Text Representation, Retrieval, and Understanding with Knowledge Graphs

Chenyan Xiong

CMU-LTI-18-016

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

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

Abstract

This dissertation aims to improve text representation, retrieval, and understanding with knowledge graphs. Previous information retrieval systems were mostly built upon bag-of-words representations and frequency-based retrieval models. Effective as they are, word-based representations and frequency signals only provide shallow text understanding and have various intrinsic challenges. Utilizing entities and their structured semantics from knowledge graphs, this dissertation goes beyond bag-of-words and improves search with richer text representations, customized semantic structures, sophisticated ranking models and neural networks.

This thesis research starts by *enriching query representations* with entities and their textual attributes. It first presents query expansion methods that better represent the query with words from entity descriptions. Then it develops a supervised latent-space ranking model that connects query and documents through related entities from the knowledge graph. It also provides a novel supervised related entity finding technique in the entity search setup.

Then this dissertation presents our *entity-oriented search* framework that represents query and documents with *entity-based text representations* and matches them in the entity space. We construct a *bag-of-entities* model that represents texts using automatically linked entities with a customized linking strategy. Ranking with bag-of-entities can be done either solely with discrete match—as in classic retrieval models—or by our Explicit Semantic Ranking approach that soft matches the query and documents with continuous knowledge graph embeddings. The entity-based text representations are then combined with word-based representations in a *word-entity duet* representation method. In the duet, query and documents are represented by both bag-of-words and bag-of-entities; the ranking of them goes through both in-space matches and cross-space matches which together incorporates various types of semantics from knowledge graphs. The duet framework also introduces a hierarchical ranking model that learns the linking of entities and the ranking of documents jointly from relevance labels.

This thesis research concludes with a neural entity salience estimation model that provides a deeper text understanding capability. We developed a Kernel Entity Salience Model that better estimates the importance of entities in text with distributed representations and kernel-based interactions. Not only does it improve the salience estimation accuracy, it can also be used to estimate the importance of query entities in documents, which provides effective ranking features that transfer the model's deeper text understanding capability to improve retrieval.

With the effective usage of entities, their structured semantics, customized semantic grounding techniques and novel machine learning models, this dissertation formulates a new entity-oriented search paradigm that overcomes the limitation of bag-of-words and frequency based retrieval. The better text representation, retrieval, and understanding ability provided by this dissertation is a solid step towards the next generation of intelligent information systems.

Acknowledgements

I never thought I would be a Ph.D. when entering collage. After three years of death march for the college entrance exam, I thought I was done studying after undergraduate. No more textbooks, no more exams, go get a real job.

It was 2008 when I thought I should try academia. I was working on my internship project about automatic entity extraction from queries. I read a lot of papers from information retrieval conferences and implemented many in the production environment. I was excited that those papers are really interesting and (some) are indeed useful. However, I was also unhappy because I found that, compared to implementing existing technologies, I am more into creating my own. And I found the best way to learn creating new technologies—to pursue a Ph.D. myself.

Sometimes it feels surreal that a simple decision took ten years of execution. There are many times that I did not think I had a chance at all. I was very lucky and blessed. Much of this blessedness come from the wonderful environment and people around me.

First and foremost, I wish to express my deepest gratefulness to my advisor Jamie Callan. I learned a lot from Jamie in the past six years, not only about how to conduct research, but also how to be a researcher. I hope our collaboration continues and we can keep pushing the boundaries together.

I wish to thank the rest of my Ph.D. committee, Bruce Croft, William Cohen, and Tie-Yan Liu, for their insightful suggestions and comments. This dissertation benefits a lot from their advices.

I wish to share my special gratitude to Tie-Yan Liu. As my role model and hero since the beginning of my research career, I learned a lot from Tie-Yan's vision, dedication to impactful research, and the never-ending pursuit to greatness. I will keep learning from him and hope I will become a better researcher myself.

I wish to thank Professor Yi-dong Shen, for his guidance and the warmest supports for me during my Ph.D. study. It is a bless to have Prof. Shen as my advisor in my master study.

I also would like to thank Russell Power, for hosting me a wonderful internship in Allen Institute, for helping me start in deep learning, and for his countless supports as a mentor and as a friend. My research would be less interesting without him.

I am grateful to my friends, students, and collaborators. I especially would like to mention Oren Etzioni, Taifeng Wang, Zhiyuan Liu, Yiqun Liu, Kristian Balog, Eduard Hovy, Teruko Mitamura, Yiming Yang, Laura Dietz, Edgar Meij, Jeff Dalton, Bhavana Dalvi, Guoqing Zheng, Yubin Kim, Anagha Kulkarni, Reyyan Yeniterzi, Jing Chen, Zhuyun Dai, Zhengzhong Liu, Di Wang, Xiaohua Yan, William Wang, Miaomiao Wen, Zi Yang, Yin Xu, Michael Belmonte, Faegheh Hasibi, Cheng Luo, Van Dang, Bin Wu, Zhenghao Liu, Liang Du, Xuan Li, Hongyu Li, Ziqi Wang, Wenkui Ding, Meiwen Ouyang, and many many more. I benefited and learned a lot from our interactions in research and in personal life.

I own a great deal to my parents who provide me the environment to pursue my aspirations. It is not easy to be the parents of a Ph.D. student who sometimes is absent from the real world.

This dissertation is dedicated to my wife, Mengqiao Liu. Without her tremendous supports and understanding, this dissertation work would be less enjoyable and probably will not finish on time.

To My Family.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Ad hoc search is the core task of information retrieval. Consider the following query and document that a search engine may encounter:

> Query: "*Carnegie Mellon Location*"
>
> Document: "*Carnegie Mellon University is a private research university in Pittsburgh, Pennsylvania. Founded in 1900 by Andrew Carnegie as the Carnegie Technical Schools, the university became the Carnegie Institute of Technology in 1912 and began granting four-year degrees. In 1967, the Carnegie Institute of Technology merged with the Mellon Institute of Industrial Research to form Carnegie Mellon University."* [1]

The goal of ad hoc search is to find the relevant documents that satisfy the information needs behind the query, such as the example query-document pair. In a typical ranking system, this process can be divided into two steps: *representation*, which transfers the natural language query and documents into computer understandable formats, and *ranking*, which ranks documents by modeling their relevance to the query. In modern information retrieval, the representation and ranking are mostly done through *bag-of-words*.

1.1 Information Retrieval by Bag-of-Words

In bag-of-words based search engines, the query and documents are represented by discrete vectors, whose dimensions correspond to terms in the texts. For example, the bag-of-words for the example query can be "{Carnegie:1, Mellon:1, Location:1}", where the weights are the importance of the corresponding words in the text. In the bag-of-words representation, texts are broken into terms and the terms are considered independent with each other. These simplifications make bag-of-words efficient to construct and maintain. It scales up easily to the large amount of query traffic and web pages.

With the bag-of-words representation, *ranking models* are able to leverage *term-level statistics* to model the relevance between query and documents based on their relevance to the query.

[1] `https://en.wikipedia.org/wiki/Carnegie_Mellon_University`

Standard term-level statistics include how many times the query terms appear in the document's vector (term frequency), the frequency of query terms in the entire corpus (document frequency), and the document length. Term-level statistics have been used by both unsupervised retrieval models and learning to rank models. Unsupervised retrieval models (e.g. query likelihood, vector space models, BM25, and DPH [29]) developed various ways to combine these term-level statistics to produce document rankings. Learning to rank models leverage features from unsupervised retrieval and document qualities, and combine them with supervised learning [60].

The bag-of-words based search engines are a huge success. Millions of users are using them per day. However, their effectiveness is mostly a result of the development of the ranking models in recent decades; the representation part stays almost the same. The overlooked representation part limits the information available to the ranking models. For example, despite being derived from variant theories, current unsupervised retrieval models all rely on the same set of term-level statistics: term frequency (TF), inverse document frequency (IDF) and document length. Learning to rank models utilize all kinds of machine learning models, but their features are almost the same: retrieval models based on term-level statistics, document qualities, and perhaps also user feedbacks in the commercial environment. Only the fancy ranking models themselves did not overcome the limitation of the isolated word-based representation space.

Many techniques have been developed to address the limitation of bag-of-words representation. For example, query expansion techniques expand the query with a larger set of related terms, with the hope that the expansion terms can better retrieve relevant documents. Sequential dependency model (SDM) takes phrases into consideration and matches query and documents not only by their words but also ngrams. However, within the bag-of-words scenario, the term vector based representation is only an approximation about how search engine users understand texts. Search engine users generate queries and read returned documents with their prior knowledge that is external to the query or corpus. Also, they understand the texts structurally instead of by flat term vectors. Individual term counts and statistic models provide an empirical approximation that works well, but the gap between user's and search engine's understanding of query and documents is inevitably there.

1.2 Information Retrieval with Human Knowledge

Another way to bridge the gap between users and search systems is to incorporate human knowledge. Since centuries ago, librarians have been using *controlled vocabularies*, such as subject headings and thesauri, to describe, index, and retrieve books. Controlled vocabularies are a set of manually selected terms that refer to real world concepts like 'Education' and 'University'. The controlled vocabulary terms are often grouped into a carefully designed *ontology*, which partitions the target domain into tree-structured categories. For example, a possible top-down path in the ontology is 'Education' $\Rightarrow$ 'University' $\Rightarrow$ 'Private university'. The earliest search engines adopted the librarian's approach and represented documents by controlled vocabularies [83]. In these retrieval systems, human editors manually annotate documents with terms from a controlled vocabulary, and the retrieval of documents is done in the controlled vocabulary space.

Controlled vocabularies based representation has several advantages. The terms in controlled vocabularies are manually selected informative units, for example, concepts of a certain domain. The vocabularies are controlled to be clean and of real-world meaning. The design of the ontology and the categorization of terms are done by experts using their domain knowledge. They provide search engines meaningful external information to refer to during retrieval. The annotation of controlled vocabularies to documents is done by editors according to their understandings about the documents. The vocabulary mismatch problem is also less severe as both query and documents are grounded manually to a small controlled vocabulary. The controlled vocabulary based search systems can retrieve documents accurately with simple retrieval models. They are still being used by search engines in several domains [66].

The involvement of human experts makes controlled vocabulary representation more intelligent but also restricts the scale of its usage. The manual construction of a controlled vocabulary requires substantial expert efforts. Thus, the coverage of controlled vocabularies over general domain texts is limited—it is impossible to manually assign controlled vocabulary terms to the nearly infinite number of documents in modern search engines. The focus of ranking model research in recent decades is also mainly on bag-of-words representations. It was unclear whether and how those new ranking models can improve controlled vocabulary based search engines.

Recently, large scale *knowledge bases* or *knowledge graphs* have emerged. Knowledge bases store human knowledge into computer-understandable format in graph-like structures. Each node in the knowledge graph records a basic information unit, called 'object' or 'entity'. An object or entity can be a named entity (e.g., Carnegie Mellon University), a concept (e.g., University), or anything that corresponds to real-world stuff. The edges in the knowledge graph connect entities by their relationships or link entities to their attributes. The information represented by an edge is usually called 'fact'. A fact can be an attribute (e.g., the enrollment of Carnegie Mellon), a category (e.g., CMU is a University), or a relationship to another entity (e.g., CMU locates in Pittsburgh). Knowledge bases share the similar goal of storing human knowledge into structured formats with controlled vocabularies, but they are also different in many perspectives. Modern knowledge bases are curated by community efforts (e.g., Wikipedia), semi-automatically by both human and machines (e.g., Google's Knowledge Vault [35]), or automatically by information extraction systems (e.g., NELL [19]). They are carefully curated thus with higher quality than document corpus, but are usually less precise than controlled vocabularies which are manually created by domain experts. The annotation of entities to texts can also be done automatically using entity linking systems [20, 47]. The automatic nature makes it possible to obtain knowledge graphs at large scale with richer information (e.g., Freebase), and to annotate large web corpora (e.g., Google's FACC1 annotation [39]). This brings a new opportunity to explore knowledge bases' ability in information retrieval, but the noises and contradictions due to automation also raise new challenges to address.

1.3 Thesis Research

This dissertation aims to improve the representation, retrieval, and understanding of texts using knowledge graphs. It starts by enriching the original word-based text representations in search engines with entities and their attributes in knowledge graphs. Then it presents new entity-based text representations that represent and match query and documents directly in the entity space.

After that, this thesis research develops a word-entity duet paradigm that combines the word-based and the entity-based text representations as well as machine learning models to handle the uncertainties in entity-based representations. The last part of this thesis research presents a new path towards text understanding through entities. It develops a graph-based representation that leverages neural networks to model the interactions between entities and words in the texts, which better estimates the importance/salience of entities in texts and also improves search accuracy.

The Definition of Knowledge Graph.

This thesis research uses a flexible definition of knowledge graph—any data resource that contains entities and some information about the entities is considered as a knowledge graph. There is no restriction on the specific format or structure on knowledge graphs in this dissertation.

Our scope of entities includes named entities, general entities, or any thing that refers to a concept. Name entities can be people names, brands, locations, etc. General entities include common things, for example, 'table', 'laptop', and 'university'. The concept can be a real-world concept such as 'physics' or a fictional concept such as 'cyberpunk'. Practically, anything that has its own meaning can be considered an entity, for example, all Wikipedia entries are entities in this thesis research.

The information about entities, which this dissertation refers to as 'knowledge graph semantics', also has a wide scope. There is no restriction of the specific format, origin, or aspect of the information, as long as it is about the entity. This definition includes the entity attributes in a typical knowledge base, e.g., the names, aliases, description and types of Freebase entities, are information about. The entity relations are also included. In addition to those standard knowledge graph semantics, weak-structured information such as bag-of-words or embeddings of entities are also feasible knowledge graph semantics. The knowledge graph semantics can be expert-written or automatically extracted; they do not have to be precise, exactly-defined or structurally-organized.

This dissertation thus defines knowledge graph as *a data resource that contains entries ('entities') that have their own meanings and also information ('knowledge graph semantics') about those entries*. Following this definition, a web corpus is not a knowledge graph as there is no 'meaningful entry'; a list of controlled vocabulary words is not a knowledge graph as there is no 'additional information about those entries'. But controlled vocabularies, modern knowledge graphs, and automatically constructed semantic resources are all considered as knowledge graphs in this dissertation, as long as they store information (knowledge graph semantics) around meaningful nodes (entities).

Enriching Word-Based Retrieval with Knowledge Graphs.

The first focus of this thesis research is to enrich the *word-based text representation* of queries. Queries are usually short and carelessly written, unable to fully describe user's information needs behind them. We utilize the information about entities in knowledge graphs to enrich the word-based query representations. For example, entities such as Carnegie Mellon University, Andrew Carnegie, and Language Technologies Institute are linked with their descriptions, attributes, types and related entities. This knowledge can be useful in explaining various queries about Carnegie Mellon. Many existing techniques can be used to find useful entities for a query. Query annotations provide entities that directly appear in the query string. Entity search retrieves entities that are related to the query. Document annotations provide entities that connect to the query through retrieved documents, which are also useful under the pseudo relevance feedback assumption.

We start by using query expansion, a widely used technique, to expand queries with terms from relevant entity's textual descriptions. The carefully written textual descriptions of knowledge base entities can help explain corresponding queries. For example, the description of Carnegie Mellon University contains terms 'research', 'computer', 'Turing', and 'award', all useful for query 'CMU'. Our method first finds relate entities for a given query, for example, Carnegie Mellon University, Pittsburgh and Computer Science for query 'CMU', using entity search and document annotations. Terms that frequently appear in their descriptions, or fall into similar categories with the query are selected to expand the query. The expansion terms from knowledge base turned out to be very effective in both improving the accuracy of existing ranking models and reducing the risk of query expansion. This result motivates us to further explore the potential of knowledge bases in information retrieval.

Knowledge graph entities bring much richer information than just textual descriptions. To better make use of them, we develop `EsdRank`, a framework that connects query and documents through such external semi-structured data. In `EsdRank`, entities that appear in the query, are retrieved by the query, and frequently appear in query's top retrieved documents are selected to enrich the query. EsdRank framework makes it possible to use all kinds of information associated with these entities to improve document ranking. Recall our example query 'Carnegie Mellon Location', the relevant documents should have strong connections to the entity Carnegie Mellon University. The connection is not only about text similarity, but can also be determined by whether the document contains query entities and whether it falls into similar categories with the query entities. To make use of the new and heterogeneous evidence introduced by query entities, a novel learning to rank model is developed to learn the connection from query to entities and the rank of documents jointly. The new entity based query representation adds information retrieval with richer external evidence and a sophisticated but suitable learning to rank model, thus significantly improving the state-of-the-art of document ranking.

An essential step in our knowledge based query representations is to find relevant entities for the query. In our previous systems, it was done by existing automatic approaches. One of them is entity search, which provides entities that are useful for our systems, but also found to be a bottleneck of the ranking performance. We address this problem by developing our own entity search system using learning to rank. The facts about an entity in the knowledge base are grouped into the fields of a virtual document. Text similarity features between the query and entity's virtual document are extracted and used in learning to rank models. This leads to significantly better entity search accuracies on several test collections.

Representing Texts with Entities

The second part of this thesis research presents the *entity-based* text representations, which, instead of *enriching* the original word-based representations, directly represent texts using their related entities. We human beings do not understand texts through individual words. We see "Carnegie Mellon University" together as one concept and do not break it down into three individual words as in the bag-of-words model. If search engines can process and match texts in the concept level, it will automatically avoid some strong assumptions introduced by bag-of-words. In fact, it is how search was done in the early controlled vocabularies based search engines. However, they have been taken over by bag-of-words due to their inability to scale.

This part of this thesis research first builds *entity-based* text representations by revisiting the controlled vocabulary based text representations, with larger knowledge bases and automatic entity linking systems. Given the text, in the query or the document, we construct a *bag-of-entities* vector from its automatically linked entity annotations. For example, the query "Carnegie Mellon Location" is now represented by the entities "Carnegie Mellon" and "Location". We relax the precision constraint of entity linking systems and show that the coverage of bag-of-entities with automatically entity annotation can provide sufficient coverage on general domain texts. Matching texts in the entity space provide advantages of automatic chunking, i.e. "Carnegie Mellon" is considered as one term, and also automatic synonym resolution, i.e. "CMU" is matched to "Carnegie Mellon" as both are linked to the same entity. Although the accuracy of the automatic annotations is not high, exact-matching query and documents with bag-of-entities can already outperform standard word-based retrieval.

The entity-based representations provide much more opportunities than exact-matching bag-of-entities. Entities come with structured semantics in the knowledge graphs. They are not isolated terms in a dictionary. The thesis research then developed *Explicit Semantic Ranking* (`ESR`), which uses the structured semantics associated with entities to match query and documents. With some prior knowledge, we know that "CMU" is a University and is located in Pittsburgh. It helps us formulate queries and understand documents. To utilize such structured semantics from knowledge graphs, `ESR` embeds them as distributed representations of entities which allows matching the entities in the query and document softly in the embedding space. Thus the relevance connections can be made through entities that are not the same but related to each other. Our analyses find that the inability to "understand" such semantics of entities is the major cause of failures in an online academic search engine. Our experiments on its online search logs show that `ESR` is able to address those word-based search failures and significantly improves the ranking accuracy.

Combining Word-Based and Entity-Based Text Representations

The third part of this dissertation presents the *word-entity duet representation*, a framework to fuse word-based and entity-based text representations. As shown by our previous progress, bag-of-entities provides a different way to represent texts and introduces novel ranking signals that are not available in word-based retrieval systems. On the other hand, entities are only part of the texts, and the entity-based representations are automatically constructed thus not perfect. The development of entity-based ranking has also provided various novel ranking signals. It is also time to study how they interact and compensate with the original word-based ranking signals from modern information retrieval.

The word-entity duet framework provides a systematic view for ranking with the two representations. It first allocates the classic word-based ranking features and the new entity-based ranking features (from ESR) as the in-space matching signals within the word or the entity space. This formulation also introduces new cross-space matching between bag-of-words and bag-of-entities through the textual attributes of entities. For example, documents that discuss a lot of "CMU" culture probably can be matched with the description of the entity "CMU" in the query. All the four-way interactions cover different aspects of query-document relevance and can be combined by learning to rank models. We further developed a hierarchical ranking model to handle the noisy entity annotations in the query—the major bottleneck in many entity-based search systems. Its attention mechanism helps shift the ranking model away from noisy entities and is essential for noisy queries.

Instead of treating the entity linker as a black box and only trying to demote its errors, we propose the JointSem method that learns how to link entities and how to rank documents with entities jointly using the hierarchical ranker. JointSem introduces a soft-linking scheme that keeps multiple possible entity candidates for the query and uses the ranking labels to learn the weights (attention) on each possible candidate. It delays the annotation decision until the entity-based ranking under the guidance of the ranking model. The soft linking provide more flexibility for the ranking model to fix some linking errors that would have been made by hard linking. It is also more robust on ambiguous queries without enough signal. For example, "CMU" can also be "Central Michigan University". Keeping both "Carnegie Mellon" and "Central Michigan" and letting the ranking model to decide globally is more effective than just picking one of the two in the linking step.

From Text Representation to Understanding

The last part of this thesis research moves *from representation to understanding* by better-estimating entity salience. In the bag-of-words, bag-of-entities, or the word-entity duet representations, the importance of terms, words or entities, is largely defined by their frequencies in the text. However, many cases *frequency is not equal to importance* and breaking texts into isolated terms only provides shallow text understanding. It has been a long desired goal in information retrieval to go beyond "bag-of-terms" and model the interactions between them to better estimate term importance.

We revisit this idea with the advantage of entities, structured semantics, and neural networks. We develop a Kernel Entity Salience Model (KESM) that estimates the importance (salience) of entities in the texts not only by their frequencies but also by their interactions. The interactions are modeled by KESM in the embedding space, through our new kernel interaction model that learns multi-level interaction patterns between terms. The interaction patterns are then used by KESM to estimate the importance of entities for the text.

