼	local	0.926	0.906▼
movies	0.575	0.325▼	movies	0.851	0.719▼
music	0.791	0.660▼	music	0.934	0.903▼
news	0.339	0.232▼	news	0.748	0.677▼
travel	0.797	0.608▼	travel	0.947	0.894▼
video	0.290	0.246▼	video	0.701	0.658

(a) AP

(b) NDCG

Table 4.5: Target-trained results. Given access to target-vertical training data, a model that uses all features (portable + non-portable) performs significantly better than one that is given access to only the non-portable ones. A ▼ denotes a significant degradation in performance at the $P < 0.05$ level.

With respect to pseudo-training data parameter N , we observe that the optimal value of N seems to be vertical-dependent. There may be two reasons for this. First, the base model performance (column 5 in Table 4.3) is also vertical-dependent. Given a fixed value of N across verticals, pseudo-labels from some verticals may be noisier than others.

³This is a “cheating” experiment, as we assume no target-vertical training data. However, we do this for the purpose of error analysis.

Second, the optimal value of N for a given vertical may correlate with the vertical’s prior. That is, it may depend on the number of queries for which the vertical is *truly* relevant.

4.8 Summary

Maximizing the return on editorial investment is an important aspect of any system requiring training data. We presented an ensemble of approaches which significantly improve prediction of a new target vertical using only source-vertical training data. Our results demonstrate that *model portability*, the ability of a model to generalize across different target verticals, requires careful attention to *feature portability*, the ability of a *feature* to correlate with vertical relevance across different target verticals. We found that those features which seemed to be the most portable—and hence most important for a portable model—were query-vertical features as opposed those that are independent of the candidate vertical. Conversely, when we tried to adapt a model for a specific target, the least portable features appeared to be those most important for the adapted model to consider.

Furthermore, we showed that a portable solution can be used to build a target-specific one at no additional editorial cost. Our best approach adapted a portable model using its own target-vertical predictions. This approach consistently outperformed both the base model and a competitive alternative which does not require adaptation. Results also showed that, given available resources, human-annotation on the new target vertical remains the best alternative.

This work could be extended in several directions. In terms of portability, vertical balancing may be improved by modeling the similarity (in terms of predictive evidence) between source verticals. In terms of adaptability, further improvements may be achieved by modeling the similarity between each source vertical and the target vertical. In the absence of target-vertical training data, we may be able to measure this source-to-target similarity using query-traffic data.

Vertical Results Presentation: Evaluation Methodology

Aggregated web search is a two-part task. Up to this point, we have focused on *vertical selection*—deciding *which* verticals are relevant to the query. In this and the next chapter we focus on *vertical results presentation*—deciding *where* in the Web results to present the vertical results. Together, these two sub-tasks combine to produce the end-to-end system output: the final presentation of results. An example *presentation* for the query “pittsburgh” is shown in Figure 5.1. One natural question is: how good is this presentation of results? Suppose we swapped *maps* and *news*. Would that presentation be better? And if so, would the improvement be *noticeable*? The first step towards addressing the presentation task is to determine what the system should do. In other words, we need an evaluation methodology that can help us determine the quality of a particular arrangement of Web and vertical results. Not only is an evaluation methodology essential for comparing systems, it is *also* essential for model building (i.e., for automatically tuning a system to produce high-quality output).

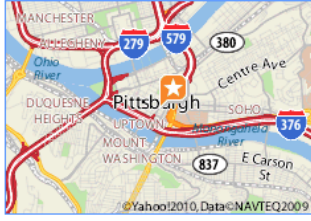
Aggregated web search results are typically evaluated based on user feedback, collected either implicitly (e.g., by observing user clicks and skips [28, 63, 67, 78, 101]) or explicitly (e.g., by directly asking users which presentations they prefer [121]). Most of this prior work, however, focuses on the integration of at most a *single* vertical into the Web results, either assuming a fixed location for the vertical [28, 63, 67] (e.g., above the Web results), or a few alternative locations [101, 121] (above, in the middle of, or below the Web results). We want an evaluation methodology that can handle *multiple* verticals in *multiple* positions. Ponnuswami *et al.* [78] focus on the integration of potentially multiple verticals throughout the Web results. However, they use metrics that measure performance for each vertical *independently* (vertical click-through rate and coverage). While vertical-specific metrics are informative, we want an evaluation metric that can determine the quality of a particular presentation as a whole.

Our objective in this chapter is to propose and empirically validate a methodology for aggregated web search evaluation. This includes evaluating not only *which* verticals are presented/suppressed (as we do in Chapters 3 and 4), but also *where* they are presented. As mentioned above, the number of possible presentations for a given query is large. In spite of this, we want an evaluation metric that uses a relatively small number of human relevance judgements to evaluate *any* possible presentation for a given query.

¹This work was published in ECIR 2011 with co-authors Fernando Diaz, Jamie Callan, and Ben Carterette [5].

pittsburgh Search

maps



Pittsburgh, PA 15219 Map
maps.yahoo.com

[View Larger Map](#) | [What's Nearby?](#) | [Satellite Map](#)

Get Directions

From

Make this my default location

web

City of Pittsburgh, Pennsylvania - Pghgov.com
City of Pittsburgh municipal website ... Public Works Pothole Patrol - When Public Works crews are not responding to winter weather conditions, they will be on ...
www.city.pittsburgh.pa.us - [Cached](#)

news

Pittsburgh - News Results

[Recounting Abraham Lincoln's only trip to Pittsburgh, 150 years ago](#)
Pittsburgh Post-Gazette - 10 hours ago

[Pittsburgh, Lincoln crossed paths in trip to D.C.](#), Pittsburgh... - 10 hours ago

[Pittsburgh-Villanova Preview](#) AP via Yahoo! Sports - Feb 11 01:02pm

web

Pittsburgh - Wikipedia, the free encyclopedia
[Etymology](#) | [History](#) | [Geography](#) | [Demographics](#)

Pittsburgh is the second-largest city in the U.S. Commonwealth of Pennsylvania and the county seat of Allegheny County. Regionally, it anchors the largest urban area of Appalachia and...

en.wikipedia.org/wiki/Pittsburgh - [Cached](#)

images

Pittsburgh - Image Results




Figure 5.1: An example presentation of vertical results for “pittsburgh”, where the *maps* vertical is presented above, the *news* vertical within, and the *images* vertical below the top Web results. Consistent with the major commercial search engines (i.e., Bing, Google, and Yahoo!), we assume that same-vertical results must appear grouped together—horizontally (e.g., *images*) or vertically (e.g. *news*)—within a in the output presentation.

The proposed methodology is defined by two main components. The first component is the prediction of a ground truth or *reference* presentation. The *reference* presentation marks the best possible presentation a system can produce given the query (and a set of layout constraints). Because the set of all possible presentations is prohibitively large, we do not attempt to identify the *reference* by eliciting judgements directly on full presentations. Instead, we take a piece-wise, bottom-up approach. In other words, we collect human judgements on sequences of Web and vertical results that must appear grouped together in the final presentation. Then, we use these piece-wise judgements to derive the *reference* presentation. Finally, we propose that any possible presentation of results can be evaluated based on its distance (using a rank-based distance metric) to the reference. This rank-based distance metric is the second major component. To validate our evaluation metric, we empirically test whether the metric correlates with user preferences. More specifically, we show assessors pairs of presentations and ask whether they prefer

one over the other. In cases where the assessors state a preference, we verify whether the preferred presentation is the one scored superior by our metric.

5.1 Related Work in Aggregated Web Search Evaluation

Vertical results presentation focuses on deciding *where* to present vertical results (if at all). In this chapter, we focus on evaluation. How can we determine the quality of a particular presentation of results from zero or more verticals? Any evaluation metric should be grounded on user behavior. Several studies investigate user behavior with aggregated web search interfaces, for example, by considering click-through behavior and user preference behavior. We review this related work in Section 5.1.1. Subsequently, in Section 5.1.2, we describe existing alternatives for aggregated web search evaluation.

5.1.1 Understanding User Behavior

Several studies (most by Sushmita *et al.* [87, 100, 101, 102]) investigate how users interact with aggregated web search interfaces and how these affect search quality, both from the user's perspective and in terms of task completion.

Sushmita *et al.* [87, 102] present several analyses of a large commercial query-log, with clicks on Web and cross-vertical content. Several trends were found. First, most sessions (about 90%) had clicks on a single type of content (either Web content or content from the same vertical). Second, most sessions with clicks on multiple types of content included clicks on Web content. Finally, as might be expected, users clicked more on highly ranked content irrespective of its source. However, the distribution was flatter for *images* results. That is, click-through rate for *images* results was less affected by rank. This may be due to the visually-appealing nature of image thumbnails, typically used to represent *images* results. From these query-log analyses, one might draw the following conclusions. First, Web results continue to be highly desired. Second, predicting *where* in the Web results to embed a particular (relevant) vertical is an important task (users tend to examine those ranked higher). Third, click-through behavior may be vertical-dependent.

Sushmita *et al.* [100] conducted a small task-oriented user study with two types of interfaces: a tabbed interface, where the user could access results from different verticals by clicking on different tabs, and an aggregated interface, where the user was presented the top results from each vertical within the search results page. The aggregated search interface was *static*—each vertical was allocated a specific query-independent slot. Users were given a topic and asked to compile as much relevant content about the topic as possible. A few trends were observed. The aggregated interface was associated with more clicks. In other words, participants inspected a greater number of results using the aggregated interface. Also, the amount and diversity of content collected by participants (i.e., perceived as being relevant) was greater for the aggregated interface. The user study suggests that users interact with vertical content more when the vertical results are readily visible within the search results page than when the vertical results are accessible

using tabs.

In a different user study, Sushmita *et al.* [101] investigated user click-through behavior using a *dynamic* aggregated search interface, similar to the one modeled in this dissertation. This work focused on presentations with one of three verticals (*images*, *news*, and *video*) embedded in three positions relative to the Web results (above the first Web results, between Web results 5 and 6, and below the last Web result). They found a correlation between click-through rate and both the relevance of the vertical and its rank. More surprisingly, perhaps, they found a click-through bias in favor of *video* results. That is, users clicked more on *video* irrespective of its relevance and rank. Zhu and Carterette [121] conducted a similar study with the *images* vertical. They found a preference for *images* ranked high for queries likely to have image intent. From these studies we can draw the conclusion that users prefer relevant verticals ranked high. Thus, a dynamic aggregated interface, where a vertical’s position is query-dependent, may be more appropriate than a static one.

Our proposed formulation of the task and our evaluation metric are consistent with these studies. More specifically, we always present Web results, we do not assume a fixed slot for a particular vertical, and we propose an evaluation metric that punishes mistakes at the top of the ranking more than at the bottom.

5.1.2 Aggregated Web Search Evaluation

Most research on aggregated web search focuses on vertical selection—predicting which verticals are relevant. Besides the vertical selection work presented in this dissertation (Chapters 3 and 4), Li *et al.* [67] evaluate classifiers for two verticals: *shopping* and *jobs*. Performance was measured in terms of precision and recall for each vertical independently. Diaz [28] and König *et al.* [63] focused on predicting when to display news results (always displayed above the Web results) and evaluated in terms of correctly predicted clicks and skips. Diaz [29] investigated single-vertical selection in the presence of user feedback. Evaluation was conducted using a simulated stream of queries, where each query impression was associated with at most a *single* relevant vertical. Performance was measured in terms of correctly predicted queries, weighing false positives less than false negatives. Their assumption is that users are more dissatisfied by not seeing a relevant vertical than by seeing a non-relevant one (i.e., a user can easily skip over a non-relevant vertical to get to the desired Web results).

The above work assumes at most a *single* relevant vertical per query and, either implicitly or explicitly, assumes a fixed presentation template—a single vertical presented above the Web results or not at all [28, 29, 63, 67]. For the purpose of vertical results presentation evaluation, this work has several limitations. First, even if at most a *single* vertical were truly relevant for every query, its optimal positioning may depend on its *degree* of relevance and the relevance of the Web results. Second, in practice, the system can only guess when a vertical is relevant. Thus, displaying multiple verticals at different ranks is a way of resolving contention between verticals. The best presentations are those that rank the true relevant vertical higher. Finally, though prior research suggests

that single-vertical intent is more common than multiple-vertical intent [100], in some situations, a user may actually want results from multiple verticals simultaneously. For all these reasons, our goal is an evaluation methodology that can handle an arbitrary number of verticals embedded throughout the Web results.

Punnuswami *et al.* [78] propose a method for vertical results presentation with multiple verticals and slotting positions. Their approach is to train independent vertical classifiers (one per vertical) and to assign each vertical to a slot using its prediction confidence value and a set of slot-specific thresholds. In their approach, each vertical is assigned to a slot *independently* of any other vertical. In a similar fashion, performance for each vertical was evaluated independently of any other. Evaluation was conducted in an operational setting based on implicit user feedback. Their evaluation methodology is to allocate a small percentage of query-traffic to one of two conditions: the control condition, which uses the existing baseline system, and the experimental condition, which uses the proposed system. Then, performance (for a specific vertical and slot) is measured in terms of *coverage* and *vertical click-through rate*. Coverage measures the percentage of queries for which the vertical is assigned to the slot. Vertical click-through rate measures the percentage of *those* queries for which the user clicks on the vertical. The goal for the experimental system is to improve click-through rate while maintaining an acceptable level of coverage.¹

Evaluating based on click data has the advantage that the implicit judgements are provided by real users within the context of a real search. On-line evaluation, however, is time-consuming and can degrade the user experience if the experimental system is worse than the baseline. Recent work proposes methods for collecting on-line user-interaction data once and using this data to perform *multiple* rounds of off-line testing [66, 77]. The objective of this work is to perform off-line testing while avoiding a system bias. That is, we do not want off-line evaluation to favor systems that are similar to the one used to collect the user interaction data. The basic idea is to collect the user interaction data in a completely unbiased fashion, where every system output is *equally* probable. Then, given a particular model (with a fixed parameter configuration), evaluation can be done by considering only those outputs that are identical to what the system would have produced given the same query. Results show that metrics computed in this off-line fashion closely approximate those computed in an on-line setting using the same experimental system [66, 77].

Whether evaluation is on-line or off-line (i.e., using cached results, features, and user clicks), evaluating based on user interaction data requires a live system with users. Moreover, because clicks (and skips) are noisy, they require *many* users. Our goal is a evaluation methodology that does not suffer from this cold-start problem.

In this work, we evaluate alternative search results by presenting them to users side-by-side and asking which they prefer. Several works elicit preference judgements on pairs of search results. Thomas and Hawking [103] validated the side-by-side compar-

¹If click-through rate improves, but coverage decreases significantly, then the experimental system is simply being more conservative in predicting the vertical, which may not be considered an improvement.

ison approach by presenting assessors with pairs of results of different quality (e.g., Google results 1-10/11-20, or overlapping sets 1-10/6-15). As expected, users typically preferred results 1-10 over 11-20 or 6-15. Sanderson *et al.* [82] used a similar interface with Amazon’s Mechanical Turk to validate a set of test-collection-based metrics: Normalized Discounted Cumulative Gain (NDCG), $\mathcal{P}@10$, and Mean Reciprocal Rank (MRR). NDCG agreed the most with user preferences, obtaining 63% agreement overall and 82% agreement for navigational queries. We use a similar methodology for validating our metric.

5.2 Layout Assumptions

In order to formally define the *vertical results presentation* task, we must first assume a set of layout constraints. As previously mentioned, our goal in this chapter is an evaluation metric that can measure the quality of *any* possible presentation of results for a given query. The following layout constraints define the set of *all* possible presentations.

First, we assume that vertical results (if any) can only be embedded or *slotted* in specific positions relative to the top Web results (we focus on the top 10 Web results). Specifically, we assume 4 vertical slotting positions: above the first Web result, between Web results 3-4, between Web results 6-7, and below the last Web result. A similar assumption is made in prior work [28, 101, 121]. Effectively, this divides the top 10 Web results into three *blocks* of Web results, denoted as w_1 , w_2 , and w_3 . Multiple verticals can be presented in a particular slot, that is, above w_1 , between w_1 and w_2 , between w_2 and w_3 , or below w_3 . Web results within the same Web block cannot be partitioned in any presentation.

Second, we assume that, if a vertical is presented, a fixed set of its top results must be presented. This assumption simplifies the vertical results presentation task. If a vertical is presented, the system does not have to decide which of its top results to present. Instead, the task is to determine whether to present a vertical given its top results, and, if so, where. Each vertical is associated with a minimum and maximum number of top results that must be presented if the vertical is presented. If the vertical retrieves fewer than its minimum number of results, it is not a candidate vertical.

Third, if a vertical is presented, then its top results must be grouped together—vertically (e.g., *news*) or horizontally (e.g., *images*)—in the final presentation. This assumption is consistent with all major commercial search providers and is at least partly motivated by the fact that results from different verticals are presented differently. For example, image results are presented as thumbnail images; news results are presented using their title, news-source, and age or publication date; and local business results are presented using the name of the business, its contact information, and their location is often displayed geographically on a map. Combining our second and third assumptions, every *candidate* vertical (i.e., every vertical that retrieves at least its minimum number of results for the query) forms its own block of results. As with our Web blocks, results from the same vertical block cannot be partitioned in any presentation.

Fourth, we assume that Web blocks w_{1-3} are always displayed and are done so in their original order. Web results are not re-ranked relative to each other. Finally, we

assume that users prefer to not see results from non-relevant verticals, even below the last Web block: w_3 . Non-relevant vertical results should be suppressed entirely from view.

5.3 Task Formulation: Block Ranking

Given our layout assumptions, we can formulate the vertical results presentation task as one of ranking blocks of Web and vertical results, as shown in Figure 5.2. A *block* is defined as a sequence of Web or same-vertical results that must be presented together in the final aggregated results. If a block is presented, then all the results within the block must be presented and must be grouped together. If a block is suppressed, then all the results within the block must be suppressed.

Let \mathcal{B}_q denote the set of blocks associated with query q . \mathcal{B}_q will always include all three Web blocks (w_{1-3}) and will include one block for each vertical that retrieves results. The goal of block ranking is to produce an ordering of \mathcal{B}_q , denoted by σ_q . Suppressed verticals will be handled using an imaginary “end of search results” block, denoted by *eos*. Blocks that are ranked above *eos* are displayed and those ranked below *eos* are suppressed. Let $\sigma_q(i)$ denote the rank of block i in σ_q . We say σ_q is a *partial* ranking because all blocks ranked below *eos* (i.e., those that are suppressed) are effectively tied at rank $\sigma_q(\text{eos}) + 1$.

Our objective is an evaluation measure that can determine the quality of *any* possible presentation σ_q for query q . Our solution is to evaluate alternative presentations based on their distance (using a rank-based distance metric) to a “gold standard” or *reference* presentation, denoted by σ_q^* . We assume that σ_q^* defines the best possible block-ranking for query q . Given the prohibitively large number of possible presentations for a given query, we do not elicit human judgements directly on alternative presentations. Instead, σ_q^* is derived from judgements on individual blocks (we describe this in more detail below). Any evaluation metric should be grounded on user preference data. We validate the metric by making sure that the metric agrees with human preferences stated on pairs of presentations.

5.4 Metric-Based Evaluation Methodology

Our metric-based evaluation methodology is depicted in Figure 5.3. The primary goal is to derive a *reference* presentations for query q , which we denote by σ_q^* . Then, we propose that any arbitrary presentation σ_q can be evaluated based on its distance (using a ranked based-distance metric) to σ_q^* .

Our methodology proceeds in 4 steps. First, given query q (and our set of layout constraints), we compose a set of blocks associated with the query, denoted by \mathcal{B}_q . Set \mathcal{B}_q will include all three Web blocks w_{1-3} and one block for each vertical that retrieves more than the minimum number of results (Figure 5.3 (a-b)). Then, we generate σ_q^* using human relevance judgements on individual blocks. Prior work on document-ranking

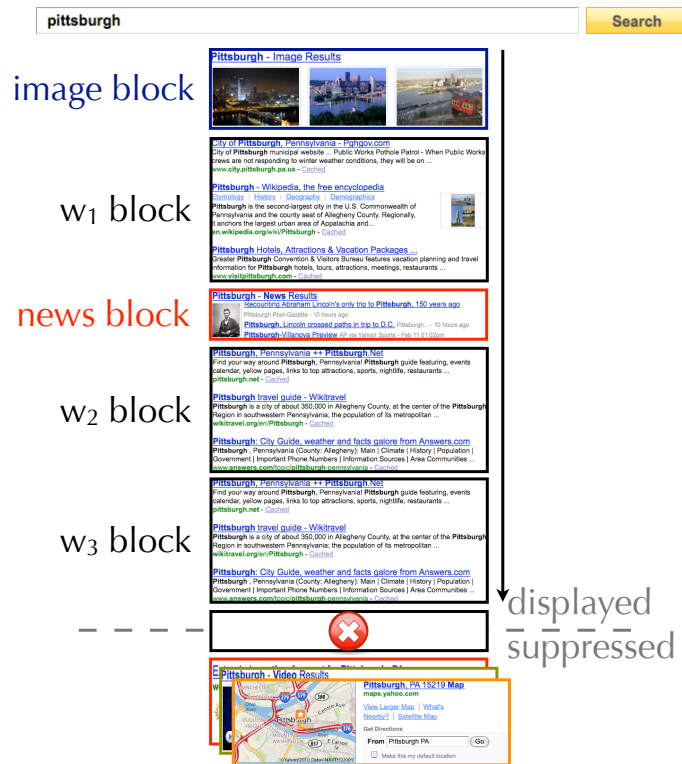


Figure 5.2: The vertical results presentation task can be formulated as *block ranking*.

evaluation found that assessors work faster and have higher agreement when providing pairwise preference judgements than when providing absolute judgements [18]. We assume this is also true for blocks of Web and vertical results. Thus, the second step is to collect pairwise preference judgements on all block-pairs in \mathcal{B}_q (Figure 5.3 (c)). More specifically, each block-pair $i, j \in \mathcal{B}_q$ is presented to assessors who are asked to state a preference. We denote these pairwise preference judgements by π_q . Third, given block-pair preference judgements π_q , we use a voting method to derive the *reference* presentation σ_q^* (Figure 5.3 (d)). Finally, we propose that *any* alternative presentation σ_q for q can be evaluated using a rank-based metric to measure its distance to σ_q^* (Figure 5.3 (e)). We describe these steps in more detail in the next sections.

5.4.1 Collecting Block-Pair Preference Judgements

Given \mathcal{B}_q , our methodology is to collect pairwise preference judgements on *all* block-pairs $i, j \in \mathcal{B}_q$, with the exception of pairs of Web blocks. We do not collect preference judgements between Web block pairs because assume that Web blocks must be presented in their natural order, that is $\sigma_q^*(w_1) < \sigma_q^*(w_2) < \sigma_q^*(w_3)$.

Each pairwise preference judgement consists of a triplet of the form (q, i, j) , composed of query q and block pair i, j presented side-by-side in random order. A screenshot of the assessment interface is shown in Figure 5.4. Assessors were given three choices: i

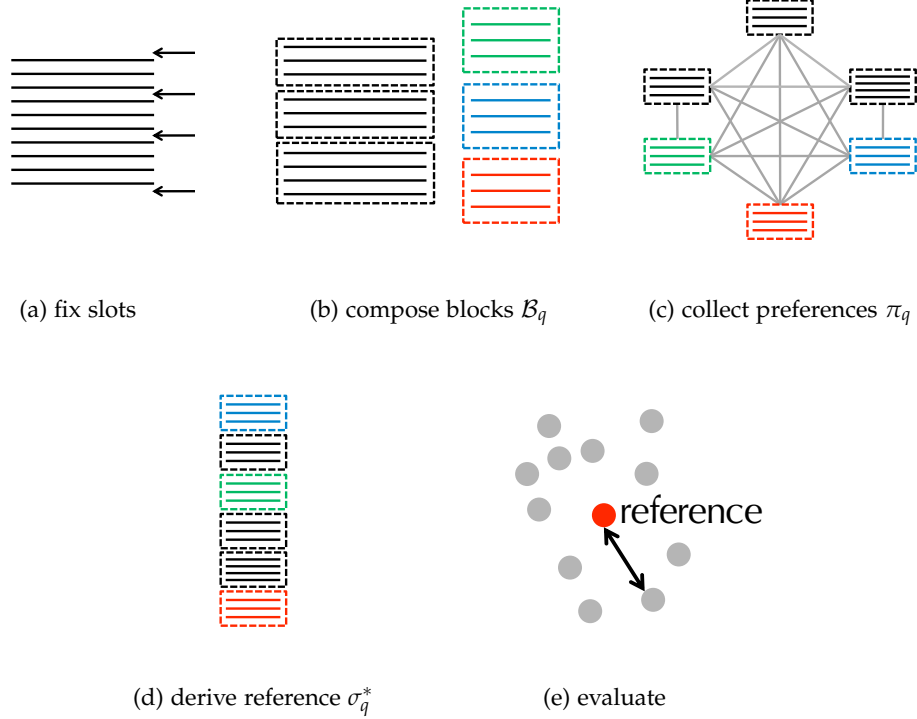


Figure 5.3: Approach Overview.

is better j , j is better i , and both are bad. We omitted the choice that both i and j are equally good to prevent assessors from abstaining from difficult decisions. We interpret the assessor selecting “both are bad” as evidence that both i and j should be suppressed for query q . These preference judgements, denoted by π_q , are the raw input to the voting method that derives the *reference* presentation σ_q^* .

Let $\pi_q(i, j)$ denote the strength with which block i is preferred over block j (note that $\pi_q(i, j) \neq \pi_q(j, i)$). In general, $\pi_q(i, j)$ can be set differently depending on the particular choice of assessors. For example, suppose assessor reliability is known. Then, we can collect one judgement per block-pair and set $\pi_q(i, j)$ according to the reliability of an assessor who prefers block i over j . In a different setting, for example, when using non-expert annotators, we can collect *redundant* judgements per block-pair and set $\pi_q(i, j)$ equal to the number of assessors who prefer i over j .

5.4.2 Deriving the Reference Presentation

There exist many voting methods for aggregating item preference data into a single ranking. In this work, we used the Schulze voting algorithm because of its widespread adoption and ease of implementation [85]. The Schulze algorithm is described in Algorithm 1. The input to the algorithm is \mathcal{B}_q and preference judgements π_q and the output is the *reference* presentation σ_q^* .

query: dominican republic maps

The user is looking for a map of the Dominican Republic.

left	right
<p>Dominican republic maps - Book Results</p> <p>Dominican Republic Gary Prado Chandler and Liza Prado Chandler - 2005</p> <p>Dominican Republic and Haiti Paul Clammer, Michael Grosberg, and Jens Porup - 2008</p> <p>Dominican Republic, 2nd Sarah Cameron - 2004</p>	<p>Dominican Republic Road Map, Detailed Maps, Travel, Tourist Dominican Republic road map, detailed maps, travel, tourist, street, topographic, city, for independent travelers. See everything! Never get lost. www.maps2anywhere.com/Maps/Dominican_Republic_map.htm</p> <p>Dominican Republic maps from Omnimap.com, world's leading map ... Omnimap.com offers the best selection of maps of Dominican Republic, plus over 250,000 maps and guidebooks for the world, GPS maps, travel ... www.omnimap.com/catalog/int/dominirp.htm</p> <p>Caribbean-On-Line: Dominican Republic: Maps Dominican republic maps. The Borch Map of the Dominican Republic is a folding, plastic ... This map features a large map of the Dominican ... www.caribbean-on-line.com/islands/dr/drm.shtml</p> <p>Map of Dominican Republic Map by Area @ Maps.com Map Store Maps of Dominican Republic from Maps.com -- Maps.com are the world's largest map store featuring Dominican Republic maps along with ... www.maps.com/dominican-republic-map.aspx?cid=1425,1704</p>

Which set of results would best satisfy the user?

- LEFT is BETTER
- RIGHT is BETTER
- BOTH are BAD

Figure 5.4: Block-pair assessment interface. Two blocks are displayed randomly side-by-side (in this case w_3 and w_3) and the assessor is asked which would best satisfy a user for the given query and description: the left block, the right block, or neither.

The general idea behind the Schulze voting method is the following. Let $\pi_q(i, j)$ denote the strength with which block i is preferred over block j . We say that i directly defeats j if $\pi(i, j) > \pi(j, i)$. A *beatpath* from i to j is defined as a direct or indirect defeat from i to j . An *indirect* beatpath from i to j is a sequence of direct defeats from i to j . For example, if i directly defeats k and k directly defeats j , then this is an *indirect* beatpath from i to j . The strength of an indirect beatpath corresponds to strength associated with the weakest direct defeat in the beatpath. Finally, we say that i defeats j if the strongest (direct or indirect) beatpath from i to j is stronger than the one from j to i . Blocks are then ranked by their number of defeats.

As previously noted, the aggregation task is not only ranking blocks, but also deciding which vertical blocks to suppress. Suppressed verticals are handled using the imaginary *eos* block. The *eos* block is treated by the Schulze method the same as any non-imaginary block. Every time an assessor selected that both i and j are bad, we incremented the value of $\pi(eos, j)$ and $\pi(eos, i)$. Also, recall that we assume that Web blocks (w_{1-3}) are always presented and never re-ranked. This constraint was imposed by setting $\pi(eos, w_*) = \pi(w_x, w_y) = 0$, where $x > y$, and by setting $\pi(w_*, eos) = \pi(w_x, w_y) = N$, where $x < y$ and N is some large number (we used $N = 1000$).

Input: blocks \mathcal{B}_q and pairwise preferences π_q
Output: reference presentation σ_q^*

```

foreach  $i \in \mathcal{B}_q$  do
  foreach  $j \in \mathcal{B}_q$  do
    if  $i \neq j$  then
      if  $\pi_q(i, j) > \pi_q(j, i)$  then
         $p(i, j) \leftarrow \pi_q(i, j) - \pi_q(j, i)$ 
      else
         $p(i, j) \leftarrow 0$ 
      end
    end
  end
end
foreach  $i \in \mathcal{B}_q$  do
  foreach  $j \in \mathcal{B}_q$  do
    if  $i \neq j$  then
      foreach  $k \in \mathcal{B}_q$  do
        if  $i \neq k$  and  $j \neq k$  then
           $p(j, k) \leftarrow \max(p(j, k), \min(p(j, i), p(i, k)))$ 
        end
      end
    end
  end
end
foreach  $i \in \mathcal{B}_q$  do
  foreach  $j \in \mathcal{B}_q$  do
    if  $i \neq j$  then
      if  $p(i, j) > p(j, i)$  then
         $wins(i) \leftarrow wins(i) + 1$ 
      end
    end
  end
end
 $\sigma_q \leftarrow \text{SortDescending}(wins)$ 

```

Algorithm 1: The Schulze Voting Algorithm.

5.4.3 Measuring Distance from the Reference

Our proposed method is to evaluate *any* possible presentation σ_q by measuring its distance to the *reference* σ_q^* . We used a rank-based distance metric. Possibly the most widely used rank-based distance metric is Kendall's tau (K), which counts the number of dis-

cordant pairs between two rankings,

$$K(\sigma^*, \sigma) = \sum_{\sigma^*(i) < \sigma^*(j)} [\sigma(i) > \sigma(j)],$$

where $\sigma(i)$ denotes the rank of element i in σ . Kendall’s tau treats all discordant pairs equally regardless of position. In our case, however, we assume that users scan results from top-to-bottom. Therefore, we care more about a discordant pair at the top of the ranking than one at the bottom. For this reason, we used a variation of Kendall’s tau proposed by Kumar and Vassilvitskii [64], referred to as *generalized* Kendall’s tau (K^*), which can encode positional information using element weights.

To account for positional information, K^* models the cost of an *adjacent* swap, denoted by δ . In traditional Kendall’s tau, $\delta = 1$, irrespective of rank. Adjacent swaps are treated equally regardless of position. In our case, however, we would like discordant pairs at the top to be more influential. Let δ_r denote the cost of an adjacent swap between elements at rank $r - 1$ and r . We used the DCG-like cost function proposed in Kumar and Vassilvitskii [64],

$$\delta_r = \frac{1}{\log(r)} + \frac{1}{\log(r+1)}$$

which is defined for $2 \leq r \leq n$. Given rankings σ^* and σ , element i ’s displacement weight $\bar{p}_i(\sigma^*, \sigma)$ is given by the average cost (in terms of adjacent swaps) it incurs in moving from rank $\sigma_q^*(i)$ to rank $\sigma_q(i)$,

$$\bar{p}_i(\sigma^*, \sigma) = \begin{cases} 1 & \text{if } \sigma^*(i) = \sigma(i) \\ \frac{p_{\sigma^*(i)} - p_{\sigma(i)}}{\sigma(i)^* - \sigma(i)} & \text{otherwise} \end{cases},$$

where $p_r = \sum_2^r \delta_r$. The K^* distance is then given by,

$$K^*(\sigma^*, \sigma) = \sum_{\sigma^*(i) < \sigma^*(j)} \bar{p}_i(\sigma^*, \sigma) \bar{p}_j(\sigma^*, \sigma) [\sigma(i) > \sigma(j)].$$

A discordant pair’s contribution to the metric is equal to the product of the two element weights.

5.5 Methods and Materials

One important property of any evaluation measure is that it should correlate with user preference data. In other words, aggregated search results preferred by humans should be judged superior by the metric. In this section, we describe a user study that was conducted to empirically validate the metric. In the next few sections, we first describe our block-pair assessment interface. Block-pair assessments were collected in order to derive reference presentations for a set of queries. Then, we describe our verticals and queries. Finally, we describe our user study, where assessors were presented pairs of presentations (shown side-by-side) and asked to state a preference.

5.5.1 Block-Pair Preference Assessment

Given query q , we collected pairwise preference assessments for all pairs $i, j \in \mathcal{B}_q$ (except between pairs of Web blocks). These assessments were collected in order to derive a reference presentation for each query, denoted by σ_q^* .

All block-pair judgements were collected using Amazon’s Mechanical Turk (AMT). Workers were compensated US\$ 0.01 for each judgement. Because we used untrained, non-expert assessors, we collected *redundant* judgements for each block-pair. Each block-pair was judged by 4 different assessors. Thus, we set $\pi_q(i, j)$ equal to the number of assessors who preferred block i over block j . Likewise, we set $\pi_q(eos, i)$ equal to the number of assessors who selected that block i was bad (i.e., should be suppressed) in conjunction with another block j .

Following Sanderson *et al.* [82], quality control was done by including 150 “trap” HITs (a Human Intelligence Task is a task associated with AMT). Each trap HIT consisted of a triplet (q, i, j) where either i or j was taken from a query other than q . We interpreted an assessor preferring the set of extraneous results as evidence of malicious or careless judgement. If an assessor failed more than 2 trap HITs, *all* their judgements were removed from the judgement pool.

As previously mentioned, while collecting block-pair judgements, in addition to the query, assessors were given a topic description to help disambiguate the user’s intent. In a preliminary experiment, we observed an improvement in inter-annotator agreement from giving assessors topic descriptions. We were careful, however, to not explicitly mention vertical intent. For example, for the query “pressure cooker”, we stated: “The user plans to buy a pressure cooker and is looking for product information.” We did not say: “The user is looking for shopping results.”

5.5.2 Verticals

We focused on a set of 13 verticals constructed using freely-available search APIs provided by eBay (*shopping*), Google (*blogs*, *books*, *weather*), Recipe Puppy (*recipes*), Yahoo! (*community Q&A*, *finance*, *images*, *local*, *maps*, *news*), Twitter (*micro-blogs*), and YouTube (*video*). Figure 5.5 shows a few examples. A screenshot of *every* vertical is given in Appendix A.

Each vertical was associated with a unique presentation of results. For example, *news* results were associated with the article title and url, the news source title and url, and the article’s publication date. The news block also included an optional thumbnail image associated with the top result. *Shopping* results were associated with the product name, condition (e.g., new, used), and price, and also included an optional thumbnail image associated with the top result. *Local* results were associated with the business name and url, its address and telephone number, and the number of reviews associated with it, and included a map. Each vertical was associated with a minimum and maximum number of top results (e.g., 1-4) from which to construct a block.

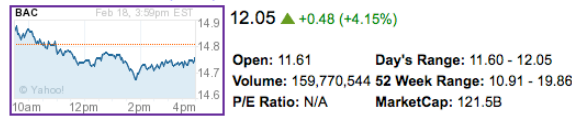
Gardening books - Book Results



[Garden Wisdom & Know-How Everything You Need to Know to Plant, Grow, and Harvest](#)
Rodale Gardening Books and Judy Pray - 2010
[The well-tended perennial garden planting & pruning techniques](#)
Tracy DiSabato-Aust - 2006
[Landscape architecture magazine](#)
American Society of Landscape Architects - 1917


(a) books

Bank of America C (BAC)



(b) finance

Key west florida fishing trips - Local Results



[Key West Fishing Adventures 0 reviews](#)
[www.fishinkeys.com](#)
(305) 393-0566 - 1801 N Roosevelt Blvd, Key West, FL


[Charters Key West - Family Fishing, Sport Fishing 0 reviews](#)
[lethalweaponcharters.com](#)
(305) 744-8225 - 1401 1st St, Key West, FL

[Key West Sheraton Suites 27 reviews](#)
[www.sheratonkeywest.com](#)
(305) 292-9800 - 2001 S Roosevelt Blvd, Key West, FL

[Keys to Key West Incorporated 0 reviews](#)
[gotothekeys.com](#)
(305) 292-7702 - 5555 College Rd, #202, Key West, FL

(c) local

I like to move it move it - Video Results



[Will.I.Am - "I Like to Move It" Madagascar 2: Escape 2 Africa](#) 3:08
[Reel 2 Real - I Like To Move It HQ \[1994\]](#) 3:50
[Happy Feet Like to Move It](#) 2:46
[will.i.am Official Madagascar 2 Music Video: I Like To Move It](#) 2:47

(d) video

Figure 5.5: Example vertical blocks.

5.5.3 Queries

Our evaluation was conducted on a set of 72 queries from two different sources: the AOL query log and Google Trends. Google Trend queries cover recent events and topics currently discussed in news articles, blogs, and on Twitter (e.g., “us open fight”). AOL queries cover more persistent topics likely to be relevant to verticals such as *local* (e.g., “cheap hotels in anaheim ca”), *recipe* (e.g., “cooking ribs”), and *weather* (e.g., “marbella weather”). Queries were selected manually in order to ensure coverage across our set of 13 verticals. The complete list of queries and descriptions is given in Appendix B.

5.5.4 Empirical Metric Validation

The goal of our user study is to determine whether our metric (the K^* distance between σ_q and σ_q^*) agrees with user preferences. That is, whether presentations that are preferred by human assessors are also judged superior by the metric. Assessors were shown pairs of presentations side-by-side (along with the query and its description) and were asked to state a preference (“left is better”, “right is better”). We assumed that assessors would have difficulty deciding between two bad presentations. Therefore, to reduce the cognitive load, we also gave assessors a “both are bad” option (as it turns out, however, this option was seldom used). Our hypothesis is that, if one presentation is preferred over the other, the metric will agree with the stated preference. In other words, the preferred presentation will be closer to σ_q^* in terms of K^* . Significance was tested using a sign test, where the null hypothesis is that the metric selects the preferred presentation randomly with uniform probability.

For this analysis, we also used Amazon’s Mechanical Turk, using trap HITs to detect unreliable assessors. As before, assessors were compensated with US\$ 0.01 per judgement. Given that presentation-pair assessments require a greater effort than block-pairs, presentation-pair assessments took much longer to be completed by AMT workers. Block-pairs were assessed in a couple of days, while presentation-pairs in a couple of weeks. A screenshot of our presentation-pair assessment interface is shown in Figure 5.6.

Conducting this analysis required a method for selecting pairs of presentations to show assessors. One alternative is to sample pairs uniformly from the set of all presentations. However, we were particularly interested in pairs of presentations from *specific* regions of the metric space. For example, does the metric agree with user preferences when one presentation is close to the reference (presumably high quality) and the other is far (presumably low quality)? Does it agree when both are close to the reference or both are far? To investigate these questions, we sampled presentation-pairs using a binning approach.

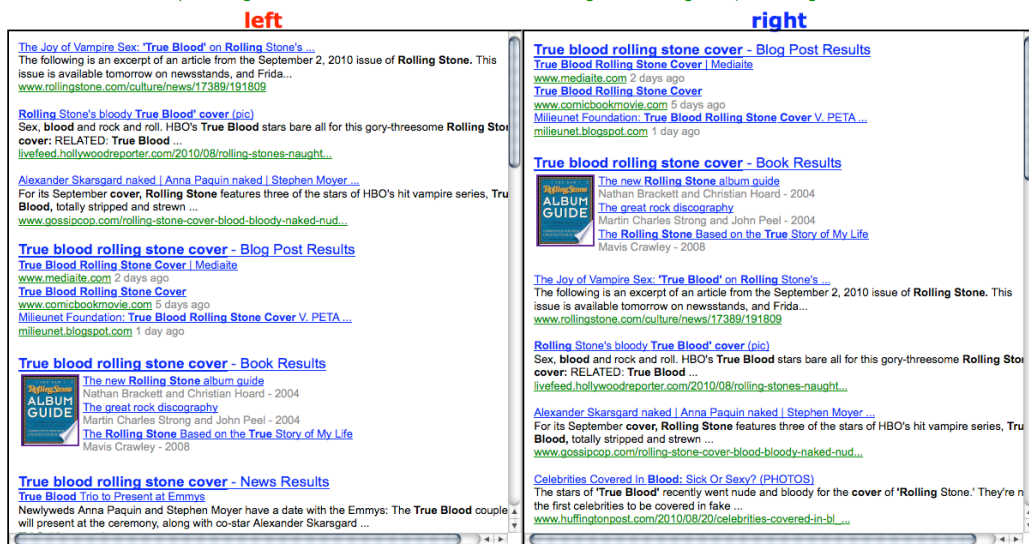
For each query, we enumerated every possible presentation and divided these into three bins: a presumably high-quality bin (\mathcal{H}), a medium-quality bin (\mathcal{M}), and a low-quality bin (\mathcal{L}). The binning was done based on the metric value. The metric distribution is such that this produces bins where $|\mathcal{H}| < |\mathcal{M}| < |\mathcal{L}|$. The \mathcal{H} bin is the smallest and contains those presentations that are nearest to σ_q^* . The \mathcal{L} bin is the largest and contains those presentations that are furthest from σ_q^* . For each query, we sampled 4 presentation-pairs from each of the six bin-combinations ($\mathcal{H}\text{-}\mathcal{H}$, $\mathcal{H}\text{-}\mathcal{M}$, $\mathcal{H}\text{-}\mathcal{L}$, $\mathcal{M}\text{-}\mathcal{M}$, $\mathcal{M}\text{-}\mathcal{L}$, and $\mathcal{L}\text{-}\mathcal{L}$) and collected 4 redundant judgements per presentation-pair. In total, this resulted in 1,728 presentation-pairs and 6,912 individual preference judgements.

5.6 Experimental Results

Our evaluation methodology is to collect preference judgements on block-pairs for a given query, to use these to derive a *reference* block-ranking, and, finally, to evaluate

query: true blood rolling stone cover

The user wants to know the public's general reaction to the new cover of "Rolling Stone" magazine, featuring the cast of TV show "True Blood".



Which set of results would best satisfy the user?

- LEFT is BETTER
- RIGHT is BETTER
- BOTH are BAD

Figure 5.6: Presentation-pair assessment interface. Two presentations are displayed randomly side-by-side and the assessor is asked which they prefer: the left presentation, the right presentation, or neither.

alternative block-rankings by measuring their distance to the *reference*. To validate this methodology, we present a user study where we verify whether block-rankings that are preferred by assessors are scored superior by the metric.

Before presenting our metric validation results (i.e., the level of agreement between the metric and the preferred block-ranking), we present inter-assessor results (i.e., the level of agreement between assessors). We collected two different types of judgements from assessors: block-pair judgements (used to derive the reference presentation) and presentation-pair judgements (used to validate the metric). Inter-assessor agreement results on *block*-pairs are given in Section 5.6.1. Inter-assessor agreement results on *presentation*-pairs are given in Section 5.6.2. Finally, in Section 5.6.3, we present our metric-validation results.

5.6.1 Assessor Agreement on Block-Pair Judgements

The total number of AMT workers who contributed block-pair assessments was 120. Of these, 2 workers had their assessments removed due to failing more than 2 "traps". For the remaining 118/120, worker contribution followed a power-law distribution. That is, about 20% of the workers (24/118) completed about 80% of all block-pair assessments

(9,856/12,293) .

We report inter-annotator agreement in terms of Fleiss’ Kappa (κ_f) [37] and Cohen’s Kappa (κ_c) [22], both which correct for agreement due to chance. Fleiss’ Kappa measures the (chance-corrected) agreement between *any* pair of assessors over a set of triplets. Cohen’s Kappa measures the (chance-corrected) agreement between a *specific* pair of assessors over a *common* set of triplets. Compared to Cohen’s Kappa, Fleiss’ Kappa is convenient because it ignores the identity of the assessor-pair. It is designed to measure agreement over instances labeled by different (even disjoint) sets of assessors. However, precisely because it ignores the identity of the assessor-pair, Fleiss’ Kappa is dominated by the level of agreement between the most active assessors, which we know to be a selected few (i.e., 20% of the workers completed 80% of the judgements). To compensate for this, in addition reporting Fleiss’ Kappa, we report the Cohen’s Kappa agreement for all assessors-pairs with at least 100 triplets in common.

The Fleiss’ Kappa agreement over *all* triplets was $\kappa_f = 0.656$, which is considered *substantial* agreement based on Landis and Koch [65]. In terms of Cohen’s Kappa agreement, there were 25 pairs of assessors with at least 100 triplets in common. Of these, 5 (20%) had *moderate* agreement ($0.40 < \kappa_c \leq 0.60$), 16 (64%) had *substantial* agreement ($0.60 < \kappa_c \leq 0.80$), and the remaining 4 (16%) had *perfect* agreement ($0.80 < \kappa_c \leq 1.00$). Overall, assessor agreement on block-pairs was high. We interpret this as evidence that assessors did not have difficulty providing preferences for pairs of Web and vertical blocks.

In terms of sources of disagreement, a pair of assessors could disagree in two ways: by having one assessor state a preference and the other state the *opposite* preference or by having one state a preference and the other state that “both are bad”. We may view the first type of disagreement as more severe—the assessors prefer the blocks in the opposite order. To examine which type of disagreement was more common, we used a *weighted* version of Cohen’s Kappa [23] and penalized the first type of disagreement more than the second. This increased the Cohen’s Kappa agreement for 20/25 assessors with at least 100 common triplets. Therefore, when assessors disagreed, it was typically *not* because they preferred the blocks in the opposite order, but rather because one assessor stated a preference and the other stated “both are bad”.

5.6.2 Assessor Agreement on Presentation-Pair Judgements

Assessor agreement on presentation-pairs is given in Table 5.1. Overall, assessor agreement on presentation-pairs was low ($\kappa_f = 0.216$), which is considered *fair* agreement [65]. It was significantly lower than assessor agreement on block-pairs. This may not be surprising. Agreement on presentation-pairs requires that assessors agree on the relative costs associated with different types of errors: false-positives (displaying a non-relevant vertical), false-negatives (suppressing a relevant vertical), and ranking errors (displaying a relevant vertical in the wrong position). The cost associated with each error type may be highly user-specific. More than 4 judgements per presentation-pair may be required to see greater convergence.

Although assessor agreement on presentation-pairs was low, a few trends are worth noting. First, agreement was particularly low (nearly random) on \mathcal{H} - \mathcal{H} pairs ($\kappa_f = 0.066$). This is because the \mathcal{H} bin corresponds to those presentations closest to σ_q^* based on K^* , which focuses on discordant pairs within the top ranks. As it turns out, about half of all \mathcal{H} - \mathcal{H} pairs had the same top-three blocks and *all* pairs had the same top-two blocks. The low inter-assessor agreement may be explained by users primarily focusing on the top results, perhaps rarely scrolling down to see the results below the “fold”. Alternatively, it may be that assessors have a hard time distinguishing between good presentations. Further experiments are required to determine the exact cause of disagreement.

Assessor agreement was the highest for \mathcal{H} - \mathcal{M} and \mathcal{H} - \mathcal{L} . This means that assessors agreed more on pairs where one presentation was close the σ_q^* (based on K^*) and the other was far. Agreement was lower on \mathcal{M} - \mathcal{M} , \mathcal{M} - \mathcal{L} , and \mathcal{L} - \mathcal{L} pairs. We revisit these points below.

bins	κ_f
\mathcal{H} - \mathcal{H}	0.066
\mathcal{H} - \mathcal{M}	0.290
\mathcal{H} - \mathcal{L}	0.303
\mathcal{M} - \mathcal{M}	0.216
\mathcal{M} - \mathcal{L}	0.179
\mathcal{L} - \mathcal{L}	0.237
all	0.216

Table 5.1: The Fleiss’ Kappa agreement on presentation-pairs.

5.6.3 Metric Validation Results

Given the low level of inter-assessor agreement on presentation-pairs, rather than focus on the metric’s agreement with each *individual* preference, we focus on the metric’s agreement with a *majority* preference. We present results for two levels of majority preference: a majority preference of 3/4 or greater and a perfect (4/4) majority preference. These results are presented in Table 5.2. The “pairs” column shows the number of presentation pairs for which the level of majority preference was observed. The “% agreement” column shows the percentage of these pairs for which the metric agreed with the majority preference. Notice that, across bin-combinations, column “pairs” correlates with inter-assessor agreement (Table 5.1).

The metric’s agreement with the majority preference was 67% on pairs where at least 3/4 assessors preferred the same presentation and 73% on pairs where all (4/4) assessors preferred the same presentation (both significant at the $p < 0.005$ level). Agreement with each individual preference (not in Table 5.2) was 60% (also significant at the $p < 0.005$ level).

bins	majority preference	pairs	% agreement
all	3/4 or greater	1151	67.07% [▲]
\mathcal{H} - \mathcal{H}	3/4 or greater	164	60.37% [▲]
\mathcal{H} - \mathcal{M}	3/4 or greater	210	81.90% [▲]
\mathcal{H} - \mathcal{L}	3/4 or greater	204	84.31% [▲]
\mathcal{M} - \mathcal{M}	3/4 or greater	184	57.61% [△]
\mathcal{M} - \mathcal{L}	3/4 or greater	187	50.80%
\mathcal{L} - \mathcal{L}	3/4 or greater	202	63.37% [▲]
all	4/4	462	72.51% [▲]
\mathcal{H} - \mathcal{H}	4/4	47	65.96% [△]
\mathcal{H} - \mathcal{M}	4/4	95	87.37% [▲]
\mathcal{H} - \mathcal{L}	4/4	97	91.75% [▲]
\mathcal{M} - \mathcal{M}	4/4	75	58.67%
\mathcal{M} - \mathcal{L}	4/4	71	54.93%
\mathcal{L} - \mathcal{L}	4/4	77	63.64% [△]

Table 5.2: Metric agreement with majority preference. Significance is denoted by \triangle and \blacktriangle at the $p < 0.05$ and $p < 0.005$ level, respectively.

Possibly the most important trend observed in these results is that the metric’s agreement with the majority preference was higher on pairs where there was greater consensus between assessors. Across all bin-combinations, the metric’s agreement with the majority preference was higher on presentation-pairs that had a perfect (4/4) majority preference than on pairs that had a (3/4) majority preference or greater. This is a positive result if we primarily care about pairs where one presentation was strongly (or even unanimously) preferred over the other. This same trend can also be seen across bin-combinations. The metric’s agreement with the majority was the highest on \mathcal{H} - \mathcal{M} and \mathcal{H} - \mathcal{L} pairs (82-92%). These were also the bin-combinations with the highest inter-assessor agreement ($\kappa_f = 0.290$ for \mathcal{H} - \mathcal{M} and $\kappa_f = 0.303$ for \mathcal{H} - \mathcal{L}).

The stronger agreement between the metric and the majority preference for \mathcal{H} - \mathcal{M} and \mathcal{H} - \mathcal{L} pairs as well as the higher inter-assessor agreement also means that, on average, the reference presentation σ_q^* was good. Assessors strongly preferred presentations close to σ_q^* over presentations far from σ_q^* in terms of K^* . The metric’s agreement on \mathcal{H} - \mathcal{H} pairs (60-66%) was lower. However, as discussed above, these pairs were very similar in terms of the top-ranked blocks and assessor agreement was *also* low.

Finally, the metric was less reliable for \mathcal{M} - \mathcal{M} , \mathcal{M} - \mathcal{L} , and \mathcal{L} - \mathcal{L} pairs. However, again, inter-assessor agreement was also lower for pairs in these bin-combinations ($\kappa_f = 0.216$, $\kappa_f = 0.179$, and $\kappa_f = 0.237$, respectively). Inter-assessor agreement (and the metric’s agreement with the majority preference) was lower when neither presentation was close to σ_q^* .

5.7 Discussion

We examined the queries for which the metric’s agreement with the majority preference was the lowest. In some cases, assessors had a strong preference towards a particular vertical, but only when seen in the context of a full presentation, not during the block-pair assessment phase. For example, for the query “ihop nutritional facts”, assessors favored presentations with *images* ranked high. For the query “nikon coolpix”, assessors favored presentations with *shopping* ranked high. For the queries “san bruno fire”, “learn to play the banjo”, “miss universe 2010”, and “us open fight”, assessors favored presentations with *video* ranked high.

Compared to some of the other verticals, the *images*, *shopping*, and *video* verticals are visually appealing (i.e., all include at least one thumbnail image). Prior research found a click-through bias in favor of visually appealing verticals (e.g., *video*) [101]. It may be that this type of bias affected assessors more on presentation-pairs (i.e., where the vertical is embedded within less visually appealing results) than on block-pairs (where the vertical is shown in isolation). If accounting for such a bias is desired, then future work might consider incorporating more context into the block-pair assessment interface. For example, one possibility would be to show presentation-pairs where the only difference is that the vertical-pair is swapped, as in Figure 5.7. This block-pair assessment interface would allow the direct comparison between two blocks and would present the blocks in the context of other results.

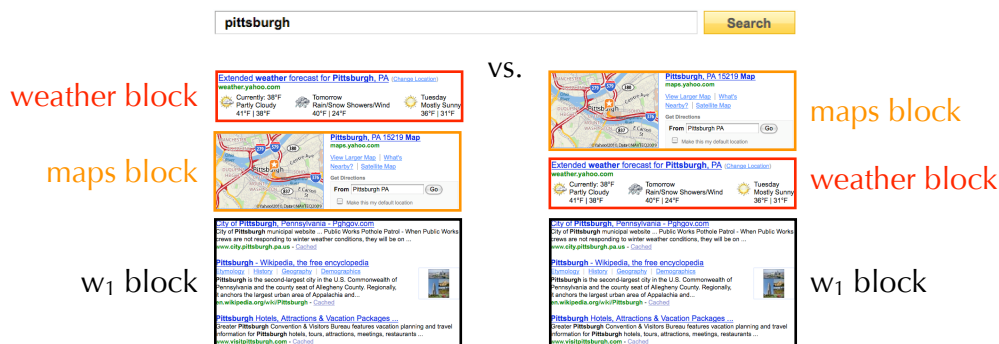


Figure 5.7: Context-aware block-pair assessment interface.

One important component of this work was the use of Amazon’s Mechanical Turk to obtain relevance judgements (pairwise preferences for block-pairs and presentation-pairs). As previously noted, we found that worker participation tends to follow a power-law: 20% of the assessors complete 80% of the assessments. At first, this may seem undesirable. We may prefer greater diversity of opinion. This trend, however, has a major benefit for doing quality control: 80% of the assessments are made by assessors who contribute many assessments, and thereby provide enough evidence to assess their reliability (for example, using traps). The difficulty, however, lies in determining the reliability of the 20% produced by assessors who contribute only a few assessments. Thus, one way to improve quality control might be to increase the barrier of entry for

assessors, for example, using qualification tests. Assessors should be given an incentive to provide either no assessments or many.

Our formulation of the presentation task as *block-ranking* assumes a one-dimensional presentation of Web and vertical results (e.g., Figure 5.1). This assumption is motivated by prior research which suggests that users interact with vertical results more when these are directly displayed within the Web results [100]. This is also the most common presentation strategy used by commercial search providers (e.g., Bing, Google, Yahoo!). While we assume a one-dimensional display, there are vertical presentation strategies that are *multi-dimensional*. Yahoo!, for example, sometimes presents multiple verticals horizontally using a tabbed display above the first Web result (e.g., Figure 5.8).

Our assumption of a one-dimensional display affects two different components in our evaluation methodology. First, it affects how we derive the reference presentation. The Schulze voting algorithm [85] is used to produce a ranking of blocks based on their number of pairwise defeats over other blocks. Under the assumption of a one-dimensional display, there is a direct correspondence between the reference *block-ranking* and the reference *presentation* (thus, we refer to both as simply the *reference*). Our assumption, which is common to most IR research, is that users scan results from top-to-bottom. Second, it affects our evaluation metric. We use a variant of Kendall's tau, which measures the distance between pairs of (one-dimensional) rankings.

Extending our evaluation methodology to handle multi-dimensional displays would require re-thinking both of these components. It may be possible, however, to continue modeling the reference presentation as a ranking of blocks and to continue using a rank-based metric to measure distance from the reference block-ranking. Extending the first component (i.e., deriving the reference presentation) would require learning an *indirect* correspondence between a reference block-ranking and how blocks should be arranged in a *multi-dimensional* display. For example, in Figure 5.8, it may be that users typically examine results first from left-to-right and then from top-to-bottom (this a simple scenario). If so, then this provides a recipe for arranging blocks given a reference block-ranking. Extending the second component (i.e., measuring distance from the reference block-ranking) would require learning the weights that users assign to different types of discordant pairs. These weights would have to be a function of the multi-dimensional template. For example, in Figure 5.8, it may be that horizontal discordant pairs are less important than vertical discordant pairs. Then, it might be possible to incorporate these weights into the generalized Kendall's tau distance metric [64].

5.8 Summary

In this chapter, our objective was a methodology for evaluating the end-to-end output of an aggregated search system. The general idea is to evaluate the quality of an arbitrary presentation σ_q by measuring its distance (using K^*) to a reference presentation σ_q^* . Our method for deriving σ_q^* focuses on preference judgements on block-pairs, where a block is a sequence of Web or same-vertical results that must appear grouped together (verti-

The image shows a search results page for "michelle obama". At the top, there is a search bar with the text "michelle obama" and a yellow "Search" button. Below the search bar, the page is divided into two main sections: "news" and "web".

The "news" section features a vertical purple bar on the left with the word "STORIES" written vertically. The main content includes a news article titled "And Michelle Obama's Park City Swag Bag Contained ..." from Park Record, dated Jul 29 02:46pm. The article includes a small image of a McDonald's meal and a brief summary: "Michelle Obama hauled in more than campaign donations for her husband when she visited the Park City area on Tuesday for a fundraiser. [Full Story >](#)". Below the article are "Related Stories" with links to other news items.

The "web" section features a vertical purple bar on the left with the word "web" written vertically. The main content includes a Wikipedia entry for "Michelle Obama" with a small portrait photo and a link to "en.wikipedia.org/wiki/Michelle_Obama - Cached". Below the Wikipedia entry are two links from "www.whitehouse.gov" regarding Michelle Obama's commitments.

On the right side of the page, there are three vertical bars: a red one labeled "IMAGES", a blue one labeled "VIDEOS", and an orange one labeled "TWITTER". Lines connect these labels to the corresponding vertical bars. Below these bars, the words "images", "video", and "twitter" are written in red, blue, and orange respectively.

Figure 5.8: Example of a multi-dimensional vertical display. Given the query “michelle obama”, Yahoo! presents *news*, *images*, *video*, and *twitter* within a horizontal tabbed display. Other verticals can appear embedded within the Web results below.

cally or horizontally) in the final ranking. The proposed methodology has the following advantages.

- It does not require a live system with (many) users.
- It is cost-effective. With fewer than about 400 judgements per query, we can evaluate the quality of *any* presentation of results. Furthermore, we show that block-pair judgements can be collected using a large pool of untrained and inexpensive assessors.
- It is portable. The methodology facilitates the construction of a portable test collection: a set of queries with relevance judgements (on block-pairs). The methodology is portable in the sense that these relevance judgements can be distributed across researchers and used to directly compare systems.
- It is general. We used a particular interface for assessing block-pairs, a particular voting method for deriving the *reference*, and a particular rank-similarity metric for measuring distance from the *reference*. Future work may consider others.

- It is grounded in assessor behavior. We empirically show that the metric correlates with user preferences, especially on presentation-pairs where one presentation was strongly preferred across multiple assessors.

Several open questions remain. Assessor agreement on presentation-pairs was low. Further experiments are needed to understand why. It may be, for example, that users assign a different cost to different types of errors (false positives, false negatives, ranking errors). Also, in some cases, assessors favored a particular vertical only when seen within the context of *other* results. There may be preferential biases that affect presentation-pair judgements more than block-pair judgements. Accounting for these presentation-level biases in the block-pair judgements may require re-thinking the block-pair assessment interface.

Vertical Results Presentation: Approaches

End-to-end aggregated web search requires not only deciding *which* verticals to are relevant (*vertical selection*), but also deciding *where* in the Web results to present them (*vertical results presentation*). If we assume that vertical results must be presented in specific positions relative to the Web results (i.e., above, below, and in between certain Web results), then the vertical results presentation task can be cast as *block-ranking*. A *block* is defined as a short sequence of Web or vertical results that must appear grouped together in the output presentation. In the previous chapter, we presented an evaluation methodology for block-ranking. Based on this methodology, the objective is to produce a block-ranking σ_q that approximates a “gold-standard” or *reference* ranking σ_q^* . A user study was conducted to empirically verify that when assessors prefer one block-ranking over another, the preferred block-ranking is closer (in terms of a rank-based distance metric) to σ_q^* . This methodology facilitates not only evaluation, but also model learning. A set of labeled queries (each with a known σ_q^*) can be used to train a model to produce output that approximates σ_q^* . In this Chapter, we develop supervised approaches to block-ranking.

Casting block-ranking as a supervised machine learning problem has two major challenges. First, different verticals focus on different types of media (e.g., news, images, videos) and serve different types of search goals (e.g., search for local businesses, items for sale, weather information). Therefore, results from different verticals are associated with different types of meta-data, which are likely to influence their relevance to a query. For example, *news* results are associated with a publication date, *local* results are associated with a geographical location, and *community Q&A* results are associated with a number of suggested answers. If we want to use these meta-data as features, then block-ranking requires approaches that can handle an inconsistent feature representation across blocks from different verticals. Secondly, even if a feature is common to multiple verticals, it may not be equally predictive. For example, the number of results retrieved by the vertical search engine may be predictive for *news*. However, this type of information is meaningless for a vertical that retrieves at most a *single* result, such as *weather*. Alternatively, a feature may be predictive for different verticals, but in the *opposite* direction. For example, the query-term “pics” may be positive evidence for *images*, but negative evidence for *shopping*. Likewise, a city name in the query may be positive evidence for *local*, but negative evidence for *finance*. Thus, block-ranking may require approaches that can exploit a *vertical-specific* predictive relationship between features and relevance. We evaluate methods that address both challenges in different ways.