The kernel entity salience model is first tested in the task of estimating the importance/salience of entities in a document, a preliminary step of text understanding. Leveraging the large amount of training data in the task, the end-to-end trained KESM is much more effective in predicting which entity is important in a document than frequency-based and feature-based methods. For example, given a new article about CMU, even if the entity "CMU" is only mentioned several times, the graph-based representation is able to better predict its salience based on all other entities and words in the document, based on the interactions between those terms in the embedding space, as modeled by the kernels.

The better text understanding capability from KESM also improves ranking accuracy. The public ranking benchmarks are often in limited scale and thus prevent end-to-end learning of complex neural networks. We use the pre-trained model in the salience task and use it to estimate the importance of query entities in candidate documents. The model promotes documents that focus on the query concepts instead of just mentioning them. For example, it can promote documents that are exclusively about "Carnegie Mellon" for the query "CMU" because the target entity is more important in the document, compared to documents that contain a list of many universities. It demonstrates that our model successfully converts the text understanding ability learned from the salience task to search. It effectively models the interactions and consistency of query entities in documents, which have been long desired goals in information retrieval and very challenging tasks with the bag-of-terms framework.

Contributions

This dissertation aims to better represent and understand texts and improve search engines with knowledge graphs. It enriches the original word-based text representations, builds entity-based text representations, and also develops a duet framework that systematically fuses the two representations for query and documents. A series of new ranking approaches are developed to incorporate the semantics from knowledge graphs in the search systems, including query expansion, latent learning to rank, embedding-based soft match models and joint learning models. This thesis research also goes beyond bag-of-terms representation by modeling the interactions between terms using embeddings and kernels. The formulated graph-based representations provide a deeper text understanding ability that generalizes well across modeling entity salience and ranking documents based on the interactions of query entities within them.

The effectiveness of this thesis research has been demonstrated generally across various scenarios, including general domain web search, medical search, and academic search, on highly competitive academic benchmarks and also online search logs. The knowledge graphs utilized range from classical controlled vocabularies, modern large scale knowledge graphs, to an automatically constructed knowledge graph. The methods presented in this dissertation achieve the state-of-the-art in various search tasks, including but not limited to query expansion, unsupervised retrieval, and feature-based learning to rank. Recently, the techniques developed in this dissertation have also been successfully adopted in deep learning models and improved state-of-the-art neural ranking models.

This dissertation research has provided a new entity-oriented search paradigm that effectively integrates entities and their semantics from knowledge graphs to information retrieval systems. It is now almost a disadvantage not to consider this thesis research when working in core ranking research. This dissertation research also goes beyond bag-of-terms representations using knowledge graphs and neural networks. Our Kernel Entity Salience Estimation model shows deeper text understanding abilities that can also be generalized to improve real information retrieval tasks. Previously it is very challenging for such fine-grain text processing techniques to improve search accuracy. The landscape this thesis research sets up provides a wide range of opportunities for future research in utilizing structured knowledge, developing ranking models, and constructing new knowledge graphs that better fits the needs of real-world applications. We believe this dissertation has pointed out several promising paths towards the future intelligent systems and will inspire more Ph.D. dissertation research in the near future.

The rest of this dissertation organizes as follows. A brief overview of knowledge bases and related works are presented in Chapter 2. Our research about *enriching word-based query representations* is in Chapter 3, with query expansion with knowledge base in Session 3.1, EsdRank with query entities in Session 3.2, and relevant entity finding in Session 3.3. The research about *entity-based text representations* is in Chapter 4. It includes the bag-of-entities model in Session 4.1 and the Explicit Semantic Ranking model in Session 4.2. The research about *word-entity duet* is presented in Chapter 5, including the duet representation in Section 5.1 and the joint model for linking and ranking in Section 5.2. Chapter 6 presents the last part of this dissertation research that improves *text understanding* with entity salience estimation. The last chapter concludes and discusses the impacts of this dissertation.

Chapter 2

Background and Related Work

This chapter first provides some background about controlled vocabulary based information retrieval. Then it gives an overview of knowledge graphs, semantic grounding techniques, and related works.

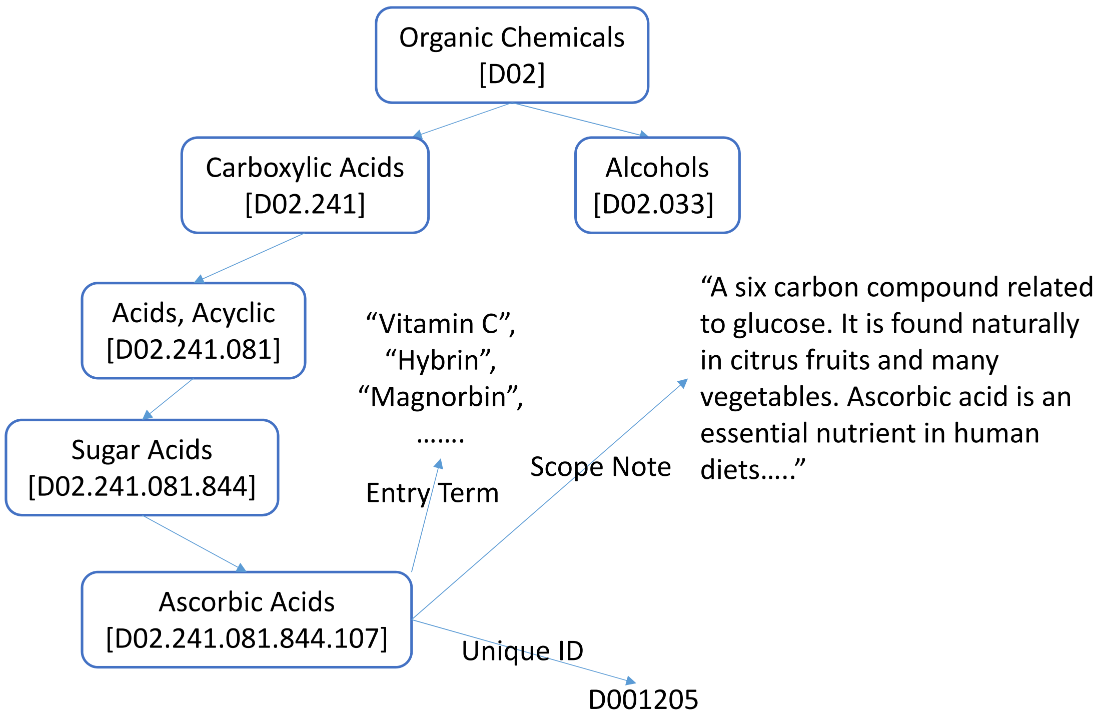
2.1 Controlled Vocabularies and Information Retrieval

The use of controlled vocabularies can date back to at least two thousand years ago, when the librarians in Egypt use them to organize books in the Library of Alexandria, 300 BCE. Instead of using all words, controlled vocabularies restrict themselves to a much smaller set of informative terms. These terms are carefully selected to represent real-world objects, for example, concepts, common names, and domain terminologies. The controlled vocabularies are still being used by most libraries and also on the Internet. Famous representatives include World Bank Thesaurus, Medical Subject Headings (MeSH), and Library of Congress Subject Headings (LCSH).

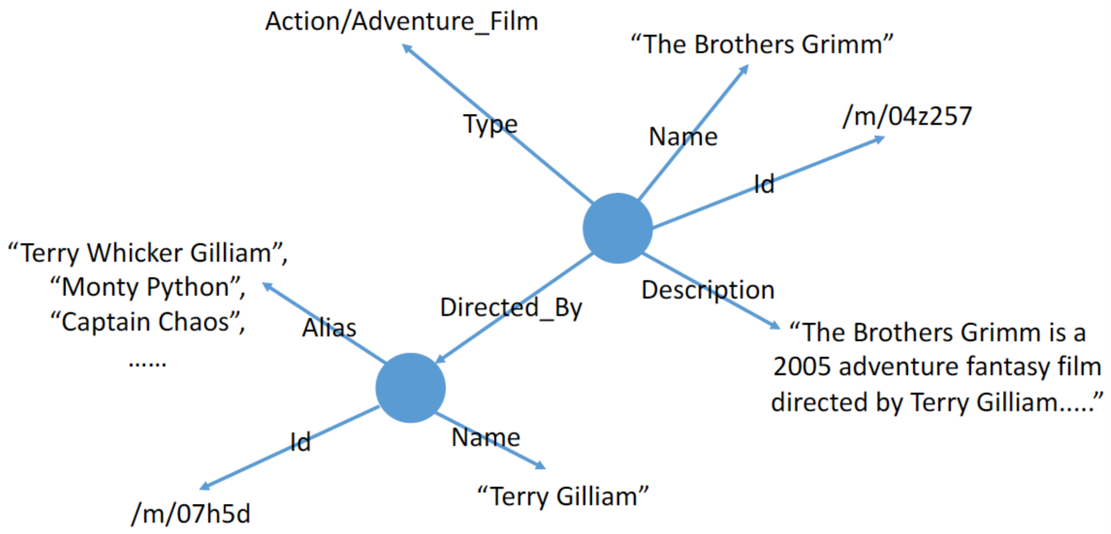
Although different controlled vocabulary datasets may make different choices on what to include, there are some common ones among various datasets. Synonym, or alias, is one of the first things to include in a controlled vocabulary, since one significant use of controlled vocabulary is to resolve language variety. Ontology, or taxonomy, is another important component of many controlled vocabularies. Based on domain experts' understanding, ontology partitions the domain knowledge into a tree level structure, to organize, index, and retrieve target information. Textual descriptions are also often included to define and explain a controlled vocabulary term. The description helps users understand the term and facilitates the usage for domain experts. Figure 2.1a illustrates a part of MeSH's ontology [1] and some facts associated with the term "Ascorbic Acids", including its aliases ("Entry Term") and description ("Scope Note").

Information retrieval researchers inherited the librarians' approach and built controlled vocabulary based search engines. In the earliest search engines, documents were represented by terms from controlled vocabularies, e.g., thesaurus or subject headings. The annotation of controlled vocabularies to documents was performed manually by domain experts, using their understandings and prior knowledge. The documents were then indexed based on their annotations, and the retrieval was done by matching them with the query in the controlled vocabulary space.

As full-text search became popular, controlled vocabularies continued to be used in some


Organic Chemicals [D02]
Carboxylic Acids [D02.241]
Alcohols [D02.033]
Acids, Acyclic [D02.241.081]
Sugar Acids [D02.241.081.844]
Ascorbic Acids [D02.241.081.844.107]
"Vitamin C", "Hybrin", "Magnorbin", …….
Entry Term
Scope Note
"A six carbon compound related to glucose. It is found naturally in citrus fruits and many vegetables. Ascorbic acid is an essential nutrient in human diets….."
Unique ID
D001205

(a) A part of the MeSH ontology. The attributes of "Ascorbic Acids" are shown.


Action/Adventure_Film
"The Brothers Grimm"
/m/04z257
Type
Name
Id
Description
"The Brothers Grimm is a 2005 adventure fantasy film directed by Terry Gilliam….."
Directed_By
"Terry Whicker Gilliam", "Monty Python", "Captain Chaos", ……
Alias
Id
/m/07h5d
Name
"Terry Gilliam"

(b) A subgraph of Freebase. Example relations and attributes around the entities "The Brothers Grimm" and "Terry Gilliam" are illustrated.

Figure 2.1: Examples of a controlled vocabulary (MeSH) and a knowledge base (Freebase)

systems. The use of controlled vocabularies is almost a necessity in medical search engines. Medical queries are often about diseases, treatments, and genes. Their names may not convey their domain-specific meanings. Typical procedures include using controlled vocabularies as alternative representations to match query and document [66, 78], and expanding queries using synonyms [65]. There is also rich literature about overcoming vocabulary mismatch by adding synonyms and related concepts to queries and documents [57]. These approaches can be useful in enterprise search and domain-specific search environments [41], or improving recall of relevant documents in general search environments [57]. One can also improve the ranking accuracy by combining controlled vocabulary and bag-of-words [78].

Nevertheless, bag-of-words plays a more crucial role in most current search engines, because the use of controlled vocabularies faces many obstacles, especially in general domains and large scale search environments. It is expensive for experts to construct a large enough controlled vocabulary that has sufficient coverage in general domains, if ever possible. The construction of controlled vocabulary representations requires annotations. In search environments where the sizes of corpora are enormous, manual annotation is not feasible. One on-going research topic is to automatically annotate texts to controlled vocabulary ontologies using large scale multi-class classification techniques [40]. Another on-going research topic is how to automatically leverage the information in controlled vocabulary in search engines. For example, synonyms of medical terms are very important for medical search, in which vocabulary mismatch problem is severe. However, the results of the TREC Genomics Track illustrate the difficulty of using controlled vocabularies in medical search; most systems require human efforts to be successful [87]. One particular challenge is how to pick and weight the synonyms correctly [65]. The ontology and cross-reference of controlled vocabulary terms provide structural information that connects individual terms. Using these connections intuitively should be able to improve the matching between query and documents; however, the experiments in recent works only show mixed results [51, 57]. The importance of controlled vocabulary based search engines nowadays is mostly in special domains serving domain experts who can formulate high-quality structured queries, for example, in the medical domain and the legal domain.

2.2 Knowledge Graph Overview

Knowledge Graphs, or Knowledge Bases, are collections that store human knowledge in computer-understandable formats. The collections can be manually or semi-automatically curated, for example, Freebase [13], YAGO [89], and DBPedia [56], or automatically constructed, for example, NELL [19] and OpenIE [5]. They are usually organized semi-structurally as a graph. For example, Figure 2.1b shows a sub-graph of Freebase. In the graph, a node is an entity (object), with a unique Machine Id. An edge in the graph links an entity to its attribute or another entity. There are many kinds of edges in Freebase, representing different facts. For example, in Figure 2.1b, "The Brothers Grimm" is connected to "Terry Gilliam" by a "Directed_by" edge, showing that the movie "The Brothers Grimm" is directed by the director "Terry Gilliam"; the two nodes also associated with its attributes such as aliases (synonyms), types (category) and textual descriptions.

The knowledge graph is usually stored as RDF triples. Each triple of subject-predicate-object corresponds to a head, edge, and tail in the knowledge graph. The head is an entity, or object, which can be a named entity, general domain entity, or just a noun phrase. The edge stores the type of the connection. Its type can be chosen from a vocabulary predefined manually by domain experts, called closed schema (e.g., in Freebase and DBpedia). It can also be verb phrases automatically extracted by information extraction techniques, called open schema (e.g., in NELL and OpenIE). The tail can be a related entity, or an attribute such as name, description or category of the subject entity.

The modern knowledge graphs make different choices with classic controlled vocabularies in recording real-world semantics. Controlled vocabularies are carefully edited by domain experts, more precise but mainly designed for specific domains at a smaller scale. Modern knowledge bases choose a looser schema to facilitate semi-automatic or automatic construction, which also introduces noise and contradictions. For example, MeSH, a widely used medical controlled vocabulary, contains about 27 thousand descriptors (terms), while Freebase contains more than 58 million entities. They also favor different semantic information. Controlled vocabularies focus more on the ontologies. For example, MeSH has a carefully defined ontology that partitions medical knowledge into a thirteen-level hierarchical tree. In comparison, although storing general domain knowledge at much larger scale, Freebase only has a two-level ontology. But modern knowledge bases include a wider range of attributes and relationships. For example, the entity Carnegie Mellon University is associated with 640 facts from 74 types in Freebase, while most MeSH terms only have less than ten attributes.

The large scale, richness, and flexibility of modern knowledge graphs bring a new opportunity to reconsider their potential in information retrieval. However, through these divergences, modern knowledge bases, and classic controlled vocabularies share the same spirits: storing human knowledge in structured formats. They both organize around semantically informative objects: controlled vocabulary terms or knowledge graph entities. The information they store also overlaps: ontologies or type systems, scope notes or descriptions, references or relations. From the perspective of full-text search engines, they are all external information to the user, query, and corpus.

This thesis research uses a broader definition of knowledge graphs that includes all such external and semi-structured collections, with the goal of developing general solutions to improve the representation, retrieval, and understanding of texts. We also use a more general scope of ‘entity’ which is the ‘basic’ information unit or object in such structured semantic collections. It includes concepts, noun phrases, named entities, controlled vocabulary terms, and general entities. In the rest of this chapter, we will first discuss the related semantic grounding techniques that align knowledge graphs to texts, and then the related work in utilizing knowledge graphs to improve full-text search engines.

2.3 Related Semantic Grounding Techniques

Another widely used technique in this dissertation is *semantic grounding*, which aims to automatically ground natural language texts to semantic structures, such as knowledge graphs. It enables us to automatically find related entities for the query and documents, instead of requiring manual annotations as in the classic controlled vocabulary based search engines. This section provides an overview of related grounding techniques.

Entity search aims to find the related entities in knowledge graphs for a given text query. The query format can vary from keywords to natural language questions [3, 45], but the information needs target knowledge graph entities. For example, the query can be "Carnegie Mellon University", "List of Universities in Pittsburgh", or "What are the best Universities in Computer Science", while the targets of them all include the entity "CMU".

Various techniques have been proposed to tackle entity search. On recent public benchmarks, although the information associated with entities comes in a structured format, the most effective approaches are to convert the attributes, types, and relations to textual fields of the entity and apply full-text retrieval models on them [3]. Along this line of research, Zhiltsov et al. developed the fielded sequential dependence model (FSDM) for entity search, which groups the attributes of an entity into five manually defined fields and applies the sequential dependence model on them [112]. Nikolaev et al. applied supervised learning to tune the parameters in the fielded sequential dependence model [76]. Hasibi et al. matches query and entities using the entity annotations on their texts [44], which is the bag-of-entities model for entity search and was developed in parallel with our bag-of-entities model for full-text search (Section 4.1).

Entity linking aims to recognize the appearance of entities in text and link them to corresponding entries in the knowledge graph. For example, given the first sentence of our example document in Chapter 1, an entity linking system may annotate it as "Carnegie Mellon University is a private research university in Pittsburgh, Pennsylvania.", with the underlined phrases linked to corresponding entities in the knowledge graph.

A standard way to perform entity linking is to first match ngrams in the text to names and aliases of entities in the knowledge graph (*spotting*), and then pick the right one for each spot from all entities have the same name (*disambiguation*). The linking process considers various information from the knowledge graph and the text. The decision is made based on both local evidence about the entity and the ngram, and the global evidence across the entire text and all other possible entity links. Famous entity linking systems include (but are not limited to as there are so many) TagMe [38], DBPedia spotlight [68], Wikification [72] and S-MART [106]. Linking long documents is the first focus of entity linking research [47], while research about linking on shorter text such as tweets and queries has also emerged recently [20, 43]. After years of development, now entity linking systems can be reliable and fast enough to annotate knowledge graphs as large as Freebase to large scale web corpora such as ClueWeb09 and ClueWeb12 (e.g., Google's FACC1 annotation [39]).

This dissertation explores and experiments with various entity search and entity linking techniques to find related entities for the query or documents. Throughout this thesis research, we studied the applicability of existing grounding techniques in information retrieval tasks, found various discrepancies between the existing grounding techniques, and developed our own ones based on the needs of information retrieval applications, for example, in Section 3.3, Section 4.1, and Chapter 5.

2.4 Related Work

Recently, the rapid developments of knowledge graphs, automatic grounding techniques, and machine learning techniques have drawn wide attention in improving the intelligence of information systems using structured semantics. This dissertation is one of the first to study the capability of modern knowledge graphs in information retrieval tasks. During this thesis research, there have been several parallel research from other groups that share similar focuses with us. Interestingly, the parallel progress aligns rather well with the first half of this dissertation: first to enrich existing word-based retrieval systems and then to construct entity-based text representations.

The first group of related work includes Entity Query Feature Expansion [31] and Latent Entity Space [61]. They stay within the modern information retrieval framework and contribute by bringing extra ranking signals from knowledge graphs to an existing word-based ranking system.

The Entity Query Feature Expansion (EQFE) approach by Dalton et al. enriches the ranking features using information from entities related to the query [31]. They study several ways to align Freebase entities to a query using query annotation, textual similarity, an entity context model, and annotations from top retrieved documents. The name, alias and type fields of these entities are considered as possible expansion candidates. The combination of different linking methods, expansion fields, and hyperparameters in the expansion are enumerated to get various expanded queries. These expansion queries are then used to calculate ranking scores for each document using a query likelihood or sequential dependency model. A learning to rank model uses these ranking scores as features for each document and produces final document rankings. Their results on news retrieval (Robust 04) and web search (TREC Web Track)‘ are among one of the first to demonstrate that knowledge graphs can compete with state-of-the-art bag-of-words based ranking methods on general domain search tasks.

The Latent Entity Space (LES) approach developed by Liu et al. utilizes entities to build latent connections between query and documents [61]. They use entities manually labeled to a query as a hidden layer between query and documents. The latent entities provide alternative connections between query and documents. With them, the ranking of documents is determined not only by query-document matching, but also by the textual similarities between those documents to latent entities, and between latent entities to the query. This evidence is incorporated in an unsupervised latent space language model to rank documents. Their experiments on multiple test collections and in the TREC Web Track competition [62] provide evidence of knowledge graphs' potential in unsupervised retrieval.

The second group of related work includes Entity-based Language Models [80] and Semantics-Enabled Language Model [37]. They construct entity-based text representations using entity annotations and match query and document in the entity space.

The Entity-based Language Models (`ELM`) approach by Raviv et al. uses language models to exact-match the entity-based text representations of query and documents [80]. They use standard entity linking systems, for example, TagMe [38] and Wikifier [79], to annotate the texts in the query and document automatically. The annotated entities form entity-based representations the same as words for bag-of-words. Standard language models are built upon the entity-based representations. They explore various ways to integrate the entity linking confidence information to the language models, as well as how to combine the entity-based language models with the original word-based ones. Their experiments demonstrate that the result language models are more effective than the phrase-based sequential dependency model, that entities provide extra information than enumerating phrases in the unsupervised exact match settings.