Consistent with our work on vertical selection (Chapters 3 and 4), we evaluate approaches that rank blocks as a function of a set of features. To this end, we exploit various types of features, including, for example, properties of the query, vertical-specific click-through data, and the text-similarity between the query and the results presented in the block. We partition features into two general classes: *pre-retrieval* and *post-retrieval* features. During vertical results presentation, we assume that the system has already issued the query to each vertical, or at least to those predicted relevant during vertical selection. Thus, post-retrieval features can be derived directly from the vertical results. We investigate the cost-benefit of post-retrieval features. Are they useful for predicting *where* a vertical should be presented? Can they *also* be used to re-evaluate vertical selection decisions in light of the vertical results? Answering these questions affects, for example, whether the end-to-end system should issue the query to as many verticals as possible (or to those which post-retrieval features help the most), or whether it should cache post-retrieval features for future impressions of the query.

6.1 Formal Task Definition

The goal of block ranking is to produce a ranking of Web and vertical blocks in response to a query. A *block* is defined as a short sequence of Web or same-vertical results that must be grouped together—vertically (e.g., *blogs*, *news*) or horizontally (e.g., *images*, *video*)—in the output presentation.

Let \mathcal{B}_q denote the set of blocks associated with query q . Consistent with the set of layout constraints given in Section 5.2, set \mathcal{B}_q will include: all three Web blocks w_{1-3} , one block for each vertical that retrieves results, and the imaginary “end of search results” block, denoted by *eos* (explained below).

Let σ_q^* denote the gold-standard or *reference* block-ranking. σ_q^* is assumed to be the optimal ordering of \mathcal{B}_q for query q . Formally, the objective of block ranking is to predict a block-ranking σ_q that approximates σ_q^* . The quality of the approximation can be measured using a rank-based distance metric, such as Kendall’s tau. In our experiments, we measure the quality of σ_q using the *generalized* Kendall’s tau distance between σ_q and σ_q^* , denoted as $K^*(\sigma_q, \sigma_q^*)$. Generalized Kendall’s tau punishes ranking mistakes at the top of σ_q more severely than at the bottom [64].

The imaginary *eos* block is used to mark the end of the visible results. Let $\sigma_q(i)$ denote the rank of block i in σ_q . Blocks ranked above *eos* are considered to be displayed to the user and those ranked below *eos* are considered to be suppressed. Because the ranking of blocks below rank $\sigma_q(\textit{eos})$ is inconsequential to the user (i.e., it is not observed), for the purpose of comparing σ_q with σ_q^* , all blocks ranked below *eos* in σ_q are considered tied at rank $\sigma_q(\textit{eos}) + 1$.

Aggregated web search involves predicting which verticals are likely to be relevant (vertical selection) and predicting where to present them, or whether to suppress them, for example, in light of their results (vertical results presentation). In this chapter, we focus on vertical results presentation. In order to separate vertical selection performance

from our evaluation, we assume that the query is issued to every available vertical. That is, \mathcal{B}_q will include one vertical block for *every* vertical that retrieves results.

6.2 Related Work

Most previous research on vertical results presentation assumes at most a *single* relevant vertical and a fixed presentation template. As previously mentioned in Section 3.7, König *et al.* [63] and Diaz [28] focused on predicting whether to present news results above the Web results (also known as slot 0) or not at all. In this chapter, we consider potentially *multiple* verticals simultaneously and cast the problem as block-ranking rather than assume a pre-specified slot for the vertical results.

Ponnuswami *et al.* [77, 78] is the only published research that considers potentially multiple verticals at different positions. More specifically, this work considered three slotting positions: *top* (above the Web results), *middle* (between Web results 3-4), and *bottom* (below the last Web result). Their proposed framework combines vertical-specific binary classifiers as follows. During the training phase, each classifier is trained to predict whether its vertical should be presented in the *top* slotting position. Ponnuswami *et al.* [78] show that training data for such a binary classifier can be generated from implicit user feedback—by presenting random presentations to users and observing clicks and skips. Then, at test time, first an upstream vertical selection component predicts which verticals to present (in any of the three slotting position). Then, each classifier described above is used to predict whether its vertical (if selected) should be presented in the *top* slotting position. Finally, using two threshold parameters, each selected vertical is assigned to the *top*, *middle*, or *bottom* position based on its classifier’s prediction confidence value. As previously mentioned, one challenge in vertical results presentation is that different verticals may be associated with different types of predictive evidence. Ponnuswami *et al.* [78] addressed this challenge by learning a *different* binary classifier per vertical. Thus, each classifier can use a different set of features. The underlying assumption is that confidence values from these different binary classifiers are directly comparable. We make a similar assumption in our work on vertical selection (Chapter 3) and in this chapter evaluate a similar classification framework for vertical results presentation.

The work above casts vertical results presentation as a classification task. Additionally, in this chapter, we investigate whether it can also be cast as a learning-to-rank task. In machine learning, learning-to-rank (LTR) algorithms learn to order items as a function of a set of features. In document ranking, features may be derived from the query (e.g., whether the query contains a domain name [56]), the document (e.g., the URL’s PageRank [80]), and the query-document pair (e.g., the document’s BM25 score [80]). Existing LTR methods can be classified into three types. *Point-wise* methods (e.g., Gradient Boosted Decision Trees [38]) learn to predict a document’s relevance grade independent of other documents. *Pair-wise* methods (e.g., RankSVM [56]) learn to predict whether one document is more relevant than another. *List-wise* methods (e.g., AdaRank [114]) di-

rectly optimize an IR evaluation measure such as NDCG, which considers the quality of the ranking as a whole. In addition to document-ranking, LTR methods have been successfully applied to other tasks, such as ranking alternative language translations [31], ranking related news stories [71], and ranking 3D protein structures for a given amino acid sequence [32]. No prior work, however, has applied LTR methods to the task of ranking blocks of Web and vertical results.

6.3 Features

We propose machine learning approaches that rank blocks as a function of a set of features. To this end, we use various types of features which we believe are predictive of a particular block’s relevance to a query. These features can be divided into two general classes.

1. *Pre-retrieval features* can be generated *before* the query is issued to the Web or vertical search engine. These features include, for example, the topic of the query or whether the query contains a particular named-entity type (e.g., the name of a person, product, or location). Pre-retrieval features are independent of the block.
2. *Post-retrieval features* must be generated *after* the query is issued to the Web or vertical search engine. These features include, for example, the total number results retrieved by the block’s search engine or the average text-similarity between the query and the results presented in the block.

The goal of vertical selection (Chapters 3 and 4) is to predict which verticals (if any) are likely to have relevant content. Thus, vertical selection uses only pre-retrieval features. In vertical results presentation, however, our assumption is that query has already been issued to those verticals selected. Therefore, the system also has access to post-retrieval features.

We investigate the importance of post-retrieval features. Are they useful for block-ranking? If so, are they predictive for some verticals more than others? As previously stated, the vertical results presentation task includes deciding which selected verticals to present (rank above *eos* in σ_q) and which to suppress (rank below *eos* in σ_q). Are post-retrieval features useful for predicting which selected verticals to present/suppress in light of their results? Next, we motivate and describe each feature. A summary of our features is presented in Section 6.3.3.

6.3.1 Pre-retrieval Features

Pre-retrieval features can be generated before the query is issued to any search engine. Thus, pre-retrieval features are independent of the block under consideration.

Named-Entity Type Features

These binary features correspond to named-entity types possibly appearing in the query. Queries were automatically annotated using the BBN Identifinder named-entity tagger [8]. Named-entity features include *location* (possibly predictive for *local*, *maps*, and *weather*), *product* (possibly predictive for *shopping*), *person* (possibly predictive for *news* and *images*), and *organization* (possibly predictive for *finance*). In total, we focused on 24 named-entity types. Each binary feature equals 1 if the named-entity type appears at least once in the query and 0 otherwise.

Category Features

Some verticals may be topically focused. Thus, knowing the general topic of the query may help in block ranking. We focused on 30 topical categories, derived as follows. First, we selected 150 categories from the Open Directory Project (ODP) hierarchy and crawled Web documents associated with these ODP nodes. Then, in order to reduce the number of category features, these 150 categories were clustered into 30 clusters. Clustering was done using complete-link agglomerative clustering [72]. The distance between ODP categories was computed using the symmetric Kullback-Leibler divergence [53] between category language models. We used unigram language models with add-one smoothing to avoid zero probabilities. Let θ_i denote the language model associated with cluster/category i . We set the i^{th} category feature for query q according to,

$$\text{cat}_q(i) = \frac{1}{\mathcal{Z}} \prod_{w \in q} P(w|\theta_i),$$

where \mathcal{Z} normalizes across clusters/categories.

Click-through Features

User clicks are often viewed as surrogates for relevance. The queries associated with clicks on a particular document convey the types of information needs the document satisfies. Likewise, the queries associated with clicks on vertical results convey the types of information needs the vertical satisfies. Our click-through features harness this type of evidence by considering the similarity between the query and those associated with clicks on vertical content (or content very similar to the vertical's).

Click-through data was derived from the AOL query-log. We derived one click-through feature per vertical as follows. First, for each vertical, we manually selected a set of Web domains which we believe have content closely related to the vertical. The vertical-to-domain mapping is given in Table 6.1. Then, we constructed vertical-specific (unigram) language models using all queries (allowing duplicates) associated with click-events on the vertical's corresponding domains. Finally, given a query, we generate one feature per vertical based on the query generation probability given the vertical's query-based language model. Given query q , we set the click-through feature for vertical i

according to,

$$\text{click}_q(i) = \frac{1}{Z} \prod_{w \in q} P(w|\theta_i),$$

where Z normalizes across verticals.

Vertical-Intent Features

Users often express vertical-intent *explicitly* using keywords such as “news” (for the *news* vertical), “pics” (for the *images* vertical) or “buy” (for the *shopping* vertical). The goal of our explicit vertical-intent features is to determine vertical relevance based on how often the query co-occurs with keywords used in explicit requests for the vertical. For example, given the query “britney spears”, we may predict that *images* is relevant because users often issue the query “britney spears pics”. Co-occurrence statistics were derived from the AOL query-log.

Vertical-intent features were generated as follows. First, we manually associate each vertical with a small set of keywords which we believe are often used in explicit requests for the vertical. For example, the *images* vertical was associated with “picture(s)”, “photo(s)”, “pic(s)”, and “image(s)”. The complete vertical-to-keyword-set mapping is given in Table 6.2. Then, to measure the affinity between the query and a particular vertical, we use the chi-squared statistic to measure the lack of independence between two events: the occurrence of the query in the AOL query-log and the occurrence of any of the vertical’s vertical-intent keywords. To reduce sparsity (particularly for long queries), we compute the chi-squared statistic for each query-term individually and use the geometric average. The geometric average (rather than the arithmetic average) was used to favor queries with terms that *consistently* co-occur with keywords used in explicit requests for the vertical.

6.3.2 Post-retrieval Features

Post-retrieval features must be generated after the query is issued to the Web or vertical search engine. Because they are generated from the vertical (or Web) search engine results, different block-types are associated with a different set of post-retrieval features, though some features are common to multiple block-types.

Hit Count Features

This feature considers the number of results retrieved from the Web or vertical search engine. For some verticals, an abundance of query-related content in the index may be predictive of its relevance. This may be true, for example, for *news*, where the rate of content production may correlate with the rate of content demand. However, we do not expect this feature to be useful for every vertical. For example, the number of retrieved results contributes no useful information for verticals that retrieved at most a *single* result (*finance*, *maps*, and *weather*).

vertical	urls
blogs	blogger.com
	wordpress.org
	typepad.com
	qawker.com
	ibiblio.org
books	blogspot.com
	*books.com
community Q&A	answers.yahoo.com
	howstuffworks.com
finance	finance.yahoo.com
	hoovers.com
images	picasa.google.com
	flickr.com
local	local.yahoo.com
	yellowpages.com
	citysearch.com
maps	maps.google.com
	maps.yahoo.com
	mapquest.com
news	news.yahoo.com
	nytimes.com
	wsj.com
	cnr.com
	foxnews.com
	ap.org
recipes	cbsnews.com
	allrecipes.com
	cooks.com
	foodnetwork.com
shopping	*recipe.com
	craigslist.com
	target.com
	ebay.com
	walmart.com
	shopping.yahoo.com
	amazon.com
	sears.com
overstock.com	
video	*videos.com
	youtube.com
	video.google.com
	movies.yahoo.com
weather	weather.com
	weather.yahoo.com

Table 6.1: Vertical-to-URL mapping. URLs correspond to either the actual vertical (e.g., the *shopping* vertical was constructed using the eBay search API) or content related to the vertical (e.g., <http://www.cooks.com> contains recipes and is thus related to the *recipe* vertical). A * denotes a “wildcard” operation.

vertical	keyword set
blogs	blog(s)
books	book(s)
community Q&A	how, when, where, who, why, what
finance	stock(s)
images	picture(s), photo(s), pic(s), image(s)
local	city name
maps	map(s)
news	news
recipes	recipe(s)
shopping	shop, shopping, buy, sale(s), product(s), review(s), price(s)
video	video(s)
weather	weather, forecast

Table 6.2: Vertical-to-keyword mapping.

Temporal Features

Some verticals may be highly time sensitive. Prior work, for example, shows that recency is important in news search [28]. Temporal information was available for four verticals: *news*, *blogs*, *community Q&A*, and *twitter*. Our assumption is that, for these verticals, users care primarily about recent results. If this is true, then the average age of results presented in the block should affect its relevance. Temporal features are generated as follows. For each individual vertical result, we measure the elapsed time (in hours) between the current and created date/time. We included four features for each time-sensitive vertical: the minimum, maximum, mean, and standard deviation of the elapsed time across results within the block.

Text-Similarity Features

The goal of these features is to characterize the text-similarity between the query and the results presented in the block. The challenge in deriving text-similarity features is that results from different sources (i.e., results from the Web search engine and from different verticals) are associated with *different* sets of textual representations. For example, each Web result is associated with three representations: a title, URL, and summary snippet. Each *community Q&A* result is associated with two representations: a question and an optional “best answer”. Each *weather* result is associated with a single representation: the location, which we define as the concatenation of the city, state, and country of the weather forecast. The complete set of representations associated with each block-type is given in Table 6.3.

Text-similarity features are generated for a query-block pair in two steps. In the first step, for each result within the block, we measure the text similarity between the query and each representation associated with the result. We use four different text-similarity

measures: (1) the cosine similarity between the query and the representation, (2) the maximum number of query-terms appearing consecutively in the representation, (3) the percentage of query-terms appearing in the representation, and (4) the percentage of the representation-terms corresponding to a query-term. Similar text-similarity features were used in prior learning-to-rank research (for document ranking) [56, 70].¹

The second step depends on whether the block-type is associated with a single result per block (e.g., *weather*, *finance*, and *maps*) or multiple results per block (e.g., *news*, *local*, and *shopping*). For block-types with a single result, we simply include our four similarity measures for each of its text representations. For block-types with multiple results, for each query-representation similarity measure, we use the minimum, maximum, mean, and the standard deviation across results within the block.

block-type	representations
blogs	url, title
books	title, author(s)
community Q&A	question, best answer, subject
finance	company
images	url, title
local	url, title, location
maps	location
news	url, title, summary snippet, news source
recipes	url, title
shopping	url, title
twitter	tweet
video	title
weather	location
web	url, title, summary snippet

Table 6.3: Block-types along with their textual representations.

6.3.3 Summary of Features

Table 6.4 provides a summary of our features and the block-types for which each feature group is available. Pre-retrieval features are independent of the block. Thus, every block-type is associated with the same set of 80 pre-retrieval features: 24 named-entity type features, 30 category features, 13 click-through features, and 13 vertical intent features. Notice that we added all 13 click-through features (one per vertical) and all 13 vertical intent features (one per vertical) to every block’s feature representation, irrespective of

¹In previous chapters, we use KL-divergence (KLD) to measure text-similarity. Here, we use the cosine distance for the following reason. KLD requires smoothing both language models to avoid zero probabilities. Both queries and our representations are terse (most terms occur once or zero times). Therefore, smoothing would significantly change their language models. The cosine similarity does not require smoothing. The query and the representation can be completely disjoint.

its type. Click-through features were not generated for Web blocks w_{1-3} . A language model derived from queries with clicks on Web content would simply look like a background language model. Likewise, vertical-intent features were not generated for Web blocks w_{1-3} . Because Web results are always presented by default, users typically do not explicitly request Web content using query keywords.

Post-retrieval features, as opposed to pre-retrieval features, are derived from the Web or vertical search engine results or directly from those results presented in the block. Therefore, different types of blocks are associated with a different set of post-retrieval features. Some post-retrieval features are available for multiple block-types (e.g., query-summary text-similarity features are available for *news* and w_{1-3}). Others, however, are available for a single type of block (e.g., query-tweet similarity features are available only for *twitter*). Table 6.4 presents the number of features that each group contributes for each block-type and the total feature count.

6.4 Block-Ranking Approaches

As previously mentioned, casting block-ranking as a supervised machine learning problem is associated with two main challenges. First, different types of blocks are associated with different features. Second, even when a feature is common to multiple block-types, it may have a *type-specific* relationship with relevance. We propose three general approaches which address both challenges in different ways.

6.4.1 Classification Approach

Our classification framework takes the form of n independent binary classifiers (one per vertical). We choose to use logistic regression due to its prediction accuracy and training speed on large-scale classification tasks [68].

Each binary classifier is trained to predict whether a particular vertical should be presented (ranked above *eos*) or suppressed (ranked below *eos*). While training the classifier for vertical v , a query is considered a positive instance if v is ranked above *eos* in the *reference* ranking σ_q^* and a negative instance otherwise. To compensate for class imbalance (verticals are more often suppressed), each positive training instance is weighted according to the number of negative instances in the training set and vice-versa.

The classification approach produces a block-ranking by assigning vertical blocks to slots. Consistent with our layout constraints, we assume four vertical slotting positions relative to the Web results: slot s_1 (above w_1), slot s_2 (between w_1 and w_2), slot s_3 (between w_2 and w_3), and slot s_4 (between w_3 and *eos*). In the output ranking, a slot may contain zero or more vertical blocks.

The classification approach combines all n vertical-specific binary classifiers as follows. First, the query, represented as a vector of features, is submitted to each candidate vertical’s classifier. Each classifier outputs a prediction probability that its vertical should be presented (i.e., ranked above *eos* in the predicted ranking σ_q). Let $P(\sigma_q(v) < \sigma_q(eos))$

	pre-retrieval					post-retrieval										total
	hit count	temporal	url	title	summary	location	question	subject	answer	company	author	tweet				
blogs	80	1	4	16	16								117			
books	80	1		16	16						16	16	129			
community Q&A	80	1	4				16	16	16				133			
finance	80									4			84			
images	80	1		16	16								113			
local	80	1		16	16	16							129			
maps	80					4							84			
news	80	1	4	16	16	16							133			
recipes	80	1		16	16								113			
shopping	80	1		16	16								113			
twitter	80	1	4										85			
video	80	1		16	16								113			
weather	80					4							84			
w_1	80	1		16	16	16							129			
w_2	80	1		16	16	16							129			
w_3	80	1		16	16	16							129			

Table 6.4: Summary of Features. Pre-retrieval features are independent of the block under consideration. Thus, every block-type is associated with the same set of 80 pre-retrieval features. Post-retrieval features, on the other hand, are derived from the Web or vertical search engine or directly from those results presented in the block. Therefore, different types of blocks are associated with a different set of post-retrieval features. Some features are common to multiple block-types, and others are unique to a particular type. Text-similarity features are marked in gray.

denote the prediction probability that v should be presented. Then, as shown in Figure 6.1, each candidate vertical is assigned to a slot (or is suppressed) using four threshold parameters τ_{1-4} . Vertical v is assigned to slot x if $P(\sigma_q(v) < \sigma_q(eos)) \geq \tau_y \forall x \leq y$. In other words, vertical v is assigned to the highest slot for which v 's prediction probability is greater than or equal to all thresholds below it. At this point, vertical blocks assigned to the same slot are tied. Finally, ties are broken by ordering vertical blocks within the same slot by descending order of prediction probability. This approach focuses on assigning verticals to slots. It might not, however, do a good job in ordering vertical blocks within a slot.

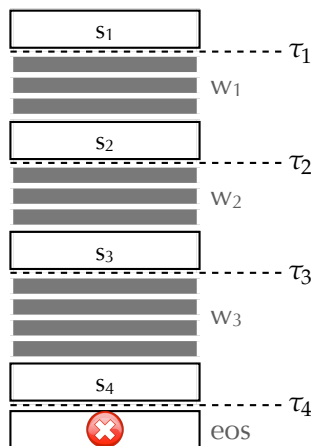


Figure 6.1: Vertical slotting positions. The classification approach predicts an output presentation by assigning verticals to slots s_{1-4} using threshold parameters τ_{1-4} . Ties (i.e., verticals assigned to the same slot) are broken using the prediction probability.

The classification approach addresses the two challenges mentioned above by training a different binary classifier per vertical. Each classifier adopts its own feature representation, which is unique to the vertical, and learns a vertical-specific relationship between features and block relevance.

6.4.2 Voting Approach

In the classification approach, each independent binary classifier is trained to predict whether a particular vertical should be presented or suppressed. Our voting approach also combines independent binary classifiers. However, these are trained to make more fine-grained predictions. Independent binary classifiers are trained to predict the relative ordering between pairs of blocks of a particular type. Each classifier is trained to predict whether block i (of a particular type) should be ranked above or below block j (of another particular type) for a given query.

The voting approach uses one binary classifier per block-type pair. Given n verticals

and m non-vertical block-types, the total number of classifiers is given by $\binom{n+m}{2} - \binom{m}{2}$. The second term accounts for the fact that Web results are always presented and always ranked in their original order (i.e., $\sigma_q(w_1) < \sigma_q(w_2) < \sigma_q(w_3) < \sigma_q(eos)$). Thus, it is not required to learn a classifier to determine the relative ordering between pairs of non-vertical blocks. In our case, $n = 13$ and $m = 4$, which results in 130 independent binary classifiers. The large number of binary classifiers used by this method may be viewed as a disadvantage. Later, we describe one way to reduce this number.

The training phase proceeds as follows. Recall that \mathcal{B}_q denotes the set of block associated with query q and includes one block per candidate vertical, all three Web blocks (w_{1-3}), and the *eos* block. While training a classifier specific to a block-type pair, the query is considered a positive or negative instance depending on the pair’s relative rank. Certain block-type pairs occur more frequently in one order versus the other. Therefore, to correct for class imbalance, each positive training instance is weighted according to the number of negative instances in the training set and vice-versa.

To predict a block-ranking for query q , first, every block-pair $i, j \in \mathcal{B}_q$ is submitted to the appropriate classifier, depending on the type of i and the type of j . Let $P(\sigma_q(i) < \sigma_q(j))$ denote the prediction probability that i should be ranked above j . This probability can be treated as a preference score between i and j . The voting approach produces the output block-ranking σ_q by combining these preference scores as input to the Schulze voting method [85].

As in the classification approach, each binary classifier is associated with a unique feature representation. More specifically, each classifier is associated with three sets of features: one set of pre-retrieval features, which are independent of the block-types under consideration, and two separate sets of post-retrieval features (one set specific to each type in the block-type-pair).

The voting approach addresses both challenges mentioned above (block-type-specific features and a potentially different predictive relationship across types) by training a different binary classifier per block-type pair. Each classifier adopts its own feature representation and learns a predictive relationship that is specific to the block-type-pair.

One limitation of this configuration is the large number of independent binary classifiers. An alternative to learning one model per block-type pair is to learn one model per vertical/non-vertical block-type pair. This results in four models per vertical: three which predict the vertical’s relevance compared to each Web block (w_{1-3}) and one which predicts its relevance compared to *eos* (i.e., it predicts whether to display/suppress the vertical). As before, given a query, every block-pair $i, j \in \mathcal{B}_q$, where one is a vertical- and the other a non-vertical block, is submitted to the appropriate classifier. Then, the output prediction probabilities are combined as the input to the Schulze voting method in order to derive σ_q .

6.4.3 Learning to Rank Approaches

Block-ranking can also be cast as a learning-to-rank (LTR) problem. Many different LTR methods have been proposed. In this work, we adopt a *pairwise* learning to rank

approach. Pairwise approaches optimize over the set of pairwise preferences implicit in the training data. More specifically, we adopt a method that solves the classic RankSVM optimization problem, first proposed by Joachims [56].

RankSVM learns a linear model $f_{\mathbf{w}}$ parameterized by feature weight vector \mathbf{w} . Let $\phi(i, q)$ denote a feature generating function that outputs a feature vector for the block i and query q . At test time, given q , the predicted score for block i is given by $f_{\mathbf{w}}(i, q) = \langle \mathbf{w}, \phi(i, q) \rangle$. Finally, the output ranking σ_q is inferred by sorted score.

In contrast with the previous two approaches, casting block-ranking as an LTR problem requires training a *single* model $f_{\mathbf{w}}$ to predict a block’s rank irrespective of its type. In our situation, this is problematic because different block-types are associated with different features and those that are common to *multiple* block-types may be predictive for some types more than others, or predictive in the opposite direction. Next, we propose three LTR variants which address these challenges in different ways. Each variant makes a different assumption about how features may be correlated with block relevance across different block-types.

Equally Correlated Features

One alternative is to assume that each feature is equally predictive of block relevance (in the same direction) independent of the block-type. The feature representation is as follows. Pre-retrieval features are independent of the block. This model uses a *single* copy of each pre-retrieval feature. Post-retrieval features are block-specific (i.e., they are generated directly from the block or the block’s search engine results). Similar to pre-retrieval features, this approach *also* uses a *single* copy of each post-retrieval feature. If a block is not associated with a particular post-retrieval feature, then the feature is zeroed-out in that instance. Consider, for example, our post-retrieval features which determine the text-similarity between the query and the summary snippets presented in the block. These features are only associated with *news* and Web blocks w_{1-3} . Therefore, if the block is not one of these types, all these features are zeroed-out.

This approach assumes that features are equally correlated with relevance irrespective of the block-type. Once trained, model $f_{\mathbf{w}}$ will apply the *same* weight to a feature independent of the instance’s block-type. We denote this LTR variant as LTR-G because it assumes a vertical-*general* relationship between features and relevance.

Uniquely Correlated Features

This approach makes the opposite assumption as the previous one. It assumes that every feature—whether it is a pre- or post-retrieval feature—is uniquely correlated with relevance across different block-types. The feature representation is as follows. We make a separate, block-type-specific copy of each feature. So, for example, given 17 block-types (13 verticals + 3 Web blocks and the *eos* block), we make 17 copies of each pre-retrieval feature (one per block-type). Given an instance, all copies are zeroed-out except for those corresponding to the instance’s block-type. For post-retrieval features, we make one copy

per block-type for which the feature is available. Consider, for example, our temporal features, which are available for blocks from *blogs*, *community Q&A*, *news*, and *twitter*. We make 4 copies of each temporal feature.

This approach assumes that features are correlated differently with relevance depending on the block-type. Once trained, model f_w will apply a *different* weight to a feature, depending on the instance’s block-type. While this added flexibility may be advantageous, the increased number of features may introduce predictive noise and result in overfitting. Thus, this LTR variant may require more training data than LTR-G. We denote this LTR variant as LTR-S because it assumes a *vertical-specific* relationship between features and relevance.

Equally and Uniquely Correlated Features

The previous two approaches make opposite assumptions: features are either equally correlated or uniquely correlated with relevance for different block-types. A third alternative is to make neither assumption *a priori*, but to give the algorithm the freedom to exploit both types of relationships using training data.

For this approach, we maintain a single copy of each pre- and post-retrieval feature which is shared across all block-types. As before, if an instance’s block-type is not associated with a shared feature, the feature is zeroed-out for that instance. In addition to these shared features, we make one block-type-specific copy of each pre- and post-retrieval feature. Given an instance, all copies corresponding to types other than the instance’s block-type are zeroed-out. The canonical feature representation for this approach is the union of features used by the previous two approaches.

This approach makes no assumption about how a feature is correlated with relevance across block-types. If a feature is equally correlated across block-types, then the algorithm has the flexibility of exploiting this relationship by assigning a large (positive or negative) weight to the version of the feature which is shared across types. Alternatively, if a feature is correlated differently for different block-types, then the algorithm can exploit a type-specific relationship by assigning a large positive weight to some copies of the feature and a large negative weight to others. We denote this LTR variant as LTR-GS because it has the flexibility of learning a vertical-specific and vertical-general relationship for every feature. Of all three LTR variants, LTR-GS is associated with the largest number of features and may therefore need to most training data to avoid overfitting.

6.5 Methods and Materials

We train algorithms to predict a block-ranking σ_q that approximates the *reference* block-ranking σ_q^* and compare methods based on the quality of their approximation on previously unseen queries. More specifically, we evaluate each predicted block-ranking σ_q based on its generalized Kendall’s tau distance to the *reference* σ_q^* .

Evaluation was conducted using 10-fold cross-validation. Unless otherwise stated,

statistical significance is tested using a one-tailed paired t-test (at the $p < 0.05$ level) by comparing performance across test queries (i.e., the concatenation of all 10 test-folds). We evaluate block-ranking performance using a set of 13 verticals and 1070 queries.

6.5.1 Verticals and Queries

We focused on the same set of 13 verticals used in Chapter 5, which were constructed using freely available APIs from eBay (*shopping*), Google (*blogs, books, weather*), Recipe Puppy (*recipes*), Yahoo! (*community Q&A, finance, images, local, maps, news*), Twitter (*micro-blogs*), and YouTube (*video*).

A set of queries for model learning and evaluation was collected using sampling. Different situations call for different methods for query sampling. In our case, we are interested not only in performance across queries, but also across verticals. Queries were sampled from two different sources: the AOL query-log and Google Trends.² The AOL query-log contains about 20M queries issued to the AOL search engine over a period of three months in 2006. Being from 2006, AOL queries may not cover current topics for which verticals like *news* and *twitter* are relevant. For this reason, we also sampled queries from Google Trends. Google Trends provides the 20 most popular queries issued to the Google Web portal in the last hour (after removing spam queries) and also provides daily aggregates, which are available for dates in the past.

A total of 1,070 queries were sampled in two phases. During the first phase, 500 queries were sampled uniformly at random from the AOL query log (rejecting duplicates). As it turns out, after collecting assessments for these queries, only 43 had at least one vertical displayed (i.e., ranked above *eos*) in σ_q^* . In other words, for these 500 queries, assessors had a strong preference towards Web results. Thus, during the second phase, the sampling was biased in favor of queries with at least one relevant vertical (and potentially multiple).

Our biased sampling approach proceeds as follows. Let $P(v|q)$ denote the relevance of vertical v to query q and let \mathcal{Q} denote the set of all queries. To ensure coverage across all verticals, first, a vertical v is selected in a round-robin fashion. Then, a query is sampled from \mathcal{Q} according to,

$$\frac{P(v|q)}{\sum_{q \in \mathcal{Q}} P(v|q)}$$

We now describe how to we estimate $P(v|q)$ without using human judgements. In order to guide users towards more successful searches, commercial search provides often suggest queries that are related to the current search. In some cases, the set of recommended queries provides evidence that a particular vertical is relevant to the query. For example, as shown in Figure 6.2, given the query “pittsburgh”, Bing recommends the query “pittsburgh photos”. This recommended query suggests that users who issue the query “pittsburgh” often want to see image results.