The Semantics-Enabled Language Model (`SELM`) by Ensan et al. enriches `ELM` with soft matches between entities [37]. Besides the exact match signals provided by the overlap of the entity-based representations, `SELM` introduces the correlation scores between entities to form soft matches between query and documents. The correlation scores are provided by the entity linkers, originally used to disambiguate entities. `SELM` aggregates the correlation scores between entities in the query and candidate documents under standard retrieval models. The soft match signals are then integrated with existing ranking signals by unsupervised retrieval models. Experiments show that the soft matches through entity correlation scores provide useful ranking signals.

Among those related approaches, `EQFE` and `LES` are related to our approach in Chapter 3. `LES` is the unsupervised version of `EsdRank` in Section 3.2. It often requires manual cleaning on the query entities so is not a suitable baseline for our fully-automatic and supervised methods [61]. `EQFE` is a main baseline throughout this dissertation. On the other hand, `ELM` and `SELM` fall into the category of entity-based text representations in Chapter 4. `ELM` was developed in parallel and is almost identical to the bag-of-entities model in Section 4.1. `SELM` is a special case of Explicit Semantic Ranking which is compared against in Section 4.2. It also requires manual cleaning of annotation noise.

Besides the related work about utilizing knowledge graphs in full-text search discussed above, there is other research related to the specific topics of the individual sections in this dissertations. These related works are discussed in detail in the corresponding sections: Section 3.1 discusses the related work in query expansion; Section 3.3 presents the related work in entity search with more details; Section 4.2 describes the related work in academic search and soft match based retrieval; and Chapter 6 presents the related work in term importance estimation.

2.5 Summary

This chapter first reviews controlled vocabularies and their usage in information retrieval. Then it provides an overview of modern knowledge graphs, their divergence and commonness with controlled vocabularies, and the scope of knowledge graphs in this dissertation. After that, it discusses the related semantic grounding techniques—entity search and entity linking. The last section presents several related works about using knowledge graphs to improve information retrieval.

Chapter 3

Enriching Query Representations with Knowledge Graphs

This chapter presents how to enrich query representations with knowledge graphs. As the first part of this dissertation research, we stayed in the traditional word-based retrieval framework and focused on improving it with knowledge graphs. Specifically, this chapter focuses on the query part, which is often short and not carefully written, thus ambiguous or insufficient to describe the information needs. The rich semantic information stored in knowledge bases, for example, synonyms, ontologies, entities, and relationships, gives search engine a different perspective to understand the query and provides a new opportunity for improving the retrieval accuracy.

In Section 3.1 we develop several query expansion methods that select terms from related entities in a knowledge base to expand the original query. Then Section 3.2 presents our `EsdRank` framework that connects query and documents using entities in a novel learning to rank model. In both sections, we used existing methods to find related entities for the query, and we found the quality of them plays a vital role in determining the final ranking performances. In last part of this chapter, Section 3.3 presents our research about how to better find related entities for a query with entity search.

3.1 Query Expansion with Knowledge Base

Query expansion techniques, which generate expansion terms to enhance the original query, have been widely used to find better term based query representations. This section presents a simple and effective method of using *Freebase*, one of the large scale modern knowledge graphs, to improve query representation using query expansion techniques[1]. We decompose the problem into two components. The first component identifies query-specific entities for query expansion. We present implementations that retrieve entities directly or select entities from retrieved documents. The second component uses information about these entities to select potential query expansion terms. We present implementations that select words with a tf.idf method or using category information. Finally, a supervised model is trained to combine information from multiple sources

[1]Chenyan Xiong and Jamie Callan. *Query Expansion with Freebase*. In Proceedings of the First ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR 2015) [98].

for better expansion.

Our experiments on the TREC Web Track ad-hoc task demonstrate that all our methods, when used individually, are about 20% more effective than previous state-of-the-art query expansion methods, including Pseudo Relevance Feedback (PRF) on Wikipedia [105] and supervised query expansion [17]. In addition to these improvements, experimental results show that our methods are more robust and have better *win/loss* ratios than state-of-the-art baseline methods, reducing the number of damaged queries by 50%. This makes query expansion using Freebase more appealing, because it is well-known that most query expansion techniques are 'high risk / high reward' insofar as they often damage as many queries as they improve, which is a considerable disadvantage in commercial search systems. The supervised model also successfully combines evidence from multiple methods, leading to 30% gains over the previous state-of-the-art. Besides being the first to improve query expansion this much on the widely used ClueWeb09 web corpus, the methods presented here are also fully automatic.

Section 3.1.2 discusses our new methods of using Freebase for query expansion. Experimental methodology and evaluation results are described in Sections 3.1.3 and 3.1.4 respectively. The last part of this section summarizes contributions.

3.1.1 Related Work in Query Expansion

Queries are usually short and not written carefully, which makes it more difficult to understand the intent behind a query and retrieve relevant documents. A common solution is query expansion, which uses a broader set of related terms to represent the user's intent and improve the document ranking.

Among various query expansion techniques, Pseudo Relevance Feedback (PRF) algorithms are the most successful. PRF assumes that top ranked documents for the original query are relevant and contain good expansion terms. For example, Lavrenko et al.'s RM model selects expansion terms based on their term frequency in top retrieved documents, and weights them by documents' ranking scores:

$$s(t) = \sum_{d \in D} p(t|d) f(q,d)$$

where D is the set of top retrieved documents, $p(t|d)$ is the probability that term t is generated by document d's language model, and $f(q,d)$ is the ranking score of the document provided by the retrieval model [54]. Later, Metzler added inverse document frequency (IDF) to demote very frequent terms:

$$s(t) = \sum_{d \in D} p(t|d) f(q,d) \log \frac{1}{p(t|C)} \tag{3.1}$$

where $p(t|C)$ is the probability of term t in the corpus language model C [34].

Another famous PRF approach is the Mixture Model by Tao et al. [92]. They assume the terms in top retrieved documents are drawn from a mixture of two language models: a query model θ_q and a background model θ_B. The likelihood of a top retrieved document d is defined as:

$$\log p(d|\theta_q, \alpha_d, \theta_B) = \sum_{t \in D} \log(\alpha_d \, p(t|\theta_q) + (1 - \alpha_d) \, p(t|\theta_B)).$$

α_d is a document-specific mixture parameter. Given this equation, the query model θ_q can be learned by maximizing the top retrieved documents' likelihood using EM. The terms that have non-zero probability in θ_q are used for query expansion.

Although these two algorithms have different formulations, they both focus on term frequency information in the top retrieved documents. So do many other query expansion algorithms [26, 55, 70, 108]. For example, Robertson et al.'s BM25 query expansion selects terms based on their appearances in relevant (or pseudo-relevant) documents versus in irrelevant documents [81]. Lee et al. cluster PRF documents and pick expansion terms from clusters [55]. Metzler and Croft include multi-term concepts in query expansion and select both single-term concepts and multi-term concepts by a Markov Random Field model [70].

The heavy use of top retrieved documents makes the effectiveness of most expansion methods highly reliant on the quality of the initial retrieval. However, web corpora like ClueWeb09 are often noisy, and documents retrieved from them may not generate reasonable expansion terms [8, 75]. Cao et al.'s study shows that top retrieved documents contain as many as 65% harmful terms [17]. They then propose a supervised query expansion model to select good expansion terms. Another way to avoid noisy feedback documents is to use a high-quality external dataset. Xu et al. proposed a PRF-like method on top retrieved documents from Wikipedia, whose effectiveness is verified in TREC competitions [75, 105]. Kotov and Zhai demonstrated the potential effectiveness of concepts related to query terms in ConceptNet for query expansion, and developed a supervised method that picks good expansion concepts for difficult queries [52].

Another challenge of query expansion is its 'high risk / high reward' property; often as many queries are damaged as improved. This makes query expansion risky to use in real online search service because users are more sensitive to failures than successes [23]. Collins-Thompson et al. [26] address this problem by combining the evidence from sampled sub-queries and feedback documents. Collins-Thompson also proposes a convex optimization framework to find a robust solution based on previous better-on-average expansion terms [24, 25]. The risk is reduced by improving inner difference between expansion terms, and enforcing several carefully designed constraints to ensure that expansion terms provide sufficient coverage of query concepts.

3.1.2 Expansion Using Freebase

In this section, we introduce our methods of using Freebase for query expansion. We first discuss our unsupervised expansion methods utilizing different information from Freebase. Then we propose a supervised query expansion method to combine evidence from our unsupervised methods.

3.1.2.1 Unsupervised Expansion Using Freebase

We perform unsupervised query expansion using Freebase in two steps: finding related entities and selecting expansion words. In finding related entities, we develop implementations that retrieve entities directly, or select them from annotations in top-ranked documents. In selecting expansion words, we also present two implementations: one uses the tf.idf information from entity descriptions; the other uses similarity of the query and the word distributions in Freebase's categories.

Formally, given a query q, and a ranked list of documents from initial retrieval $D = \{d_1, ...d_j..., d_N\}$, the goal of the entity finding step is to generate a ranked list of Freebase entities $E = \{e_1, ...e_k..., e_K\}$, with ranking scores $r(E) = \{r(e_1), ...r(e_k)..., r(e_K)\}$. The goal of word selection is to find a set of expansion words $W = \{w_1, ...w_i..., w_M\}$ and their scores $s(W) = \{s(w_1), ...s(w_i)..., s(w_M)\}$ from related entities using their descriptions $g(E) = \{g(e_1), ...g(e_k)..., g(e_K)\}$ and Freebase categories $C = \{c_1, ...c_u..., c_U\}$.

Finding Related Entities

Our first method retrieves entities directly. The query q is issued to the Google Freebase Search API[2] to get its ranking of entities E with ranking scores $r_s(E)$. The ranking score ranges from zero to several thousand, with a typical long-tailed distribution. We normalize them so that the ranking scores of each query's retrieved entities sum to one.

Our second approach selects related entities from the FACC1 annotations [39] in top retrieved documents. It is a common assumption that top retrieved documents are a good representation of the original query. Intuitively the entities that frequently appear in them shall convey meaningful information as well. We utilize such information by finding entities that are frequently annotated to top retrieved documents.

Specifically, for a query q's top retrieved documents D, we fetch their FACC1 annotations, and calculate the ranking score for entity e_k as:

$$r_f(e_k) = \sum_{d_j \in D} tf(d_j, e_k) \log \frac{|F|}{df(e_k)}. \tag{3.2}$$

In Equation 3.2, $tf(d_j, e_k)$ is the frequency of entity e_k in document d_j's annotations, and $df(e_k)$ is the total number of documents e_k is annotated to in the whole corpus. $|F|$ is the total number of documents in the corpus that have been annotated in the FACC1 annotation. $\frac{|F|}{df(e_k)}$ in Equation 3.2 serves as inverse document frequency (IDF) to demote entities that are annotated to too many documents. $r_f(e_k)$ is normalized so that ranking scores of each query's entities sum to one.

Selecting Expansion Words from Related Entities

We develop two methods to select expansion words from related entities.

PRF Expansion: The first method does tf.idf based Pseudo Relevance Feedback (PRF) on related entities' descriptions. PRF has been successfully used with Wikipedia articles [8, 75, 105]. It is interesting to see how it works with Freebase.

[2]The Google Search Freebase API is deprecated.

Given the ranked entities E and their ranking scores $r(E)$, the PRF score of the word w_i is calculated by:

$$s_p(w_i) = \sum_{e_k \in E} \frac{tf(g(e_k), w_i)}{|g(e_k)|} \times r(e_k) \times \log \frac{|G|}{df(w_i)} \tag{3.3}$$

where $tf(g(e_k), w_i)$ is the word frequency of w_i in the description of e, $|g(e_k)|$ is the length of the description, $df(w_i)$ is the document frequency of w_i in the entire Freebase description corpus G. $|G|$ is the total number of entities in Freebase that have a description.

Category-based Expansion: Our second word selection method uses Freebase categories. Freebase organizes entities in a multi-level ontology tree, which conveys the topic information of entities as well as their texts. We use the highest level in the ontology tree, such as $/people$ and $/movie$, to make sure sufficient instances exist in each category. There are in total $U = 77$ first level categories in Freebase. The descriptions of entities in these categories are training data to learn the language models used to describe these categories.

Our second approach selects words based on the similarities of their category distributions with the query's, which reflect their relatedness with the query in view of the Freebase ontology. The distribution of a word in Freebase categories is estimated using a Naive Bayesian classifier. We first calculate the probability of a word t_i generated by a category c_u via:

$$p(w_i|c_u) = \frac{\sum_{e_k \in c_u} tf(g(e_k), w_i)}{\sum_{e_k \in c_u} |g(e_k)|}$$

where $e_k \in c_u$ refers to the entity e_k belongs to the category c_u.

Assuming uniform prior distribution over Freebase categories C, the probability of word w_i belonging to category c_u is:

$$p(c_u|w_i) = \frac{p(w_i|c_u)}{\sum_{c_u \in C} p(w_i|c_u)}.$$

With the same uniform prior, the category distribution of a query q is:

$$p(c_u|q) = \frac{p(q|c_u)}{\sum_{c_u \in C} p(q|c_u)}.$$

Then, with the assumption that the categories of query words are independent with each other

$$p(q|c_u) = \prod_{w_i \in q} p(w_i|c_u).$$

These derivations together provide us the multinomial categorical distributions of a word w_i: $p(C|w_i) = \{p(c_1|w_i), ...p(c_u|w_i)..., p(c_U|w_i)\}$ and the query q: $p(C|q) = \{p(c_1|q), ...p(c_u|q)..., p(c_U|q)\}$.

The category-based expansion score $s_c(w_i)$ of the word w_i for the query q is then calculated by the negative Jenson-Shannon divergence between their categorical distributions, $p(C|w_i)$ and $p(C|q)$:

$$s_c(w_i) = -\frac{1}{2}\text{KL}(p(C|q)||p(C|q, w_i)) - \frac{1}{2}\text{KL}(p(C|w_i)||p(C|q, w_i))$$

Table 3.1: Unsupervised Query Expansion Methods Using Freebase.

	Found by Search	Found by FACC1
Select by PRF	`FbSearchPRF`	`FbFaccPRF`
Select by Category	`FbSearchCat`	`FbFaccCat`

where:

$$p(C|q, w_i) = \frac{1}{2}(p(C|q) + p(C|w_i))$$

and KL$(\cdot||\cdot)$ is the KL divergence between two distributions. $s_c(w_i)$ is the expansion score for a word w_i. We use a min-max normalization to re-range all $s_c(w_i)$ into $[0, 1]$.

As a result, we have two methods to find related Freebase entities to a query, and two methods to select expansion words from related entities. They together form four unsupervised expansion methods, as listed in Table 3.1.

3.1.2.2 Supervised Expansion Using Freebase

Different related entity finding and word selection algorithms have different strengths. Entity search finds entities that are directly related to the query by keyword matching. FACC1 annotation provides entities that are more related in meanings and does not require exact textual matches. In expansion word selection, PRF picks words that frequently appear in entities' descriptions. The category similarity method selects words that have similar distributions with the query in Freebase's categories. They together provide three scores describing the relationship between a query-word pair:

- `FbSearchPRF`: Pseudo Relevance Feedback score in retrieved entities;
- `FbFaccPRF`: Pseudo Relevance Feedback score in top retrieved documents' FACC1 annotations;
- `FbSearchCat` & `FbFaccCat`: The category-based expansion score using query-word category distribution similarity. `FbSearchCat` and `FbFaccCat` differ in candidate words but provide the exact same expansion scores.

The three scores are used as features for a supervised model, `FbSVM`, which learns how to select better expansion words. All words in related entities' descriptions are used as candidates for query expansion. The ground truth score for a candidate word is generated by its influence on retrieved documents, when used for expansion individually. If a word increases the ranking scores of relevant documents, or decreases the ranking scores of irrelevant documents, it is considered to be a good expansion word, and vice versa.

The influence of a word t_i over retrieved documents is calculated as:

$$\begin{aligned} y(w_i) = & \frac{1}{|D^+|} \sum_{d_j \in R} (f(q + w_i, d_j) - f(q, d_j)) \\ & - \frac{1}{|D^-|} \sum_{d_j \in D^-} (f(q + w_i, d_j) - f(q, d_j)) \end{aligned}$$

where D^+ and D^- are the sets of relevant and irrelevant documents in relevance judgments. $f(q, d_j)$ is the ranking score for document d_j and query q in the base retrieval model. $f(q{+}w_i, d_j)$ is the ranking score for d_j when the query is expanded using expansion word t_i individually. Binary labels are constructed using $y(w)$. Words with $y(w) > 0$ are treated as good expansion words and the rest as bad expansion words.

Our training label generation is a little different than Cao et al.'s [17]. Their labels were generated by a word's influence on documents' ranking positions: if relevant documents are moved up, or irrelevant documents are pushed down by a word, it is considered a good expansion word, otherwise a bad one. In comparison, we use influence on ranking scores which reflect an expansion word's effectiveness more directly. Our preliminary experiments also confirm that both their method and our method work better with our ground truth labels.

We used a linear SVM classifier to learn the mapping from the three features of a word w to its binary label. To get the expansion weights, we used the probabilistic version of SVM in the LibSVM [21] toolkit to predict the probability of a word being a good expansion word. The predicted probabilities are used as words' expansion scores, and those words with highest scores are selected for query expansion.

3.1.2.3 Ranking with Expansion Words

We use the selected expansion words and their scores to re-rank the retrieved documents with the RM model [54]:

$$f^*(d_j, q) = \alpha_q f(q, d_j) + (1 - \alpha_q)(\sum_{w_i \in W} s(w_i) f(w_i, d_j))). \tag{3.4}$$

In Equation 3.4, $f^*(q, d_j)$ is the final ranking score to re-rank documents. $f(q, d_j)$ and $f(w_i, d_j)$ are the ranking scores generated by the base retrieval model, e,g, BM25 or query likelihood, for query q and the expansion word w_i respectively. α_q is the weight on the original query. W is the set of selected expansion words and $s(w_i)$ is the expansion score of the word w_i. Expansion scores are normalized so that the scores of a query's expansion words sum to one.

3.1.3 Experimental Methodology

This section introduces our experimental methodology, including dataset, retrieval model, baselines, hyper-parameters, and evaluation metrics.

Dataset: Our experiments use ClueWeb09, TREC Web Track 2009-2012 adhoc task queries and the relevance judgments provided by TREC annotators. This dataset models a real web search scenario: queries are selected from the search log from Bing, and ClueWeb09 is a widely used web corpus. ClueWeb09 is known to be a hard dataset for query expansion [8, 31, 75], because it is much noisier than carefully edited corpora like the Wall-street Journal, news, and government web sets.

We use Category B of ClueWeb09 and index it using the Indri search engine [88]. Typical INQUERY stopwords are removed before indexing. Documents and queries are stemmed using the Krovetz stemmer [53]. Spam filtering is very important for ClueWeb09; we filter the 70% most spammy documents using the Waterloo spam score [27].

We retrieved Freebase entities and fetched their descriptions using the Google Freebase API on July 16th, 2014. Entity linking from documents to entities are found in FACC1 annotation [39]. Corpus statistics such as word IDF and categories' language models were calculated from the April 13th, 2014 Freebase RDF dump.

Retrieval Model: We use Indri's language model [34] as our base retrieval model. The ranking score of a document is the probability of its language model generating the query. Dirichlet smoothing is applied to avoid zero probability and incorporate corpus statistics:

$$p(q|d_j) = \frac{1}{|q|} \sum_{w_i \in q} \frac{tf(d_j, w_i) + \mu p(w_i|\mathcal{C})}{|d_j| + \mu}, \tag{3.5}$$

where $p(w_i|\mathcal{C})$ is the probability of seeing word w_i in the whole corpus, and μ is the parameter controlling the smoothing strength, set to the Indri default: 2500.

Baselines: We compare our four unsupervised expansion methods (as listed in Table 3.1) and the supervised method described in Section 3.1.2.2 (`FbSVM`) with several baselines. The first baseline is the Indri language model (`IndriLm`) as in Equation 3.5. All relative performances and Win/Loss evaluations of other methods are compared with `IndriLm` if without specific reference. Our second baseline is the Sequential Dependency Model (`SDM`) [69], a strong competitor in TREC Web Tracks.

We also include two well-known state-of-the-art query expansion methods as baselines. The first one is Pseudo Relevance Feedback on Wikipedia (`RmWiki`) [8, 75, 105]. We indexed the Oct 1st 2013 Wikipedia dump using the same setting we used for ClueWeb09. Standard Indri PRF with the IDF component [75] was performed to select expansion words.

The other query expansion baseline is `SVMPRF`, a supervised query expansion method [17]. We extracted the ten features described in their paper and trained an SVM classifier to select good expansion words. We used our word-level ground truth labels as discussed in Section 3.1.2.2, because their model performs better with our labels. Following their paper, the RBF kernel was used, which we also found necessary for that method to be effective.

For clarity and brevity, we do not show comparison with other methods such as RM3 [54], Mixture Model [92], or EQFE [31] because they all perform worse on ClueWeb09 than `RmWiki` and `SDM` in our experiment, previous TREC competitions [75], or in their published papers.

Parameter Setting: Hyper parameters in our experiment, including the number of expansion words (M), number of entities (K) in Freebase linked for expansion, and number of PRF documents for `RmWiki` and `SVMPRF`, are selected by maximizing the performance on training folds in a five-fold cross validation. The number of expansion words is selected from $\{1, 3, 5, 10, 15, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$, the number of entities is selected from $\{1, 3, 5, 10, 15, 20, 30, 40, 50\}$ and the number of PRF documents is selected from $\{5, 10, 15, 20, 25, 30\}$.

Parameters of SVM in supervised expansion (`FbSVM` and `SVMPRF`) are selected by another five-fold cross validation. In each of the five folds of the outside cross-validation that were used to select expansion parameters, we performed a second level cross-validation to select the parameters of SVM. The explored range of cost c of linear kernel and RBF kernel is $\{0.01, 0.1, 1, 10, 100\}$. The range of γ in RBF kernel is $\{0.1, 1, 10, 100, 1000, 10000\}$.

To keep the experiment tractable, other parameters were fixed following conventions in previous work [17, 75, 105]. The weight of the original query w_q is 0.5, and the re-rank depth is

Table 3.2: Performance of unsupervised expansion using Freebase. Relative gain is calculated using ERR over `IndriLm`. Win/Loss/Tie is the number of queries helped, hurt, and not changed comparing with `IndriLm`. †, ‡, § and ¶ mark the statistic significant improvements ($p < 0.05$) over `IndriLm`, `SDM`, `RmWiki` and `SVMPRF` respectively. The best results in each column are marked **bold**.

Method	**MAP@20**	**NDCG@20**	**ERR@20**	**Relative Gain**	**Win/Loss/Tie**
`IndriLm`	0.357	0.147	0.116	NA	NA
`SDM`	$0.387^{\dagger}$	$0.166^{\dagger}$	$0.122^{\dagger}$	5.52%	58/27/115
`RmWiki`	0.362	$0.161^{\dagger}$	0.114	−1.70%	67/72/61
`SVMPRF`	0.367	$0.158^{\dagger}$	0.125	8.00%	63/72/65
`FbSearchPRF`	$\mathbf{0.436}^{\dagger,\ddagger,\S,\P}$	$\mathbf{0.186}^{\dagger,\ddagger,\S,\P}$	$\mathbf{0.152}^{\dagger,\ddagger,\S,\P}$	30.80%	84/30/86
`FbSearchCat`	$0.421^{\dagger,\ddagger,\S,\P}$	$0.182^{\dagger,\ddagger,\S,\P}$	$0.144^{\dagger,\ddagger,\S,\P}$	23.99%	67/43/90
`FbFaccPRF`	$0.428^{\dagger,\ddagger,\S,\P}$	$0.184^{\dagger,\ddagger,\S,\P}$	$0.145^{\dagger,\ddagger,\S,\P}$	24.71%	97/55/48
`FbFaccCat`	$0.400^{\dagger,\S,\P}$	$0.173^{\dagger}$	$0.136^{\dagger,\ddagger,\S}$	17.25%	88/67/45

Table 3.3: Query level Win/Loss/Tie comparison between unsupervised query expansion methods. Each cell shows the number of queries helped (Win), damaged (Loss) and not changed (Tie) by row method over column method.

	`FbSearchPRF`	`FbSearchCat`	`FbFaccPRF`	`FbFaccCat`
`FbSearchPRF`	NA/NA/NA	73/47/80	82/74/44	95/65/40
`FbSearchCat`	47/73/80	NA/NA/NA	72/86/42	87/72/41
`FbFaccPRF`	74/82/44	86/72/42	NA/NA/NA	84/67/49
`FbFaccCat`	65/95/40	72/87/41	67/84/49	NA/NA/NA

1000. We chose re-ranking instead of retrieval again in the whole index because the latter is costly with the large set of expansion words and did not show any significant difference in our experiments. When using FACC1 annotations to link entities, we used the FACC1 annotations in the top 20 retrieved documents provide by `IndriLm`. The candidate words for `FbSVM` were generated from the top 20 retrieved entities and top 20 linked FACC1 annotations. To reduce noise in entity descriptions, we ignored words that contained less than three characters.

Evaluation Metric. Our methods re-ranked the top retrieved documents, so we mainly focus evaluation on the top 20 documents in the re-ranked list. We chose ERR@20 as our main evaluation metric, which is the primary metric of the TREC Web Track adhoc task. We also show the evaluation results for MAP@20 and NDCG@20.

3.1.4 Evaluation Results

This section first evaluates the unsupervised expansion methods. Then it presents results from the supervised expansion method. It concludes with case studies and discussions.

3.1.4.1 Performance of Unsupervised Expansion

The average performances on MAP, NDCG and ERR are shown in Table 3.2. The relative gain and Win/Loss ratio are compared with IndriLm on ERR. Statistical significance tests are performed using the permutation test. Labels †, ‡, § and ¶ indicate statistical significance ($p < 0.05$) over IndriLm, SDM, RmWiki and SVMPRF respectively.

Our unsupervised expansion methods outperform all state-of-the-art baselines by large margins for all evaluation metrics. All the gains over IndriLm are statistically significant, while SVMPRF and RmWiki are only significantly better on NDCG. Three of the methods, FbSearchPRF, FbSearchCat and FbFaccPRF, are significantly better than all baselines. FbFaccCat's improvements do not always pass the statistical significance test, even when the relative gains are almost 10%. This reflects the high variance of query expansion methods, which is addressed in Section 3.1.4.3.

Comparing the performances of our methods, linking entities by search works better than by FACC1 annotations, and selecting expansion words by PRF works better than using category similarity. One possible reason is that entities from FACC1 annotation are noisier because they rely on the quality of top retrieved documents. Also, the category similarity suffers because suitable categories for query or words may not exist.

We further compare our unsupervised expansion methods at the query level. The results are shown in Table 3.3. Each cell shows the comparison between the method in the row and the method in the column. The three numbers are the number of queries in which the row method performs better (win), worse (loss), and equally (tie) with the column method respectively. The results demonstrate that our methods do perform differently. The two most similar methods are FbSearchPrf and FbSearchCat, doing the same on 80 queries out of 200. But 36 queries have no returned entities from the Google Search API, on which two methods retreat to IndriLm. Otherwise, our four unsupervised methods perform the same for at most 49 queries.

These results showed the different strengths of our unsupervised methods. The next experiment investigates whether they can be combined for further improvements by a supervised method.

3.1.4.2 Performance of Supervised Expansion

The performance of our supervised method FbSVM, which utilized the evidence from our unsupervised methods, is shown in Table 3.4. To investigate whether the combination of multiple sources of evidence is useful, we conduct statistical significance tests between FbSVM with our unsupervised methods. †, ‡, § and ¶ indicates statistical significance in the permutation test over FbSearchPRF, FbSearchCat,FbFaccPRF and FbFaccCat correspondingly.

The results demonstrate that evidence from different aspects of Freebase can be combined for further improvements: FbSVM outperforms IndriLm by as much as 42%. Statistical significance is observed over our unsupervised methods on $NDCG$, but not always on MAP and ERR. We have also run statistical significance tests between FbSVM and all other baselines, which are all statistically significant as expected.

FbSVM and SVMPRF differ in their candidate words and features. FbSVM selects words from Freebase, while SVMPRF selects from web corpus. FbSVM uses features from Freebase's related

Table 3.4: Performance of supervised expansion using Freebase. Relative gain and Win/Loss/Tie are calculated comparing with `IndriLm` on ERR. †, ‡, § and ¶ mark the statistically significant improvements over `FbSearchPRF`, `FbSearchCat`,`FbFaccPRF` and `FbFaccCat` respectively. Best results in each column are marked **bold**.

Method	MAP@20	NDCG@20	ERR@20	Relative Gain	Win/Loss/Tie
`FbSearchPRF`	0.436	0.186	0.152	30.80%	84/30/86
`FbSearchCat`	0.421	0.182	0.144	23.99%	67/43/90
`FbFaccPRF`	0.428	0.184	0.145	24.71%	97/55/48
`FbFaccCat`	0.400	0.173	0.136	17.25%	88/67/45
`FbSVM`	**0.444**	**0.199**†,‡,§,¶	**0.165**‡,§,¶	42.42%	96/63/41

Table 3.5: Candidate word quality from top retrieved documents (Web Corpus) and related entities' descriptions (Freebase). Good and bad refer to the number of words that have positive and negative influences on ranking accuracy respectively. Their fractions are shown in percentages.

Source	Good Words		Bad Words	
`Web Corpus`	9,263	41.4%	13,087	58.6%
`Freebase`	19,247	39.6%	29,396	60.4%

entities' descriptions and categories, while `FbSVM` uses word distribution and proximity in top retrieved documents from web corpus. Table 3.5 shows the quality of candidate words from two sources. Surprisingly, Freebase' candidate words are slightly weaker in quality (39.4% vs. 41.4%), and there are more of them. However, `FbSVM`'s classification Precision is about 10% relatively better than `SVMPRF`, as shown in Table 3.6. The Recall of both methods is low due to the large number of 'good' expansion terms available in the top retrieved documents or entity descriptions. In general, `FbSVM` picks more good expansion words given the larger number of good candidate words in Freebase.

Nevertheless, the relative improvements of `FbSVM` over our best performing unsupervised method `FbSearchPRF` are not as high as expected. Our preliminary analysis shows that one possible reason is the features between query and words are limited, i.e., only three dimensions. Another possible reason is the way of using the supervised information (document relevance judgments). Document relevance judgments are used to generate labels at the word level using heuristics, while the final document ranking is still computed using unsupervised retrieval models. A more powerful machine learning framework seems necessary to better utilize Freebase information. This would be a good topic for further research.

Table 3.6: Classification performance of supervised query expansion.

Method	Precision	Recall
`SVMPRF`	0.5154	0.0606
`FbSVM`	0.5609	0.0400

3.1.4.3 Query Level Analysis

A common disadvantage of query expansion methods is their high variance: they often hurt as many queries as they helped. To evaluate the robustness of our methods, we compare the query level performance of each method versus `IndriLm` and record the Win/Loss/Tie numbers. The results are listed in the last columns of Tables 3.2 and 3.4. Table 3.2 shows that `SDM`, which is widely recognized as effective across many datasets, is reasonably robust and hurts only half as many queries as it helps. It also does not change the performance of 115 queries partly because 53 of them only contain one word on which nothing can be done by `SDM`. In comparison, `RmWiki` and `SVMPRF` hurt more queries than they help, which is consistent with observations in prior work [23].

Our methods have much better Win/Loss ratios than baseline query expansion methods. When selecting words using PRF from related entities' descriptions, `FbSearchPRF` and `FbFaccPRF` improve almost twice as many queries as they hurt. The variance of word selection by category is higher, but `FbSearchCat` and `FbFaccCat` still improve at least 30% more queries than they hurt. Linking by object retrieval has slightly better Win/Loss ratios than by FACC1 annotation, but it also helps a smaller number of queries. One reason is that for some long queries, there is no object retrieved by Google API.

More details of query level performance can be found in Figure 3.1. The x-axis is the bins of relative performances on ERR compared with `IndriLm`. The y-axis is the number of queries that fall into corresponding bins. If the performance is the same for a query, we put it into 0 bin. If a query is helped by 0 to 20%, we put it into bin 20%, etc. Figure 3.1 confirms the robustness of our methods. Especially for FbSearchPRF and FbFaccPRF, more queries are helped, fewer queries are hurt, and much fewer queries are extremely damaged.

`FbSVM`'s robustness is average among our expansion methods, and is better than `RmWiki` and `SDM`. Fewer queries fall into bin 0, as it is rare that none of our evidence affects a query. However, the number of damaged queries is not reduced. One possible reason is that the ground truth we used to train the SVM classifier is the individual performance of each candidate word, and only the average performance is considered in model training/testing. As a result, our model might focus more on improving average performance but not on reducing risk.

3.1.4.4 Case Study and Discussion

To further understand the properties of our object linking and word selection methods, Table 3.7 lists the queries that are most helped or hurt by different combinations of methods. The $\uparrow$ row shows the most helped queries and $\downarrow$ row shows those most hurt[3]. The comparison is done on `ERR` compared to `IndriLm` too.

Table 3.7 shows the different advantages of linking by object search and FACC1 annotation. For example, the query 'fybromyalgia' is damaged by `FbFaccPRF`, while improved by `FbSearchPRF` and `FbSearchCat`. The FACC1 annotation leads to a set of weakly related entities, like doctors and organizations focused on diseases, which generate overly-general expansion words. Instead, object search is precise and returns the object about 'fybromyalgia'.

[3]More details including related entities and expansion words are available at `http://boston.lti.cs.cmu.edu/appendices/ICTIR2015/`.

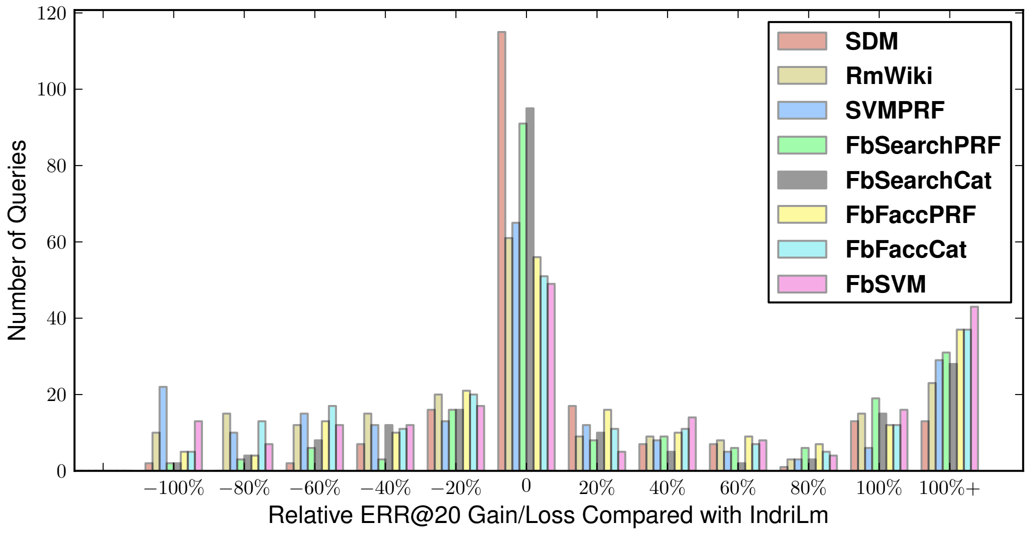


Figure 3.1: Query level relative performance. The x-axis is the bins of relative performance on ERR compared with `IndriLm`. Y-axis is the number of queries that fall into each bin. Bin 0 refers to queries that were not changed, 20% refers to queries that improved between $(0\%, 20\%]$, etc. The left-to-right ordering of histograms in each cell corresponds to the top-to-bottom ordering of methods shown in the key.

Sometimes the generality of FACC1 annotations can help instead. For example, for query ‘rock art’ whose main topic is about rock painting, object search links to entities about rock music, while FACC1 annotation is more general and links to related entities for both rock painting and rock music.

Our two word selection methods also have different behaviors. An exemplary case is the query ‘computer programming’, on which `FbSearchPRF` and `FbFaccPRF` perform very well, while `FbSearchCat` and `FbFaccCat` do not. The related entities of two methods are both reasonable: object search mostly links to programming languages, and FACC1 annotation brings in programming languages, textbooks, and professors. With the good quality of related entities, PRF selects good expansion words from their descriptions. However, category similarity picks words like: ‘analysis’, ‘science’, ‘application’ and ‘artificial’, which are too general for this query. The granularity of the Freebase ontology’s first level is too coarse for some queries, and lower levels are hard to use due to insufficient instances. Nevertheless, when the related entities are noisy, like for the query ‘sore throat’, the category information helps pick more disease-related words using the ‘/medicine’ category and provides better performance.

Some queries are difficult for all methods. For example, ‘wedding budget calculator’ contains the entities ‘wedding’, ‘budget’ and ‘calculator’, but refers to the concept ‘wedding budget’ and how to calculate it. Similar cases are ‘tangible personal property tax’ and ‘income tax return online’, whose meanings cannot be represented by a single Freebase object.

There are also queries on which Freebase is very powerful. For example, the query ‘UNC’ asks for the campuses of the University of North Carolina. Freebase contains multiple entities about UNC campuses, and campuses of other related universities, which generate good expan-

Table 3.7: The queries most helped and hurt by our methods. ↑ row shows the five most-helped queries for each method, and ↓ shows the most-hurt queries.

	`FbSearchPRF`	`FbSearchCat`
↑	porterville	unc
	hobby stores	porterville
	fybromyalgia	fybromyalgia
	computer programming	bellevue
	figs	figs
↓	von willebrand disease	rock art
	website design hosting	espn sports
	403b	ontario california airport
	ontario california airport	computer programming
	rock art	bobcat
	`FbFaccPRF`	`FbFaccCat`
↑	signs of a heartattack	porterville
	computer programming	idaho state flower
	figs	bellevue
	idaho state flower	flushing
	hip fractures	atari
↓	wedding budget calculator	poem in your pocket day
	poem in your pocket day	ontario california airport
	fybromyalgia	computer programming
	ontario california airport	bobcat
	becoming a paralegal	blue throated hummingbird

sion words. Freebase is also very effective for ‘Figs’, ‘Atari’, ‘Hoboken’ and ‘Korean Language’, whose meanings are described thoroughly by related Freebase entities.

To sum up, our object linking and word selection methods utilize different parts of Freebase and thus have different specialties. In object linking, object search is aggressive and can return the exact object for a query, when there are no ambiguities. FACC1 annotation relies on top retrieved documents and usually links to a set of related entities. Thus, it is a better choice for queries with ambiguous meanings. In word selection, Pseudo Relevance Feedback via tf.idf directly reflects the quality of related entities. It performs better when the related entities are reasonable. In contrast, category similarity offers a second chance to pick good expansion words from noisy related entities, when proper category definition exists for the query. `FbSVM` provides a preliminary way to combine the strength from different evidence and does provide additional improvements. Next in Section 3.2 we develop a more sophisticated method that better use supervision and richer evidence from Freebase, which further improves the ranking accuracy.

3.1.5 Summary of Query Expansion with Knowledge Base

This section uses Freebase, a large general domain knowledge graph, to improve query representation using query expansion techniques. We investigate two methods of identifying the entities associated with a query, and two methods of using those entities to perform query expansion. A supervised model combines information derived from Freebase descriptions and categories to select words that are effective for query expansion. Experiments on the ClueWeb09 dataset with TREC Web Track queries demonstrate that these methods are almost 30% more effective than strong state-of-the-art query expansion algorithms. In addition to improving average performance, some of these methods have better win/loss ratios than baseline algorithms, with 50% fewer queries damaged. To the best of our knowledge, this work is the first to show the effectiveness of Freebase for query expansion on the widely used ClueWeb09 web corpus.

3.2 EsdRank: Connect Query-Documents through Entities

The last section discussed our methods for enriching *word-based* query representations with query expansion techniques. Effective as they are, only the words in entity descriptions are used and only term frequency and ontology information from Freebase are utilized. This section presents `EsdRank`, a new technique that learns to connect the query and documents directly in the learning to rank framework[4]. `EsdRank` treats entities from a knowledge graph as latent entities connecting query and documents. The information from the knowledge graph is used as features between the query and entities. The document ranking evidence used in LeToR research is used as features between entities and documents. Instead of treating the features about query-entity and features about entity-document individually, `EsdRank` uses a *latent-listwise* LeToR model, Latent-ListMLE. The model treats the entities as a latent layer between query and documents, and learns how to handle the features between query, entities, and documents in one unified procedure directly from document relevance judgments.

One major challenge in using external data is to find related entities for a query and documents. Several methods have been used by prior research [31, 105] and by our prior work (Section 3.1), but it is not clear how each of them contributes to final performance. This section explores three popular methods to select related entities from external data, including query annotation, entity search, and document annotation. To investigate their effectiveness, we apply `EsdRank` with related entities generated by these methods on Freebase [13], and also the classic medical controlled vocabulary, MeSH [1], which is also considered as a special domain knowledge graph with a smaller scale, in web search and medical search respectively. Experiments on TREC Web Track and OHSUMED datasets show EsdRank's significant effectiveness over state-of-the-art ranking baselines, especially when using entities from query annotations. Experiments also show that the effectiveness not only comes from the additional information from external data, but also our Latent-ListMLE model that uses it properly.

In the rest of this section, Section 3.2.1 discusses EsdRank, its Latent-ListMLE ranking

[4] Chenyan Xiong and Jamie Callan. *EsdRank: Connecting Query and Documents through External Semi-Structured Data.* In Proceedings of the 24th ACM International Conference on Information and Knowledge Management (CIKM 2015) [97].

model, related entity selection, and features. The experimental methodology and evaluation results are described in Sections 3.2.2 and 3.2.3 respectively. The last part of this section summarizes the contribution of `EsdRank`.

3.2.1 EsdRank

EsdRank is intended to be a general technique for using external semi-structured data to improve ranking. External data elements are modeled as *entities*. An entity could be a term from another corpus or a knowledge graph entity. The evidence from the query, external data, and the corpus is incorporated as features to express the relationship between query, entity and document. A ranking model is then learned to rank documents with these entities and evidence.

In the first part of this section, we propose a novel latent listwise learning to rank model, Latent-ListMLE, as our ranking model. Latent-ListMLE handles the entities as a latent space between query and documents. The evidence between query-entity and entity-document is naturally expressed as features connecting the query to the latent space, and then connecting latent space to documents.

Prior research found that a major challenge in using external data is to find related entities [20, 31, 105]. In the second part of this section, we explore three popular related entity selection methods used in prior research. In the final part of this section, we describe the features used to connect query, entities and documents.

3.2.1.1 Latent-ListMLE

Given a query q and an initial set of retrieved documents D, a set of entities $E = \{e_1, ..., e_j, ..., e_m\}$ related to q and D is produced by one of the related entity selection methods as described in Section 3.2.1.2. Features that describe relationships between query q and entity e_j are denoted as vector v_j. Features that describe relationships between entity e_j and document d_i are denoted as u_{ij}. The goal of Latent-ListMLE, like other learning to rank methods, is to re-rank D, but with the help of the related entities E and feature vectors $U = \{u_{11}, ..., u_{ij}, .., u_{nm}\}$ and $V = \{v_1, ..., v_j, ..., v_m\}$.