²<http://www.google.com/trends>

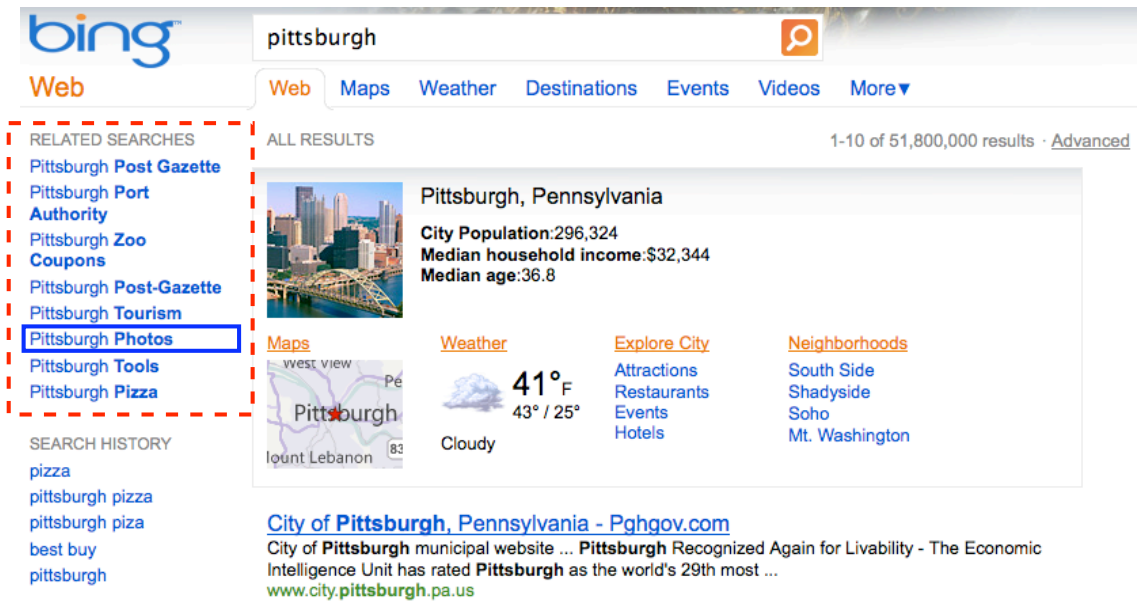


Figure 6.2: Given the query “pittsburgh”, a search engine recommends a set of related queries (red, dashed box). The related query “pittsburgh photos” (blue, solid box) suggests that users who issue the query “pittsburgh” often want to see photos (i.e., it suggests that the *images* vertical is relevant).

Each vertical was associated with a set of keywords believed to be used in explicit requests for the vertical. Let \mathcal{T}_v denote the set of keywords associated with v and let \mathcal{Q}_q^s denote the set of queries suggested by Bing in response to q . $P(v|q)$ was approximated according to,

$$P(v|q) \approx \frac{1}{|\mathcal{Q}_q^s|} \sum_{q' \in \mathcal{Q}_q^s} \mathcal{I}(|q' \cap \mathcal{T}_v| > 0)$$

where $q' \cap \mathcal{T}_v$ denotes the intersection between terms in q' and \mathcal{T}_v . Queries were sampled from a large set of queries: 20K queries sampled uniformly from the AOL query-log, all Google Trends daily aggregates from a year preceding the second round of assessments, and all Google Trends queries (scraped hourly) from 3 days preceding the second round of assessments.

6.5.2 Block-Pair Preference Assessment

As described in Section 5.4, given q , the reference presentation σ_q^* is derived from human preference judgements on block-pairs. More specifically, it is derived from preference judgements on *all* block pairs $i, j \in \mathcal{B}_q$. These preference judgements, denoted by π_q , are the raw input to the Schulze voting method, which produces σ_q^* . A block-pair judgment is defined by a triplet (q, i, j) .

As in Chapter 5, block-pair judgments were collected using Amazon’s Mechanical Turk (AMT). Assessors were compensated with US\$ 0.01 per block-pair. The block-pair assessment interface was similar to the one used in our user study, with three main differences. First, rather than include a single block-pair assessment per HIT (Human Intelligence Task) we included five block-pair assessments per HIT and priced HITs at US\$ 0.05.³ Second, assessors were not given a topic description for the query. Instead, they were instructed to interpret results based on their best guess of the hypothetical user’s intent. We purposely omitted topic descriptions in order to model query ambiguity and diversity in the aggregated results. That is, we want query ambiguity to be reflected in the vertical’s presented in σ_q^* . We expected, however, that omitting the topic description would reduce inter-assessor agreement, particularly for queries without a obvious relevant vertical. Third, in order to annotate more queries, we collected 3 redundant judgements per triple (q, i, j) instead of 4. Let $\pi_q(i, j)$ denote the strength with which block i is preferred over j given q . We set $\pi(i, j)$ equal to the number of assessors who prefer i over j given q . $\pi(eos, i)$ was set equal to the number of assessors who stated that i is bad in conjunction with another block j .

To do quality control, 10% of all block-pair assessments were traps. In a trap assessment, the assessor is given a triplet (q, i, j) , but either block i or j is taken from a query other than q . If the assessor selects the extraneous block as the preferred block, we consider it a *failed* trap. Assessors who failed more than 2 traps had all their judgements removed from the pool and resubmitted to AMT.

Block-Pair Assessment Results

Because we collected redundant block-pair judgements, we can measure inter-assessor agreement. Inter-assessor agreement in terms of Fleiss’ Kappa was $\kappa_f = 0.524$, which is considered *moderate* agreement based on Landis and Koch [65]. We interpret this level of agreement high enough to suggest that assessors did not find the assessment task or the interface confusing. Agreement on this round of assessments, however, was lower than in Chapter 5.4 ($\kappa_f = 0.656$). There may be two reasons for this. First, queries were sampled automatically, whereas in Chapter 5 they were selected manually. It may be that these manually selected queries had a more obvious relevant vertical, which resulted in higher agreement. On this round of assessments, agreement on the 500 queries that were sampled uniformly from the AOL query log was $\kappa_f = 0.502$. Agreement on the 570 queries that were sampled using biased sampling was $\kappa_f = 0.546$. Thus, agreement is higher when the query is biased towards at least one vertical. Secondly, assessors were not given topic descriptions, and, therefore, may have adopted different interpretations for the same query.

³In addition to compensating assessors for their work, using AMT requires paying Amazon a 10% commission, with a minimum commission fee of \$ 0.001. Thus, pricing HITs at US\$ 0.01 results in paying Amazon a 50% rather than a 10% commission per HIT.

6.5.3 Modeling Bias Towards Vertical Results

In certain scenarios, one may prefer a system that is biased towards vertical results. In other words, the aggregated web search provider may want to tune the system so that it more frequently presents vertical results and/or presents them higher in the output presentation (i.e., ranks them higher in σ_q). This may occur, for example, if the search company has contractual obligations with vertical providers or advertisers (prior research shows that such contractual obligations occur [78]).

Vertical bias can be modeled in our evaluation framework using vertical *pseudo-votes*, a parameter p in the range $[0, \infty]$. As previously noted, $\pi_q(i, j)$ corresponds to the number of assessors who prefer i over j given q . A vertical bias can be introduced by incrementing this value by some number of pseudo-votes p , but only if i corresponds to a vertical block. If i and j correspond to blocks from different verticals, this method increments both $\pi_q(i, j)$ and $\pi_q(j, i)$ by an equal number of pseudo-votes. This has the effect that, after producing σ_q^* using the Schulze voting method, vertical results are ranked higher in σ_q^* . However, the ranking of verticals relative to each other is unchanged. Modeling vertical bias using vertical pseudo-votes changes σ_q^* and, therefore, changes model learning and evaluation. Rather than select a single pseudo-vote parameter p , we learn and evaluate models across several values of p (i.e., $p = \{0, 1, 2, 3, 4, 5\}$). Table 6.5 presents the number of queries for which a type of block was presented within the top 3 ranks of σ_q^* for different values of p . Across values of p , the verticals ranked higher in σ_q^* were *video*, *local*, *news*, *blogs*, and *community Q&A*. This is consistent with prior work that found a bias in favor of *video* [101].

Given no vertical bias ($p = 0$), only 190/1070 (18%) queries had at least one vertical presented in σ_q^* (i.e., ranked above *eos*). This number may seem low. Bing, for example, presents at least one vertical for 652/1070 (61%) of our queries. There are several possible explanations for this difference. Commercial search providers often use implicit feedback (i.e., clicks and skips) as features or targets for prediction [28, 78]. Gathering this feedback data requires presenting a vertical more often than it is relevant. Additionally, as explained above, commercial search providers may have strong incentives to show verticals often. These may include contractual obligations with advertisers (who advertise in the vertical) and alleviating traffic from the Web search engine. Finally, presenting verticals is a type of diversification in light of query ambiguity. We might have observed more verticals presented if we would have collected a greater number of redundant preference judgements per block-pair.

6.5.4 Evaluation Metric

Given query q , the objective in block-ranking is to produce a ranking of blocks σ_q that approximates the reference ranking σ_q^* . Thus, our primary evaluation metric is the Generalized Kendall's tau distance between the predicted block-ranking σ_q and the reference block-ranking σ_q^* , denoted as $K^*(\sigma_q^*, \sigma_q)$ [64].

Generalized Kendall's tau is closely related to Kendall's tau (K), which measures the

block-type	pseudo-votes p					
	0	1	2	3	4	5
blogs	5	13	38	80	139	193
book	0	1	3	6	12	23
community Q&A	4	10	26	52	80	115
finance	5	8	11	14	16	20
image	5	10	19	39	61	83
local	10	17	24	32	46	60
map	1	3	5	8	12	15
news	8	26	53	86	105	123
recipe	1	3	7	16	22	32
shopping	4	8	18	33	40	66
twitter	0	0	0	2	5	12
video	21	44	100	208	286	360
weather	6	11	12	13	14	14
w_1	1070	1070	1066	1045	1013	947
w_2	1067	1059	1027	940	852	749
w_3	1003	927	801	636	507	398
any vertical	67	143	269	434	563	672

Table 6.5: Number of queries (out of 1070) in which each block-type was presented within the top 3 ranks of σ_q^* as a function of pseudo-votes.

number of discordant pairs between two rankings of the same set of elements,

$$K(\sigma^*, \sigma) = \sum_{\sigma^*(i) < \sigma^*(j)} [\sigma(i) > \sigma(j)],$$

where $\sigma(i)$ denotes the rank of element i in σ . For our purposes (i.e., to measure the distance between σ_q and σ_q^*), Kendall's tau has a major limitation: it considers *all* discordant pairs equally important. However, in aggregated web search users typically scan results from top to bottom. Thus, discordant pairs at the top of the ranking should be more influential. Generalized Kendall's tau (K^*) accounts for positional information using element weights. Let δ denote the cost of an *adjacent* swap. In traditional Kendall's tau, $\delta = 1$, irrespective of rank. Adjacent swaps are treated equally regardless of rank. Let δ_r denote the cost of an adjacent swap between elements at rank $r - 1$ and r . As suggested by Kumar and Vassilvitskii [64], adjacent swaps at the top of the ranking can be made more influential by using the following DCG-like cost function,

$$\delta_r = \frac{1}{\log(r)} + \frac{1}{\log(r+1)}$$

defined for $2 \leq r \leq n$. Given rankings σ^* and σ , element i 's displacement weight $\bar{p}_i(\sigma^*, \sigma)$ is given by the average cost (in terms of adjacent swaps) it incurs in moving

from rank $\sigma_q^*(i)$ to rank $\sigma_q(i)$,

$$\bar{p}_i(\sigma^*, \sigma) = \begin{cases} 1 & \text{if } \sigma^*(i) = \sigma(i) \\ \frac{p_{\sigma^*(i)} - p_{\sigma(i)}}{\sigma(i)^* - \sigma(i)} & \text{otherwise} \end{cases},$$

where $p_r = \sum_2^r \delta_r$. The K^* distance is then given by,

$$K^*(\sigma^*, \sigma) = \sum_{\sigma^*(i) < \sigma^*(j)} \bar{p}_i(\sigma^*, \sigma) \bar{p}_j(\sigma^*, \sigma) [\sigma(i) > \sigma(j)].$$

A discordant pair's contribution to the metric is equal to the product of the two element weights.

In Chapter 5, we present a user study that shows that when assessors strongly prefer one block-ranking over another, the metric judges the preferred ranking as superior.

In its current form, K^* has two limitations: lower is better, which is uncommon for evaluation metrics, and its range is query-dependent—its highest value is a function of the number of blocks associated with the query. Since we are interested in average performance across queries, the metric's range should be query-independent. Both limitations can be addressed by scaling K^* to the range $[-1, +1]$ according to,

$$K_{\text{scaled}}^*(\sigma^*, \sigma) = \frac{K_{-}^*(\sigma^*, \sigma) - K^*(\sigma^*, \sigma)}{K_{-}^*(\sigma^*, \sigma) + K^*(\sigma^*, \sigma)}$$

where,

$$K_{-}^*(\sigma^*, \sigma) = \sum_{\sigma^*(i) < \sigma^*(j)} \bar{p}_i(\sigma^*, \sigma) \bar{p}_j(\sigma^*, \sigma) [\sigma(i) < \sigma(j)].$$

This normalization is analogous to the normalized form of regular Kendall's tau,

$$K(\sigma^*, \sigma) = \frac{C - D}{C + D},$$

where C is the number of concordant pairs and D is the number of discordant pairs and $C + D$ is the total number of pairs. From hereon, we refer to K_{scaled}^* simply as K^* .

6.5.5 Implementation Details

Classification and Voting Approaches

The classification framework requires tuning threshold parameters τ_{1-4} , which are used to slot verticals based on each classifier's prediction confidence value. In addition to these parameters, logistic regression requires tuning cost factor C , which determines the cost of misclassifying a training set instance. Instead of using a single held-out validation set, these 5 parameters were tuned using 10-fold cross-validation. More specifically, they were tuned for each train/test pair individually by doing a *second* level of 10-fold cross-validation on each primary fold's training set. An exhaustive search was used to find the parameter values with the best average performance across secondary train/test

pairs. We did not tune parameter C for each vertical individually as doing so would have resulted in an intractable number of parameter settings. The voting approach also uses logistic regression models: one per block-type-pair. The C parameter was tuned as described above (i.e., not individually for each block-type pair). In both approaches, models were trained using the LibLinear toolkit.⁴

Learning-to-Rank Approaches

As shown in Table 6.5, most queries have Web blocks w_{1-3} ranked high and verticals ranked low (or suppressed). This is a problem if during training the LTR model learns that the best alternative is to present w_{1-3} and suppress all verticals. In the classification and voting approach, we balanced the training data using instance weighting. That is, more weight was allocated to training instances from the minority class. Casting block-ranking as a learning-to-rank problem may also require some form of instance weighting.

Our approach to instance weighting is as follows. Let σ_q^w denote a block-ranking which presents Web blocks w_{1-3} in their original order and suppresses all verticals. Given a training query q , $-K^*(\sigma_q^*, \sigma_q^w)$, which is in the range $[-1, +1]$, measures the distance between σ_q^* and σ_q^w . If $-K^*(\sigma_q^*, \sigma_q^w)$ is close to $+1$, it means that σ_q^* has verticals presented in the top ranks. If $-K^*(\sigma_q^*, \sigma_q^w)$ is close to -1 , it means that σ_q^* has Web blocks w_{1-3} presented in the top ranks and verticals presented in the low ranks or suppressed. Our approach to instance weighting is to replicate queries in the training set proportional to this distance. This has the effect that queries which deviate from σ_q^w are oversampled. The amount of replication is controlled using parameter α . First, we scale $-K^*(\sigma_q^*, \sigma_q^w)$ to zero min and unit max (using queries from the training set). Then, we multiply this value by α . Given training query q , the number of *additional* copies of q in the training set is given by $-\alpha K_{\min\text{-max}}^*(\sigma_q^*, \sigma_q^w)$. Note that when $\alpha = 0$, no additional copies of q are added to the training set. We evaluate the effect of parameter α in Section 6.7.1.

In addition to parameter α , RankSVM has a regularization parameter λ . Both parameters were tuned using two levels of 10-fold cross-validation. An exhaustive search was used to find the parameter values with the best average performance across secondary train/test pairs. We trained LTR models using the sofia-ml toolkit.⁵

6.6 Experimental Results

We evaluate three general approaches to block-ranking: the classification approach, the voting approach, and the LTR approach. One limitation of the voting approach is that it requires a large number of independent binary classifiers. To address this limitation, we propose a second version of the voting approach that trains a binary classifier only for block-type-pairs where one type corresponds to a vertical block-type and the other corresponds to a non-vertical block-type. Both are included in this evaluation. We also

⁴<http://www.csie.ntu.edu.tw/~cjlin/liblinear/>

⁵<http://code.google.com/p/sofia-ml/>

include all three variants of the LTR approach, with parameter α tuned using cross-validation.

Results in terms of average $K^*(\sigma_q^*, \sigma_q)$ are presented in Table 6.10. As mentioned, pseudo-vote parameter p controls for vertical bias. The higher its value, the greater the bias towards verticals ranked high. Rather than choose a single parameter p , results are presented for $p = \{0, 1, 2, 3, 4, 5\}$. When $p = 0$, the σ_q^* 's used for training and evaluation have no vertical bias. The second row in Table 6.10 (WEB) corresponds to a degenerate baseline that suppresses all vertical results and simply presents w_{1-3} above *eos* in their original order.

	$p = 0$	$p = 1$	$p = 2$	$p = 3$	$p = 4$	$p = 5$
WEB	0.982 Δ	0.959 $\Delta\nabla$	0.896 $\nabla\nabla$	0.773 $\nabla\nabla\nabla$	0.656 $\nabla\nabla\nabla$	0.520 $\nabla\nabla\nabla$
classification $\Delta\nabla$	0.950 $\nabla\nabla$	0.943 $\nabla\nabla$	0.924 \blacktriangle	0.878 \blacktriangle	0.824 $\blacktriangle\wedge$	0.775 $\blacktriangle\wedge$
voting (all pairs) $\blacktriangle\nabla$	0.984 Δ	0.960 $\Delta\nabla$	0.898 $\nabla\nabla$	0.830 $\nabla\nabla$	0.744 $\nabla\nabla$	0.734 $\nabla\nabla$
voting (only v-w pairs)	0.980 $\Delta\nabla$	0.956 $\Delta\nabla\nabla$	0.898 $\nabla\nabla$	0.822 $\nabla\nabla$	0.760 $\nabla\blacktriangle\nabla$	0.734 ∇
LTR-G $\wedge\nabla$	0.980 Δ	0.967 $\Delta\blacktriangle$	0.932 \blacktriangle	0.871 \blacktriangle	0.798 $\nabla\blacktriangle$	0.754 $\nabla\blacktriangle$
LTR-S	0.986 $\Delta\wedge$	0.970 $\Delta\blacktriangle$	0.937 $\Delta\blacktriangle$	0.883 $\blacktriangle\wedge$	0.838 $\Delta\blacktriangle\wedge$	0.792 $\Delta\blacktriangle\wedge$
LTR-GS	0.986 $\Delta\wedge$	0.973 $\Delta\blacktriangle\wedge$	0.936 $\Delta\blacktriangle$	0.882 $\blacktriangle\wedge$	0.843 $\Delta\blacktriangle\wedge$	0.795 $\Delta\blacktriangle\wedge$

Table 6.6: Block-ranking results in terms of average $K^*(\sigma_q^*, \sigma_q)$. Statistical significance was tested using a one-tailed paired t-test, comparing performance across pairs of queries. A $\Delta(\nabla)$ denotes significantly better(worse) performance compared to the classification approach, a $\blacktriangle(\blacktriangledown)$ denotes significantly better(worse) performance compared to the voting (all pairs) approach, and a $\wedge(\vee)$ denotes significantly better(worse) performance compared to the LTR-G approach.

Table 6.10 shows several meaningful trends. Performance for all methods decreases with greater values of p (this was expected for WEB). One possible reason is the following. As p increases, the number of queries with top-ranked verticals increases. This means that the room for error also increases. When p is low, ranking w_{1-3} above all verticals is a reliable and effective strategy.

When $p = 0$, WEB performs well. This is not surprising given that 94% of all queries (1003/1070) had Web blocks w_{1-3} in the top-three ranks of σ_q^* (see Table 6.5). In fact, the only two approaches that significantly outperform WEB when $p = 0$ are LTR-S and LTR-GS. Significance with respect to WEB is not shown in Table 6.10. For values of $p \geq 3$, *all* block-ranking approaches significantly outperform WEB. Thus, given a vertical bias, any of the three general block-ranking approaches is better than presenting only Web results.

Both voting approaches perform similar to each other. This is interesting because the second version (which learns one binary classifier per vertical/non-vertical block-type) uses many fewer binary classifiers than the first (which learns one classifier per block-type). The performance for both voting approaches, however, deteriorates for $p \geq 2$.

The classification approach does surprisingly well considering that it uses a just one binary classifier per vertical. Its performance is competitive across all values of p , show-

ing that it can be effectively trained to fit different degrees of vertical bias.

Comparing among the three LTR variants, performance is significantly worse for LTR-G. It never performs better and often performs significantly worse than both LTR-S and LTR-GS, particularly for values of $p \geq 3$. For values of $p \geq 4$, LTR-G performs significantly worse than the classification approach. The improvement of LTR-S and LTR-GS over LTR-G reveals the importance of exploiting vertical-specific evidence. Imposing the constraint that features must be similarly correlated with relevance across different block-types degrades performance. LTR-G and LTR-S perform better by giving the learning algorithm the flexibility to exploit a block-type-specific relationship between features and block relevance.

Compared to the classification approach, LTR-S and LTR-GS both perform better. The improvement is significant for all values of p , except $p = 3$, where all three perform at the same level. We view this as a positive result in favor of casting block-ranking as a learning-to-rank task. In contrast with the classification approach, both of these LTR variants perform better and require training only a single model (rather than one per vertical). Relative to each other, LTR-S and LTR-GS are statistically indistinguishable across all values of p (statistical significance not shown in Table 6.10). We provide an explanation for this in Section 6.7.3.

6.7 Discussion

Several questions remain. Our results show that casting block-ranking as a learning-to-rank problem is a good idea. In addition to allowing the model to learn a vertical-*specific* relationship between features and relevance, we argue that instance weighting during LTR training is also important. In Section 6.7.1, we test the importance of parameter α on LTR performance.

Our approach is to rank blocks as a function of a set of features. Different features have a different cost. For example, post-retrieval features require issuing the query to the vertical search engine. Thus, we want to know each feature group’s contribution to overall performance (Section 6.7.2) as well as performance for each vertical (Section 6.7.3).

Making a block-type-specific copy of each feature (as in the LTR-S and LTR-GS approaches) is one way to allow an LTR model to learn a block-type specific relationship between features and relevance. In Section 6.7.4, we evaluate a second alternative, which is to include features that identify the block-type and use a learning algorithm that can exploit feature interactions.

In Section 5.6, we found that our metric’s ability to predict the preferred block-ranking is the highest (in the 80-90% range) when one block-ranking is close to σ_q^* (i.e., in the \mathcal{H} bin) and the other is far from σ_q^* (i.e., in the \mathcal{M} or \mathcal{L} bin). Taking a highly conservative view of the metric, we might consider two alternative presentations *indistinguishable* to users if they fall under bin-combinations other than $\mathcal{H}\text{-}\mathcal{M}$ and $\mathcal{H}\text{-}\mathcal{L}$. In Section 6.7.5, we compare the voting, classification, and LTR methods under this conservative assumption.

6.7.1 Effect of Parameter α on LTR variants

The goal of parameter α is to focus LTR training on those queries that have verticals ranked high. Given enough evidence in favor of a particular vertical, we want to the LTR model to overcome the prior probability that the vertical is ranked low. Our method for instance weighting is to replicate queries in the training set proportional to the K^* distance between the reference block-ranking σ_q^* and one that presents w_{1-3} and suppresses all verticals. Parameter α controls the amount of replication.

We are interested in the effect of parameter α on performance. Each LTR variant is evaluated under two conditions. In the first condition, $\alpha = 0$, which corresponds to *no replication*—each training query q appears just once in the training set. In the second condition, α is tuned by sweeping across $\alpha = \{0, 10, 25, 50\}$. Notice that when α is tuned, $\alpha = 0$ (no replication) is *also* a parameter choice. The results with α tuned are identical to those in Table 6.10.

Table 6.7 presents results (in terms of average $K(\sigma_q^*, \sigma_q)$) for all three LTR variants under these two conditions. As a second point of reference, we also present results for the classification approach.

	$p = 0$	$p = 1$	$p = 2$	$p = 3$	$p = 4$	$p = 5$
classification	0.950	0.943	0.924	0.878	0.824	0.775
LTR-G, $\alpha = 0$	0.983 Δ	0.967 Δ	0.923	0.839 ∇	0.775 ∇	0.701 ∇
LTR-S, $\alpha = 0$	0.986 Δ	0.966 Δ	0.928	0.863 ∇	0.798 ∇	0.723 ∇
LTR-GS, $\alpha = 0$	0.986 Δ	0.967 Δ	0.940 Δ	0.870	0.812	0.743 ∇
LTR-G, α tuned	0.980 $\nabla\Delta$	0.967 Δ	0.932 \blacktriangle	0.871 \blacktriangle	0.798 $\blacktriangle\nabla$	0.754 $\blacktriangle\nabla$
LTR-S, α tuned	0.986 Δ	0.970 Δ	0.937 $\blacktriangle\Delta$	0.883 \blacktriangle	0.838 $\blacktriangle\Delta$	0.792 $\blacktriangle\Delta$
LTR-GS, α tuned	0.986 Δ	0.973 $\blacktriangle\Delta$	0.936 Δ	0.882 \blacktriangle	0.843 $\blacktriangle\Delta$	0.795 $\blacktriangle\Delta$

Table 6.7: Block-ranking results in terms of $K(\sigma_q^*, \sigma_q)$. A \blacktriangle (\blacktriangledown) denotes a significant improvement(drop) in performance for an LTR variant with α tuned compared to its counterpart for which α is forced to zero. A Δ (∇) denotes significantly better(worse) performance compared to the classification approach.

Before comparing LTR variants across both conditions, it should be noted that the relative performance between LTR variants when $\alpha = 0$ (rows 3-5) is consistent with their relative performance when α is tuned (rows 6-8). That is, LTR-S and LTR-GS outperform LTR-G and their improvement increases with p . Due to their superior performance, we focus the discussion on LTR-S and LTR-GS. For both approaches, tuning α (vs. setting it to zero) has either no effect or significantly improves performance. For $p \geq 4$, setting $\alpha = 0$ degrades performance even compared to the classification approach. Thus, casting block-ranking as a learning-to-rank task requires not only providing the learning algorithm the flexibility to exploit a block-type-specific predictive relationship between features and relevance, it also requires re-weighting training-phase instances in order to overcome the prior probability that verticals are ranked low.

6.7.2 Feature Contribution to Overall Performance

	$p = 0$	$p = 1$	$p = 2$	$p = 3$	$p = 4$	$p = 5$
all feats.	0.950	0.943	0.924	0.878	0.824	0.775
no ne type	0.952 (0.29%)	0.946 (0.23%)	0.920 (-0.47%)	0.877 (-0.17%)	0.826 (0.14%)	0.778 (0.49%)
no category	0.953 (0.38%)	0.944 (0.02%)	0.916 (-0.83%)▼	0.889 (1.21%)▲	0.827 (0.28%)	0.775 (0.03%)
no click	0.946 (-0.42%)	0.940 (-0.40%)	0.908 (-1.71%)▼	0.863 (-1.69%)▼	0.808 (-1.97%)▼	0.759 (-2.05%)▼
no vert. int.	0.949 (-0.06%)	0.939 (-0.44%)	0.914 (-1.15%)▼	0.879 (0.12%)	0.824 (0.00%)	0.768 (-0.87%)
no hit count	0.952 (0.28%)	0.946 (0.22%)	0.920 (-0.51%)	0.875 (-0.41%)	0.826 (0.24%)	0.785 (1.32%)
no location	0.949 (-0.06%)	0.940 (-0.33%)	0.921 (-0.31%)	0.877 (-0.16%)	0.826 (0.23%)	0.775 (0.08%)
no temporal	0.948 (-0.19%)	0.950 (0.66%)▲	0.924 (0.02%)	0.880 (0.22%)	0.822 (-0.24%)	0.774 (-0.02%)
no text-sim.	0.947 (-0.28%)	0.945 (0.17%)	0.915 (-0.94%)▼	0.868 (-1.14%)▼	0.805 (-2.34%)▼	0.744 (-3.97%)▼
no post-ret.	0.949 (-0.10%)	0.942 (-0.19%)	0.916 (-0.86%)▼	0.864 (-1.62%)▼	0.805 (-2.29%)▼	0.748 (-3.44%)▼

(a) classification approach

	$p = 0$	$p = 1$	$p = 2$	$p = 3$	$p = 4$	$p = 5$
all feats.	0.986	0.970	0.937	0.883	0.838	0.792
no ne type	0.986 (-0.02%)	0.970 (0.05%)	0.938 (0.19%)	0.888 (0.57%)	0.835 (-0.42%)	0.802 (1.28%)
no category	0.985 (-0.10%)	0.970 (-0.03%)	0.940 (0.31%)	0.879 (-0.46%)	0.835 (-0.35%)	0.794 (0.21%)
no click	0.985 (-0.04%)	0.970 (0.00%)	0.939 (0.26%)	0.877 (-0.67%)	0.836 (-0.20%)	0.791 (-0.13%)
no vert. int.	0.986 (-0.03%)	0.973 (0.33%)▲	0.938 (0.12%)	0.886 (0.31%)	0.839 (0.16%)	0.797 (0.60%)
no hit count	0.984 (-0.15%)	0.970 (-0.01%)	0.934 (-0.26%)	0.887 (0.41%)	0.834 (-0.45%)	0.790 (-0.23%)
no location	0.986 (-0.01%)	0.971 (0.13%)	0.938 (0.12%)	0.885 (0.15%)	0.833 (-0.65%)	0.805 (1.61%)▲
no temporal	0.986 (0.01%)	0.970 (0.04%)	0.937 (0.05%)	0.884 (0.13%)	0.837 (-0.10%)	0.797 (0.65%)
no text-sim.	0.985 (-0.09%)	0.970 (0.04%)	0.933 (-0.41%)	0.874 (-1.04%)	0.816 (-2.60%)▼	0.765 (-3.44%)▼
no post-ret.	0.986 (-0.02%)	0.968 (-0.18%)	0.937 (0.08%)	0.863 (-2.29%)▼	0.813 (-2.95%)▼	0.762 (-3.79%)▼

(b) LTR-S

Table 6.8: Feature ablation results for the classification approach and the LTR-S approach. A ▲(▼) denotes a significant improvement(drop) in performance compared to the model with access to all features.

A feature ablation study was conducted to test each feature group’s contribution to overall performance, measured in terms of average $K(\sigma_q^*, \sigma_q)$. The analysis is conducted for both the classification and the LTR-S approach because these were the two general approaches that performed the best in Section 6.6. Each feature group was individually omitted and this model was compared to a model with access to all features. Because features may be correlated, a non-significant drop in performance does not necessarily mean that the feature group contributes no useful information.

Results are presented in Table 6.8 for the classification approach and the LTR-S approach. The top four feature groups are pre-retrieval features. The next four features are post-retrieval features. The last row corresponds to a model that ignores *all* post-retrieval features.

Table 6.8 shows several interesting results. The LTR-S approach appears to be slightly more robust to missing features. One possible reason is the following. The classification approach trains one independent binary classifier per vertical. The LTR-S approach, on the other hand, trains a single model. It may be that training a single model allows

the LTR-S approach to better shift its focus towards block-types for which it is more confident.

The features with the greatest drop in performance (at least for $p \geq 4$) are text-similarity features, which are a type of post-retrieval feature. This shows the importance of deriving evidence directly from those results presented in the block. Also, it suggests the importance of issuing the query to as many vertical search engines as possible (in order to derive this type of evidence) or caching these post-retrieval features for future impressions of the query. Text-similarity features may have contributed the most to performance because many of the block-types often ranked high in σ_q^* were associated with text-rich information. These included Web blocks w_{1-3} , *news*, *blogs*, and *community Q&A*.

Finally, omitting most feature groups did not result in a significant drop in performance. There may be two reasons for this. First, features may be correlated. Second, most verticals are rarely ranked high in σ_q^* . It may be that some features are highly predictive for these minority verticals, but this may not have a noticeable effect on the average $K^*(\sigma_q^*, \sigma_q)$. We explore this further in the next section by using a different metric to measure per-vertical performance.

6.7.3 Feature Contribution to Per-Vertical Performance

Evaluating a feature’s contribution to a particular vertical is not trivial. Suppose, for example, that we omit temporal features and want to evaluate the effect on the *news* vertical. One possibility might be to compare the *news* vertical’s predicted rank and its ideal rank across a set of queries. However, omitting temporal features may affect *other* verticals as well. And, if so, then ranking mistakes for those verticals would displace *news* from its ideal rank.

For this reason, we focus our analysis on the classification approach, which trains one binary classifier per vertical. Each vertical-specific classifier is trained to predict whether the vertical should be displayed (ranked above *eos*) or suppressed (ranked below *eos*) in σ_q . Performance for a particular vertical is evaluated by considering the quality of confidence values produced by the vertical’s corresponding classifier. Let \mathcal{Q}_v denote the queries for which v is a candidate vertical. We evaluate the quality of confidence values output from v ’s classifier by computing the average precision (AP) metric on a ranking of \mathcal{Q}_v (ranked in descending order of confidence value that v should be presented). In computing AP, a ranked query q is considered “relevant” if $\sigma_q^*(v) < \sigma_q^*(eos)$ and “non-relevant” otherwise. Results are presented in Table 6.9. We limit our discussion to the case where $p = 5$ because it is associated with verticals ranked higher in σ_q^* .

Table 6.9 shows several noteworthy trends. First, performance across verticals varied widely (see row “all features”). The best-performing verticals were *weather* (AP=0.938), *finance* (AP=0.638), and *news* (AP=0.594). Interestingly, both *weather* and *finance* were minority verticals. Both were presented (i.e., ranked above *eos* in σ_q^*) for only 14 and 21 queries, respectively. In spite of having few positive instances for training, performance for both these verticals was good. As it turns out, *weather* was easy because *every* query

	community	Q&A	blogs	book	finance	image	local	map	news	recipe	shopping	twitter	video	weather
all features	0.284		0.431	0.088	0.638	0.323	0.546	0.419	0.594	0.48	0.377	0.053	0.571	0.938
no ne type	10.15%▲		2.65%▲	34.41%▲	-18.69%▼	9.19%▲	-3.97%▼	-1.99%	13.04%▲	6.97%▲	1.35%	-14.19%▼	2.14%▲	4.55%▲
no category	15.49%▲		-6.61%▼	7.77%	21.95%▲	6.61%▲	-8.67%▼	-0.36%	21.92%▲	18.10%▲	-32.93%▼	-0.91%	3.35%▲	-17.31%▼
no click-through	2.32%▲		0.32%	-1.29%	-14.35%▼	-9.23%▼	-10.44%▼	-16.86%▼	3.27%▲	-0.01%	-8.88%▼	1.99%	-4.48%▼	-50.11%▼
no vertical intent	7.80%▲		2.60%▲	1.97%	0.00%	-19.34%▼	7.37%▲	-74.42%▼	2.73%	-3.63%▼	-0.65%	0.31%	4.41%▲	-1.11%▼
no hit count	5.53%▲		-0.61%▼	3.65%▲	2.44%▲	0.81%	-2.89%▼	-1.94%▼	-4.21%▼	3.78%▲	0.21%	-5.90%▼	-0.75%▼	-0.45%▼
no location	0.00%		0.00%	0.00%	0.00%	0.00%	-8.84%▼	13.12%▲	0.00%	0.00%	0.00%	0.00%	0.00%	-17.68%▼
no temporal	0.12%		-0.43%▼	0.00%	0.00%	0.00%	0.00%	0.00%	0.51%	0.00%	0.00%	-5.71%▼	0.00%	0.00%
no text-similarity	-8.44%▼		-12.35%▼	-31.16%▼	-3.38%▼	2.47%	-16.93%▼	0.00%	-48.88%▼	0.00%	-8.02%▼	-26.78%▼	-12.11%▼	0.00%
no post-retrieval	-6.03%▼		-14.20%▼	-31.07%▼	-3.71%▼	2.20%	-18.04%▼	13.15%▲	-45.26%▼	3.78%▲	-10.96%▼	-27.64%▼	-12.46%▼	-17.81%▼

Table 6.9: Feature contribution to per-vertical performance, based on A.P. Statistical significance is tested using a one-tailed paired t-test, comparing across cross-validation test-folds. A ▲ (▼) denotes a significant improvement(drop) in performance compared to the model with access to all features.

for which it was presented had the query term “weather” (e.g., “weather louisville ky”). This explains why the most predictive feature for *weather* was the click-through feature (i.e., the query’s similarity to those with clicks on weather-related content, many which contain “weather”). Similarly, *finance* was easy because 10/21 queries for which it was presented had the query term “stock(s)” (e.g., “boeing stock”). In other words, performance for *weather* and *finance* was high because their queries often had explicit vertical intent, which is easy to detect.

It is interesting that *news* was among the best performing verticals. This result is inconsistent with our vertical selection results in Chapter 3 (see Table 3.5). The most useful features for *news* were text-similarity features, which are a type of post-retrieval feature. Thus, while it is difficult to detect that a query is newsworthy using only pre-retrieval evidence (as in Chapter 3), useful evidence can be harnessed by issuing the query to the *news* vertical.

The worst performing verticals were *twitter* (AP=0.053), *books* (AP=0.088), and *community Q&A* (AP=0.284). Both *twitter* and *books* were minority verticals, while *community Q&A* was fairly common. Predicting *twitter* may require predicting that the query is about a trending topic, which we did not explicitly address. Many queries for which *twitter* was presented were sampled from Google Trends, however, we purposely did not use this information as evidence. Predicting *books* and *community Q&A* seems difficult. Queries for which these verticals were relevant had no clear pattern. For *books*, for example, some queries had the keyword “book” (e.g., “books on giraffes”), some corresponded to a book title (e.g., “lolita”), some corresponded to an author name (e.g., “dr. phil”), and, finally, others corresponded to encyclopedic information needs (e.g., “wedding cake ideas”, “why don’t babies sleep at night”, “perennials”). *Community Q&A* queries showed a similar pattern. This may explain why text-similarity features were the only ones to significantly improve performance for both. In the absence of a clearly predictive query-pattern, it seems useful to derive evidence directly from the block.

Features contributed to performance differently for different verticals. For example, vertical intent features, which exploit vertical-related keywords, were predictive for *maps*. Many queries for which *maps* was relevant had the keyword “map(s)”. Similarly, vertical-intent features were predictive for *images* because many queries for which *images* was relevant had the keywords “photo(s)”, “pic(s)”, and “picture(s)”. Category features were predictive for *shopping* because one of our category clusters was related to the shopping ODP node.

Different features also hurt performance for different verticals. For example, named-entity type features hurt performance for *books* and *news*. Category features hurt performance for *community Q&A*, *finance*, *news*, and *recipe*. Click-through features hurt performance for *community Q&A* and *news*.

The only features that did not significantly hurt performance for *any* vertical were text-similarity features. In the previous section, text-similarity features had the greatest contribution to overall performance. In this analysis, text-similarity features were the most consistently predictive for different verticals.

To summarize, this analysis reveals several important results. First, as might be expected, some verticals are more difficult than others. This is consistent with results presented in Chapters 3 and 4. Second, different features were predictive for different verticals. This may suggest why LTR-S and LTR-GS perform at the same level. Because most features help some verticals and hurt others, there is little to be gained from learning a vertical-*general* relationship between features and relevance. Finally, text-similarity features did not hurt performance for *any* vertical and improved performance for several. For some verticals, the improvement from including text-similarity features was substantial (an 86% improvement for *news*).⁶ Therefore, in practice, it may be worth determining which verticals gain the most from text-similarity evidence. Then, the vertical selection component can be tuned so that those verticals are selected more often.

6.7.4 LTR Model with Cross-Product Features

The success of LTR-S and LTR-GS over LTR-G reveals that an LTR approach to block-ranking requires allowing the model the flexibility to exploit a block-type-specific relationship between features and block relevance. With a linear ranker (i.e., RankSVM), one solution is to make a block-type-specific copy of each feature. A second potential alternative is to use an LTR model that can exploit feature interactions that somehow convey block-type information. In this section, we investigate the performance of a RankSVM model with a full cross-product feature representation. We denote this LTR variant as LTR-C.

The input feature representation for LTR-C is identical to the one used by LTR-G. Internally, however, LTR-C learns on all conjunctions of two features. As with all our LTR variants, the set of features input to LTR-C include 17 binary features which serve to identify the block-type (recall that we have 17 block-types: 13 verticals, 3 Web blocks, and the *eos* block). In this respect, the set of features used internally in LTR-C can be viewed a superset of those used by LTR-S—the feature representation adopted by LTR-S corresponds to the set of two-feature conjunctions that include one of these 17 block-type-identifying features.

One might expect LTR-C to perform well because it has access to the same feature representation as LTR-S as well as *other* types of feature interactions. On the other hand, however, the (internal) feature representation is the largest of all LTR variants, which may lead to over-fitting (considering our small amount of training data).

Table 6.10 shows performance for the LTR-C model (in terms of average K^* -distance to the reference) compared to LTR-G and LTR-S. LTR-C performs essentially at the same level as LTR-G (it performs significantly better for $p = 4$). LTR-C performs significantly worse than LTR-S. There may be two reasons for this. First, it may be that the only two-feature conjunctions modeled by LTR-C that are useful are those that are modeled by LTR-S (the rest may be simply introducing noise). Second, it may be necessary to model interactions

⁶Table 6.9 shows a 45.26% decrease in performance from omitting text-similarity features. This corresponds to a 85.63% improvement from including them.

more complex than conjunctions. Gradient Boosted Decision Trees [38], used in Chapter 4, may be one alternative to model more complex interactions. Further experiments are needed to determine why LTR-C does not perform better.

	$p = 0$	$p = 1$	$p = 2$	$p = 3$	$p = 4$	$p = 5$
LTR-S	0.986	0.970	0.937	0.883	0.838	0.792
LTR-G	0.980	0.967	0.932	0.871	0.798	0.754
LTR-C	0.981 [∇]	0.966 [∇]	0.938	0.869 [∇]	0.814 ^{∇▲}	0.760 [∇]

Table 6.10: Block-ranking performance in terms of average $K^*(\sigma_q^*, \sigma_q)$. Statistical significance was tested using a one-tailed paired t-test, comparing performance across pairs of queries. A $\Delta(\nabla)$ denotes significantly better(worse) for LTR-C vs. LTR-S. A $\blacktriangle(\blacktriangledown)$ denotes significantly better(worse) performance for LTR-C vs. LTR-G.

6.7.5 Binning Analysis

Our main evaluation metric is the K^* -distance between the predicted block-ranking σ_q and the reference block-ranking σ_q^* . In Chapter 5, we tested this metric’s agreement with user preferences on pairs of alternative block-rankings. We found that the metric’s agreement with the assessors’ majority preference is high (in the 80-90% range) when one block-ranking is close to σ_q^* (i.e., in the \mathcal{H} bin) and the other is far from σ_q^* (i.e., in the \mathcal{M} or \mathcal{L} bin). Conversely, the metric’s agreement is low (in the 50-65% range) when both block-rankings are close to σ_q^* or both are far from σ_q^* (see Table 5.2). As might be expected, inter-assessor agreement followed a similar trend (see Table 5.1). One might interpret these results by concluding that presentation pairs which fall under bin-combinations other than $\mathcal{H}\text{-}\mathcal{M}$ and $\mathcal{H}\text{-}\mathcal{L}$ are indistinguishable to users—they are either both equally good or both bad enough that their difference does not matter. In this section, we compare approaches under this assumption.

To conduct this analysis, we take the following approach. We binarize the metric value (i.e., K^* -distance) to equal ‘1’ if the predicted block-ranking σ_q is in the \mathcal{H} bin and ‘0’ otherwise. Note that if we average this binary variable across queries, it corresponds to the average number of queries for which the approach predicts a block-ranking in the \mathcal{H} bin. To compare approaches against each other, we use a one-tailed *paired* t-test (paired on queries) on this binary metric. The *null* hypothesis is that both approaches predict block-rankings that are indistinguishable (i.e., both block-rankings correspond to a ‘1’ or both correspond to a ‘0’). Given a query, one approach will outperform another only if it predicts a block-ranking in the \mathcal{H} bin (i.e., outputs a ‘1’) and the other predicts one in the \mathcal{M} or \mathcal{L} bin (i.e., outputs a ‘0’).

Our results are given in Table 6.11 for the voting approach, the classification approach, and the LTR-G and LTR-S LTR variants. We present results for different values of pseudo-votes p . The table should be read as follows. Column ‘avg’ gives the average number of queries for which the approach predicts a block-ranking in the \mathcal{H} bin. Subsequent columns give the significance test results. Cell c_{ij} corresponds to the p -value

associated with the test that the approach in row i outperforms that in column j . A ‘-’ is inserted if i does not outperform j .

Tables 6.11 present several noteworthy results. In general, the trends are the same as in Section 6.6. All approaches perform worse as p increases. As we argued previously, this is probably because as p increases there is more room for error (i.e., simply presenting w_{1-3} in the top ranks becomes less effective). LTR-S performs either at the same level or better than the voting and classification approaches. Furthermore, it performs equal to or better than LTR-G, which reaffirms that casting block-ranking as a learning-to-rank task requires learning a type-specific relationship between features and block relevance.

One difference in this analysis is that LTR-S and LTR-G perform at the same level when $p = 0$ (no vertical bias). In Section 6.6 LTR-S significantly outperforms LTR-G. When $p = 0$, both approaches are equal if we consider only those block-rankings that are likely to be noticeably better. This result, combined with the fact that LTR-S and LTR-G diverge with greater values of p (observed here and in Section 6.6), shows that LTR-S’s advantage over LTR-G comes from being able to rank verticals more effectively.

In summary, our most robust solution, LTR-S, performs relatively well when $p \leq 3$. Performance for LTR-S (and all other approaches) drops as we increase the bias in favor of vertical results (e.g., when $p = 5$). There are several possible reasons for this. It may be that more training data is necessary to learn how to rank verticals more effectively. It may also be that when $p = 5$, the noise-to-signal ratio is too high. As we introduce a greater bias, we deviate more from the choices made by our assessors. Future extensions of this work should consider these possibilities. Additionally, the proposed approaches may be improved by considering different types of evidence or by exploiting predictive relationships between different verticals.

6.8 Summary

In this chapter, we proposed and evaluated three general machine learning approaches to block-ranking—ordering blocks of Web and vertical results in response to a query. The block-ranking task is associated with two main challenges. First, blocks from different verticals have a unique feature representation. Second, similar features may be correlated with relevance differently for different verticals. Our three proposed approaches address these challenges in different ways.

The best overall performance was obtained by casting block-ranking as a learning-to-rank (LTR) problem. However, our results show that, to be successful, the LTR model requires two things. First, the feature representation must allow for the model to learn a vertical-*specific* predictive relationship between features and block relevance. That is, even if a feature is shared among block’s from different verticals, the feature representation must allow the model to weight the feature as positive evidence for some verticals and negative evidence for others. Second, during training, the model must be biased in favor of training queries with verticals ranked high, which are the minority. We show that ignoring either of these requirements significantly hurts performance.

		$p = 0$			
	avg	vote	class	LTR-G	LTR-S
vote	0.972	-	0.000	-	-
class	0.868	-	-	-	-
LTR-G	0.974	0.353	0.000	-	-
LTR-S	0.979	0.072	0.000	0.083	-

		$p = 1$			
	avg	vote	class	LTR-G	LTR-S
vote	0.928	-	0.000	-	-
class	0.864	-	-	-	-
LTR-G	0.936	0.153	0.000	-	-
LTR-S	0.950	0.001	0.000	0.004	-

		$p = 2$			
	avg	vote	class	LTR-G	LTR-S
vote	0.778	-	-	-	-
class	0.803	0.032	-	-	-
LTR-G	0.862	0.000	0.000	-	-
LTR-S	0.876	0.000	0.000	0.029	-

		$p = 3$			
	avg	vote	class	LTR-G	LTR-S
vote	0.646	-	-	-	-
class	0.693	0.002	-	-	-
LTR-G	0.732	0.000	0.001	-	-
LTR-S	0.764	0.000	0.000	0.002	-

		$p = 4$			
	avg	vote	class	LTR-G	LTR-S
vote	0.507	-	-	-	-
class	0.586	0.000	-	-	-
LTR-G	0.599	0.000	0.182	-	-
LTR-S	0.676	0.000	0.000	0.000	-

		$p = 5$			
	avg	vote	class	LTR-G	LTR-S
vote	0.472	-	-	-	-
class	0.505	0.027	-	-	-
LTR-G	0.512	0.020	0.318	-	-
LTR-S	0.567	0.000	0.000	0.000	-

Table 6.11: Binning evaluation results.

Consistent with the central theme of the dissertation, various types of evidence were integrated as input features. A set of feature ablation studies showed that different features improve performance for different verticals, which is another result in favor of evidence-integration. The features that contributed the most to overall performance (and those that were *consistently* useful across different verticals) were text-similarity features, derived from the similarity between the query and the results presented in the block. These are a type of post-retrieval feature. Thus, they can only be generated for those verticals predicted relevant during vertical selection. Their contribution to performance suggests the importance of issuing the query to a vertical when possible (or at least to those which benefit the most) or caching this type of evidence for future impressions of the query or those similar to it.

The relevance of a particular vertical (and therefore its target rank) may be a function of *other* verticals presented. For example, suppose that a user issues the query “nikon cool pix” and wants to know what the camera looks like. Both the *images* and the *video* vertical may be relevant. However, presenting both of them in the top ranks may be redundant. This is a phenomenon we did not investigate in this chapter. Addressing this issue would first require a change in our block-pair assessment interface. Block-pair judgements were collected outside the context of other vertical results (we raised this issue in Section 5.7). Thus, novelty/diversity may not be reflected in our derived reference presentations (used for training and testing).

Suppose, however, that we have training data that reflects (positive/negative) correlations between different verticals. There are several possibilities, which future research might consider. One is to filter redundancies during vertical selection using a two-stage classification framework where the predicted relevance of each vertical is used as a feature in each vertical-specific classifier. A second alternative is to identify query-intent classes at a higher level than vertical-relevance and to associate each class with a particular set of verticals. A third alternative is to address redundancy as a post-process to block-ranking. A simple solution, for example, would be modify the predicted block-ranking using Maximal Marginal Relevance [17], where vertical similarity can be estimated using any of the metrics proposed in Hong *et al.* [48]. A final alternative is to use a learning-to-rank algorithm that favors diversity in the predicting ranking [81].

Resource Selection in a Homogeneous Environment

A consistent trend in the experiments presented so far in the dissertation is the importance of evidence integration in aggregated web search. Evidence-integration plays a critical role in deciding *which* verticals to issue the query to (Chapters 3 and 4) as well as in deciding *where* to present the vertical results (Chapter 6). However, up to this point, we have focused on aggregated web search. Is evidence integration also useful in a more traditional—more homogeneous—federated search environment?

In this chapter, we propose and evaluate a classification-based framework for resource selection in a homogeneous federated search environment, where collections contain similar types of (text-rich) documents and use similar retrieval algorithms. Most prior resource selection methods, such as CORI [14] and ReDDE [94], were designed with a homogeneous environment in mind and a large body of published work empirically confirms their effectiveness in this type of environment [14, 90, 94, 95, 96, 98, 104].

Consistent with the main theme of the dissertation, we explore a variety of sources of evidence believed to be correlated with resource relevance. We use features similar to those presented in previous chapters and explore a few new ones. Most importantly, we explore evidence derived from click-through data. Once in operation, a federated search system has access to implicit user feedback on aggregated results from previously seen queries. Given a new query, one potentially useful source of evidence is the query's similarity to those previously seen queries that have resulted in a click on a document from a particular resource.

In Chapter 4, we presented models that attempt to make maximal use of already-available training data (collected for a set of existing verticals) to learn a predictive model for a new vertical. In this chapter, we propose another cost-effective method for training a classification-based resource selector. In contrast to the methods in Chapter 4, however, the method proposed here uses no editorial input of any kind. For this reason, we refer to it as *zero-judgement training*.

The idea behind zero-judgement training is the following. Given n candidate resources, the goal of resource selection is to decide which k resources to issue the query to. We refer to a (federated) retrieval that merges content from $k < n$ resources as a *partial-dataset retrieval* and one that merges content from all $k = n$ resources as a *full-dataset retrieval*. At the core, our zero-judgement training method is based on the fol-

¹This work was published in CIKM 2009 with co-authors Fernando Diaz and Jamie Callan [2].

lowing assumption: given a query, an effective partial data-set retrieval will resemble a full-dataset retrieval. If we assume this to be true, then, given k , the resource selection objective can be viewed as that of selecting those k resources whose merged ranking more closely approximates a full-dataset merge. More specifically, because users scan results from top-to-bottom, the objective is to select those k resource that contribute the greatest number of results to the top ranks of a full-dataset merge. Given this objective, our zero-judgement training method, then, is to generate training data from a set of full-dataset retrievals conducted off-line. Put differently, we train a classification system to predict a collection’s inclusion in the top ranks of a full-dataset retrieval.

We present a thorough evaluation of the classification-based approach and several state-of-the-art resource selection methods on three federated search testbeds which were constructed in-house. This allowed us to investigate several questions not yet addressed in the dissertation. How does a classification-based approach perform with hundreds of candidate resources? Our three testbeds consisted of 30, 250, and 1000 collections, where the first testbed had the largest collections and the third has the smallest collections. Recall that the classic resource selection approach is to derive evidence *exclusively* from sampled documents. Another question we explore is: what is the effect of resource representation quality on resource selection effectiveness across algorithms? Compared to the classification approach, how do these methods perform when given access to fairly complete representations (i.e., large sample sets relative to the collection size)? How do they perform when given access to impoverished representations (small sample sets relative to the collection size)? Is there more to gain from evidence-integration—from the classification-based approach’s ability to easily integrate evidence as features—in one scenario vs. the other?

7.1 Formal Task Definition

Given a set of n candidate collections and a query q , the goal of resource selection is to choose k collections from which to retrieve documents. Consistent with most prior work in resource selection, we assume that k , or the number of collections to select, is given. We expect that, on average, a retrieval that merges content from $k = n$ collections will be superior to one that merges content from $k < n$ collections. Therefore, given k , our goal is to select those collections whose merged ranking more closely approximates a ranking that merges content from all n collections.

In this chapter, we are primarily concerned with the quality of the end-to-end output: the merged results. In order to focus on resource selection (rather than merging), we simulate merging by assuming access to each (retrieved) document’s centralized index score. That is, we assume access to the retrieval score that would be given to a (retrieved) document if the query were issued to an index of all collection content.

7.2 Classification-based Approach

Our classification approach takes the form of n independent, one-vs-all logistic regression models (one per collection). We trained logistic regression models using the LibLinear toolkit.¹ At test time, given a query, each classifier makes a binary prediction with respect to its collection. Then, we prioritize all n collections based on each classifier’s confidence of a positive prediction, $P_i(y = 1|q)$. Given this ranking of n collections, we select the top k .

Training collection-specific classifiers requires training data in the form of binary judgements on collections. If \mathcal{Q} denotes the set of queries and \mathcal{C} the set of target collections, we require a function of the form,

$$\mathcal{F} : \mathcal{Q} \times \mathcal{C} \rightarrow \{+1, -1\},$$

which maps query-collection pairs to $+1$, if C_i is relevant to q , and -1 , otherwise.

As previously noted, our objective is to learn a model that selects collections based on their contribution to a full-dataset retrieval. This is based on the assumption that, on average, a full-dataset retrieval, where $k = n$, outperforms a partial dataset retrieval, where $k < n$. Given a full-dataset retrieval of query q , we generate true (positive/negative) labels for every collection based on each collection’s contribution to the top (full-dataset) ranks. More specifically, we consider the query a *positive* example for the collection if the collection contributes more than τ documents to the top T full-dataset results. Otherwise, we consider the query a negative example.

7.3 Features

We investigate three types of features. Consistent with Chapter 3, we explore *corpus features*, derived from sampled documents, and *query category features*, derived from the query topic. Additionally, we explore *click-through features*, derived from queries with clicks on documents from the collection.

7.3.1 Corpus Features

As in Chapter 3, corpus features are derived from sampled collection documents. In this work, however, instead of using a *single* collection scoring method as a type of feature, we adopted *three* existing resource selection methods: CORI [14], Seo and Croft’s geometric average approach [86] (GAVG), and ReDDE.top, the same ReDDE variant used in Chapter 3. Although these three methods derive evidence from the same source (i.e., collection samples), they model different phenomena. CORI and GAVG model the similarity between the query and collection text. They differ in that CORI models the collection as one large query-independent bag of words, while GAVG focuses on those collection documents most similar to the query. In contrast, ReDDE.top approximates

¹Available at: <http://www.csie.ntu.edu.tw/~cjlin/liblinear/>

each collection’s average retrieval score given a full-dataset retrieval. Because they model different phenomena, these resource scoring methods may contribute complementary evidence. Thus, we included features from all three. Next, we describe these in more detail.

CORI

CORI adapts INQUERY’s inference net document ranking approach to ranking collections [14]. Here, all statistics are derived from sampled documents rather than the full collection. We used the CORI score with respect to each collection as a feature, for a total of n CORI score features.

Geometric Average (GAVG)

Seo and Croft’s approach [86] issues the query to the centralized sample index and scores collection C_i by the geometric average query likelihood from its top m samples,

$$\text{GAVG}_q(C_i) = \left(\prod_{d \in \text{top } m \text{ from } S_i} P(q|d) \right)^{\frac{1}{m}},$$

where $P(q|d)$ is the query likelihood score and S_i denotes C_i ’s sample set. If fewer than m documents from S_i match the query, the product above is padded with $P_{\min}(q|d)$, the retrieval’s minimum query likelihood. We used the GAVG score with respect to each collection as a feature, for a total of n GAVG score features.

ReDDE.top

Like GAVG, given a query, ReDDE.top conducts a retrieval from a centralized sample index, producing a retrieval score for every sampled document. Then, it assumes that each scored sampled document within the top N represents some number of documents in its original collection with the same score. Finally, by normalizing across collections, it scores each collection by the average retrieval score in its estimated distribution of scores. More details are presented in Section 3.3.1.

We used two sets of ReDDE.top features, one set using $N = 100$ and a second set using $N = 1,000$, for the following reason. The first set accumulates scores from the top 100 sampled documents. However, a collection with no samples in the top 100 receives a score of zero. This is problematic if the number of collections with a non-zero score is less than k , the number of collections to select. To increase the number of collections with a non-zero ReDDE.top score, we used a second set of ReDDE.top features setting $N = 1,000$. We use $2n$ ReDDE.top features: n features corresponding to the ReDDE.top score for each collection setting $N = 100$ and another n features corresponding to the ReDDE.top score for each collection setting $N = 1,000$.

7.3.2 Query Category Features

If collections are topically-focused, then a potentially useful source of evidence is the topic of the query. We built a set of binary query-topic classifiers as follows. First, we selected a set of 166 topics from the Open Directory Project (ODP) hierarchy and crawled Web documents associated with each of these ODP nodes.² Then, these document sets were used to train logistic-regression classifiers (one per category) using unigram features. Because queries are terse, instead of applying each trained classifier directly on the query string (represented as a unigram vector), we apply each classifier to those documents in the centralized sample index. Finally, we classify the query using a retrieval from this index. We set the value of category feature y_i according to,

$$\text{CAT}_q(y_i) = \frac{1}{\mathcal{Z}} \sum_{d \in \mathcal{R}_{S,q}^N} P(q|d) \left(\frac{P(y_i|d)}{\sum_{y_j \in \mathcal{Y}} P(y_j|d)} \right), \quad (7.1)$$

where $P(y_i|d)$ is category y_i 's confidence value on document d and the normalizer $\mathcal{Z} = \sum_{d \in \mathcal{R}_{S,q}^N} P(q|d)$. For these features, we set $N = 100$.

We use 166 query category features (one per topical category). In Chapter 6 we also used query-category features, but, in order to reduce the number of features, we clustered ODP nodes into 30 categories. In this work, we used a larger number of category features for two reasons. First, we focus on testbeds with hundreds of resources (i.e., 250 and 1000 resources). Thus, using more fine-grained topics seems useful for distinguishing between these. Second, as we show later, our zero-judgement training method can be used to produce a fairly large training set (i.e., 75,000) queries. Thus, reducing the number of category features (at the expense of possibly losing discriminative category granularity) seems undesirable.

7.3.3 Click-through Features

Once in operation, a resource selection system has access to user feedback in the form of clicks on collection documents. A click on a document can be viewed as a surrogate for document relevance. We view a click on a document as a surrogate for collection relevance, in favor of the collection from which the document originates. Click-through features exploit the similarity between the query and those previously seen queries that have resulted in a click on a document from the collection.

We model click-events as follows. For a collection, C_i , let Q_i denote the set of queries (allowing duplicates) associated with a click event on a document from C_i . We index each Q_i as an individual pseudo-document in a corpus of n documents. Then, given a query, we use the retrieval score of each Q_i as a feature. This results in n click-through features (one per collection).

²<http://www.dmoz.org>

7.3.4 Summary of Features

The total number of features varies with the number of candidate resources (denoted by n), which varies across testbeds. CORI and GAVG contribute n features each (one per resource). ReDDE.top contributes $2n$ features (n with setting $N = 100$ and another n with setting $N = 1,000$). In addition to these, we use 166 query-category features and n click-through features (one per resource).

7.4 Methods and Materials

In this section, we describe our experimental methodology: our experimental testbeds, evaluation queries, evaluation metrics, and baselines for comparing the performance of the proposed classification-based framework. Also, we describe a few implementation details pertaining to our zero-judgement training method and some of our features.

7.4.1 Evaluation Testbeds

The TREC GOV2 test collection is a large crawl of the “.gov” portion of the Web, containing about 25M documents.³ The GOV2 corpus was used to construct 3 experimental federated search testbeds, varying the number of target collections: 1,000, 250, and 30. We refer to these testbeds as *gov2.1000*, *gov2.250*, and *gov2.30*, respectively. We constructed the *gov2.1000* testbed following the procedure described in Fallen and Newby [34]. While the GOV2 corpus consists of about 17,000 unique hosts (e.g., www.epa.gov), the largest 1,000 hosts contain about 90% of the GOV2 collection (i.e., about 22M documents). The *gov2.1000* testbed was constructed by treating each of the largest 1,000 hosts as a separate collection.