Latent-ListMLE treats E as the latent space between q and D and uses V, U as features to describe the relationships between query-entity, and entity-document. We will first revisit ListMLE [96], the listwise learning to rank model which Latent-ListMLE is built upon. Then we discuss the construction, learning, and ranking of Latent-ListMLE.

ListMLE Revisited

ListMLE defines the probability of a ranking (a list) being generated by a query in a parametric model. Maximum likelihood estimation (MLE) is used to find parameters that maximize the likelihood of the best ranking(s). However, the sample space of all possible rankings is the permutation of all candidate documents D, which is too large. One contribution of ListMLE is that it reduces the sample space by assuming the probability of a document being ranked at position i is independent of those ranked at previous positions.

Specifically, with a likelihood loss and a linear ranking model, ListMLE defines the probability of a document d_i being ranked at position i as:

$$p(d_i|q, S_i) = \frac{\exp(w^T x_i)}{\sum_{k=i}^{n} \exp(w^T x_k)}, \tag{3.6}$$

where $S_i = \{d_i \ldots d_n\}$ are the documents that were not ranked in positions $1 \ldots i-1$, x_i is the query-document feature vector for d_i, and w is the parameter vector to learn.

Equation 3.7 defines the likelihood of a given ranking $\vec{D}$ of candidate documents.

$$p(\vec{D}|q; w) = \prod_{i=1}^{n} \frac{\exp(w^T x_i)}{\sum_{k=i}^{n} \exp(w^T x_k)}. \tag{3.7}$$

The parameter vector w is learned by maximizing the likelihood for all given queries q_k and their best rankings $\vec{D}_k^*$ given document relevance judgments:

$$\hat{w} = \arg\max_{w} \prod_{k} p(\vec{D}_k^*|q_k; w). \tag{3.8}$$

This is an unconstrained convex optimization problem that can be solved efficiently by gradient methods.

Latent-ListMLE Construction

Latent-ListMLE extends ListMLE by adding a latent layer in the ranking generation process. The latent layer contains related entities E as possible representations of the original query q.

With the latent entities E, the ideal generation process of a ranking $\vec{D}$ is to first sample a ranking of entities $\vec{E}$, and then sample document ranking $\vec{D}$ based on $\vec{E}$. However, this process is also impractical due to the huge sampling space. Similarly to ListMLE, we assume that the probabilities of picking entities and documents at each position are independent with those at previous positions. Thus, the generative process is redefined to be:

For each position from 1 to N:

1. Sample e_j from multinomial distribution $Multi(E|q)$, with probability $p(e_j|q)$; and
2. Sample d_i from multinomial distribution $Multi(S_i|e_j)$, with probability $p(d_i|e_j, S_i)$.

We further define:

$$p(e_j|q) = \frac{\exp(\theta^T v_j)}{\sum_{k=1}^{m} \exp(\theta^T v_k)} \tag{3.9}$$

$$p(d_i|e_j, S_i) = \frac{\exp(w^T u_{ij})}{\sum_{k=i}^{n} \exp(w^T u_{kj})}, \tag{3.10}$$

where v_j is the query-entity feature vector for e_j; u_{ij} is the entity-document feature vector between d_i and e_j; m is the number of entities; and θ and w are the model parameters.

In this generative process, the latent layer is the sampled entities produced by query q, and the document ranking probability is conditioned on the sampled entities instead of the query. With this extension, the probability of picking d_i at position i is:

$$\begin{aligned}
p(d_i|q, S_i) &= \sum_{j=1}^{m} p(d_i|e_j, S_i)p(e_j|q) && (3.11)\\
&= \sum_{j=1}^{m} \frac{\exp(w^T u_{ij})}{\sum_{k=i}^{n} \exp(w^T u_{kj})} \frac{\exp(\theta^T v_j)}{\sum_{k=1}^{m} \exp(\theta^T v_k)} && (3.12)
\end{aligned}$$

and the probability of ranking $\vec{D}$ given q is:

$$p(\vec{D}|q; w, \theta) = \prod_{i=1}^{n} \sum_{j=1}^{m} p(d_i|e_j, S_i)p(e_j|q). \tag{3.13}$$

Latent-ListMLE also uses query-document features by adding a 'query node' e_0 to E that represents query q. The features relating query q to e_0 are set to 0, thus, e_0 is the 'origin point' for related entities. The features relating e_0 to documents are typical LeToR query-document features. The combination of query-document features and entity-document features is done by treating them as individual dimensions in U, and setting missing feature dimensions' values to zero. So the dimensions referring to entity-document similarities for the query node are set to zero, and vice versa.

Our idea of representing the query via related entities ($p(e|q)$) is similar to query expansion. In fact, if we use expansion terms as our related entities, and discard the document ranking part, Latent-ListMLE becomes a supervised query expansion method. On the other hand, if we only use the query node as the related entity, Latent-ListMLE is exactly the same as ListMLE. The difference is that, in Latent-ListMLE, the connections from query to latent layer, and from latent layer to documents are learned together in one unified procedure, which finds the best combination of query representation ($p(e|q)$) and document ranking ($p(d|e)$) together, instead of only focusing on one of them.

Learning

The parameters w and θ are learned using MLE with given queries and their best rankings. To keep the notation clear, we present the derivation with one training query q, without loss of generality.

For a training query q and its best ranking $\vec{D^*}$ derived from relevance labels, their log likelihood is:

$$\begin{aligned}
l(\vec{D^*}|q; w, \theta) &= \log p(\vec{D^*}|q; w, \theta) && (3.14)\\
&= \sum_{i=1}^{n} \log \sum_{j=1}^{m} \left(\frac{\exp(w^T u_{ij})}{\sum_{k=i}^{n} \exp(w^T u_{kj})} \times \frac{\exp(\theta^T v_j)}{\sum_{k=1}^{m} \exp(\theta^T v_k)} \right). && (3.15)
\end{aligned}$$

The goal of MLE is to find parameters w^*, θ^* such that:

$$w^*, \theta^* = \arg\max_{w,\theta} l(\vec{D^*}|q; w, \theta). \tag{3.16}$$

Directly maximizing Equation 3.15 is difficult due to the summation of the latent variables inside the log, thus we use the EM algorithm to solve this optimization problem.

The **E step** finds the posterior distribution of hidden variable e_j for each ranking position given the current parameters θ_{old} and w_{old}.

$$\begin{aligned}\pi(e_j|d_i, q) &= p(e_j|d_i, q; \theta_{old}, w_{old}) \\ &= \frac{p(d_i|e_j, S_i; w_{old})p(e_j|q; \theta_{old})}{\sum_{k=1}^{m} p(d_i|e_k, S_i; w_{old})p(e_k|q; \theta_{old})}.\end{aligned}$$

The **M step** maximizes the expected log complete likelihood.

$$\begin{aligned}E(\tilde{l}) =& E_{\pi(\vec{E}|\vec{D^*},q)} \log p(\vec{D^*}, \vec{E}|q; w, \theta) \\ =& \sum_{i=1}^{n}\sum_{j=1}^{m} \pi(e_j|d_i, q) \log p(d_i|e_j, S_i)p(e_j|q) \\ =& \sum_{i,j} \pi(e_j|d_i, q) \log \frac{\exp(w^T u_{ij})}{\sum_{k=i}^{n} \exp(w^T u_{kj})} \frac{\exp(\theta^T v_j)}{\sum_{k=1}^{m} \exp(\theta^T v_k)}.\end{aligned}$$

We use gradient ascent to maximize the expectation. The gradients are:

$$\frac{\partial E(\tilde{l})}{\partial w} = \sum_{i,j} \pi(e_j|d_i, q)\{u_{ij} - \frac{\sum_{k=i}^{n} u_{kj} \exp w^T u_{kj}}{\sum_{k=i}^{n} \exp w^T u_{kj}}\} \tag{3.17}$$

$$\frac{\partial E(\tilde{l})}{\partial \theta} = \sum_{i,j} \pi(e_j|d_i, q)\{v_j - \frac{\sum_{k=1}^{m} v_k \exp(\theta^T v_k)}{\sum_{k=1}^{m} \exp(\theta^T v_k)}\}. \tag{3.18}$$

The E step is very efficient with the closed form solution. The M step is an easy convex optimization problem. Intuitively, the E step finds the best assignment of entity probabilities under current parameters and best document rankings, thus transferring document relevance information to latent entities. The M step learns the best parameters that fit the entity probabilities provided by the E step. EM iterations are guaranteed to improve the likelihood until convergence. However, the overall optimization is not convex and has local optima. In practice, the local optima problem can be suppressed by repeating the training several times with random initial w and θ, and using the result that converges to the largest likelihood.

Ranking

Given a learned model, a query, and an initial ranking, a new ranking is constructed by picking the document that has the highest probability $p(d_i|q, S_i)$ at each position from 1 to n. The complexity of picking one document is $O(nm)$, thus the total cost of ranking n documents with m related entities is $O(n^2m)$, which is slower than ListMLE's $O(n \log n)$ ranking complexity. We can restrict the number of documents n (e.g. 100) and related entities m (e.g. < 5) to maintain reasonable efficiency.

3.2.1.2 Related Entities

How to find related entities E given query q and documents D is important for using external data. Many options have been proposed by prior research [31, 62, 77, 105], but it is not clear which is the most reliable. This research studies the following three popular automatic methods to find related entities.

Query Annotation selects the entities that directly appear in the query. Based on specific entity types in the external data, one can choose corresponding techniques to 'annotate' them to the query, for example, entity linking techniques [20] to find entities that appear in the query, or all query terms when the entities are terms from an external corpus.

Entity Search selects the entities that are textually similar to the query. Search engines can be used to find such entities. We can build an index for all the entities in external data, in which each document is the textual data about the entity, for example, name, alias, description and context words of an entity. Then textually similar entities can be retrieved by running the query on the index.

Document Annotation selects the entities that appear in retrieved documents D. The terms in retrieved documents have been widely used in query expansion. The entities that are annotated to D were also useful in prior work [31]. This method introduces entities that are more indirectly related to the query, such as 'President of United States' for the query 'Obama family tree'. A typical method to score and select entities from retrieved documents is the RM3 pseudo relevance feedback model [54].

3.2.1.3 Features

Representing the relationship between query and related entities is the major focus of prior research in using external data. We explore the following query-entity features in EsdRank.

Features between Query and Entities

Entity Selection Score features are the scores produced by the entity selection step: Annotator confidence, entity ranking score, and RM3 model score $s(q, e)$.

Textual Similarity features cover the similarities between the query and the entity's textual fields. For example, one can use coordinate matches, BM25 scores, language model scores, and sequential dependency model (SDM) scores between the query and the entity's textual fields, such as name, alias (if any) and descriptions if using entities.

Ontology Overlap features use the ontology, a common type of information for some external semi-structured datasets. When an ontology is available, one can build a multiclass classifier that classifies the query into the ontology, and use the overlaps with the entity's ontology as features.

Entity Frequency is the number of documents in the corpus the entity appears in (e.g. term) or is annotated to (e.g. entity). This feature is query independent and distinguishes frequent entities from infrequent ones.

Similarity with Other Entities are the max and mean similarity of this entity's name with the other related entities'. These features distinguish entities that have similar names with other related entities from those that do not.

Features between Entities and Documents

In learning to rank, rich query-document ranking features, such as multiple retrieval algorithms on different document fields, have been shown very important for ranking performance [59]. Our entity-document features are similar to query-document features widely used in LeToR research. But entities may have richer contents than queries, enabling a larger set of features.

Textual Similarity features measure the similarity of a document and an entity's textual fields. BM25, language model, SDM, coordinate match, and cosine similarity in the vector space model are used to calculate similarities between all combinations of an entity's textual fields and a document's fields. These retrieval models are applied by using the entity to 'retrieve' the document. The fusion of these similarity features provides multiple ways to express the entity-document similarities for EsdRank.

Ontology Overlap features are the same as the ontology overlap features between query and entity. The same multiclass classifier can be used to classify documents, and overlaps in the entity's and document's top categories can be used as features.

Graph Connection features introduce information about the relationships in the external data and document annotations. If such graph-like information is available, one can start from the entity, traverse its relations, and record the number of annotated entities reached at each step (usually within 2 steps) as features. These features model the 'closeness' of the entity and the document's annotations in the external data's relationship graph.

Document Quality features are commonly used in learning to rank. We use classic document quality features such as document length, URL length, spam score, the number of inlinks, stop word fraction, and whether the document is from Wikipedia (web corpus only).

3.2.1.4 Discussion

EsdRank provides general guidance about how to use external semi-structured data in ranking. When an external dataset is available, to use it in ranking, one can first use entity selection methods to find related entities for each query, and then extract query-entity and entity-document features. Although the detailed entity selection methods and features may vary for different datasets, we list some common related entity selection methods and features that have been used widely in prior research to start with. One can also derive new related entity selection methods and features based on other available information.

Related entities and features are handled by our latent listwise LeToR model, Latent-ListMLE. It models the entities as the latent space. As a result, the evidence is decoupled into a query-entity part and an entity-document part, which is easier to learn than mixed together (as shown in Section 3.2.3.2). Instead of solely focusing on the query-entity part, or the document ranking part, Latent-ListMLE learns them together using EM. Our experiments show that the additional evidence from external data and Latent-ListMLE are both necessary for EsdRank's effectiveness.

3.2.2 Experimental Methodology

EsdRank is intended to be a general method of using external semi-structured data, and experiments test it with two types of semi-structured data and search tasks: Web search using the Freebase knowledge base, and medical search using the MeSH controlled vocabulary. The former uses a newer type of semi-structured data, a noisy corpus, and queries from web users; the latter uses a classic form of semi-structured data, a clean corpus, and queries from domain experts. Datasets and ranking features are determined by these two choices, as described below.

Data. Experiments on web search using Freebase were conducted with ClueWeb09-B and ClueWeb12-B13, two web corpora used often in IR research. Each corpus contains about 50 million English web documents. The 2009-2013 TREC Web Tracks provided 200 queries with relevance assessments for ClueWeb09 and 50 queries with relevance assessments for ClueWeb12. We used a snapshot of Freebase provided by Google on Oct 26th, 2014.[5]

Experiments on medical search using the MeSH controlled vocabulary were conducted with the OHSUMED corpus, a medical corpus used often in IR research. It contains abstracts of medical papers that are manually annotated with MeSH terms. The OHSUMED dataset includes 106 queries with relevance assessments. We used the 2014 MeSH dump provided by the U.S. National Library of Medicine (NIH).[6]

Indexing and Base Retrieval Model. The ClueWeb and OHSUMED corpora were indexed by the Indri search engine, using default stemming and stopwords. ClueWeb documents contain title, body and inlink fields. OHSUMED documents contain title and body fields.

Freebase and MeSH also were indexed by Indri, with each entity treated as a document containing its associated text fields (name, alias, description). Entities without descriptions were discarded. Retrieval was done by Indri using its default settings.

Spam filtering is important for ClueWeb09. We removed the 70% spammiest documents using Waterloo spam scores [27]. Prior research questions the value of spam filtering for ClueWeb12 [31], thus we did not filter that dataset.

Related Entities. The three related entity generation methods in Section 3.2.1.2 are used:

Query annotation (`AnnQ`) was done by TagMe [38], one of the best systems in the Short (Query) Track at the ERD14 workshop competition [20]. TagMe annotates the query with Wikipedia page ids. Wikipedia page ids were mapped to Freebase entities using their Wikipedia links and to MeSH controlled vocabulary terms using exact match.

Entity search (`Osrch`) was done with the Indri search engine for both external semi-structured datasets. We investigated using Google's Freebase Search API to search for Freebase entities and PubMed's MeSH query annotations to annotate MeSH entities. Neither was significantly better than Indri search or TagMe annotation. Thus, we only report results with public solutions.

Google's FACC1 dataset provided annotations of Freebase entities in ClueWeb documents[7] [39]. The OHSUMED dataset contains MeSH annotations for each document. Document annotation (`AnnD`) entities are generated by first retrieving documents from the ClueWeb or OHSUMED corpus, and then using the RM3 relevance model [54] to select annotated entities

[5]https://developers.google.com/freebase/data

[6]http://www.nlm.nih.gov/mesh/filelist.html

[7]http://lemurproject.org/clueweb09/

Table 3.8: Query-Entity and Entity-Document Features. 'Web' and 'Medical' list the feature dimensions in each corpus. 'Field combinations' refers to combinations of an entity's and a document's fields. An entity's fields include name, alias, description, query minus name, query minus alias, and query minus description. ClueWeb documents (for Fb) contain title, body and inlink fields. OSHUMED documents (for MeSH) contain title and body fields.

Query-Entity Feature	**Web**	**Medical**
Score from Query Annotation	1	1
Score from Entity Search	1	1
Score from Document Annotation	1	1
Language model of entity's description	1	1
SDM of entity's description	1	1
BM25 of entity's description	1	1
Coordinate match of entity's text fields	3	3
Top 3 categories overlap	1	1
Entity's idf in corpus's annotation	1	1
Similarity with other entities	2	2
Total Dimensions	13	13

Entity-Document Feature	**Web**	**Medical**
Language model of field combinations	18	12
SDM of field combinations	18	12
Cosine correlation of field combinations	18	12
Coordinate match of field combinations	18	12
BM25 of field combinations	18	12
Top 3 cateogories overlap	1	1
Is annotated to document	1	1
Connected in graph at 1,2 hop	2	0
Total Dimensions	94	62

from these documents.

Thus, the experiments consider query annotation (`EsdRank-AnnQ`), entity search (`EsdRank-Osrch`) and document annotation (`EsdRank-AnnD`) methods to select related entities.

Feature Extraction. We extract features described in Section 3.2.1.3. Our feature lists are shown in Tables 3.8–3.9.

For entity-document text similarity features, each entity field is dynamically expanded with a virtual field that contains any query terms that the field does not contain. For example, given an entity with name 'Barack Obama' and query 'Obama family tree', the text 'family tree' is added to the entity as a virtual field called 'query minus name'. Similar expansion is performed for 'query minus alias' and 'query minus description' fields. As a result, Latent-ListMLE also knows which part of the query is not covered by an entity. This group of features is very important for long queries, in order to make sure documents only related to a small fraction of the query are not promoted too much.

Table 3.9: Query-Document and document quality features used by EsdRank and learning to rank baselines. 'Web' and 'Medical' indicate the number of features in each corpus.

Feature	Web	Medical
Language model of doc's fields	3	2
SDM of doc's fields	3	2
Cosine correlation of doc's fields	3	2
Coordinate match of doc's fields	3	2
BM25 of doc's fields	3	2
Spam Score	1	0
Number of inlinks	1	0
Stop word fraction	1	1
URL length	1	0
Document length	1	1
Is Wikipedia	1	0
Total Dimensions	21	12

The ontology features use the linear multiclass SVM implementation of SVMLight [48] as the classifier. The Freebase classes are the 100 most frequent (in the FACC1 annotation) bottom categories in Freebase's two-level ontology tree. The classes for MeSH are all of the top level categories in MeSH's ontology. The training data is the descriptions of entities in each class. Entities are classified by their descriptions. Documents are classified by their texts. Queries are classified using voting by the top 100 documents that Indri retrieves for them [16]. The accuracy of Multiclass SVM in cross-validating on training data is above 70%.

The graph connection features between Freebase entities and ClueWeb documents are calculated using Freebase's knowledge graph. We treat all edges the same, leaving further exploration of edge type to future work. We only use the reachable at zero hop for OHSUMED, that is whether the MeSH term is annotated to the document.

Evaluation Metrics. We use ERR@20 and NDCG@20, which are the main evaluation metrics in the TREC Web Track ad hoc task. Besides these two metrics that focus on the top part of the ranking, we also use MAP@100, which considers the quality over the entire reranked section.

Statistical Significance was tested by the permutation test (Fisher's randomization test) with p-value < 0.05, which is a common choice in IR tasks.

Hyper-Parameters and Training. We follow standards in prior work to set hyper-parameters. The number of documents retrieved and re-ranked is set to 100. The same set of documents is used to generate document annotation `AnnD`. All supervised ranking models are evaluated on the testing folds in 10-fold cross validation. Cross-validation partitioning is done randomly and kept the same for all experiments. To suppress the local optima problem, in each cross-validation fold Latent-ListMLE is trained ten times with random initialization. The training result with the largest likelihood on training data is used for testing. In order to maintain reasonable ranking efficiency and avoid too many noisy entities, the number of related entities from `AnnD` and `Osrch` are restricted to at most 3. This restriction does not change `AnnQ` as none of our queries has more than 3 annotations.

Baselines. The first baseline is the Sequential Dependency Model (SDM) [70] approach which is widely recognized as a strong baseline. For ClueWeb datasets, we find that Dalton et al's SDM results obtained with Galago[8] are stronger than our SDM results obtained with Indri, thus we use their SDM results as a baseline. We also use EQFE rankings provided by them as the second baseline on ClueWeb data sets. On OHSUMED, we use the Indri query language to obtain SDM results. EQFE is not included for OHSUMED as it is specially designed for knowledge bases. The third and fourth baselines are two state-of-the-art learning to rank baselines: RankSVM (pairwise) [49] and ListMLE (listwise) [96]. Their features are shown in Table 3.9 and are trained and tested exactly the same as EsdRank.

Three of our baseline algorithms were used widely in prior research; the fourth (EQFE) is included because it has important similarities to EsdRank. We could have included other methods that use external data, for example, Wikipedia for query expansion [15, 105], Wikipedia as one resource for query formulations [10], and MeSH as an alternate document representation [66]. These methods mainly focus on using external data to better represent the query; they rank documents with unsupervised retrieval models. In recent TREC evaluations, supervised LeToR systems that use rich ranking features have shown much stronger performance than unsupervised retrieval systems. Thus, we mainly compare with LeToR models with document ranking features, e.g., those in Table 3.9, which we believe are the strongest baselines available now.