Testbeds *gov2.250* and *gov2.30* were constructed by clustering hosts in the *gov2.1000* testbed into 250 and 30 clusters, respectively, as follows. First, to represent hosts, we randomly sampled 1,000 documents from each. We define a host’s vocabulary by all term-stems (using the Porter stemmer [79]) appearing at least 10 times in its document sample. Host-specific language models were constructed using maximum likelihood without smoothing. The distance between hosts was computed using the Jeffrey divergence between their respective language models [53], also known as the symmetric Kullback-Leibler divergence,

$$D_J(\theta_i || \theta_j) = \sum_w (P(w|\theta_i) - P(w|\theta_j)) \log_2 \left(\frac{P(w|\theta_i)}{P(w|\theta_j)} \right).$$

We used average-link agglomerative clustering, iteratively merging clusters according to their hosts’ average pair-wise similarity. We seeded the clustering by first combining hosts belonging to the same government entity (e.g., [nih](http://nih.gov), [usgs](http://usgs.gov), [usda](http://usda.gov), [epa](http://epa.gov), [uspto](http://uspto.gov), [nasa](http://nasa.gov)).

Figure 7.1 shows each testbed’s collection size distribution. The *gov2.1000* and *gov2.250* testbeds have a few large collections and many small collections, while *gov2.30* has many

³http://ir.dcs.gla.ac.uk/test_collections/gov2-summary.htm

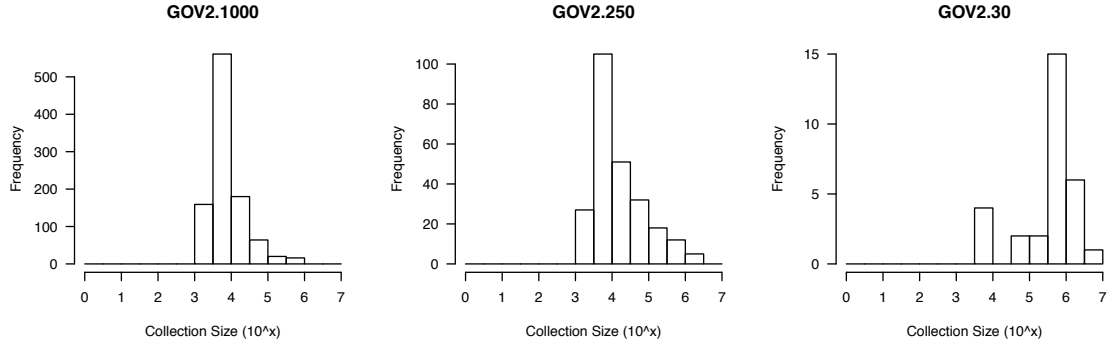


Figure 7.1: Collection size distribution of our three experimental testbeds.

large collections and a few small ones. In the *gov2.1000* testbed, 720 (72%) collections have fewer than 10,000 documents and 438 (44%) have fewer than 5,000 documents. In the *gov2.250* testbed, 131 (66%) collections have fewer than 10,000 documents. In the *gov2.30* testbed, 24 (80%) collections have more than 1M documents.

7.4.2 Queries

We evaluated on TREC queries 701-850, used in the ad-hoc retrieval task of the Terabyte Track from 2004, 2005, and 2006. Recall that we are missing about 10% of the GOV2 collection in our testbeds, corresponding to those documents in GOV2 not originating from the 1,000 largest hosts. In spite of these missing documents, all queries had at least one relevant document in our testbeds except query number 703, which has no relevant documents in the entire GOV2 collection.

7.4.3 Evaluation Metrics

We are interested in the quality of document rankings produced by selecting only a few collections and combining their documents into a single ranked list. For this reason, we evaluate in terms of precision at different cut-off points, $\mathcal{P}@\{5, 10, 30\}$, when selecting between 1-5 collections. To focus evaluation on resource selection rather than results merging, we assume access to a function that provides the score that a centralized retrieval would have provided for every document retrieved. Access to such a function is realistic in a cooperative setting, for example, where resources divulge corpus statistics (e.g., number of documents, average document length, IDF values, etc.) to facilitate document score normalization and merging. Thus, given a set of collections selected, we combine their documents into a single ranked list according to each document’s “global” retrieval score.

7.4.4 Implementation Details

Zero-Judgement Training

As described in Section 7.2, we train a classification system to select collections based on their contribution to the top-ranks of a full-dataset retrieval. During training, for a given collection, the query is considered a positive instance if the collection contributes more than τ documents to the top T ranks of a full-dataset retrieval. We set $T = 30$ because we evaluate merged results in terms of $\mathcal{P}@\{5, 10, 30\}$ and we set $\tau = 3$ in order to ignore collections that contribute only a few documents to the top 30. We do not claim that this configuration is optimal. Another alternative, for example, would have been to train different models using $T = \{5, 10, 30\}$ when evaluating based on $\mathcal{P}@\{5, 10, 30\}$, respectively.

Our “training data” consists of full-dataset retrievals which select and merge content from *all* collections. There are multiple possibilities for selecting a set of “training” queries (e.g., using a query-log or generating artificial queries from the collection text). However, one requirement is that there be enough positive instances for training for every collection. In other words, for each collection, there should be a sufficient number of queries with hits in the collection.

In this work, training queries were sampled from the AOL query-log, which consists of about 20M queries that were issued by about 650K users to the AOL search engine over a period of three months in 2006. Recall that our three experimental testbeds consist of clusters of hosts from the “.gov” domain (1,000 singleton host clusters in the case of the *gov2.1000* testbed). Click events in the AOL query-log are uniquely identified by user ID, query, date/time and host URL (i.e., for the host associated with the document clicked). Therefore, it is possible to identify all AOL click events associated with any one of our 1,000 hosts. For each host, we estimate a query multinomial using the query’s relative frequency in its click events. A set of 75,000 queries was sampled (without replacement) using a two-step iterative processes. First, a host is sampled uniformly from the set of 1,000 hosts. Then, a query is sampled from the host’s query multinomial. Hosts were sampled uniformly to favor coverage across hosts and, thereby, coverage across collections in our three testbeds. Queries were sampled according to their relative frequency in click events in order to favor popular queries likely to have hits in the collection.

Features Implementation

Some of our features (i.e., CORI, GAVG, ReDDE.top) required sampling documents from every collection. Documents were sampled from each collection uniformly without replacement. In some federated search environments, collection documents may only accessible to the system via a search interface. If this is the case, prior work suggests that high-quality samples (similar to those obtained using uniform sampling) can be obtained using query-based sampling [16].

Click-through features required simulating click events on collection documents.

Click-through data collected over time was simulated also using the AOL query-log. We collected a total of 305,236 click events associated with our 1,000 “.gov” hosts. About 25% of hosts had no AOL click events and about 35% had fewer than 50. Click events associated with a query in our test set (described later) were omitted from the set of queries used for training and from those used to simulate click-through data.

7.4.5 Single-Evidence Baselines

The classification approach was evaluated against six single-evidence baselines, including one for every type of feature used in the classification approach.

ReDDE.top was used as a single-evidence baseline as follows. First collections are prioritized by ReDDE.top score using $N = 100$. A second priority list is constructed using $N = 1,000$. If the first priority list has fewer than k collections, the remaining collections are selected from the second priority list.

In addition to ReDDE.top, we evaluated against the original version of ReDDE [94], which is described in detail in Section 2.3. We used the version of ReDDE referred to as *modified* ReDDE in Si and Callan [94]. As we did with ReDDE.top, modified ReDDE creates two priority lists: one by setting ReDDE parameter $\tau = 0.0005$ and a second one by setting $\tau = 0.003$. Given k , modified ReDDE selects collections from the first priority list only if the collection has a ReDDE mass greater than 0.10. If more collections are needed to complete k collections, the remaining collections are selected from the second priority list.

Our category baseline (denoted as CATS) scores collections based on the similarity between the topical profile of the query and the topical profile of the collection. The query’s topical profile is given by normalizing Equation 7.1 across categories, such that $\sum_{y_j \in \mathcal{Y}} \text{CATS}_q(y_j) = 1$. The collection’s topical profile is defined by,

$$P(y_j|C_i) = \frac{1}{|S_i|} \sum_{d \in S_i} \frac{P(y_j|d)}{\sum_{y_k \in \mathcal{Y}} P(y_k|d)}$$

where S_i denotes the set of documents sampled from collection C_i . The similarity between the query and collection topical profiles is given by the Bhattacharya distance between these two distributions [7],

$$\mathcal{B}(q, C_i) = \sum_{y_k \in \mathcal{Y}} \sqrt{P(y_k|q) \times P(y_k|C_i)}.$$

Finally, we also evaluated a baseline approach that scores collections by query likelihood given its click-through queries, denoted by CLICK, as described in Section 7.3.3.

7.5 Experimental Results

As previously mentioned, one of our objectives is to investigate the effect of resource representation quality on resource selection performance across algorithms. In particular, we focus on sample set quality. Four of our single-evidence baselines (i.e., CORI,

P@5								
k	full	cori	gavg	redde.top	redde	cats	click	classification
1	0.569	0.224	0.405	0.360	0.166	0.192	0.183	0.392 (-3.31%)
2	0.569	0.315	0.446	0.447	0.275	0.256	0.239	0.436 (-2.40%)
3	0.569	0.372	0.479	0.489	0.336	0.302	0.277	0.482 (-1.37%)
4	0.569	0.405	0.483	0.506	0.380	0.321	0.322	0.506 (0.00%)
5	0.569	0.417	0.495	0.529	0.395	0.336	0.337	0.510 (-3.55%)
P@10								
k	full	cori	gavg	redde.top	redde	cats	click	classification
1	0.534	0.188	0.331	0.321	0.150	0.152	0.147	0.355 (7.30%)
2	0.534	0.264	0.390	0.394	0.248	0.215	0.194	0.399 (1.19%)
3	0.534	0.323	0.423	0.436	0.302	0.261	0.228	0.446 (2.47%)
4	0.534	0.359	0.438	0.457	0.344	0.285	0.270	0.458 (0.15%)
5	0.534	0.380	0.442	0.484	0.364	0.302	0.281	0.468 (-3.33%)
P@30								
k	full	cori	gavg	redde.top	redde	cats	click	classification
1	0.452	0.113	0.201	0.206	0.102	0.095	0.091	0.224 (8.68%)
2	0.452	0.167	0.266	0.268	0.168	0.139	0.124	0.281 (4.59%)
3	0.452	0.217	0.305	0.312	0.206	0.170	0.152	0.319 (2.51%)
4	0.452	0.247	0.319	0.337	0.248	0.194	0.185	0.339 (0.53%)
5	0.452	0.266	0.325	0.362	0.275	0.205	0.195	0.352 (-2.60%)

Table 7.1: Results for experimental condition *gov2.1000.1000*.

GAVG, ReDDE.top, and ReDDE) focus exclusively on evidence derived from collection samples. The classification-based approach used these as input features and, thus, is also indirectly affected by sample set quality. To investigate the effect of sample set quality on performance, every method was evaluated on all three testbeds under two conditions: using sample sets of 300 samples per collection and using sample sets of 1000 samples per collection. We denote the resulting six experimental conditions as *gov2.x.y*, where x denotes the testbed and y denotes the sample set size. For instance, *gov2.1000.1000* denotes the condition which uses the *gov2.1000* testbed and sample sets of 1000 samples per collection.

Results are presented based on $\mathcal{P}@\{5, 10, 30\}$. Tables 7.1-7.3 show results across our three testbeds when sampling 1,000 documents from each collection. Tables 7.4-7.6 show results when sampling 300 documents from each collection. In addition, to evaluate the overall performance of federated search, we present results from centralized retrieval (denoted as “full”), from a single index of all n collections combined. The classification approaches’s percent improvement is with respect to the *best* single-evidence baseline. Statistical significance is with respect to *all* single-evidence baselines. Significance, using a paired t-test on queries, is denoted with a \triangle at the $p < 0.05$ level and a \blacktriangle at the $p < 0.005$ level.

The classification-based approach either significantly outperforms or is statistically indistinguishable from the *best* single-evidence baseline in all cases. In the *gov2.1000.1000* condition, the GAVG and ReDDE.top baselines perform at the same level as the classification approach. We investigate how this experimental condition favors these methods in the next section.

From the performance of our single-evidence baselines, we notice two trends. First, all baselines that derive evidence from sampled documents (i.e., CORI, GAVG, ReDDE.top,

P@5								
<i>k</i>	full	cori	gavg	redde.top	redde	cats	click	classification
1	0.569	0.137	0.294	0.326	0.238	0.174	0.220	0.419 (28.40%)[▲]
2	0.569	0.228	0.328	0.408	0.360	0.242	0.303	0.494 (21.05%)[▲]
3	0.569	0.291	0.360	0.432	0.417	0.272	0.340	0.497 (14.91%)[▲]
4	0.569	0.323	0.374	0.475	0.483	0.313	0.364	0.505 (4.44%)
5	0.569	0.357	0.389	0.489	0.503	0.345	0.388	0.515 (2.40%)
P@10								
<i>k</i>	full	cori	gavg	redde.top	redde	cats	click	classification
1	0.534	0.105	0.248	0.283	0.209	0.142	0.188	0.371 (31.35%)[▲]
2	0.534	0.186	0.293	0.363	0.311	0.201	0.262	0.452 (24.40%)[▲]
3	0.534	0.248	0.330	0.394	0.372	0.229	0.291	0.460 (16.70%)[▲]
4	0.534	0.282	0.338	0.432	0.430	0.266	0.308	0.477 (10.58%)[▲]
5	0.534	0.293	0.350	0.438	0.457	0.297	0.334	0.487 (6.46%)[△]
P@30								
<i>k</i>	full	cori	gavg	redde.top	redde	cats	click	classification
1	0.452	0.068	0.158	0.197	0.143	0.090	0.130	0.265 (34.13%)[▲]
2	0.452	0.124	0.196	0.272	0.230	0.132	0.182	0.343 (26.19%)[▲]
3	0.452	0.168	0.233	0.309	0.283	0.151	0.213	0.359 (16.05%)[▲]
4	0.452	0.204	0.245	0.337	0.331	0.187	0.227	0.372 (10.22%)[▲]
5	0.452	0.226	0.262	0.344	0.353	0.208	0.246	0.382 (8.38%)[▲]

Table 7.2: Results for experimental condition *gov2.250.1000*.

P@5								
<i>k</i>	full	cori	gavg	redde.top	redde	cats	click	classification
1	0.569	0.281	0.302	0.322	0.295	0.323	0.298	0.370 (14.52%)
2	0.569	0.380	0.403	0.419	0.428	0.384	0.374	0.447 (4.39%)
3	0.569	0.434	0.446	0.456	0.447	0.427	0.421	0.487 (6.76%)
4	0.569	0.462	0.468	0.487	0.472	0.451	0.454	0.499 (2.48%)
5	0.569	0.474	0.472	0.503	0.491	0.482	0.460	0.507 (0.80%)
P@10								
<i>k</i>	full	cori	gavg	redde.top	redde	cats	click	classification
1	0.534	0.246	0.264	0.269	0.246	0.280	0.255	0.318 (13.67%)
2	0.534	0.332	0.348	0.361	0.368	0.340	0.335	0.393 (6.56%)
3	0.534	0.391	0.387	0.403	0.392	0.384	0.374	0.438 (8.49%)[△]
4	0.534	0.426	0.415	0.442	0.415	0.407	0.413	0.461 (4.41%)
5	0.534	0.445	0.429	0.462	0.444	0.433	0.423	0.471 (2.03%)
P@30								
<i>k</i>	full	cori	gavg	redde.top	redde	cats	click	classification
1	0.452	0.181	0.188	0.185	0.167	0.195	0.176	0.220 (12.87%)
2	0.452	0.253	0.262	0.261	0.269	0.267	0.241	0.304 (13.32%)[△]
3	0.452	0.309	0.294	0.304	0.304	0.302	0.280	0.346 (11.71%)[△]
4	0.452	0.339	0.326	0.337	0.328	0.313	0.309	0.361 (6.47%)
5	0.452	0.353	0.341	0.358	0.345	0.334	0.320	0.377 (5.25%)

Table 7.3: Results for experimental condition *gov2.30.1000*.

and ReDDE) performed better using 1,000 vs. 300 samples per collection. This shows that these methods are sensitive to the sampled set size. They perform better with more evidence, which is consistent with previous evaluations [92]. Second, the relative performance between single-evidence methods varied across experimental conditions. In the *gov2.1000.1000* condition, GAVG and ReDDE.top clearly outperform CATS and CLICK in all cases. This is not true in the *gov2.30.300* condition. When $k = 1$, in the *gov2.30.300* condition, CATS and CLICK both outperform GAVG and ReDDE.top. These approaches derive evidence from different sources. GAVG and ReDDE.top derive evidence exclu-

P@5								
<i>k</i>	full	cori	gavg	redde.top	redde	cats	click	classification
1	0.569	0.209	0.323	0.303	0.125	0.200	0.183	0.383 (18.26%)
2	0.569	0.306	0.358	0.407	0.221	0.263	0.239	0.427 (4.95%)
3	0.569	0.340	0.399	0.450	0.283	0.301	0.277	0.463 (2.99%)
4	0.569	0.370	0.427	0.466	0.313	0.313	0.322	0.482 (3.46%)
5	0.569	0.381	0.440	0.478	0.350	0.333	0.337	0.490 (2.53%)
P@10								
<i>k</i>	full	cori	gavg	redde.top	redde	cats	click	classification
1	0.534	0.174	0.277	0.270	0.110	0.166	0.147	0.332 (19.90%)^Δ
2	0.534	0.260	0.317	0.358	0.191	0.224	0.194	0.392 (9.57%)
3	0.534	0.293	0.361	0.399	0.253	0.269	0.228	0.432 (8.07%)
4	0.534	0.321	0.381	0.419	0.283	0.276	0.270	0.444 (5.93%)
5	0.534	0.339	0.399	0.433	0.321	0.297	0.281	0.456 (5.27%)
P@10								
<i>k</i>	full	cori	gavg	redde.top	redde	cats	click	classification
1	0.452	0.097	0.174	0.171	0.074	0.101	0.091	0.218 (25.65%)^Δ
2	0.452	0.162	0.216	0.245	0.126	0.141	0.124	0.270 (10.52%)
3	0.452	0.191	0.249	0.284	0.172	0.176	0.152	0.304 (7.10%)
4	0.452	0.211	0.267	0.304	0.197	0.185	0.185	0.324 (6.47%)
5	0.452	0.230	0.284	0.321	0.224	0.208	0.195	0.341 (6.05%)

Table 7.4: Results for experimental condition *gov2.1000.300*.

P@5								
<i>k</i>	full	cori	gavg	redde.top	redde	cats	click	classification
1	0.569	0.125	0.212	0.274	0.228	0.111	0.220	0.391 (42.65%)[▲]
2	0.569	0.184	0.267	0.353	0.333	0.176	0.303	0.472 (33.84%)[▲]
3	0.569	0.246	0.286	0.407	0.391	0.232	0.340	0.494 (21.45%)[▲]
4	0.569	0.267	0.306	0.434	0.428	0.268	0.364	0.498 (14.86%)[▲]
5	0.569	0.290	0.319	0.462	0.447	0.279	0.388	0.518 (12.21%)[▲]
P@10								
<i>k</i>	full	cori	gavg	redde.top	redde	cats	click	classification
1	0.534	0.096	0.178	0.228	0.186	0.089	0.188	0.342 (49.71%)[▲]
2	0.534	0.161	0.234	0.304	0.270	0.156	0.262	0.427 (40.40%)[▲]
3	0.534	0.218	0.249	0.366	0.350	0.195	0.291	0.450 (22.94%)[▲]
4	0.534	0.236	0.273	0.399	0.387	0.218	0.308	0.456 (14.31%)[▲]
5	0.534	0.258	0.283	0.417	0.409	0.233	0.334	0.476 (13.99%)[▲]
P@30								
<i>k</i>	full	cori	gavg	redde.top	redde	cats	click	classification
1	0.452	0.057	0.115	0.148	0.133	0.055	0.130	0.241 (62.60%)[▲]
2	0.452	0.109	0.161	0.209	0.190	0.107	0.182	0.315 (51.13%)[▲]
3	0.452	0.149	0.182	0.253	0.251	0.138	0.213	0.333 (31.42%)[▲]
4	0.452	0.173	0.200	0.292	0.277	0.154	0.227	0.350 (19.77%)[▲]
5	0.452	0.187	0.210	0.314	0.302	0.166	0.246	0.362 (15.34%)[▲]

Table 7.5: Results for experimental condition *gov2.250.300*.

sively from sampled documents. CATS derives evidence from the topical similarity between the query and the collection. CLICK derives evidence from click-through data. Different types of evidence was particularly useful under different conditions.

Two results support the hypothesis that full-dataset retrievals can be used to harvest data for training a machine learned resource selection method. First, a full-dataset retrieval outperformed all methods, including the classification approach, in all cases. Second, the classification approach, trained on data harvested from full-dataset retrievals, performed at the same level or better than the best single-evidence baseline in all cases.

P@5								
<i>k</i>	full	cori	gavg	redde.top	redde	cats	click	classification
1	0.569	0.224	0.266	0.251	0.231	0.282	0.298	0.374 (25.68%)[▲]
2	0.569	0.317	0.322	0.350	0.342	0.353	0.374	0.450 (20.07%)[▲]
3	0.569	0.409	0.376	0.391	0.400	0.412	0.421	0.493 (16.88%)[▲]
4	0.569	0.446	0.424	0.403	0.413	0.444	0.454	0.487 (7.40%)
5	0.569	0.467	0.436	0.443	0.442	0.464	0.460	0.509 (8.91%)
P@10								
<i>k</i>	full	cori	gavg	redde.top	redde	cats	click	classification
1	0.534	0.194	0.222	0.206	0.178	0.238	0.255	0.321 (25.79%)[▲]
2	0.534	0.287	0.271	0.313	0.292	0.299	0.335	0.402 (20.04%)[▲]
3	0.534	0.361	0.327	0.353	0.338	0.352	0.374	0.442 (18.13%)[▲]
4	0.534	0.403	0.363	0.370	0.376	0.394	0.413	0.457 (10.55%)[△]
5	0.534	0.432	0.385	0.413	0.401	0.428	0.423	0.479 (10.71%)[△]
P@30								
<i>k</i>	full	cori	gavg	redde.top	redde	cats	click	classification
1	0.452	0.128	0.153	0.143	0.118	0.153	0.176	0.223 (26.75%)[▲]
2	0.452	0.206	0.198	0.227	0.200	0.218	0.241	0.312 (29.38%)[▲]
3	0.452	0.263	0.243	0.265	0.246	0.271	0.280	0.347 (24.06%)[▲]
4	0.452	0.308	0.282	0.291	0.282	0.300	0.309	0.367 (18.51%)[▲]
5	0.452	0.336	0.295	0.334	0.317	0.322	0.320	0.390 (15.97%)[▲]

Table 7.6: Results for experimental condition *gov2.30.300*.

7.6 Discussion

In this section, we further investigate the effect of resource representation quality (i.e., sample set quality) on resource selection performance across methods. Furthermore, we present results from a set of feature ablation studies which investigate whether different features are more or less predictive given varying degrees of resource representation quality.

7.6.1 Collection Representation Quality

As shown in Table 7.1, ReDDE.top and GAVG, which derive evidence from sampled documents, performed well in the *gov2.1000.1000* experimental condition. In this condition, 1,000 documents were sampled from every collection. As previously mentioned, 72% of collections in the *gov2.1000* testbed have fewer than 10,000 documents and 44% have fewer than 5,000 documents. This means that a sample set of 1,000 documents constitutes at least 20% of the full collection for about half the collections in *gov2.1000*. In other words, in the *gov2.1000.1000* condition, ReDDE.top and GAVG had access to fairly complete representations for about half the collections.

Furthermore, we would expect these methods to do well if these smaller collections frequently contain relevant documents. To examine this, we binned collections by their number of documents and determined, for each bin, the number of times a collection from the bin contains at least 10 documents relevant to a test query. These histograms are shown in Figure 7.2. In the *gov2.1000* testbed, the smallest collections, with 1,000-10,000 documents, most often contained at least 10 documents relevant to a test query. In contrast, in the *gov2.250* and *gov2.30* testbeds, the collections that most often contain

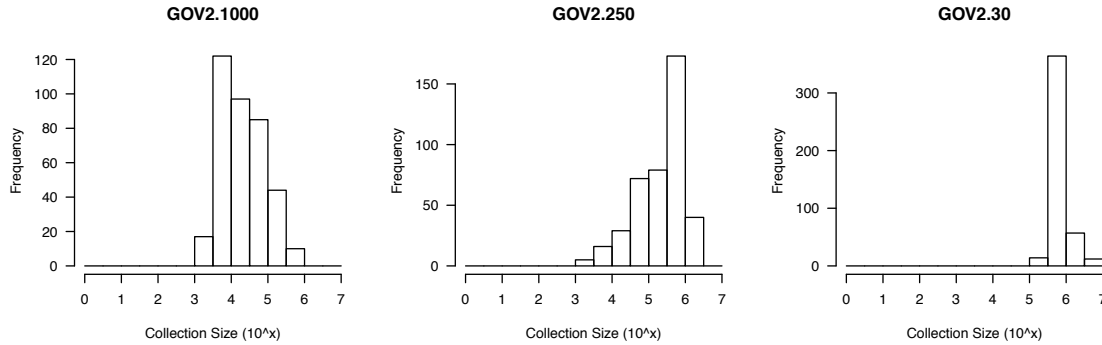


Figure 7.2: Number of instances in which a collection of a given size (bin) contributes at least 10 relevant documents to a test query.

at least 10 documents relevant to a test query have more than 100,000 documents.

To conclude, we can say that in the *gov2.1000.1000* condition, corpus-based single-evidence baselines such as ReDDE.top and GAVG benefited from having fairly complete representations (i.e. large sampled sets relative to the collection size) for those collections containing many relevant documents. Interestingly, in this condition, the classification approach performed at the same level as these methods. In other conditions, we see a more clear benefit from the kind of evidence-integration afforded by the classification approach.

7.6.2 Feature Ablation Studies

The classification approach integrates different types of evidence as input features. In this section, we conduct a set of feature ablation studies to test the contribution of evidence integration to the classification approach’s performance. We focus on experimental condition *gov2.1000.1000* and *gov2.30.300*. Our motivation is to verify that the classification approach is capable of focusing on the most reliable features under different experimental conditions. Based on the analysis from Section 7.6.1, in the *gov2.1000.1000* condition, we expect the classification approach to focus on evidence derived from sampled documents (i.e., CORI, GAVG, and ReDDE.top features). In the *gov2.30.300* condition, we expect it to focus on other types of evidence. We individually omitted each feature type (CORI, GAVG, ReDDE.top, CATS, and CLICK) and measure its contribution to performance based on the classifier’s percent decrease in precision. Significance, again, is tested using a paired t-test on queries.

Results are presented in Table 7.7. These results confirm our hypothesis. In the *gov2.1000.1000* condition, in the majority of cases, omitting ReDDE.top features leads to a significant drop in performance. This is because in the *gov2.1000.1000* condition, ReDDE.top had access to fairly complete representations for those collections with relevant content. On the other hand, in the *gov2.30.300* condition, CLICK features were more predictive, particularly in terms of $\mathcal{P}@30$.

gov2.1000.1000						
P@10						
k	all.features	no.cori	no.gavg	no.redde.top	no.cats	no.click
1	0.355	0.355 (0.00%)	0.357 (0.57%)	0.331 (-6.81%)	0.355 (0.00%)	0.354 (0.19%)
2	0.399	0.399 (0.00%)	0.393 (-1.52%)	0.383 (-4.04%)	0.385 (-3.37%)	0.401 (-0.51%)
3	0.446	0.446 (-0.15%)	0.436 (-2.26%)	0.401 (-10.23%) \blacktriangle	0.436 (-2.41%)	0.438 (-1.95%)
4	0.458	0.456 (-0.29%)	0.442 (-3.52%) \triangle	0.425 (-7.18%) \triangle	0.450 (-1.76%)	0.449 (-1.91%)
5	0.468	0.467 (-0.14%)	0.454 (-3.01%) \triangle	0.431 (-7.89%) \triangle	0.466 (-0.43%)	0.456 (-2.58%)
P@30						
k	all.features	no.cori	no.gavg	no.redde.top	no.cats	no.click
1	0.224	0.224 (0.00%)	0.227 (1.40%)	0.213 (-5.19%)	0.229 (2.20%)	0.225 (-0.20%)
2	0.281	0.281 (0.16%)	0.274 (-2.39%)	0.266 (-5.02%)	0.271 (-3.51%)	0.277 (-1.44%)
3	0.319	0.317 (-0.77%)	0.311 (-2.59%)	0.292 (-8.61%) \triangle	0.312 (-2.24%)	0.313 (-2.10%)
4	0.339	0.338 (-0.20%)	0.330 (-2.70%)	0.319 (-5.80%) \triangle	0.331 (-2.38%)	0.336 (-0.79%)
5	0.352	0.350 (-0.51%)	0.344 (-2.35%)	0.331 (-5.97%) \triangle	0.347 (-1.52%)	0.344 (-2.35%) \triangle
gov2.30.300						
P@10						
k	all.features	no.cori	no.gavg	no.redde.top	no.cats	no.click
1	0.321	0.321 (0.21%)	0.319 (-0.63%)	0.305 (-4.81%)	0.324 (1.05%)	0.279 (-13.18%)
2	0.402	0.394 (-2.00%)	0.390 (-3.01%)	0.392 (-2.50%)	0.388 (-3.51%)	0.379 (-5.84%)
3	0.442	0.438 (-0.91%)	0.428 (-3.04%)	0.423 (-4.26%)	0.431 (-2.43%)	0.435 (-1.52%)
4	0.457	0.449 (-1.76%)	0.455 (-0.44%)	0.469 (2.64%)	0.450 (-1.62%)	0.456 (-0.29%)
5	0.479	0.477 (-0.42%)	0.472 (-1.40%)	0.474 (-0.84%)	0.480 (0.28%)	0.465 (-2.81%)
P@30						
k	all.features	no.cori	no.gavg	no.redde.top	no.cats	no.click
1	0.223	0.224 (0.70%)	0.219 (-1.41%)	0.206 (-7.34%)	0.228 (2.41%)	0.191 (-14.37%)
2	0.312	0.309 (-1.22%)	0.300 (-4.08%) \triangle	0.301 (-3.51%)	0.295 (-5.52%)	0.283 (-9.46%) \triangle
3	0.347	0.338 (-2.51%)	0.333 (-3.99%)	0.335 (-3.54%)	0.330 (-4.90%) \triangle	0.319 (-8.12%) \triangle
4	0.367	0.356 (-2.93%)	0.364 (-0.79%)	0.370 (0.98%)	0.349 (-4.70%)	0.349 (-4.82%)
5	0.390	0.380 (-2.52%)	0.387 (-0.63%)	0.383 (-1.72%)	0.381 (-2.29%)	0.372 (-4.65%)

Table 7.7: Feature type ablation study. A significant drop in performance, using a paired t-test on queries, is denoted with a \triangle at the $p < 0.05$ level and a \blacktriangle at the $p < 0.005$ level.

This analysis demonstrates that the classification approach is capable on focusing on the most reliable features depending on the condition. Also, although CORI, GAVG, and ReDDE.top features derive evidence from the same source (i.e., sampled documents), they model different phenomena. Our results show that they do not contribute equally to performance. This further motivates a feature integration approach, even when the features are derived from the same source.

7.7 Summary

We evaluated a classification approach to resource selection in a homogeneous environment, one in which collections contain similar types of documents and satisfy similar types of information requests. The classification approach was compared to a number of single-evidence baselines, including three existing resource selection methods that have produced good results in previous evaluations. From this set of experiments, we draw the following conclusions.

- Most resource selection methods derive evidence from a single source of evidence, mainly sampled collection documents. Casting resource selection as a multiclass learning approach allows us to integrate multiple sources of evidence as input features. Evidence integration leads to a more robust solution. Different sources of evidence are more/less predictive under different experimental conditions. By integrating them all as input features, the classification approach is able to perform either at the same level or better than the best single-evidence baseline.
- This is the first time that a resource selection method models full-dataset retrieval *explicitly*. A classification system is trained to select collections based on their impact to a retrieval that merges content from all collections. This is significant because a lot of training data can be produced without human relevance judgements as long as the system has access to (offline) full-dataset retrievals.
- Corpus-based evidence, derived from collection samples, was integrated using three existing resource selection methods. Although they derive evidence from the same source, these three methods model different phenomena. A set feature of ablation studies (across experimental conditions) show that these methods are not redundant in terms of predictiveness. This further validates a feature integration approach, even if features are derived from the same source (in this case, sampled documents).
- In a homogeneous environment, a machine learning approach to resource selection can be trained using full-dataset retrievals, without using human relevance judgements. This result may generalize to a heterogeneous environment provided that the merging algorithm produces, on average, superior retrievals when merging results from every collections than when merging results from only a few. In fact, this may be a logical design principle for results merging algorithm development. If the results merging algorithm satisfies this criterion, it may be possible to train a resource selection algorithm from full-dataset retrievals in a heterogeneous environment, where results merging is more complex.

Conclusions

This chapter summarizes the work presented in the thesis, considers its major contributions, and discusses directions for future work.

8.1 Summary

This dissertation takes a machine-learning, feature-integration approach to the two major sub-tasks associated with federated search: *resource selection*—deciding *which* resources to issue the query to and *results merging*—deciding *where* to display the results from different resources within the final presentation.

With the exception of Chapter 7, we focused on *aggregated web search*—the prediction and integration of relevant vertical content into the Web search results. Compared to federated search environments investigated in prior research, aggregated web search is associated with two distinguishing properties: *result-type heterogeneity*—verticals retrieve very different types of results (e.g., news articles, images, videos, local business listings, weather forecasts)—and *retrieval-algorithm heterogeneity*—verticals use very different retrieval algorithms (e.g., recency is important for *news*, geographical proximity is important for *local*). These two types of heterogeneity violate some of the major assumptions made by state-of-the-art selection and merging algorithms.

8.1.1 Resource Selection

A consistent finding in all our experiments is that feature-integration is essential for effective resource selection. This is, in fact, one major limitation of existing resource selection methods—most derive evidence exclusively from (sampled) resource content and do not provide an easy way of incorporating other types of evidence. Throughout the thesis, we have demonstrated the importance of *corpus features* (derived from sampled vertical content), *query-log features*, (derived from previous queries issued directly to the vertical), *click-through features* (derived from clicks on vertical content), and *query features* (derived from properties of the query-string, independent of any vertical). In every feature ablation study, we found that no single type of feature is exclusively responsible for performance. A significant advantage of a machine learning approach is its capability of easily integrating multiple types of evidence as input features.

Our feature ablation studies also revealed a second important trend: query features, which are *not* generated from the vertical, are particularly important. In Chapter 3, for example, we found that the topic of the query (e.g., whether the query is health-related)

was the single most important type of evidence for vertical selection.¹ This is an important result because it means that any appropriate solution to vertical selection requires learning a vertical-*specific* relationship between features and vertical relevance. Different verticals will focus on different topics. Thus, exploiting query category evidence requires learning a vertical-specific relationship between the query category and the relevance of a particular vertical. For example, if the query is health-related, this is positive evidence for *health*, but negative evidence for *movies*. Other types of query evidence also require taking the identity of the vertical into consideration. The presence of the query-term “weather” (i.e., query-keyword evidence) is positive evidence for *weather*, but negative evidence for *games*. The presence of a location name in the query (i.e., named-entity type evidence) is positive evidence for *local*, but negative evidence for *finance*. A high degree of co-occurrence between the query and the term “video” (i.e., query-log evidence) is positive evidence for *video*, but negative evidence for *images*. Such vertical-*specific* predictive relationships can be automatically learned from training data, either by learning a different model for each vertical, as in Chapter 3, or by exploiting feature-interactions which consider the identity of the vertical, as in Chapter 6.

One limitation of a machine learning approach is that it requires training data. Therefore, our goal in Chapter 4 was to investigate ways of making maximal use of already-available training data (collected for a set of existing *source* verticals) to learn a predictive model for a new *target* vertical (associated with no training data).

We found that training an effective *portable* model—one that can make accurate predictions with respect to *any* vertical, including one absent of training data—requires identifying the most portable features—those consistently correlated with relevance (in the same direction) across verticals. The most portable features were found to be those generated from the vertical in question (i.e., from vertical query-traffic or from sampled vertical content). Within the set of most portable features were those corresponding to existing content-based resource selection methods. In retrospect, this result makes sense. These unsupervised resource-scoring methods were designed to use a *single* metric to make predictions with respect to *all* resources.

Conversely, we found that learning an effective *target-specific* model—one that is customized specifically for the target—requires harnessing non-portable, vertical-specific features. The challenge, however, is that harnessing target-specific features requires training data (with respect to the target). We showed that high-confidence predictions from a portable model (with respect to the target) can be used to harness these target-specific features. This result puts existing content-based resource selection methods under a new light—their portability across resources makes them a good source of target-vertical labels, which can be used to bootstrap a more effective model that harnesses target-specific evidence.

To investigate the generality of a machine-learning approach to selection, in Chapter 7 we focused on a more traditional federated search environment, associated with result-

¹This was due partly because many of our verticals were topically-focused (e.g., *autos*, *games*, *health*, *movies*, *sports*, *travel*). However, we expect this to be the case in many aggregated search environments.

type and retrieval-algorithm homogeneity. There, we found that integrating non-content based evidence, for example, derived from click-through data, is particularly important when resource representations are poor. That is, when resource sample sets (used by content-based methods such as CORI [14], GAVG [86], and ReDDE [94]) are relatively small for those collections with relevant content. In practice, we may not know when resource representations are poor. Or worse, even if we do, sampling restrictions may prevent us from constructing better ones. We showed, however, that when given access to multiple sources of evidence, a machine-learning approach is capable of shifting its focus between content-based features (derived from sampled content) and non-content-based features (derived from other information) depending on the *actual* resource representation quality. Compared to content-based methods, a feature-integration machine learning approach is more robust.

Additionally, in Chapter 7, we show that at least in a homogeneous environment, a machine-learning approach can be trained without using human-produced training data of any kind. Our *zero-judgement* training method harvests training data from retrievals that merge content from every available resource. A machine-learning method was trained to select those resource whose merged ranking approximates one that merges content from all. This training approach may be particularly useful in a dynamic environment, where changes in resource content and user interest might reduce the effectiveness of an already-trained model. Assuming access to recent queries (representative of current interests), a fresh new set of training queries can be harvested at *zero* editorial cost.

8.1.2 Results Presentation

When resources retrieve similar types of results, an appropriate presentation strategy is to interleave results from different resources in an unconstrained fashion. In an aggregated web search environment, however, *result-type heterogeneity* imposes a number of layout constraints on the final presentation. One constraint, for instance, is that same-vertical results must appear grouped together—vertically (e.g., *news*) or horizontally (e.g., *images*)—in the aggregated results. In the absence of layout constraints, existing methods cast the results presentation task as retrieval score normalization: transforming scores from different resources so that they are directly comparable and can be used to induce a merged ranking. These methods are not well-suited for aggregated web search. Even if we had some way of incorporating layout constraints, some verticals, particularly those that behave like a database look-up (e.g., *weather*, *finance*, *movie times*), are not associated with a retrieval score. A major contribution of the dissertation is the formulation of the vertical results presentation task as *block-ranking*. In this formulation, the goal is to order *sequences* of Web and vertical results that must appear grouped together in the final presentation.

Based on this formulation of the presentation task, in Chapter 5, we proposed and empirically validated a methodology for evaluating a particular presentation of results (i.e., a particular ranking of Web and vertical result-blocks). We found that, in most

cases, non-expert, inexpensive assessors are capable of making preferential judgements between pairs of blocks and that these pairwise judgements can be used to derive a ground truth or *reference* presentation. Our evaluation approach, then, is to score alternative presentations (i.e., alternative block-rankings) based on their distance (using an existing rank-based distance metric [64]) to the reference. A user study demonstrated that when assessors (strongly or unanimously) prefer a particular block-ranking over another, the metric also scores the preferred block-ranking as superior.

Finally, in Chapter 6, we investigated machine learning approaches to block-ranking. The methodology from Chapter 5 was used for model learning and evaluation. In other words, models were trained to produce output that approximates the *reference* block-ranking and were evaluated based on the quality of their approximation on unseen queries. Consistent with the rest of the dissertation, models were trained to predict a block's rank as a function of a set of features. These experiments show several important results.

The best-performing models were those that allow the algorithm to learn a *vertical-specific* relationship between features and block relevance. In other words, even if a feature is common to multiple verticals, the best models do not assume that the feature is *equally* correlated with relevance for different verticals. We presented two general approaches that can exploit a vertical-specific relationship between features and block relevance. One alternative is to learn and combine vertical-specific classifiers in order to produce a final block-ranking. A second alternative is to learn a *single* model by casting the task as a learning-to-rank (LTR) problem. The challenge with this second approach, however, is that LTR models assume that each feature has a consistent predictive relationship (with rank) across all elements being ranked (in our case, across blocks from different verticals). We show that a vertical-specific predictive relationship can be learned by making “copies” of each feature (one per vertical) so that the model can assign a different weight to different copies, depending on whether the feature has a positive or negative correlation (or no correlation) with a particular vertical's relevance. The importance of learning a verticals-specific relationship between features and vertical relevance is consistent with the results from Chapter 3, where the query-category was the most predictive source of evidence for vertical selection.

Another major finding in this work is the importance of *post-retrieval* evidence. During vertical selection, the task is to predict which verticals are relevant using only *pre-retrieval* evidence (i.e., without issuing the query to the vertical). During vertical results presentation, however, the assumption is that the query has already been issued to those verticals selected. Thus, post-retrieval features can be derived from the vertical search engine results, including those actually presented in the vertical block. Our results show that post-retrieval features contribute significantly to performance. In fact, they were the *only* features to not hurt performance for any vertical and improve performance for several. Moreover, in Chapter 3, *news* was one of the worst-performing verticals. Given access to post-retrieval features, however, it was one of the best-performing.

This result has important implications for end-to-end aggregated web search. For

some verticals, the decision to present or suppress the vertical is better informed using post-retrieval evidence. While this may not be surprising, the magnitude of the improvement is considerable for some verticals (e.g., an 86% improvement for *news*). While post-retrieval evidence never hurts performance, it does not help every vertical. For example, it had little impact for *finance* and *images*. To improve the end-to-end results (and maintain efficiency), system designers might consider identifying those verticals for which post-retrieval features are the most useful and tuning the vertical selection accordingly so that post-retrieval evidence can be generated for those verticals more often. A second alternative is to develop a framework for caching post-retrieval features for future impressions of the query or queries similar to it.

8.2 Thesis Contributions

As the field of Information Retrieval has progressed, researchers and commercial search providers have considered a wider range of search tasks. A consistent finding in empirical IR research is that different search tasks require specialized solutions. Different types of media require different representations. Different definitions of relevance require different search algorithms. Different user objectives require different ways of presenting results and different user interactions. As a result, in recent years we have seen an explosion of highly customized search services. In the context of Web search, these include search services for news, images, videos, local businesses, weather forecasts, driving directions, on-line discussions, items for sale, digitized books, scientific publications, and, more recently, even social-media interactions. This gives rise to a new challenge: How do we provide users with integrated access to all these diverse search services within a *single* search interface? This is the goal of *aggregated web search* and the main focus on this dissertation.

Aggregated web search can be viewed as a type of *federated search*, which has been studied extensively for the past two decades. However, as this dissertation shows, existing federated search solutions are not well-suited for an environment where resources retrieve very different types of results (*result-type heterogeneity*) and use very different retrieval algorithms (*retrieval algorithm heterogeneity*). This dissertation contributes new approaches to the two main tasks associated with aggregated web search: *vertical selection* and *vertical results presentation*. We show through extensive experimentation that the proposed methods are effective, robust, and generalize across federated search environments. Not only do they outperform existing approaches in a heterogeneous environment, they also outperform these in a more homogeneous federated search environment, for which existing methods were designed.

At the core of our proposed methods is the use of supervised machine learning as a means for (1) integrating different types of features and (2) learning a predictive relationship between features and vertical relevance. We believe that this new approach to federated search benefits the field in several meaningful ways. First, it places a greater emphasis on feature engineering. This is advantageous because it encourages division

of labor, which may result in better features and faster system development. Second, diversifying across features is a way of mitigating risk. Given enough high-quality training data, ineffective features should not completely devastate system performance. This added robustness may encourage the development of new types of features. Finally, while our feature-integration approach differs from existing federated search techniques, which focus on a single type of evidence, it does not ignore them. Two decades worth of federated search research has produced many theoretically-grounded resource selection methods that are highly effective in certain environments. In our approach, these can be used as input features in the same way that BM25 can be used as a feature in machine-learned document ranking. A feature-integration approach *generalizes* existing approaches.

Throughout the dissertation, we investigate various different types of features and sources of training data. We believe this is an important contribution. Prior federated search research focused almost exclusively on content-based evidence. Additionally, we show that training data can be produced, not only by expert assessors, but also non-expert, inexpensive Amazon Mechanical Turk workers. Most prior federated search research simulated a federated search environment by partitioning TREC corpora associated with a judged set of queries (usually judged by expert assessors).

While domain adaptation has been applied to various other learning problems, we present the first investigation of domain adaptation for the task of vertical selection. This investigation revealed that certain features (particularly those generated from the vertical) generalize well across verticals and can therefore be applied to a new vertical with no training data. Furthermore, they can be used to discover highly predictive vertical-*specific* features. The dissertation contributes algorithms that can re-use training data, which is expensive and time-consuming to produce. Additionally, our methods may help better allocate resources for the production of new training data. For example, after adapting a model to a new vertical, assessors can focus on annotating low-confidence queries. Alternatively, it may be more cost-effective to have assessors identify vertical-specific features, as we found those to be particularly predictive and non-portable.

While most prior work in federated search recognized the interdependence between resource selection and results merging, no work to date has exploited this interdependence to improve end-to-end performance. Our results show that there is much to gain from sharing information between these two different sub-tasks. In a homogeneous environment, retrievals that merge content from *all* available resources can be used to generate training data for resource selection. In aggregated web search, a trained vertical results presentation model can re-evaluate vertical selection decisions in light of post-retrieval evidence (derived from the vertical results). This may be a potential source of training data for vertical selection. In other words, if the vertical results presentation component suppresses a vertical based on post-retrieval evidence, this is evidence that the vertical selection component should *not* have selected it based on pre-retrieval evidence. Future research should continue investigating ways in which decisions made by one component can be a source of (unsupervised) training data for the other.

Formulating the vertical results presentation task as *block-ranking* plays two important roles in the dissertation. First, it facilitates evaluation. Our methodology is to derive a *reference* presentation for the query and to evaluate alternative presentations based on their distance (or similarity) to the *reference*. Second, it facilitates model learning. Models can be trained to produce output that approximates the *reference*. Our evaluation methodology has several advantages, which we believe will encourage other researchers to work on aggregated web search. First, it facilitates (off-line) test-collection-based evaluation. Relevance assessments and cached features can be easily distributed among researchers and alternative approaches can be directly compared. Most prior research in aggregated web search was conducted in a commercial setting, with a live system and millions of users to provide implicit feedback. On-line evaluations (using implicit feedback) are more difficult in an academic setting. Second, our evaluation methodology is inexpensive, therefore new evaluation testbeds can be developed at a reasonable cost.

Other aggregation tasks have similar layout constraints as those we defined for the vertical results presentation task. Examples include news story aggregation, aggregating content in portals like AOL, and recommending search-assistance tools to users. In all these cases, the system can assume a fixed presentation template (i.e., a fixed set of slots) and the problem can be formulated as deciding which content to surface and where to display it. Thus, our block-ranking methods and our evaluation methodology may encourage new research in these areas, especially in non-commercial environments.

8.3 New Directions

Within the area of aggregated web search, there are two major problems we did not address. First, advancing the use of supervised approaches to selection and presentation requires methods that can maintain performance given changes in vertical content and user interests. Second, one of our assumptions (which is also made by all aggregated web search work to date) is that verticals behave independently. We believe that overall performance can be substantially improved by sharing evidence across vertical results.

The work presented in this dissertation is important because it may facilitate research in other IR applications with similar assumptions and a similar task formulation. We propose work on two areas which have large potential and are currently not well understood.

8.3.1 Aggregated Search with Dynamic Content and User Interests

The aggregated web search environment is dynamic. The environment can change in three ways: a new vertical can be added to the set of candidate verticals, content can be added to (or removed from) a particular vertical, and the interests of the user population may drift, effectively changing the distribution of queries input to the system. In Chapter 4, we focus on the first type of change—the introduction of a new vertical (with no training data) to the set of candidate verticals (with training data). Further advancing

the use of supervised methods for vertical selection and presentation requires approaches that can handle the other two changes in the environment. Prior work confirmed that a change in resource content can degrade the performance of an already-tuned resource selector [93].

In machine learning, the problem of *concept drift* occurs when the relationship between features and the target class changes over time [84]. One approach, for example, is to maintain a running ensemble of classifiers, which are re-weighted, pruned, and/or augmented with new ensemble members based on new, more recent training examples [62]. Future work might consider selection and presentation approaches that can adapt to concept drift without access to new training examples. Just as Chapter 4 found that some features are portable across verticals (and therefore also predictive for a new vertical with no training data), we may find that some features are portable for the same vertical across time-periods. For example, the presence of the query-term “news” will likely be a portable feature across time-periods for *news*. A model trained on these time-portable features may be able to produce high-precision/low-recall predictions that can be used to discover features that are specifically predictive for the vertical, but only for the current time-period.

8.3.2 Improving the Coherence between Cross-Vertical Results

Throughout the dissertation, we made the assumption that verticals operate fairly independently from each other. In fact, in our formulation of the *vertical results presentation* task, we explicitly assume that the task is not to predict which results from the vertical to present, but rather to predict whether to present the vertical given its results (and if so, where). There are two issues with having verticals operate independently from each other. First, different verticals may focus on different interpretations of the query, making the aggregated results seem fragmented and incoherent. Second, while the query may be a difficult query for one vertical, it may be an easy query for another. Thus, the results from a confident vertical are a source of evidence for improving the results from a less confident vertical. In this sense, having verticals operate independently is a missed opportunity.

Figure 8.1 shows results from two verticals predicted relevant to the query “joplin” by a commercial search provider. At the time the query was issued, a tornado had recently hit the community of Joplin, Missouri. While the *news* vertical focuses on this event, the *images* vertical focuses on a different sense of the query (the singer Janis Joplin). As it turns out, each vertical focused on the query sense that is most prevalent in its collection.² One might argue, however, that the disproportionate amount of content in the news index about Joplin, Missouri provides evidence of a new query sense that is suddenly important and should be reflected in the *images* results. A better retrieval might have presented images from both senses.

²At the time, the *news* vertical had 13,900 hits for “joplin missouri” and 445 hits for “janis joplin”. On the other hand, the *images* vertical had 1,430,00 hits for “joplin missouri” and 3,050,000 hits for “janis joplin”.

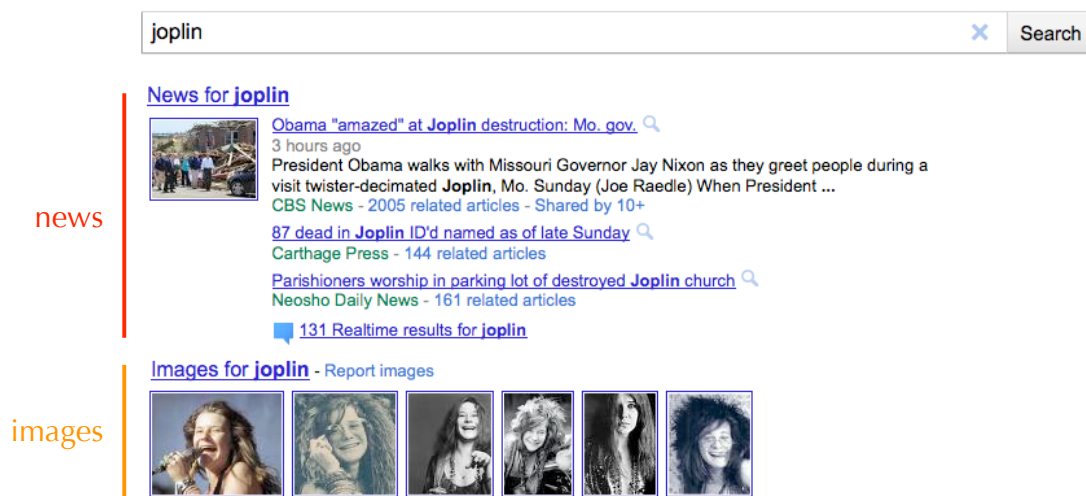


Figure 8.1: A commercial search engine presents *news* and *images* results in response the query “joplin”.

The idea is that we want to bias results from some verticals (predicted ineffective) to be similar to those from other verticals (predicted effective). This requires addressing two questions. First, how do we detect that a vertical produced an effective retrieval? Several (automatic) retrieval effectiveness predictors have been proposed in prior work, which focus, for example, on properties of the top-ranked documents [25, 27] or on the rank-stability [116, 120]. The second question is, given an effective vertical retrieval, how can we bias other vertical retrievals to be similar to it? One approach may be query-expansion—generating expansion terms from some verticals and re-submitting the query to the others. A second approach might be to bias results from different verticals to be close in a hyperlink graph.³

8.3.3 Aggregated Mobile Search

Currently, commercial mobile search providers support many of the same verticals known to desktop searchers.⁴ However, *aggregated mobile search* is associated with unique challenges and opportunities, which are not well understood. The solutions presented in this dissertation may provide a framework for, not only developing aggregated mobile search solutions, but also learning more about the problem.

One major opportunity in aggregated mobile search is the availability of rich information about the user’s current context (e.g., their local time, location and movement) and, possibly, even their current activity (e.g., are they likely at home, at work, in an unfamiliar location, or traveling?). Clues about the user’s activity can be provided by

³Often, vertical content is harvested from the Web and can therefore be associated with one or several nodes in a Web hyperlink graph.

⁴For a demonstration, see: <http://mobile.yahoo.com/search>

historical contextual data. IR researchers have long advocated that user context (including time and location) should play a more active role in search [49]. In aggregated mobile search, for example, weather results may be more relevant in the morning, maps results may be more relevant if the user is moving, traffic information may be more relevant during rush hours, and movie times may be more relevant in the evenings.

In addition to contextual information, aggregated mobile search is also unique in the way that users interact with the search results. For example, local results often present the business's phone number, which the user can click to directly place a call. This is a stronger signal of relevance than a traditional click. Also, the ability to scroll, pinch-and-zoom, and rotate (coupled with the reduced screen size) means that the system always knows precisely where the user is looking. The evidence-integration approaches presented in this dissertation may provide a framework for investigating the role of contextual information and implicit feedback signals in aggregated mobile search. Does contextual information help? Are some feedback signals more predictive than others? Are there confounding effects between the user's context and the predictiveness of a particular feedback signal?

8.3.4 Search Tool Recommendation

Just as different verticals support different search goals, different search tools and interventions are designed to assist searchers in different scenarios. For example, if a search is ineffective, the user may benefit from seeing query suggestions or potential query-expansion terms [59]. If a query is too general, a clustering of results may assist the user in narrowing their search [47, 58]. Facet-value pair recommendation can also help users narrow their search when there is evidence that document meta-data can effectively isolate the relevant set [118]. If a search is largely exploratory, the user may benefit from seeing search trails from other users who performed a similar search and identified useful content [73]. During long sessions, note-taking applications can help users keep track of search results [30]. The goal of all these search tools and interactions is to guide users towards more successful searches. The challenge, however, is that they are not always appropriate.

The problem of search tool recommendation can be cast as an aggregated search problem, where the goal is to resolve contention between different interventions rather than different verticals. Several factors affect the relevance of a particular intervention. First, different interventions are appropriate in different user scenarios, as motivated above. Second, the usefulness of a particular intervention is affected by the information available to the system (query suggestions should not be presented without evidence of their effectiveness). Third, different users may prefer different search tools. The feature-integration methods presented in the dissertation may provide a framework for addressing the problem of search-assistance tool recommendation. Potentially useful sources of predictive evidence include pre- and post-retrieval effectiveness predictors [46], user interaction data which can predict the user's intent [44] or level of frustration [36], session-level evidence [111], and user-preference information.

Bibliography

- [1] Tony Abou-Assaleh and Weizheng Gao. Geographic ranking for a local search engine. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 911–911, New York, NY, USA, 2007. ACM. 1, 1.2
- [2] Jaime Arguello, Jamie Callan, and Fernando Diaz. Classification-based resource selection. In *Proceeding of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 1277–1286, New York, NY, USA, 2009. ACM. 4.3.1, 4.5.4, 1
- [3] Jaime Arguello, Fernando Diaz, Jamie Callan, and Jean-François Crespo. Sources of evidence for vertical selection. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 315–322, New York, NY, USA, 2009. ACM. 1, 4.3.1, 4.5.4, 4.7
- [4] Jaime Arguello, Fernando Diaz, and Jean-François Paiement. Vertical selection in the presence of unlabeled verticals. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 691–698, New York, NY, USA, 2010. ACM. 1, 4.7
- [5] Jaime Arguello, Fernando Diaz, Jamie Callan, and Ben Carterette. A methodology for evaluating aggregated search results. In *Advances in Information Retrieval*, volume 6611 of *Lecture Notes in Computer Science*, pages 141–152. Springer Berlin / Heidelberg, 2011. 1
- [6] Steven M. Beitzel, Eric C. Jensen, Ophir Frieder, David D. Lewis, Abdur Chowdhury, and Aleksander Kolcz. Improving automatic query classification via semi-supervised learning. In *Proceedings of the Fifth IEEE International Conference on Data Mining*, ICDM '05, pages 42–49, Washington, DC, USA, 2005. IEEE Computer Society. 2.3.7
- [7] A. Bhattacharyya. On a measure of divergence between two statistical populations defined by probability distributions. *Bulletin of the Calcutta Mathematical Society*, 35: 99 – 109, 1943. 3.3.2, 7.4.5
- [8] Daniel M. Bikel, Richard Schwartz, and Ralph M. Weischedel. An algorithm that learns what is in a name. *Machine Learning*, 34:211–231, 1999. 6.3.1

- [9] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, EMNLP '06*, pages 120–128, Stroudsburg, PA, USA, 2006. ACL. 4.2
- [10] John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, ACL '07*, pages 187–205, Stroudsburg, PA, USA, 2007. ACL. 4.2
- [11] Leo Breiman and Jerome H. Friedman. Predicting multivariate responses in multiple linear regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 59(1):3–54, 1997. 4.2
- [12] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the seventh international conference on World Wide Web 7, WWW7*, pages 107–117, Amsterdam, The Netherlands, The Netherlands, 1998. Elsevier Science Publishers B. V. 1.2
- [13] Andrei Broder, Massimiliano Ciaramita, Marcus Fontoura, Evgeniy Gabrilovich, Vanja Josifovski, Donald Metzler, Vanessa Murdock, and Vassilis Plachouras. To swing or not to swing: learning when (not) to advertise. In *Proceeding of the 17th ACM conference on Information and knowledge management, CIKM '08*, pages 1003–1012, New York, NY, USA, 2008. ACM. 3.7
- [14] James P. Callan, Zhihong Lu, and W. Bruce Croft. Searching distributed collections with inference networks. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '95*, pages 21–28, New York, NY, USA, 1995. ACM. 1.2, 2.3.2, 2.3.5, 2.3.8, 2.4, 2.5, 3, 7, 7.3.1, 7.3.1, 8.1.1
- [15] Jamie Callan. Distributed information retrieval. In W. B. Croft, editor, *Advances in Information Retrieval*, pages 127–150. Kluwer Academic Publishers, 2000. 1
- [16] Jamie Callan and Margaret Connell. Query-based sampling of text databases. *ACM Transactions on Information Systems*, 19:97–130, 2001. 2.2, 2.2.1, 3.3.1, 3.3.1, 7.4.4
- [17] Jaime Carbonell and Jade Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '98*, pages 335–336, New York, NY, USA, 1998. ACM. 6.8
- [18] Ben Carterette and Paul N. Bennett. Evaluation measures for preference judgments. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '08*, pages 685–686, New York, NY, USA, 2008. ACM. 5.4

- [19] Rich Caruana. Multitask learning. *Machine Learning*, 28:41–75, July 1997. 4.2
- [20] James Caverlee, Ling Liu, and Joonsoo Bae. Distributed query sampling: A quality-conscious approach. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 340–347, New York, NY, USA, 2006. ACM. 2.2.1
- [21] Keke Chen, Rongqing Lu, C. K. Wong, Gordon Sun, Larry Heck, and Belle Tseng. Trada: Tree based ranking function adaptation. In *Proceeding of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 1143–1152, New York, NY, USA, 2008. ACM. 4.2, 4.3.3
- [22] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960. 5.6.1
- [23] Jacob Cohen. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological Bulletin*, 70(4):213–220, 1968. 5.6.1
- [24] Nick Craswell, Peter Bailey, and David Hawking. Server selection on the world wide web. In *Proceedings of the fifth ACM conference on Digital libraries*, DL '00, pages 37–46, New York, NY, USA, 2000. ACM. 2.2.1, 3.3.1
- [25] Steve Cronen-Townsend, Yun Zhou, and W. Bruce Croft. Predicting query performance. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '02, pages 299–306, New York, NY, USA, 2002. ACM. 3.3.1, 3.8, 8.3.2
- [26] Hal Daume III. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263. ACL, 2007. 4.2
- [27] Fernando Diaz. Performance prediction using spatial autocorrelation. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 583–590, New York, NY, USA, 2007. ACM. 8.3.2
- [28] Fernando Diaz. Integration of news content into web results. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, WSDM '09, pages 182–191, New York, NY, USA, 2009. ACM. 1, 1.2, 5, 3.7, 4.3.1, 5, 5.1.2, 5.2, 6.2, 6.3.2, 6.5.3
- [29] Fernando Diaz and Jaime Arguello. Adaptation of offline vertical selection predictions in the presence of user feedback. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 323–330, New York, NY, USA, 2009. ACM. 3.7, 4.3.1, 5.1.2