3.2.3 Evaluation Results

This section first presents experimental results about EsdRank's overall accuracy. Then it investigates the effectiveness of our Latent-ListMLE model, together with the influence of related entity quality on EsdRank's performance.

3.2.3.1 Overall Performance

Tables 3.10a, 3.10b, and 3.10c show the retrieval accuracy of several baseline methods and three versions of EsdRank on ClueWeb09, ClueWeb12, and OHSUMED datasets. The three variants of EsdRank differ only in how related entities are selected. $\dagger, \ddagger$, $\S$ and ¶ indicate statistical significance over SDM, RankSVM, ListMLE and EQFE. The change relative to ListMLE is shown in parentheses. Win/Tie/Loss is the number of queries improved, unchanged or damaged as compared to ListMLE by ERR@20. The best performing method according to each evaluation metric is marked **bold**.

On all three data sets, the best EsdRank methods outperform all baselines on all metrics. The gain over ListMLE varies from 3.33% (OHSUMED, MAP@100) to 21.85% (ClueWeb09, ERR@20), with 5–7% being typical. We note that ListMLE is a very strong baseline; there is little prior work that provides statistically significant gains of this magnitude over ListMLE on the ClueWeb datasets. These improvements show the effectiveness of external data in ranking, which is EsdRank's only difference with ListMLE.

On ClueWeb data sets, all three versions of EsdRank outperform EQFE, and the best performing version improves over EQFE by 10%-30%. Both EQFE and EsdRank use Freebase

[8]http://ciir.cs.umass.edu/downloads/eqfe/runs/

Table 3.10: Performance of EsdRank. AnnQ, Osrch and AnnD refer to the entity selection methods: Query annotation, entity search and document annotation. Relative changes compared to ListMLE are shown in parentheses. Win/Tie/Loss is the number of queries improved, unchanged or hurt, compared to ListMLE. †, ‡, § and ¶ show statistic significance over SDM†, RankSVM‡, ListMLE§ and EQFE¶. The best method for each evaluation metric is marked **bold**. SDM scores are from Dalton et al. [31] which is a highly-tuned Galago implementation and performs better than Indri's SDM in Table 3.2.

(a) ClueWeb09

Method	**MAP@100**	**NDCG@20**	**ERR@20**		**Win/Tie/Loss**
SDM	0.375	0.214	0.135	(−13.52%)	75/30/95
EQFE	0.364	0.213	0.139	(−11.21%)	77/27/96
RankSVM	0.352†	0.193†	0.147	(−6.10%)	59/37/104
ListMLE	0.410†,‡,¶	0.221†,‡	0.156†	–	–
EsdRank-AnnQ	**0.437**†,‡,§,¶	**0.237**†,‡,§,¶	**0.190**†,‡,§,¶	(**21.85**%)	88/42/70
EsdRank-Osrch	0.397†,‡,¶	0.219†,‡	0.155†	(−0.75%)	71/47/82
EsdRank-AnnD	0.403†,‡,¶	0.221†,‡	0.167†,¶	(6.70%)	75/36/89

(b) ClueWeb12

Method	**MAP@100**	**NDCG@20**	**ERR@20**		**Win/Tie/Loss**
SDM	0.293	0.126	0.091	(−25.66%)	18/7/25
EQFE	0.319	0.146†	0.106	(−13.61%)	17/7/26
RankSVM	0.383†	0.161†	0.114	(−7.48%)	16/13/21
ListMLE	0.397†,¶	0.174†,¶	0.123†	–	–
EsdRank-AnnQ	0.410†,¶	0.181†,¶	**0.133**†,‡,§,¶	(**8.39**%)	20/12/18
EsdRank-Osrch	0.391†,¶	0.167†	0.121†	(−1.26%)	22/10/18
EsdRank-AnnD	**0.440**†,‡,§,¶	**0.186**†,‡,§,¶	0.132†,¶	(7.67%)	24/14/12

(c) OHSUMED

Method	**MAP@100**	**NDCG@20**	**ERR@20**		**Win/Tie/Loss**
SDM	0.413	0.332	0.453	(−6.14%)	38/11/57
RankSVM	0.396	0.336	0.453	(−6.12%)	43/14/49
ListMLE	0.414	0.343	0.483	–	–
EsdRank-AnnQ	**0.428**‡,§	**0.355**†,‡,§	**0.511**†,‡,§	(**5.77**%)	56/16/34
EsdRank-Osrch	0.427‡,§	0.347	0.489‡	(1.29%)	44/17/45
EsdRank-AnnD	0.416	0.342	0.500†,‡	(3.59%)	44/20/42

as external data and learning to rank models to rank documents. The features between query and entity in EsdRank are very similar with those used in EQFE. Why does EsdRank perform better than EQFE?

One difference is in the entity-document features. EQFE uses the language model or SDM score between an entity's textual fields and the whole document to connect the entity to the document, while EsdRank uses a much larger feature set (Table 3.8). The entity-document features in Table 3.8 provide multiple different views of entity-document relevancy, which has been shown very effective in LeToR research [59]. More entity-document features also better propagate document relevance judgments to latent entities in Latent-ListMLE's EM training procedure, and help it learn better weights for query-entity features. We have tried EsdRank with only language model scores between an entity's textual fields and the document as entity-document features, however, results with this smaller feature set are no better than EQFE. In Section 3.2.3.2, our second experiment also shows that our Latent-ListMLE model is another essential factor for EsdRank's performance with its ability to handle features in the two parts properly.

On ClueWeb09 and OHSUMED, query annotation (EsdRank-AnnQ) performs the best with statistically significant improvements on almost all evaluations over all baselines, indicating query annotation is still the most effective in finding reliable entities. On ClueWeb12, EsdRank-AnnQ also outperforms all baselines, although less statistical significance is observed due to fewer training queries (only 50). It is interesting to see that EsdRank-AnnD performs better than EsdRank-AnnQ on ClueWeb12. This is consistent with results of EQFE [31], showing that FACC1 annotation might be of better quality for ClueWeb12 queries than ClueWeb09 queries. EsdRank-Osrch performs the worst on all three data sets. The reason is that entity search provides related entities that might be textually similar to the query, but are actually noise. The ranking models designed to rank document use assumptions that may not be suitable for entities, leaving room for improvement in future research.

The different effectiveness of EsdRank with different related entity selection methods shows the importance of related entity quality. It is also well recognized as one of the most essential factors in determining the usefulness of external data in prior research [20, 32, 62] (Section 3.1). In Section 3.2.3.3, our third experiment studies the correlation between related entities' quality and EsdRank's performance.

3.2.3.2 Effectiveness of Latent-ListMLE

To investigate the effectiveness of Latent-ListMLE, in this experiment, we compare it with ListMLE, using the same external evidence, but with features generated by feature engineering techniques similar to those in Dalton et al. [31].

Formally, for each query q and document d_i pair, we have entities $\{e_1, ..., e_j, ..., e_m\}$, query-entity features $V = \{v_1, ..., v_j, ..., v_m\}$ and entity-document features $U_i = \{u_{i1}, ..., u_{ij}, ..., u_{im}\}$. Let $|u|$ and $|v|$ be the feature dimensions for query-entity features and entity-document features. We generate $|u| \times |v|$ features $x_i = \{x_i(1,1), ..., x_i(k_1,k_2), ..., x_i(|u|,|v|)\}$ by enumerating all combination of features in U_i and V. The combination of k_1-th feature in V and k_2-th feature U_i is defined as: $x_i(k_1,k_2) = \sum_j v_j(k_1) \times u_{ij}(k_2)$, the summation of corresponding features over all possible related entities. One may notice that if we only use language model scores as entity-document features, this set up is exactly the same as EQFE. We name these 'flat' features in comparison with Latent-ListMLE's latent space model.

We put these 'flat' features into ListMLE and use the same training-testing procedure to evaluate them on our datasets. Evaluation results are shown in Table 3.11. ListMLE-AnnQ, ListMLE-Osrch and ListMLE-AnnD refer to the flat model with features from related en-

Table 3.11: Performance of `ListMLE` with flat features derived from external data. `AnnQ`, `Osrch` and `AnnD` refer to entity selection methods: Query annotation, entity search and document annotation. Relative changes and Win/Tie/Loss are compared to `ListMLE`. $\triangledown$ and $\blacktriangledown$ indicate statistic significantly weaker performance compared to `ListMLE` and `EsdRank` (which uses the same information but `Latent-ListMLE`).

(a) ClueWeb09

Method	**MAP@100**	**NDCG@20**	**ERR@20**
`ListMLE-AnnQ`	$0.389^{\blacktriangledown}$ (-4.96%)	$0.208^{\blacktriangledown}$ (-5.88%)	$0.170^{\blacktriangledown}$ (8.99%)
`ListMLE-Osrch`	$0.331^{\triangledown,\blacktriangledown}$ (-19.33%)	$0.168^{\triangledown,\blacktriangledown}$ (-23.79%)	$0.127^{\triangledown,\blacktriangledown}$ (-18.34%)
`ListMLE-AnnD`	$0.357^{\triangledown,\blacktriangledown}$ (-12.79%)	$0.197^{\triangledown,\blacktriangledown}$ (-10.63%)	0.155 (-0.61%)

(b) ClueWeb12

Method	**MAP@100**	**NDCG@20**	**ERR@20**
`ListMLE-AnnQ`	$0.355^{\blacktriangledown}$ (-10.71%)	$0.137^{\triangledown,\blacktriangledown}$ (-21.04%)	$0.098^{\triangledown,\blacktriangledown}$ (-20.37%)
`ListMLE-Osrch`	0.368 (-7.43%)	0.154 (-11.32%)	0.105 (-14.39%)
`ListMLE-AnnD`	$0.337^{\triangledown,\blacktriangledown}$ (-15.06%)	$0.149^{\triangledown,\blacktriangledown}$ (-14.38%)	$0.098^{\triangledown,\blacktriangledown}$ (-20.65%)

(c) OSHUMED

Method	**MAP@100**	**NDCG@20**	**ERR@20**
`ListMLE-AnnQ`	$0.388^{\triangledown,\blacktriangledown}$ (-6.13%)	$0.323^{\triangledown,\blacktriangledown}$ (-5.64%)	$0.444^{\triangledown,\blacktriangledown}$ (-7.94%)
`ListMLE-Osrch`	$0.400^{\blacktriangledown}$ (-3.37%)	$0.323^{\triangledown,\blacktriangledown}$ (-5.58%)	$0.445^{\triangledown,\blacktriangledown}$ (-7.84%)
`ListMLE-AnnD`	$0.397^{\triangledown,\blacktriangledown}$ (-4.12%)	0.339 (-1.14%)	0.494 (2.43%)

tities selected by query annotation, entity search and document annotation respectively. Relative performance (in brackets) and Win/Tie/loss values are compared with `ListMLE`, which uses the same model but query-document features. $\triangledown$ indicates significantly worse performance than `ListMLE` in the permutation test ($p < 0.05$). $\blacktriangledown$ indicates significantly worse performance compared to the corresponding version of `EsdRank`. For example, `ListMLE-AnnQ` is significantly worse than `EsdRank-AnnQ` if marked by $\blacktriangledown$.

Most times these flat features hurt performance, with significant drops compared to `ListMLE` and corresponding `EsdRank` versions. Even on ClueWeb09, where the most training data is available, it is typical for flat features to perform more than 10% worse than `EsdRank`. These results demonstrate the advantage of using a latent space learning to rank model instead of a flat model to handle external evidence: By modeling related entities as hidden variables in latent space, `Latent-ListMLE` naturally separates query-entity and entity-document evidence, and uses the EM algorithm to propagate document level training data to the entity level. As a result, `Latent-ListMLE` avoids feature explosion or confusing machine learning algorithms with highly correlated features, while still only requiring document relevance judgments.

Table 3.12: The performances of `EsdRank`'s variants when they agree or differ with other variants, combined from three data sets. 'Agree' is defined by at least one overlap in related entities. Relative gain, Win/Tie/Loss and statistical significance § are compared with `ListMLE` on corresponding queries. The best relative gains for each variant are marked **bold**.

(a) EsdRank-AnnQ

Versus	Group	MAP@100	NDCG@20	ERR@20	Win/Tie/Loss
`AnnD`	`Agree`	$0.541^{§}$ (**14.80%**)	$0.299^{§}$ (**12.89%**)	$0.254^{§}$ (**24.18%**)	54/14/26
	`Differ`	0.390 (0.90%)	0.252 (2.45%)	$0.286^{§}$ (8.14%)	110/56/96
`Osrch`	`Agree`	$0.495^{§}$ (7.10%)	$0.328^{§}$ (6.69%)	$0.347^{§}$ (7.53%)	74/17/39
	`Differ`	0.393 (3.75%)	$0.227^{§}$ (4.28%)	$0.238^{§}$ (15.31%)	90/53/83

(b) EsdRank-Osrch

Versus	Group	MAP@100	NDCG@20	ERR@20	Win/Tie/Loss
`AnnD`	`Agree`	0.490 (1.81%)	0.287 (1.66%)	0.210 (0.08%)	29/9/21
	`Differ`	0.388 (−1.70%)	0.243 (−0.56%)	0.258 (0.45%)	108/65/124
`AnnQ`	`Agree`	$0.482^{§}$ (**4.41%**)	$0.320^{§}$ (**4.02%**)	$0.340^{§}$ (**5.37%**)	72/16/42
	`Differ`	0.361 (−4.81%)	0.210 (−3.53%)	0.198 (−4.08%)	65/58/103

(c) EsdRank-AnnD

Versus	Group	MAP@100	NDCG@20	ERR@20	Win/Tie/Loss
`Osrch`	`Agree`	0.504 (4.71%)	0.292 (3.49%)	$0.233^{§}$ (11.41%)	30/7/22
	`Differ`	0.394 (−0.28%)	0.244 (0.09%)	0.267 (3.92%)	113/63/121
`AnnQ`	`Agree`	$0.527^{§}$ (**11.69%**)	$0.285^{§}$ (**7.74%**)	$0.248^{§}$ (**21.19%**)	50/12/32
	`Differ`	0.371 (−4.12%)	0.241 (−1.99%)	0.266 (0.47%)	93/58/111

3.2.3.3 Influence of Related Entity Quality

Prior research [20, 32, 62] (Section 3.1) and our evaluation results on `EsdRank` with different related entities all indicate the great influence of selected entities' quality on ranking accuracy. The third experiment further studies its influence on `EsdRank`'s performance. Although directly measuring the quality is hard due to the lack of entity level labels, we can indirectly characterize the quality by comparing the related entities selected by different methods, with the hypothesis that entities selected by multiple entity selectors are more likely to have higher quality.

For each variant of `EsdRank`, queries were divided into two groups (`Agree`, `Differ`) based on whether the query had at least one related entity in common with another variant. The two groups were compared using the same methodology reported earlier. The results are summarized in Table 3.12. Each row is the performance of one variant (first column), divided by overlaps with another variant (second column). Relative gains, Win/Tie/Loss, and statistical significance (§) are calculated based on comparison with `ListMLE`. Results for ClueWeb09, ClueWeb12, and OSHUMED queries are combined because their trends are similar.

The two groups clearly perform differently. On queries with common entities (`Agree`), `EsdRank` is much more effective. For example, when the less effective `AnnD` and `Osrch` entity selectors agree with the more effective `AnnQ` selector, they both outperform `ListMLE` with statistical significance. When they differ, they may be worse than `ListMLE`. Although `AnnQ` is the most effective individual method, its performance is further improved when it agrees with `AnnD`, outperforming `ListMLE` by more than 10% on all metrics. This level of improvement is close to the gains reported by Liu et al. [62] with *manual* query annotations.

The results also show that `EsdRank` behaves predictably. When simple entity selectors agree, `EsdRank` is more likely to improve retrieval quality; when they disagree, the search engine can simply retreat to `ListMLE`.

Finding related entities for query and document is still an open problem in the literature. In the SIGIR ERD'14 workshop [20], many teams built their query annotators upon TagMe and obtained about 60% accuracy. It is still not clear how to adapt full-text retrieval models to perform better entity search. Document annotation can be done with very high precision but there is still room to improve recall, even on the widely used Google FACC1 annotations. Some prior research questions whether current Freebase query annotation is too noisy to be useful [32, 62]. With the help of features between query, entity and document, `Latent-ListMLE` is able to improve retrieval accuracy even when entity selection is noisy. As researchers improve the quality of entity search and annotation, producing less noisy entities, we would expect `Latent-ListMLE`'s ranking accuracy to improve further.

3.2.4 EsdRank Summary

EsdRank is a general method of using external semi-structured data in modern LeToR systems. It uses objects from external data, which are entities in this work but can also be words, as an interlingua between query and documents. Query-entity and entity-document relationships are expressed as features. Latent-ListMLE treats entities as latent layers between query and document, and learns how to use evidence between query, entities and documents in one unified procedure. This general method can be applied to a variety of semi-structured resources, and is easily trained using ordinary query-document relevance judgments.

Experiments with two rather different types of knowledge graphs – a classic controlled vocabulary and a modern general domain knowledge graph – and three well-known datasets show that EsdRank provides statistically significant improvements over several state-of-the-art ranking baselines.

Experiments with the single-layer LeToR model, which uses exactly the same information but expressed by 'flat' features, show much weaker performance than Latent-ListMLE. This result confirms the advantage of separating evidence about query-entity and entity-document relationships with the latent space, instead of mixing them together. The single-layer approach may result in a large and highly correlated feature space, which is hard for machine learning models to deal with.

Finding related entities for query and documents is an important step for using external data in ranking. EsdRank is tested with three popular related entity selection methods: query annotation, entity search, and document annotation. The evaluation results demonstrate the essential effect of related entity quality on ranking accuracy, and suggest that query annotation is the most reliable source for related entities. Section 3.3 further discusses our work in finding related information from knowledge graphs for queries with entity search.

3.3 Learning to Rank Related Entities

Previous studies in this chapter demonstrate the influence of related entities' quality to knowledge based information retrieval systems' performance. However, entity linking and entity search are both on-going research topics themselves. There is a huge room to improve in both techniques in order to acquire better entities to represent query.

This section presents our research about using entity search to find better related entities[9]. Prior state-of-the-arts represent each entity as a structured document by grouping RDF triples of an entity in knowledge base into fields [3, 112] or a tree [64]. Then entities can be retrieved by conventional document retrieval algorithms such as BM25, query likelihood, or sequential dependency models. However, the problem is far from solved. The current P@10 of prior state-of-the-art [112] is at most 25%, introducing much noise to our ranking systems. This work studies whether it is possible to utilize supervision to improve entity search accuracy. We represent entities following the previous state-of-the-art's multi-field representations [112], extract text similarity features for query-entity pairs, and use learning to rank models trained on entity level relevance judgments to rank entities.

Experimental results on an entity search test collection based on DBpedia [3] confirm that learning to rank is as powerful for entity ranking as for document ranking, and significantly improves the previous state-of-the-art. The results also indicate that learning to rank models with text similarity features are especially effective on keyword queries.

3.3.1 Related Work in Entity Search

Ad-hoc entity search was originally a task widely studied in the semantic web community, and recently draws attentions from information retrieval researchers as knowledge bases became popular. A major challenge discovered in previous entity search research is how to represent the entities. In knowledge bases, there are many kinds of information available for each entity, stored by RDF triples with different types of predicates. Recently Neumayer et al. found it more effective to just group the facts into two fields: title (name) and content (all others), and use standard field based document retrieval models such as BM25F and fielded language model [74]. Balog and Neumayer later produced a test collection that combines various previous entity search benchmarks, with different search tasks including single entity retrieval, entity list retrieval, and natural language question answering [3]. They also provide thorough experimental studies of baselines. Their experiment results demonstrate that in ad-hoc entity search, it is effective to combine entities' facts into fields of virtual documents, treat entity search as a document retrieval problem, and apply field based document retrieval models.

More recently, Zhiltsov et al. introduce the widely used fielded sequential dependence model (FSDM) to entity search [112]. They combine entities' facts into five manually defined fields. The five-fielded representation groups entities' RDF triples more structurally than only using a name field and a body field, and is also not too complex for learning to rank models to learn.

[9] Jing Chen, Chenyan Xiong, and Jamie Callan. *An Empirical Study of Learning to Rank for Entity Search.* In Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2016) [22]. This work was done by collaborating with Jing Chen, a master student in MIIS program, through her first semester Directed Study, which I co-advised.

FSDM achieves the state-of-the-arts with FSDM on multiple entity search benchmarks, especially on keyword queries [112]. On natural language question queries, Lu et al. shows that it is more effective to keep the original graph structure of knowledge bases as question queries are more complex than short keyword queries [64]. They parse the question into a tree structure, and finds the answer by mapping the parsed question tree to entity's facts, very similar to knowledge based question-answer technique (OpenQA) [6, 107]. As our focus is more on keyword queries, this work follows ad-hoc entity search methods [3, 112] and introduces the widely used and effective learning to rank techniques from document ranking to entity search.

Another related work is Schuhmacher et al. [85], in which the authors propose a new task of ranking entities specially for web queries, from web search's point of view. They use learning to rank model to rank the entities that are annotated to top retrieved documents for a web query, instead of retrieving entities directly from the knowledge base. In this way, it is not directly comparable with prior work about entity search.

3.3.2 Learning to Rank Entities

The first question in entity search is how to represent entities. We follow Zhiltsov et al. [112] and group RDF triples into five fields: `Name`, which contains the entity's names; `Cat`, which contains its categories; `Attr`, which contains all attributes except name; `RelEn`, which includes the names of its neighbor entities; and `SimEn`, which contains its aliases. We only include RDF triples whose predicates are among the top 1,000 most frequent in DBpedia in the fields [3].

In state-of-the-art learning to rank systems for document ranking, most features are the scores of common unsupervised ranking algorithms applied to different document representations (fields). The different ranking algorithms and representations provide different views of the relevance of the document to the query. The multiple perspectives represented by these features are the backbone of any learning to rank system.