- [30] Debora Donato, Francesco Bonchi, Tom Chi, and Yoelle Maarek. Do you want to take notes?: Identifying research missions in yahoo! search pad. In *Proceedings of the 19th international conference on World wide web, WWW '10*, pages 321–330, New York, NY, USA, 2010. ACM. 8.3.4
- [31] Kevin Duh. Ranking vs. regression in machine translation evaluation. In *Proceedings of the Third Workshop on Statistical Machine Translation, StatMT '08*, pages 191–194, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. 6.2
- [32] Kevin K. Duh. *Learning to Rank with Partially-Labeled Data*. PhD thesis, University of Washington, 2009. 6.2
- [33] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi-task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '04*, pages 109–117, New York, NY, USA, 2004. ACM. 4.2
- [34] Christopher T. Fallen and Gregory B. Newby. Distributed web search efficiency by truncating results. In *Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries, JCDL '07*, pages 195–203, New York, NY, USA, 2007. ACM. 7.4.1
- [35] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, June 2008. 3.2
- [36] Henry A. Feild, James Allan, and Rosie Jones. Predicting searcher frustration. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval, SIGIR '10*, pages 34–41, New York, NY, USA, 2010. ACM. 8.3.4
- [37] J.L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382, 1971. 5.6.1
- [38] Jerome H. Friedman. Gradient function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 1999. 4.3, 4.5, 6.2, 6.7.4
- [39] Wei Gao, Peng Cai, Kam-Fai Wong, and Aoying Zhou. Learning to rank only using training data from related domain. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval, SIGIR '10*, pages 162–169, New York, NY, USA, 2010. ACM. 4.2
- [40] Luis Gravano, Héctor García-Molina, and Anthony Tomasic. The effectiveness of GIOSS for the text database discovery problem. In *Proceedings of the 1994 ACM SIGMOD international conference on Management of data, SIGMOD '94*, pages 126–137, New York, NY, USA, 1994. ACM. 2.3.3

- [41] Luis Gravano, Chen-Chuan K. Chang, Héctor García-Molina, and Andreas Paepcke. Starts: Stanford proposal for internet meta-searching. In *Proceedings of the 1997 ACM SIGMOD international conference on Management of data, SIGMOD '97*, pages 207–218, New York, NY, USA, 1997. ACM. 2.1
- [42] Luis Gravano, Héctor García-Molina, and Anthony Tomasic. Gloss: Text-source discovery over the internet. *ACM Transactions on Database Systems*, 24:229–264, 1999. 2.3.3
- [43] Honglei Guo, Huijia Zhu, Zhili Guo, Xiaoxun Zhang, Xian Wu, and Zhong Su. Domain adaptation with latent semantic association for named entity recognition. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, NAACL '09*, pages 281–289, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. 4.2
- [44] Qi Guo and Eugene Agichtein. Ready to buy or just browsing?: Detecting web searcher goals from interaction data. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval, SIGIR '10*, pages 130–137, New York, NY, USA, 2010. ACM. 3.7, 8.3.4
- [45] Qi Guo, Eugene Agichtein, Charles L. A. Clarke, and Azin Ashkan. In the mood to click? Towards inferring receptiveness to search advertising. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 01, WI-IAT '09*, pages 319–324, Washington, DC, USA, 2009. IEEE Computer Society. 3.7
- [46] Claudia Hauff, Diane Kelly, and Leif Azzopardi. A comparison of user and system query performance predictions. In *Proceedings of the 19th ACM international conference on Information and knowledge management, CIKM '10*, pages 979–988, New York, NY, USA, 2010. ACM. 8.3.4
- [47] Marti A. Hearst and Jan O. Pedersen. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '96*, pages 76–84, New York, NY, USA, 1996. ACM. 8.3.4
- [48] Dzung Hong, Luo Si, Paul Bracke, Michael Witt, and Tim Juchcinski. A joint probabilistic classification model for resource selection. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval, SIGIR '10*, pages 98–105, New York, NY, USA, 2010. ACM. 2.3.8, 6.8
- [49] Peter Ingwersen and Nick Belkin. Information retrieval in context - IRiX: workshop at SIGIR 2004 - Sheffield. *SIGIR Forum*, 38:50–52, December 2004. 8.3.3
- [50] Panagiotis G. Ipeirotis and Luis Gravano. Distributed search over the hidden web: Hierarchical database sampling and selection. In *Proceedings of the 28th international*

- conference on Very Large Data Bases, VLDB '02, pages 394–405. VLDB Endowment, 2002. 1.1, 2.2.4, 2.3.5
- [51] Panagiotis G. Ipeirotis and Luis Gravano. When one sample is not enough: Improving text database selection using shrinkage. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, SIGMOD '04, pages 767–778, New York, NY, USA, 2004. ACM. 2.3.5
- [52] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems*, 20:422–446, 2002. 4.5.3
- [53] Harold Jeffreys. An invariant form for the prior probability in estimation problems. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 186(1007):453–461, September 1946. 6.3.1, 7.4.1
- [54] Jing Jiang. *Domain Adaptation in Natural Language Processing*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana-Champaign, IL, 2008. 4.2
- [55] Jing Jiang and Chengxiang Zhai. Instance weighting for domain adaptation in nlp. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, ACL '07, pages 264–271, Stroudsburg, PA, USA, 2007. ACL. 4.2
- [56] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 133–142, New York, NY, USA, 2002. ACM. 1.2, 6.2, 6.3.2, 6.4.3
- [57] In-Ho Kang and GilChang Kim. Query type classification for web document retrieval. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR '03, pages 64–71, New York, NY, USA, 2003. ACM. 2.3.7
- [58] Weimao Ke, Cassidy R. Sugimoto, and Javed Mostafa. Dynamicity vs. effectiveness: Studying online clustering for scatter/gather. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 19–26, New York, NY, USA, 2009. ACM. 8.3.4
- [59] Diane Kelly, Karl Gyllstrom, and Earl W. Bailey. A comparison of query and term suggestion features for interactive searching. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 371–378, New York, NY, USA, 2009. ACM. 8.3.4
- [60] Jinyoung Kim and W. Bruce Croft. Ranking using multiple document types in desktop search. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 50–57, New York, NY, USA, 2010. ACM. 1.3

- [61] Soo-Min Kim, Patrick Pantel, Lei Duan, and Scott Gaffney. Improving web page classification by label-propagation over click graphs. In *Proceeding of the 18th ACM conference on Information and knowledge management, CIKM '09*, pages 1077–1086, New York, NY, USA, 2009. ACM. 4.3
- [62] J. Zico Kolter and Marcus A. Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. *Journal of Machine Learning Research*, 8:2755–2790, December 2007. 8.3.1
- [63] Arnd Christian König, Michael Gamon, and Qiang Wu. Click-through prediction for news queries. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, SIGIR '09*, pages 347–354, New York, NY, USA, 2009. ACM. 3.7, 5, 5.1.2, 6.2
- [64] Ravi Kumar and Sergei Vassilvitskii. Generalized distances between rankings. In *Proceedings of the 19th international conference on World wide web, WWW '10*, pages 571–580, New York, NY, USA, 2010. ACM. 5.4.3, 5.7, 6.1, 6.5.4, 8.1.2
- [65] J. R. Landis and G. G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174, 1977. 5.6.1, 5.6.2, 6.5.2
- [66] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the fourth ACM international conference on Web search and data mining, WSDM '11*, pages 297–306, New York, NY, USA, 2011. ACM. 5.1.2
- [67] Xiao Li, Ye-Yi Wang, and Alex Acero. Learning query intent from regularized click graphs. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '08*, pages 339–346, New York, NY, USA, 2008. ACM. 2.3.7, 3.7, 5, 5.1.2
- [68] Chih-Jen Lin, Ruby C. Weng, and S. Sathiya Keerthi. Trust region newton methods for large-scale logistic regression. In *Proceedings of the 24th international conference on Machine learning, ICML '07*, pages 561–568, New York, NY, USA, 2007. ACM. 3.2, 6.4.1
- [69] Yi Lin, Yoonkyung Lee, and Grace Wahba. Support vector machines for classification in nonstandard situations. *Machine Learning*, 46:191–202, 2002. 4.2
- [70] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Informaton Retrieval*, 3:225–331, 2009. 6.3.2
- [71] Yuanhua Lv, Taesup Moon, Pranam Kolari, Zhaohui Zheng, Xuanhui Wang, and Yi Chang. Learning to model relatedness for news recommendation. In *Proceedings of the 20th international conference on World wide web, WWW '11*, pages 57–66, New York, NY, USA, 2011. ACM. 6.2

- [72] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. 6.3.1
- [73] Meredith Ringel Morris and Eric Horvitz. Searchtogether: An interface for collaborative web search. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*, UIST '07, pages 3–12, New York, NY, USA, 2007. ACM. 8.3.4
- [74] Vanessa Murdock and Mounia Lalmas, editors. *SIGIR 2008 Workshop on Aggregated Search*, New York, NY, USA, 2008. ACM. 1, 3.7
- [75] Aditya Pal and Jaya Kawale. Leveraging query associations in federated search. In *Proceedings of the SIGIR 2008 Workshop on Aggregated Search*, 2008. 3.7
- [76] Georgios Paltoglou, Michail Salampasis, and Maria Satratzemi. Integral based source selection for uncooperative distributed information retrieval environments. In *Proceeding of the 2008 ACM workshop on Large-Scale distributed systems for information retrieval*, LSDS-IR '08, pages 67–74, New York, NY, USA, 2008. ACM. 2.2.1, 3.3.1
- [77] Ashok Kumar Ponnuswami, Kumaresh Pattabiraman, Desmond Brand, and Tapas Kanungo. Model characterization curves for federated search using click-logs: predicting user engagement metrics for the span of feasible operating points. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 67–76, New York, NY, USA, 2011. ACM. 5.1.2, 6.2
- [78] Ashok Kumar Ponnuswami, Kumaresh Pattabiraman, Qiang Wu, Ran Gilad-Bachrach, and Tapas Kanungo. On composition of a federated web search result page: Using online users to provide pairwise preference for heterogeneous verticals. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 715–724, New York, NY, USA, 2011. ACM. 5, 5, 5.1.2, 6.2, 6.5.3
- [79] M. F. Porter. An algorithm for suffix stripping. *Readings in information retrieval*, pages 313–316, 1997. 7.4.1
- [80] Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. Letor: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval*, 13: 346–374, August 2010. ISSN 1386-4564. 1.2, 6.2
- [81] Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. Learning diverse rankings with multi-armed bandits. In *Proceedings of the 25th international conference on Machine learning*, pages 784–791, 2008. 6.8
- [82] Mark Sanderson, Monica Lestari Paramita, Paul Clough, and Evangelos Kanoulas. Do user preferences and evaluation measures line up? In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 555–562, New York, NY, USA, 2010. ACM. 5.1.2, 5.5.1

- [83] Sandeepkumar Satpal and Sunita Sarawagi. Domain adaptation of conditional probability models via feature subsetting. In *Proceedings of the 11th European conference on Principles and Practice of Knowledge Discovery in Databases, PKDD 2007*, pages 224–235, Berlin, Heidelberg, 2007. Springer-Verlag. 4.2
- [84] J.C. Schlimmer and R.H. Granger. Beyond incremental processing: Tracking concept drift. In *Proceedings of the 5th National Conference on Artificial Intelligence, AAAI '86*, pages 502–507. AAAI, 1986. 8.3.1
- [85] Markus Schulze. A new monotonic, clone-independent, reversal symmetric, and condorcet-consistent single-winner election method. *Social Choice and Welfare*, 36: 267–303, 2011. 5.4.2, 5.7, 6.4.2
- [86] Jangwon Seo and W. Bruce Croft. Blog site search using resource selection. In *Proceeding of the 17th ACM conference on Information and knowledge management, CIKM '08*, pages 1053–1062, New York, NY, USA, 2008. ACM. 2.3.2, 2.3.8, 4.7, 7.3.1, 7.3.1, 8.1.1
- [87] Mounia Lalmas Shanu Sushmita, Hideo Joho and Joemon M. Jose. Understanding domain relevance in web search. In *WWW Workshop on Web Search Result Summarization and Presentation*, 2010. 5.1.1
- [88] Dou Shen, Rong Pan, Jian-Tao Sun, Jeffrey Junfeng Pan, Kangheng Wu, Jie Yin, and Qiang Yang. Q2C@UST: Our winning solution to query classification in KDDCUP 2005. *ACM SIGKDD Exploration Newsletter*, 7:100–110, December 2005. ISSN 1931-0145. 2.3.7, 3.3.1, 3.3.3
- [89] Dou Shen, Jian-Tao Sun, Qiang Yang, and Zheng Chen. Building bridges for web query classification. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '06*, pages 131–138, New York, NY, USA, 2006. ACM. 2.3.7, 3.3.1, 3.3.3
- [90] Milad Shokouhi. Central-rank-based collection selection in uncooperative distributed information retrieval. In *Proceedings of the 29th European conference on IR research, ECIR'07*, pages 160–172, Berlin, Heidelberg, 2007. Springer-Verlag. 1.1, 1.2, 2.3.1, 2.3.4, 3, 3.4.5, 7
- [91] Milad Shokouhi and Justin Zobel. Federated text retrieval from uncooperative overlapped collections. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '07*, pages 495–502, New York, NY, USA, 2007. ACM. 2.2.1, 3.3.1
- [92] Milad Shokouhi, Falk Scholer, and Justin Zobel. Sample sizes for query probing in uncooperative distributed information retrieval. In *Proceedings of the 2006 Asia Pacific Web Conference*, pages 63–75. Springer, 2006. 2.2.1, 2.2.2, 3.3.1, 7.5

- [93] Milad Shokouhi, Mark Baillie, and Leif Azzopardi. Updating collection representations for federated search. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 511–518, New York, NY, USA, 2007. ACM. 1.2, 2.2.3, 3.3.1, 8.3.1
- [94] Luo Si and Jamie Callan. Relevant document distribution estimation method for resource selection. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '03, pages 298–305, New York, NY, USA, 2003. ACM. 1.1, 1.2, 2.2.1, 2.3.3, 2.3.7, 2.5, 3, 3.3.1, 3.4.5, 3.8, 7, 7.4.5, 8.1.1
- [95] Luo Si and Jamie Callan. Unified utility maximization framework for resource selection. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, CIKM '04, pages 32–41, New York, NY, USA, 2004. ACM. 1.1, 1.2, 2.2.1, 2.3.4, 3, 3.3.1, 7
- [96] Luo Si and Jamie Callan. Modeling search engine effectiveness for federated search. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 83–90, New York, NY, USA, 2005. ACM. 1.1, 1.2, 2.2.1, 2.3.6, 3, 3.3.1, 7
- [97] Luo Si and Jamie Callan. A semisupervised learning method to merge search engine results. *ACM Transactions on Information Systems*, 21:457–491, 2003. 2.4, 2.5
- [98] Luo Si, Rong Jin, Jamie Callan, and Paul Ogilvie. A language modeling framework for resource selection and results merging. In *Proceedings of the eleventh international conference on Information and knowledge management*, CIKM '02, pages 391–397, New York, NY, USA, 2002. ACM. 1.2, 2.2, 2.3.2, 2.3.7, 3, 7
- [99] Yang Song, Nam Nguyen, Li-wei He, Scott Imig, and Robert Rounthwaite. Searchable web sites recommendation. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 405–414, New York, NY, USA, 2011. ACM. 3.7
- [100] Shanu Sushmita, Hideo Joho, and Mounia Lalmas. A task-based evaluation of an aggregated search interface. In *Proceedings of the 16th International Symposium on String Processing and Information Retrieval*, SPIRE '09, pages 322–333, Berlin, Heidelberg, 2009. Springer-Verlag. 5.1.1, 5.1.2, 5.7
- [101] Shanu Sushmita, Hideo Joho, Mounia Lalmas, and Robert Villa. Factors affecting click-through behavior in aggregated search interfaces. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 519–528, New York, NY, USA, 2010. ACM. 5, 5.1.1, 5.2, 5.7, 6.5.3
- [102] Shanu Sushmita, Benjamin Piwowarski, and Mounia Lalmas. Dynamics of genre and domain intents. In *Proceedings of the 6th Asia Information Retrieval Societies*

Conference, AAIRS '10, pages 399–409, Berlin, Heidelberg, 2010. Springer-Verlag. 5.1.1

- [103] Paul Thomas and David Hawking. Evaluation by comparing result sets in context. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, CIKM '06, pages 94–101, New York, NY, USA, 2006. ACM. 5.1.2
- [104] Paul Thomas and Milad Shokouhi. SUSHI: Scoring scaled samples for server selection. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 419–426, New York, NY, USA, 2009. ACM. 1.1, 1.2, 2.2.1, 2.3.4, 3, 3.3.1, 3.4.5, 7
- [105] Sebastian Thrun and Lorien Pratt, editors. *Learning to learn*. Kluwer Academic Publishers, Norwell, MA, USA, 1998. 4.2
- [106] Ellen M. Voorhees, Narendra K. Gupta, and Ben Johnson-Laird. Learning collection fusion strategies. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '95, pages 172–179, New York, NY, USA, 1995. ACM. 2.3.6
- [107] Stephen Wan, Cecile Paris, and Alexander Krumpholz. From aggravated to aggregated search: Improving utility through coherent organisations of an answer space. In *Proceedings of the SIGIR 2008 Workshop on Aggregated Search*, 2008. 3.7
- [108] Ji-Rong Wen, Jian-Yun Nie, and Hong-Jiang Zhang. Query clustering using content words and user feedback. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '01, pages 442–443, New York, NY, USA, 2001. ACM. 2.3.7
- [109] Jianshu Weng, Ee-Peng Lim, Jing Jiang, and Qi He. Twitterrank: Finding topic-sensitive influential twitterers. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pages 261–270, New York, NY, USA, 2010. ACM. 1, 1.2
- [110] Ian H. Witten and Timothy C. Bell. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *Transactions on Information Theory*, 37, 1991. 3.3.2
- [111] Biao Xiang, Daxin Jiang, Jian Pei, Xiaohui Sun, Enhong Chen, and Hang Li. Context-aware ranking in web search. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 451–458, New York, NY, USA, 2010. ACM. 8.3.4
- [112] Jingfang Xu and Xing Li. Learning to rank collections. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 765–766, New York, NY, USA, 2007. ACM. 2.3.8


- [113] Jinxi Xu and W. Bruce Croft. Cluster-based language models for distributed retrieval. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '99, pages 254–261, New York, NY, USA, 1999. ACM. 1.1, 1.2, 2.3.2, 3
- [114] Jun Xu and Hang Li. Adarank: A boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 391–398, New York, NY, USA, 2007. ACM. 6.2
- [115] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, ACL '95, pages 189–196, Stroudsburg, PA, USA, 1995. Association for Computational Linguistics. 4.3.3
- [116] Elad Yom-Tov, Shai Fine, David Carmel, and Adam Darlow. Learning to estimate query difficulty: Including applications to missing content detection and distributed information retrieval. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 512–519, New York, NY, USA, 2005. ACM. 8.3.2
- [117] Jian Zhang, Zoubin Ghahramani, and Yiming Yang. Flexible latent variable models for multi-task learning. *Machine Learning*, 73:221–242, December 2008. 4.2
- [118] Lanbo Zhang and Yi Zhang. Interactive retrieval based on faceted feedback. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 363–370, New York, NY, USA, 2010. ACM. 8.3.4
- [119] Zhaohui Zheng, Keke Chen, Gordon Sun, and Hongyuan Zha. A regression framework for learning ranking functions using relative relevance judgments. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 287–294, New York, NY, USA, 2007. ACM. 4.3
- [120] Yun Zhou and W. Bruce Croft. Ranking robustness: A novel framework to predict query performance. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, CIKM '06, pages 567–574, New York, NY, USA, 2006. ACM. 8.3.2
- [121] Dongqing Zhu and Ben Carterette. An analysis of assessor behavior in crowd-sourced preference judgements. In *SIGIR Workshop on Crowdsourcing for Search Evaluation*, pages 21–26, New York, NY, USA, 2010. ACM. 5, 5.1.1, 5.2

Vertical Blocks

Figures A.1- A.3 show example vertical blocks associated with the 13 verticals used in Chapters 5 and 6.

San diego earthquake - Blog Post Results
[Moderate Quake In Baja Felt In SoCal CBS Los Angeles- News ...](#)
[losangeles.cbslocal.com](#) 2 days ago
[Earthquakes Felt in San Diego County - San Diego Reviews Events ...](#)
[www.sandiegorei.com](#) 1 day ago
[Let's speculate effect of a massive earthquake on RE and CA...](#)
[piggington.com](#) 2 days ago


(a) blogs

Gardening books - Book Results
[Garden Wisdom & Know-How: Everything You Need to Know to Plant, Grow, and Harvest](#)
Rodale Gardening Books and Judy Pray - 2010
[The well-tended perennial garden planting & pruning techniques](#)
Tracy DiSabato-Aust - 2006
[Landscape architecture magazine](#)
American Society of Landscape Architects - 1917

(b) books

How to get spray paint off wood - Q & A Results
Question: How can you get spray paint off wood ? - Susan H 3 years ago
Best Answer: STRIPPER THEN POWER WASH - D.C.
[8 answers](#)
Question: Does anyone know how to remove spray paint off of a wood door?
- Jules 3 years ago
Best Answer: Since it sounds like it is fresh, try washing with a degreaser. They are strong and will take off a lot of things. I would use a brush and scrub with the grooves, not ... - Samm
[3 answers](#)
Question: How can I get spray paint off a wood fence? - Mommysharp08 1 year ago
Best Answer: If it is still relatively new, try a product called GOOF-OFF. Follow the directions on the label. - Baglady
[4 answers](#)

(c) community Q&A

Bank of America C (BAC)

12.05 ▲ +0.48 (+4.15%)
Open: 11.61 Day's Range: 11.60 - 12.05
Volume: 159,770,544 52 Week Range: 10.91 - 19.86
P/E Ratio: N/A MarketCap: 121.5B

(d) finance


Figure A.1: Example vertical blocks.

Nokia 6102 cell phone - Image Results



(e) images

Key west florida fishing trips - Local Results

 [Key West Fishing Adventures](#) 0 reviews
www.fshinkeys.com
(305) 393-0566 - 1801 N Roosevelt Blvd, Key West, FL

[Charters Key West - Family Fishing, Sport Fishing](#) 0 reviews
lethalweaponcharters.com
(305) 744-8225 - 1401 1st St, Key West, FL

[Key West Sheraton Suites](#) 27 reviews
www.sheratonkeywest.com
(305) 292-9800 - 2001 S Roosevelt Blvd, Key West, FL

[Keys to Key West Incorporated](#) 0 reviews
gotothekeys.com
(305) 292-7702 - 5555 College Rd, #202, Key West, FL

(f) local

Maps of puerto rico - Map Results



Puerto Rico
maps.yahoo.com
[View Larger Map](#) | [Satellite Map](#)

(g) maps

World series 2010 - News Results


[World Series Of Poker 2010: Final Results Posted From Las Vegas](#)
World Series Of Poker 2010: The results are in! After literally thousands of hands, the new Texas Hold 'Em Champion has been crowned. Jonathan Duhamel, ...
[Bleacher Report](#) 2 days ago
[World Series of Poker 2010 Results: First Canadian Champion](#)
[ThirdAge](#) 2 days ago
[NY Claims SF Giants Owe Them World Series Trophy Visit](#)
[CBS San Francisco](#) 13 hours ago

(h) news





Figure A.2: Example vertical blocks.

[Velveta cheese mac and cheese recipe - Recipe Results](#)
[9 Cheese Cheese Mac And Cheese Cheese Recipe](#)
[www.grouprecipes.com](#)
[Five Cheese Cheese Mac And Cheese Cheese Recipe](#)
[www.grouprecipes.com](#)
[Three Cheese Cheese Mac and Cheese Cheese with Panko Bread Crumb Topping](#)
[www.chow.com](#)

(i) recipes

[The boy who cried werewolf - Shopping Results](#)
 [KERWIN MATHEWS Movie Photo THE BOY WHO CRIED WEREWOLF](#)
\$3.5 - [eBay](#)
[The Boy Who Cried Werewolf DVD NEW! horror gore UNCUT](#)
\$13.99 Brand New - [eBay](#)
[KERWIN MATHEWS THE BOY WHO CRIED WEREWOLF PHOTO # 16](#)
\$9.99 - [eBay](#)

(j) shopping

[Walking dead season 2 - Twitter Results](#)
 [@steventkent](#): @steventkent it gets worse! <http://videogum.com/253902/ugh-alert-charlie-sheen-to-cameo-on-season-2-of-the-walking-dead/cameos/>
 0 seconds ago
 [Logan Booker](#): Season final of The Walking Dead probably best episode after the first. Shame we have to wait until next October(!) for **season 2**.
 0 seconds ago
 [@curtfletcher](#): RT @WarmingGlow: Rumor of the day: Charlie Sheen will have a cameo as a zombie in **Season 2** of "The Walking Dead."
<http://bit.ly/h9Z8tx>
 2 hours ago
 [@muthmedia](#): Well, I paid for the **season** (act. @mosesmonster did) so I might as well watch the last **2** eps. of The **Walking Dead**. (That sounds negative.)
 2 hours ago
[more results on twitter >](#)





(k) twitter

Weather for San Antonio, TX
58°F
 Current: **Sunny**
 Wind: N at 13 mph
 Humidity: 34%
[see detailed forecast](#)

Day	Wed	Thu	Fri	Sat
Temp	37°F 65°F	44°F 72°F	52°F 76°F	42°F 80°F

(l) weather

[I like to move it move it - Video Results](#)

			
Will.i.am - "I Like to Move It" Madagascar 2: Escape 2 Africa 3:08	Reel 2 Real - I Like To Move It HQ [1994] 3:50	Happy Feet Like to Move IT 2:46	will.i.am Official Madagascar 2 Music Video: I Like To Move It 2:47

(m) video

Figure A.3: Example vertical blocks.

User Study Queries and Descriptions

Set of queries and descriptions used in the user study presented in Chapter 5.

1. aperture vs photoshop: The user is looking for reviews that compare "Apple Aperture" and "Adobe Photoshop".
2. apple store pittsburgh: The user is looking for an Apple Store in Pittsburgh, PA.
3. bank of america online banking: The user is looking for information on the new features that Bank of America is adding to their online services.
4. belize map: The user is looking for a map of Belize.
5. best buy: The user is looking for financial information on the company Best Buy
6. best weather to fish in: The user is looking for recommendations on favorable weather conditions for fishing.
7. boston hotels near convention center: The user is looking for hotels in Boston that are near the convention center.
8. brussels belgium weather: The user is looking for current weather information for Brussels, Belgium.
9. calculate a golf handicap: The user is looking for instructions on how to properly calculate a "golf handicap"
10. caribou coffee: The user is looking for financial information on the company Caribou Coffee
11. cheap hotels in anaheim california: The user is looking for inexpensive hotels in Anaheim, CA.
12. cooking ribs: The user is looking for instructions on how to cook ribs.
13. craigslist suspect dies: The user wants to read about the recent death of the "Craigslist suspect".
14. culinary school charlotte north carolina: The user is looking for culinary schools in Charlotte, NC.

15. destin florida forecast: The user is looking for weather information for Destin, FL.
16. discovery channel hostage crisis: The user is looking for information on the recent hostage incident that occurred at the Discovery Channel's Headquarters.
17. dominican republic maps: The user is looking for a map of the Dominican Republic.
18. earthquake christchurch new zealand: The user is looking information on the recent earthquake in Christchurch, New Zealand.
19. eggs recall list: The user would like to find a list of brands, stores, and plants affected by the recent egg salmonella outbreak.
20. facebook places: The user would like to learn more about Facebook's new application: "Facebook Places".
21. floyd mayweather racist rant: The user is looking for a video clip of boxer Floyd Mayweather's controversial rant against opponent Manny Pacquiao.
22. fodors europe travel guide: The user would like to purchase a Fodor's travel guide for Europe.
23. giant goldfish france: The user is looking for information on the unusually large "goldfish" recently captured in France.
24. help people who have cancer and need funding: The user is looking for charitable ways of giving financial support to cancer victims.
25. home depot: The user would like to find a Home Depot store in Pittsburgh, PA.
26. hotel captain cook anchorage ak: The user is looking for the "Captain Cook" hotel in Anchorage, Alaska.
27. hotels in biloxi ms: The user is looking for hotel information in Biloxi, Mississippi.
28. how to remove glue from fabric: The user is looking for tips on how to remove glue from fabric.
29. hurricane earl path: The user is looking for forecast information on Hurricane Earl.
30. ihop nutritional facts: The user is looking for nutritional information for menu items in the restaurant IHOP.
31. instrument stores houston tx: The user is looking for stores that sell musical instruments in Houston, TX.
32. intel: The user would like to learn more about Intel's plan to acquire McAfee.

33. james lee discovery channel: The user is looking for information on "James Lee", responsible for the hostage incident at the Discovery Channel headquarters.
34. kitchen photos: The user is looking for images of kitchens.
35. learn to play the banjo: The user is looking for general information about learning to play the banjo.
36. local weather petoskey mi: The user is looking for weather information for Petoskey, MI.
37. machete race war: The user is looking for information on the controversial movie "Machete".
38. map of london england: The user is looking for a map of London, England.
39. map of wyoming: The user is looking for a map of Wyoming.
40. marbella spain weather: The user is looking for weather information for Marbella, Spain.
41. mariner energy: The user is looking for information on the recent explosion on the "Mariner Energy" oil rig.
42. marvin sapp wife dies: The user is looking for information on the recent death of the wife of singer Marvin Sapp.
43. miami above ground pool stores: The user is looking for stores that sell "above ground" swimming pools in Miami, FL.
44. michael jordan baseball stats: The user is looking for statistics on Michael Jordan's brief Baseball career.
45. miss universe 2010: The user would like to find information about the "miss universe 2010" beauty contest.
46. new ipod nano: The user is looking for reviews and pricing information for Apple's new iPod Nano.
47. nikon cool pix: The user plans to purchase a Nikon Cool Pix camera and is looking for product and pricing information.
48. nissan dealerships in albuquerque: The user is looking for Nissan dealerships in Albuquerque, New Mexico.
49. no time to die book: The user is considering purchasing the book "No Time to Die"
50. organic gardening pest: The user is looking for information on pest-control in organic farming.

51. photography books: The user is looking for books on photography.
52. pictures of bread: The user is looking for images of bread.
53. piranha 3d preview: The user would like to view a trailer for the movie: "Piranha 3D".
54. politics in the philippines: The user is looking for information on the historical and current political situation in the Philippines.
55. pot roast cooking time in oven: The user is looking for instructions on how to cook pot roast.
56. pressure cooker: The user would like to purchase a pressure cooker.
57. robert de niro photos: The user is looking for images of the actor Robert De Niro.
58. san bruno fire: The user is looking for information on the recent fire in San Bruno, CA.
59. southern home cooking: The user is looking for recipes of southern cuisine.
60. sports bars downtown chicago: The user is looking for a sports bar in downtown Chicago, IL.
61. tampa thai restaurants: The user is looking for Thai restaurants in Tampa, FL.
62. target super stores: The user is looking for Target stores in Pittsburgh, PA.
63. texas tax free weekend: The user would like to learn more about Texas' tax-free weekend sale.
64. tilapia fish recipes: The user is for recipes for Tilapia.
65. tom brady car accident: The user is looking for informaton on quaterback Tom Brady's recent car accident.
66. toyota auto parts: The user is looking for places to purchase Toyota auto parts.
67. true blood rolling stone cover: The user wants to know the public's general reaction to the new cover of "Rolling Stone" magazine, featuring the cast of TV show "True Blood".
68. ups plane crash: The user is looking for information on the recent UPS plane crash in Dubai.
69. us open fight: The user is looking for information on the recent fight that broke out between spectators at the US Open.
70. vera zvonareva: The user is looking for information on tennis player Vera Zvonareva's recent performance at the US Open.

71. watch true blood season 3 episode 10: The user would like to see a preview of the 10th episode of the 3rd season of the TV show True Blood.
72. world trade center: The user is looking for information on the latest construction plans for the future World Trade Center Memorial Site.