This approach can be applied to entity search by extracting features for query-entity pairs. We use the following ranking algorithms on each of an entity's five fields: `Language model` with Dirichlet smoothing ($\mu = 2500$), `BM25` (default parameters), `coordinate match`, `cosine correlation`, and `sequential dependency model (SDM)`. We also include Zhiltsov et al's. [112] `fielded sequential dependency model (FSDM)` score for the full document as a feature. As a result, there are in total 26 features as listed in Table 3.14.

With features extracted for all query-entity pairs, all learning to rank models developed for ranking documents can be used to rank entities. We use two widely-used LeToR models: `RankSVM`, which is an SVM-based pairwise method, and `Coordinate Accent`, which is a gradient-based listwise method that directly optimizes mean average precision (MAP). Both of these LeToR algorithms are robust and effective on a variety of datasets.

3.3.3 Experimental Methodology

This section describes our experiment methodology in studying the effectiveness of learning to rank in entity search.

Table 3.13: Entity Search Query Sets.

Query Set	**Queries**	**Search Task**
`SemSearch ES`	130	Retrieve one entity
`ListSearch`	115	Retrieve a list of entities
`INEX-LD`	100	Mixed keyword queries
`QALD-2`	140	Natural language questions

Table 3.14: Query-Entity features used in Learning to Rank Entity.

Features	**Dimension**
FSDM	1
SDM on all fields	5
BM25 on all fields	5
Language model on all fields	5
Coordinate match on all fields	5
Cosine on all fields	5

Dataset: Our experiments are conducted on the entity search test collection provided by Balog and Neumayer [3], which others also have used for research on entity retrieval [64, 112]. The dataset has 485 queries with relevance judgments on entities from DBpedia version 3.7. These queries come from seven previous competitions and are merged into four groups based on their search tasks [112]. Table 3.13 lists the four query groups used in our experiments.

Base Retrieval Model: We use the `fielded sequential dependency model` (`FSDM`) as the base retrieval model to enable direct comparison to prior work [112]. All learning to rank methods are used to rerank the top 100 entities per query retrieved by `FSDM`, as provided by Zhiltov et al. [112].

Ranking Models: `RankSVM` implementation is provided by SVMLight toolkit[10]. `Coordinate Ascent` implementation is provided by RankLib[11]. Both methods are trained and tested using five fold cross validation. We use linear kernel in `RankSVM`. For each fold, hyper-parameters are selected by another five fold cross validation on the *training* partitions only. The 'c' of `RankSVM` is selected from range $1 - 100$ using step size of 1. The number of random restarts and iterations of `Coordinate Ascent` are selected from range $1 - 10$ and $10 - 50$ respectively using step size of 1.

Baselines: The main baseline is `FSDM`, the state-of-the-art for the benchmark dataset [112]. We also include `SDM-CA` and `MLM-CA` results as they perform well in this test collection [112].

Evaluation Metrics: All methods are evaluated by MAP@100, P@10, and P@20 following previous work [112]. We also report NDCG@20 because it provides more details. Statistical significance tests are performed by Fisher Randomization (permutation) tests with $p < 0.05$.

[10]`https://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html`
[11]`http://sourceforge.net/p/lemur/wiki/RankLib/`

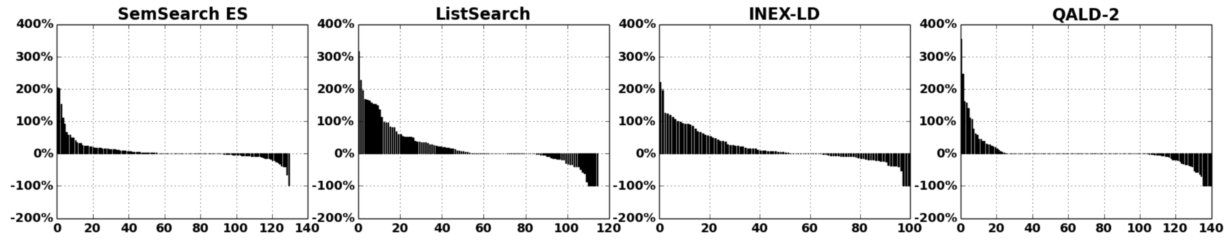


Figure 3.2: Query level relative performance in four query groups. X-axis lists all queries, ordered by relative performance. Y-axis is the relative performance of `RankSVM` comparing with `FSDM` in NDCG@20. Positive value indicates improvement and negative value indicates loss.

3.3.4 Evaluation Results

We first present experimental results for learning to rank on entity search. Then we provide analysis of the importance of features and fields, and the influence of different query tasks on LeToR models.

3.3.4.1 Overall Performance

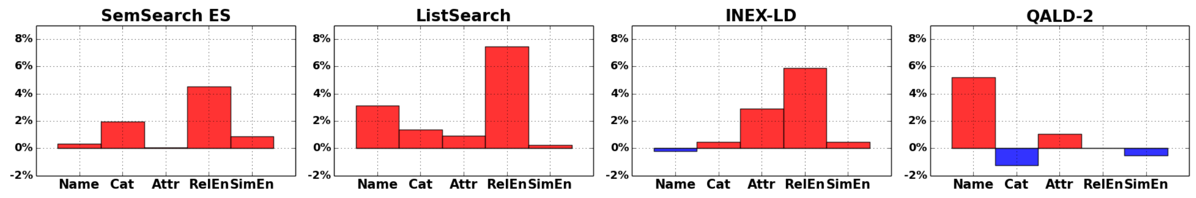
The ranking performances of learning to rank models are listed in Table 3.15. We present results separately for each query group and also combine the query groups together, shown in the `All` section of Table 3.15. Relative performances over `FSDM` are shown in parenthesis. †, ‡, § indicate statistical significance over `SDM-CA`, `MLM-CA`, and `FSDM` respectively. The best performing method for each metric is marked **bold**. Win/Tie/Loss are the number of queries improved, unchanged and hurt, also compared with `FSDM`.

The results demonstrate the power of learning to rank for entity search. On all query sets and all evaluation metrics, both learning methods outperform `FSDM`, defining a new state-of-the-art in entity search. The overall improvements on all queries can be as large as 8%. On `SemSearch ES`, `ListSearch` and `INEX-LD`, where the queries are keyword queries like 'Charles Darwin', LeToR methods show significant improvements over `FSDM`. However, on `QALD-2`, whose queries are questions such as 'Who created Wikipedia', simple text similarity features are not as strong.

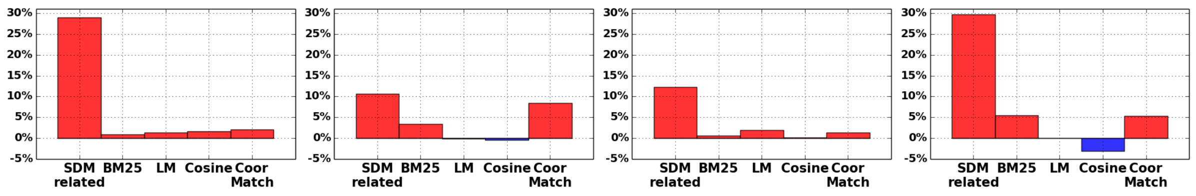
Similar trends are also found in individual query performances. Figure 3.2 compares the best learning method, `RankSVM`, with `FSDM` at each query. The x-axis lists all queries, ordered by relative performance. The y-axis is the relative performance of `RankSVM` over `FSDM` on NDCG@20. On keyword queries more than half queries are improved while only about a quarter of queries are hurt. On questions (`QALD-2`), about the same number of queries are improved and hurt. A more effective method of handling natural question queries is developed recently by Lu et al. in which queries are parsed using question-answering techniques [64]. That method achieves 0.25 in P@10, but performs worse than `FSDM` on keyword queries. Section 3.3.4.3 further studies the influence of query types on entity-ranking accuracy.

Table 3.15: Performance of learning to rank methods on entity search. Relative improvements over `FSDM` are shown in parentheses. Win/Tie/Loss show the number of queries improved, unchanged and hurt, comparing with `FSDM`. `All` section combines the evaluation results of the other four sections. $\dagger, \ddagger, \S$ indicate statistical significance over `SDM-CA`, `MLM-CA`, and `FSDM` respectively. The best method for each metric is marked **bold**.

	SemSearch ES				
	MAP@100	**P@10**	**P@20**	**NDCG@20**	**Win/Tie/Loss**
`SDM-CA`	0.254	0.202	0.148	0.355 (-29.5%)	26/15/89
`MLM-CA`	0.320†	0.250†	0.178†	0.443† (-12.0%)	30/32/68
`FSDM`	0.386†‡	0.286†‡	0.203†‡	0.503†‡ -	-
`RankSVM`	**0.410**†‡§	**0.304**†‡§	**0.213**†‡§	**0.527**†‡§ (+4.7%)	65/27/38
`Coor-Ascent`	0.396†‡	0.295†‡	0.206†‡	0.511†‡ (+1.5%)	48/32/50
	ListSearch				
	MAP@100	**P@10**	**P@20**	**NDCG@20**	**Win/Tie/Loss**
`SDM-CA`	0.197	0.252	0.202	0.296 (+1.7%)	55/23/37
`MLM-CA`	0.190	0.252	0.192	0.275 (-5.3%)	39/28/48
`FSDM`	0.203	0.256	0.203	0.291 -	-
`RankSVM`	0.224†‡§	**0.303**†‡§	**0.235**†‡§	**0.332**†‡§ (+14.3%)	61/23/31
`Coor-Ascent`	**0.225**†‡§	0.300†‡§	0.229†‡§	0.328†‡§ (+12.9%)	62/21/32
	INEX-LD				
	MAP@100	**P@10**	**P@20**	**NDCG@20**	**Win/Tie/Loss**
`SDM-CA`	0.117‡	0.258	0.199	0.284 (-0.9%)	43/7/50
`MLM-CA`	0.102	0.238	0.190	0.261 (-8.8%)	34/13/53
`FSDM`	0.111‡	0.263‡	0.214†‡	0.287‡ -	-
`RankSVM`	**0.126**‡§	**0.282**‡	**0.231**†‡§	**0.317**†‡§ (+10.6%)	55/9/36
`Coor-Ascent`	0.121‡§	0.275‡	0.224†‡	0.306†‡§ (+6.7%)	53/7/40
	QALD-2				
	MAP@100	**P@10**	**P@20**	**NDCG@20**	**Win/Tie/Loss**
`SDM-CA`	0.184	0.106	0.090	0.244‡ (-6.8%)	36/66/38
`MLM-CA`	0.152	0.103	0.084	0.206 (-21.3%)	17/78/45
`FSDM`	0.195‡	0.136†‡	0.111‡	0.262‡ -	-
`RankSVM`	0.197‡	0.136†‡	0.113†‡	0.266‡ (+1.6%)	31/74/35
`Coor-Ascent`	**0.208**‡	**0.141**†‡	**0.115**†‡	**0.278**‡ (+5.9%)	40/71/29
	All				
	MAP@100	**P@10**	**P@20**	**NDCG@20**	**Win/Tie/Loss**
`SDM-CA`	0.192	0.198	0.155	0.294 (-13.1%)	160/111/214
`MLM-CA`	0.196	0.206	0.157	0.297 (-12.1%)	120/151/214
`FSDM`	0.231†‡	0.231†‡	0.179†‡	0.339†‡ -	-
`RankSVM`	**0.246**†‡§	**0.251**†‡§	**0.193**†‡§	**0.362**†‡§ (+7.0%)	212/133/140
`Coor-Ascent`	0.245†‡§	0.248†‡§	0.189†‡§	0.358†‡§ (+5.7%)	203/131/151



(a) Influence of fields



(b) Influence of feature groups

Figure 3.3: The contribution of fields and feature groups to RankSVM's performance. X-axis lists the fields or feature groups. Y-axis is the relative NDCG@20 difference between RankSVM used with all fields or feature groups and without corresponding field or feature group. Larger values indicate more contribution.

3.3.4.2 Field and Feature Study

The second experiment studies the contribution of fields and feature groups to learning to rank models. For each field or feature group, we compare the accuracy of models when used without field or features from that group to those with all features. The change in accuracy indicates the contribution of the corresponding field or feature group. The field and feature studies for RankSVM are shown in Figures 3.3a and 3.3b respectively. The x-axis is the field or feature group studied. The y-axis is the performance difference between the two conditions (All versus held out). Larger values indicate greater contributions. Figure 3.3a organizes features by the fields they are extracted from, including Name, Cat, Attr, RelEn, and SimEn. Figure 3.3b organizes features into five groups, with one retrieval model per group. SDM related contains FSDM and SDM scores as they are very correlated.

Figure 3.3a shows that RankSVM favors different fields for different query sets. The Name field is useful for ListSearch and QALD-2, but does not contribute much to the other two query sets. RelEn provides the most gain to keyword queries, but is not useful at all for the natural language question queries in QALD-2. For feature groups we find that SDM related features are extremely important and provide the most gains across all query sets. This result is expected because all of the queries are relatively long queries and often contain phrases, which is where SDM is the most useful.

3.3.4.3 Query Type Differences

Previous experiments found that when different models are trained for different types of queries, each model favors different types of evidence. However, in live query traffic different types of queries are mixed together. The third experiment investigates the accuracy of a learning to rank entities system in a more realistic setting. The four query sets are combined into one query set,

Table 3.16: Performance of learning to rank models when queries from different groups are all trained and tested together. Relative performances and Win/Tie/Loss are compared with the same model but trained and tested separately on each query group.

	All Queries Mixed Together				
	MAP@100	**P@10**	**P@20**	**NDCG@20**	**Win/Tie/Loss**
RankSVM	0.234	0.238	0.185	0.347 (-4.3%)	153/136/196
Coor-Ascent	0.233	0.232	0.179	0.344 (-3.8%)	148/129/208

and new models are trained and tested using five fold cross validation as before.

Table 3.15 All shows the average accuracy when different models are trained for each of the four types of query. Table 3.16 shows the accuracy when a single model is trained for all types of queries. Despite being trained with more data, both learning to rank algorithms produce less effective models for the diverse query set than for the four smaller, focused query sets. Nonetheless, a single learned model is as accurate as the average accuracy of four carefully-tuned, query-set-specific FSDM models.

This results suggests that diverse query streams may benefit from query classification and type-specific entity ranking models. They may also benefit from new types of features or more sophisticated ranking models.

3.3.5 Summary of Related Entities Finding

We use learning to rank models to improve entity search. Entities are represented by multi-field documents constructed from RDF triples. How well a query matches an entity document is estimated by text similarity features and a learned model. Experimental results on a standard entity search test collection demonstrate the power of learning to rank in entity retrieval. Statistically significant improvements over the previous state-of-the-art are observed on all evaluation metrics. Further analysis reveals that query types play an important role in the effectiveness of learned models. Text similarity features are very helpful for keyword queries, but less effective with longer natural language question queries. Learned models for different query types favor different entity fields because each query type targets different RDF predicates.

3.4 Summary

This chapter focuses on enriching the word-based representations of queries using knowledge graphs. We presented several query expansion methods that select expansion terms from the descriptions of retrieved entities and document annotations, using pseudo relevance feedback and ontology similarities with the query. Then we developed the EsdRank approach to represent query directly with entities, and a latent learning to rank model that jointly reasons about the connection from query to query entities and document ranking. The experiments demonstrate the effectiveness of our methods but also found the quality of related entities is a bottleneck of the system performances. The last section provides a possible approach to improve related entity finding using learning to rank methods.

The research presented in this chapter is from the first stage of this thesis research, when we were exploring and proving the effectiveness of knowledge graphs in information retrieval tasks. Thus the methods proposed mainly aim to *improve* the existing standard word-based retrieval models. The next chapter presents a new entity-oriented search paradigm that directly constructs entity-based text representations which leads to more intrinsic approaches to integrate and utilize knowledge graph semantics in search.

Chapter 4

Entity-Based Text Representation

This chapter presents our *entity-based* text representations and ranking approaches with them. Entities are more meaningful minimum language units than individual words: when reading a document, we humans process the entity "Carnegie Mellon University" as one concept instead of breaking it down to three individual words. The idea of representing texts by entities has been widely used by controlled vocabulary based retrieval. However, it was challenging because of the needs of expensive manual annotations and lack of coverage.

This chapter revisit this idea with large scale modern knowledge graphs and automatic entity linking systems. It first constructs a *bag-of-entities* text representation using automatic entity annotations—with careful studies of their precision, coverage, and applicability with standard unsupervised retrieval models. Then it proceeds with leveraging the rich structural semantics in knowledge graphs to perform soft matches in the entity-based representation space. In addition, we also provide case studies on the needs, coverage, and effectiveness of our entity-based ranking systems in the environment of an online academic search engine: Semantic Scholar.

In the rest of this chapter, Section 4.1 presents the *bag-of-entities* representation and the unsupervised exact match retrieval models built upon it. Section 4.2 presents the *Explicit Semantic Ranking* model that soft matches query and documents with using bag-of-entities as well as the structured semantics associated with them.

4.1 Bag-of-Entities Representation

Recall that in the earliest information retrieval systems, query and documents were both represented by terms manually picked from predefined controlled vocabularies [83]. The controlled vocabulary representation conveys clean and distilled information and can be ranked accurately by simple methods. However, its usage is mainly limited to specific domains because manual annotations are required and the scale of classic controlled vocabulary dataset is usually small. Now that knowledge bases have grown to rather a large scale, and automatic entity annotation is made possible by entity linking research [20], it is time to reconsider whether a pure entity based representation can provide decent ranking performance.

This section presents a new bag-of-entities based representation for document ranking, as a heritage of the classic controlled vocabulary based representation, but with the aid of modern large scale knowledge graphs and automatic entity linking systems[1]. We represent queries and documents by their bag-of-entities constructed from the annotations provided by three entity linking systems: Google's `FACC1` [39] with high precision, `CMNS` [43] with high recall, and `TagMe` [38] with balanced precision and recall. With the deeper text understanding provided by entity linking, documents can be ranked by their overlap with the query in the entity space.

To investigate the effectiveness of bag-of-entities representations, we conducted experiments with a state-of-the-art knowledge graph, Freebase, and two large scale web corpora, ClueWeb09-B and ClueWeb12-B13, together with their queries from the TREC Web Track. Our evaluation results first confirm that current entity linking systems can provide sufficient coverage over general domain queries and documents. Then we compare bag-of-entities with bag-of-words in standard document ranking tasks and demonstrate that although the accuracy of entity linking is not perfect (about $50\% - 60\%$ on TREC Web Track queries), the ranking performance can be improved by as much as 18% with bag-of-entities representations.

4.1.1 Bag-of-Entities

We construct bag-of-entities representations for query and documents using several entity linking systems. When annotating texts, an entity linking system does not just match the n-grams with entity names, but makes the decision jointly by also considering external evidence such as the entity descriptions and relationships in the knowledge graph, and corpus statistics like commonness, linked probability and contexts of entities. As a result, representing texts by entities naturally incorporates the semantics from the linking process: Entity synonyms are aligned, polysemy in entity mentions is disambiguated, and global coherence between entities is incorporated.

We continue using Freebase as our choice of the knowledge graph, with which several entity linking systems have been developed. This work explores the following three popular ones to annotate query and documents with Freebase entities.

FACC1 is the Freebase annotation of TREC queries and ClueWeb corpora provided by Google [39]. It aims to achieve high precision, which is believed to be around 80-85%, based on a small-scale human evaluation[2].

TagMe is a system [38] widely used in entity linking competitions [20] and in our previous work (Section 3.2). It balances precision and recall, both at about 60% in various evaluations.

CMNS is an entity linking system that spots texts using surface forms from the FACC1 annotation, and links all of them to their most frequently linked entities [43]. It achieves almost 100% recall on some query entity linking datasets, but the precision may be lower than TagMe [43].

Given the annotations of a query or document, we construct its bag-of-entities vector $\vec{E}_q$ or $\vec{E}_d$, in which each dimension ($\vec{E}_q(e)$ or $\vec{E}_d(e)$) refers to an entity e in Freebase, and its weight is the frequency of that entity being annotated to the query or document.

Compared to the controlled vocabularies based retrieval, the bag-of-entities representation

[1] Chenyan Xiong, Jamie Callan, and Tie-Yan Liu. *Bag-of-Entities Representation for Ranking*. In Proceedings of the Second ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR 2016) [99].

[2]`http://lemurproject.org/clueweb09/FACC1/`

also uses entities as its basic information unit. As with controlled vocabulary terms, entities are more informative than words. However, whereas in controlled vocabularies based retrieval, the annotations were assigned manually, bag-of-entities used automatically linked entities, which is more efficient but also more uncertain. With controlled vocabularies, simple ranking methods such as Boolean retrieval work well, because the representation is clean and distilled [83], while with bag-of-words perhaps more sophisticated ranking models are required. Given the heritage to the classic controlled vocabulary based search systems, we start simple and use the following two basic ranking models to study the power of bag-of-entities representations.

Coordinate Match (`COOR`) ranks a document by the number of query entities it contains:

$$f_{\text{COOR}}(q,d) = \sum_{e:\vec{E}_q(e)>0} \mathbb{1}(\vec{E}_d(e) > 0) \tag{4.1}$$

Entity Frequency (`EF`) ranks document by the frequency of query entities in it:

$$f_{\text{EF}}(q,d) = \sum_{e:\vec{E}_q(e)>0} \vec{E}_q(e) \log(\vec{E}_d(e)) \tag{4.2}$$

$f_{\text{COOR}}(q,d)$ and $f_{\text{EF}}(q,d)$ are the ranking scores of document d for query q using coordinate match (`COOR`) and entity frequency (`EF`) respectively. $\mathbb{1}(\cdot)$ is the indicator function.

`COOR` performs Boolean retrieval, which is the most basic ranking method and often works well with controlled vocabularies. `EF` studies the value of term frequency information, another basis for document ranking. These simple models investigate basic properties of ranking with bag-of-entities, and provide understanding and intuition for the future development of more advanced ranking models.

4.1.2 Experiment Methodology

This section provides details about our experiments to study the quality of annotations and the effectiveness of bag-of-entities in ranking.

Dataset: Our experiments are conducted on TREC Web Track datasets. TREC Web Tracks use two large web corpora: ClueWeb09 and ClueWeb12. We use the ClueWeb09-B and ClueWeb12-B13 subsets. There are 200 queries with relevance judgments from TREC 2009-2012 for ClueWeb09, and 100 queries in 2013-2014 for ClueWeb12. Manual annotations provided by Dalton et al. [31] and Liu et al. [61] are used as query annotation labels.

Indexing and Base Retrieval Model: We indexed both corpora with Indri. Default stemming and stopword removal were used. Spam in ClueWeb09 was filtered using the default threshold (70%) for Waterloo spam scores. Spam filtering was not used for ClueWeb12 because its effectiveness is unclear. Indri language model with default Dirichlet smoothing ($\mu = 2500$) was used to retrieve 100 documents per query for the evaluation of entity linking and ranking. All our methods re-rank the top 100 documents retrieved by the Indri language model.

Entity Linking Systems: We used `TagMe` software provided by Ferragina et al. [38] to annotate queries and documents. It annotates text with Wikipedia entities. The same as in Section 3.2, we aligned Wikipedia entities to Freebase entities using the Wikipedia ID field in Freebase.

CMNS is implemented by ourselves, following Hasibi et al. [43]. The boundary overlaps of surface forms are resolved by only linking the earliest and then the longest one [20, 43].

FACC1 entity annotations for ClueWeb documents are provided by Google [39]. They also annotated ClueWeb09 queries' intent descriptions, but not the queries. We used the descriptions' annotations as approximations of queries' annotations, and manually filtered out entities that did not appear in the original queries to reduce disturbance. ClueWeb12's queries are not annotated by Google so we are only able to study FACC1 annotations on ClueWeb09.

Baselines: We used two standard unsupervised bag-of-words ranking models as baselines: Indri's unstructured language model (Lm) and sequential dependency model (SDM), both with default parameters: $\mu = 2500$ for Lm, and query weights $(0.8, 0.1, 0.1)$ for SDM. Typically these baselines do well in competitive evaluations such as TREC. There are better rankers, for example learning to rank methods. However, methods that combine many sources of evidence usually outperform methods that use a single source of evidence; such comparison would not reveal much about the value of the bag-of-entities representation.

Compared to Section 3.1, the experimental settings in ClueWeb09-B are kept the same, except that the relevance judgments are updated to the 'category B' setting in which only documents in ClueWeb09-B are included in the TREC qrel file. This leads to slightly different 'absolute' scores for baselines. This section also uses Indri's implementation as the SDM baseline, instead of the highly-tuned Galago implementation from Dalton et al. [31]. More comparisons with the Galago SDM implementation are presented in later sections.

There were three representations for ClueWeb09 (FACC1, TagMe and CMNS), and two for ClueWeb12 (TagMe and CMNS). COOR and EF were used to *re-rank* the top 100 documents per query retrieved by Lm. Ties were broken by Lm's score.

Evaluation Metrics: The entity annotations were evaluated by the lean evaluation metric from Hasibi et al. [43]. The ranking performance was evaluated by the TREC Web Track Adhoc Task's official evaluation metrics: ERR@20 and NDCG@20. Statistical significance was tested by the Fisher randomization test (permutation test) with $p < 0.05$.

4.1.3 Evaluation Results

This section evaluates the accuracy and coverage of entity annotations and bag-of-entities' performance in ranking.

4.1.3.1 Annotation Accuracy and Coverage

Table 4.2 shows the precision, recall, and F1 of FACC1, TagMe, and CMNS on ClueWeb queries. TagMe performs the best on ClueWeb09 queries with higher precision, while CMNS performs better on ClueWeb12 queries. The ClueWeb09 queries are more ambiguous because they needed to support the TREC Web Track's Diversity task; TagMe's disambiguation was more useful on this set. ClueWeb12 queries needed to support risk minimization research, and have been shown to be harder; both systems perform worse on them. FACC1 query annotation does not perform well as its goal was to annotate the query's description, not the query itself.

There is no gold standard entity annotation for ClueWeb documents, preventing a quantitative evaluation. Nevertheless, our manual examination confirms that FACC1 has high precision;

Table 4.1: Coverage of annotations. Freq and Dens are the average number of entities linked per query/document and per word respectively. Missed is the percentage of queries or documents that have no annotation at all. ClueWeb12 queries do not have FACC1 annotations as they are published later than FACC1.

(a) ClueWeb09

	Query			**Document**		
	Freq	Dens	Missed	Freq	Dens	Missed
FACC1	0.42	0.20	62%	15.95	0.13	30%
TagMe	1.54	0.70	1%	92.31	0.20	2%
CMNS	1.50	0.69	1%	252.41	0.55	0%

(b) ClueWeb12

	Query			**Document**		
	Freq	Dens	Missed	Freq	Dens	Missed
FACC1	NA	NA	NA	24.52	0.06	26%
TagMe	1.77	0.57	0%	246.76	0.37	0%
CMNS	1.75	0.55	0%	324.37	0.48	0%

Table 4.2: Entity linking performance on ClueWeb queries. All methods are evaluated by **Prec**ison, **Rec**all and **F1**.

	ClueWeb09 Query			**ClueWeb12 Query**		
	Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**
FACC1	0.274	0.236	0.254	NA	NA	NA
TagMe	0.581	0.597	0.589	0.460	0.555	0.503
CMNS	0.577	0.596	0.587	0.485	0.575	0.526

TagMe performs a little better on documents with more contexts; and CMNS performs worse than TagMe on documents as it only uses the surface forms.

One concern of controlled vocabulary based search systems is the low coverage on general domain queries, restricting their usage mainly to specific domains. With the much larger scale of current knowledge bases, it is interesting to study whether their coverage is sufficient to influence the majority of general domain queries. Table 4.1 shows the coverage results of our entity annotations on ClueWeb queries and their top 100 retrieved documents. Freq and Dens are the average number of entities linked per query/document, and per word respectively. Missed is the percentage of queries/documents that have no linked entity. The results show that TagMe and CMNS have good coverage on ClueWeb queries and documents. Almost all queries and documents have at least one linked entity. *The annotations are no longer sparse.* There can be up to 324 entities linked per documents on average. However, precision and coverage have not been achieved together yet. FACC1 has the highest precision but provides very few entities per document and misses many documents.

Table 4.3: Ranking performance of bag-of-entity based ranking models. FACC1, TagMe and CMNS refer to the bag-of-entity representation constructed from corresponding annotations. COOR and EF refer to the coordinate match and entity frequency ranking models. Percentages show the relative changes compared to SDM. **Win/Tie/Loss** are the number of queries improved (Win), unchanged (Tie) and hurt (Loss) comparing with SDM. † and ‡ indicate statistic significance ($p < 0.05$ in permutation test) over Lm and SDM. The best method for each metric is marked **bold**. The Indri Lm and SDM scores are higher than those in Table 3.2 because they are evaluated by the category-B relevance judgments instead of the category-A ones.

(a) ClueWeb09

Method	**NDCG@20**		**ERR@20**		**Win/Tie/Loss**
Lm	0.176	-12.92%	0.119	-5.23%	39/88/71
SDM	0.202†	–	0.126†	–	–
FACC1-COOR	0.173	-14.16%	0.126	-0.21%	64/56/78
FACC1-EF	0.167	-17.32%	0.116	-8.14%	63/51/84
TagMe-COOR	0.211†	4.55%	0.133†	5.55%	108/35/55
TagMe-EF	**0.229**†,‡	**13.71%**	**0.149**†	**18.04%**	96/24/78
CMNS-COOR	0.210†	4.08%	0.131†	4.21%	105/37/56
CMNS-EF	0.216†	6.97%	0.136	7.52%	97/22/79

(b) ClueWeb12

Method	**NDCG@20**		**ERR@20**		**Win/Tie/Loss**
Lm	0.106	-2.10%	0.086	-4.69%	28/29/43
SDM	0.108	–	0.090	–	–
FACC1-COOR	NA	NA	NA	NA	NA
FACC1-EF	NA	NA	NA	NA	NA
TagMe-COOR	0.117†,‡	8.35%	0.095†	5.02%	42/20/38
TagMe-EF	0.107	-0.90%	0.091	1.08%	42/18/40
CMNS-COOR	**0.120**†,‡	**11.03%**	0.101†	11.20%	43/22/35
CMNS-EF	0.110	2.03%	**0.102**	**12.71%**	36/20/44

These results show that the entity linking itself is still an open research problem. Precision and coverage can not yet be achieved at the same time. Thus, the ranking method must be robust and able to accommodate a noisy representation.

4.1.3.2 Ranking Performance

The ranking performances of bag-of-entities based ranking models are shown in Table 4.3. EF and COOR are used to rerank top retrieved documents using the bag-of-entities representations built from FACC1[3], TagMe, and CMNS.

On ClueWeb09, both TagMe and CMNS work well with EF and COOR. They outperform all

[3]FACC1 annotations not available for ClueWeb12 queries.

Table 4.4: Performances of bag-of-entity based ranking models on queries that are correctly annotated and not correctly annotated, compared with manual annotations. The relative performance and **Win/Tie/Loss**are calculated by comparing with SDM on the same query group. † and ‡ indicate statistic significance ($p < 0.05$ in permutation test) over Lm and SDM. The best method for each metric is marked **bold**.

(a) Correctly Annotated Query

Method	**NDCG@20**		**ERR@20**		**Win/Tie/Loss**
TagMe-COOR	$0.214^{\dagger,\ddagger}$	14.56%	$0.153^{\dagger,\ddagger}$	12.71%	59/16/17
TagMe-EF	$\mathbf{0.243}^{\dagger,\ddagger}$	**30.43%**	$\mathbf{0.178}^{\dagger,\ddagger}$	**31.19%**	53/10/29
CMNS-COOR	$0.211^{\dagger}$	4.28%	$0.146^{\dagger}$	4.89%	53/22/20
CMNS-EF	$0.240^{\dagger,\ddagger}$	18.44%	0.168	20.74%	52/11/32

(b) Mistakenly Annotated Query

Method	**NDCG@20**		**ERR@20**		**Win/Tie/Loss**
TagMe-COOR	$0.200^{\dagger}$	−3.28%	0.111	−1.93%	49/21/38
TagMe-EF	0.209	0.65%	0.118	4.30%	43/16/49
CMNS-COOR	$\mathbf{0.201}^{\dagger}$	**3.90%**	**0.112**	**3.40%**	52/17/36
CMNS-EF	0.185	−4.09%	0.100	−8.12%	45/13/47

baselines on all evaluation metrics. The best method, TagMe-EF, outperforms SDM as much as 18% on ERR@20. On ClueWeb12, COOR outperforms all baselines on all evaluation metrics by about 12%. These results demonstrate that even with current imperfect entity linking systems, bag-of-entities is a valuable representation on which very basic ranking models can significantly outperform standard bag-of-words based ranking.

Bag-of-entities' representation power correlates with the entity linking system's accuracy. ClueWeb09 queries are more ambiguous, favoring TagMe in annotation accuracy, and TagMe provides the most improvements when ranking for ClueWeb09 queries. ClueWeb12 queries have lower annotation quality, and bag-of-entities based ranking is not as powerful as on ClueWeb09 queries. FACC1's coverage is too low and can not well represent the documents. This also explains why prior research mainly uses it as a pool to select query entities [31] (Section 3.1, 3.2).

Entity linking is a rapidly developing area; improvement in the future is likely. To study how the bag-of-entities can benefit from improvements in annotation accuracy, we used the manual query annotations [31, 61] to divide queries into two groups: *Correctly Annotated*, and *Mistakenly Annotated*. Better ranking performance is expected for Correctly Annotated queries. Table 4.4 shows the ranking performances of TagMe and CMNS on the two groups in ClueWeb09. We omit FACC1 as it always hurts, and ClueWeb12 queries as there are not enough queries in either group to provide reliable observations. The relative performance, W/T/L and statistical significance over SDM are calculated on the same queries. The results are as expected: On Correctly Annotated queries, bag-of-entities provides more accurate ranking; On Mistakenly Annotated queries, the improvements are smaller, and sometimes bag-of-entities reduces accuracy.

Our experiments show that intuitions developed for bag-of-words representations do not necessarily apply directly to bag-of-entities representations. A long line of research shows that frequency based (e.g., tf.idf) ranking models are superior to Boolean ranking models. Thus, one might expect `EF` to provide consistently more accurate ranking than `COOR`, however, that is not the case in our experiments. We found that the majority of annotation errors are missed annotations, which makes entity frequency counts less reliable. However, it is rare for the entity linker to miss every mention of an important entity in a document, thus, the Boolean representation is robust to this majority type of errors.

We also examined the effectiveness of other ranking intuitions, such as inverse document frequency (idf) and document length normalization. In our bag-of-entities representations, they did not provide improvements when used individually or in language modeling and BM25 rankers. We speculate that idf had less impact because most queries contained just one or two entities, thus most of the queries were 'short' in the bag-of-entities representation; idf is known to be less important for short queries. We also speculate that the lack of improvement from document length normalization is related to the lack of improvement from frequency-based weighting (`EF`), as discussed above. Our work suggests that better ranking will require thinking carefully about models designed for the unique characteristics of entities, rather than simply assuming that entities behave like words.

4.1.4 Bag-of-Entities Summary

This section presents a new bag-of-entities representation for search. Query and documents are represented by bag-of-entities representations developed from entity annotations; ranking is performed by exact-matching them in the entity space. Experiments on TREC Web Track datasets demonstrate that bag-of-entities representations have sufficient coverage and ranking with them outperforms bag-of-words based retrieval models by as much as 18% in standard retrieval tasks.

4.2 Explicit Semantic Ranking with Entity Embedding

This section presents Explicit Semantic Ranking (`ESR`), a new ranking technique that soft matches query and documents in the entity space[4]. The structured semantics associated with entities form the majority information in knowledge graphs; only exact matching entities does not fully leverage those semantics. However, these structures are challenging to use due to their high variety and sparse nature. For example, it is often that two highly related entities have no edges between them nor share common neighbors, even in large scale general domain knowledge graphs such as Freebase. `ESR` addresses this challenge with knowledge graph embeddings that convert the sparse structure into dense, continuous representations. Then given the bag-of-entities representations of query and documents, the semantic relatedness between their en-

[4]Chenyan Xiong, Russell Power, and Jamie Callan. *Explicit Semantic Ranking for Academic Search via Knowledge Graph Embedding*. In Proceedings of the 26th International World Wide Web Conference (WWW 2017) [103]. Part of this work was done during my 2016 summer internship at the Allen Institute for Artificial Intelligence. Russell was my internship mentor.

tities can be easily calculated in the embedding space, providing soft match between query and document entities. ESR uses a two-stage pooling to generalize these entity-based matches to query-document ranking features and uses a learning to rank model to combine them.

Furthermore, this section applies our techniques in a new search domain: academic search. We provide analysis on the search intent and the error cases in the online search log of Semantic Scholar (S2), the academic search engine from Allen Institute for Artificial Intelligence. The analysis shows the importance of understanding entities in a real online production system. Then, to apply ESR in domains where large scale domain-specific knowledge graphs are not available, we propose a simple knowledge graph construction approach that automatically build the academic knowledge graph using Freebase and the S2 corpus. This section then applies ESR on this academic knowledge graph to improve the search accuracy of Semantic Scholar.

The explicit semantics from the knowledge graph has the ability to improve ranking in multiple ways. The entities and surface names help the ranking model recognize which part of a query is an informative unit, and whether different phrases have the same meaning, providing a powerful *exact match* signal. Embeddings of entities from the knowledge graph can also be leveraged to provide a *soft match* signal, allowing ranking of documents that are semantically related but do not match the exact query terms. Our experiments on S2 ranking benchmark demonstrate the effectiveness of this explicit semantics. ESR improves the already-reliable S2 online production system by more than 10%. The gains are robust, with bigger gains and smaller errors, and also favoring top ranking positions. Further analysis confirms that both exact match and soft match in the entity space provide effective and novel ranking signals. These signals are successfully utilized by ESR's ranking framework, and greatly improve the queries that S2's word-based ranking system finds hard to deal with.

The rest of this section first discusses related work in academic search and soft-match retrieval. Then it analyzes the search traffic of Semantic Scholar. After that, it presents the Explicit Semantic Ranking system, experimental methodologies, and evaluation results. The last part of this section summarizes the contributions of Explicit Semantic Ranking.

4.2.1 Related Work in Academic Search and Soft-Match Retrieval

Besides the related work discussed in Chapter 2, the research presented in this Section is also related to other work in academic search and also soft-match retrieval.

Prior research in academic search is more focused on the analysis of the academic graph than on ad-hoc ranking. Microsoft uses its Microsoft Academic Graph to build academic dialog and recommendation systems [86]. Other research on academic graphs includes the extraction and disambiguation of authors, integration of different publication resources [91], and expert finding [4, 28, 109]. The academic graph can also be used to model the importance of papers [94] and to extract new entities [2].

Soft match is a widely studied topic in information retrieval, mostly in word-based search systems. Translation models treat ranking as translations between query terms and document terms using a translation matrix [11]. Topic modeling techniques have been used to first map query and document into a latent space, and then matching them in it [95]. Word embedding and deep learning techniques have been studied recently. One possibility is to first build query and document representations using their words' embeddings heuristically, and then match them in the embedding space [93]. The DSSM model directly trains a representation model using deep neural networks, which learns distributed representations for the query and document, and matches them using the learned representations [46]. A more recent method, DRMM, models the query-document relevance with a neural network built upon the word-level translation matrix [42]. The translation matrix is calculated with pre-trained word embeddings. The word-level translation scores are summarized by bin-pooling (histograms) and then used by the ranking neural network.

4.2.2 Query Log Analysis

The `Semantic Scholar` (`S2`) launched in late 2015, with the goal of helping researchers find papers without digging through irrelevant information. The project has been a success, with over 3 million visits in its first year after launch. Its current production ranking system is based on the word-based model in ElasticSearch that matches query terms with various parts of a paper, combined with document features such as citation count and publication time in a learning to rank architecture [59]. In user studies conducted at Allen Institute, this ranking model provides at least comparable accuracy with other academic search engines.

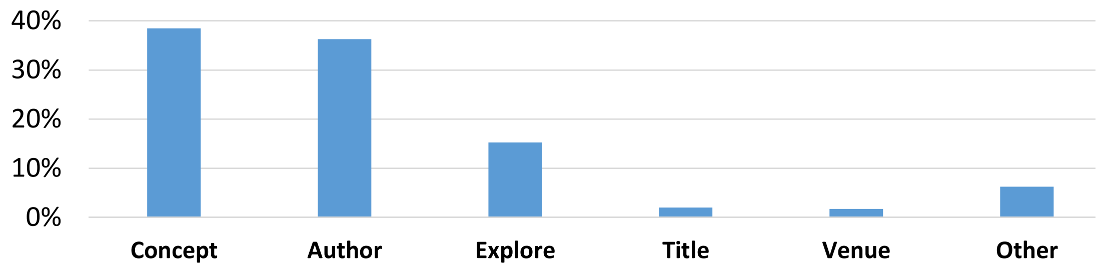
`S2`'s current ranking system is built on top of ElasticSearch's vector space model. It computes a ranking score based on the tf.idf of the query terms and bi-grams on papers' title, abstract, body text and citation context. Static ranking features are also included, for example, the number of citations, recent citations, and the time of publication. To handle the request for an author or for a particular paper by title, a few additional features match explicitly against authors and boost exact title matches. The ranking system was trained using learning to rank on a training set created at Allen Institute. Before and after release, several rounds of user studies were conducted by researchers at Allen Institute (mostly Ph.D.'s in Computer Science) and by a third party company. The results agree that on computer science queries `S2` performs at least on par with other academic search engines on the market.

The increasing online traffic in `S2` makes it possible to study the information needs in academic search, which is important for guiding the development of ranking models. For example, if users are mostly searching for paper titles, the ranking would be straightforward exact-matches; if users are mostly searching for author names, the ranking would be mainly about name disambiguation, aggregation, and recognition.

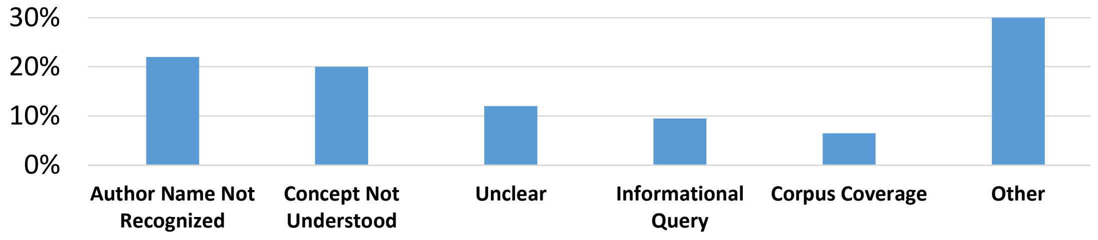
We manually labeled the intents of S2's 400 most frequent queries in the first six months of 2016. A query was labeled based on its search results and clicks. The result shows that the information needs on the head traffic can be categorized into the following categories:

- Concept: Searching for a research concept, e.g. 'deep reinforcement learning';
- Author: Searching for a specific author;
- Exploration: Exploring a research topic, e.g. 'forensics and machine learning';
- Title: Searching for a paper using its title;
- Venue: Searching for a specific conference; and
- Other: Unclear or not research related.

Figure 4.1a shows the distribution of these intents. More than half of `S2`'s head traffic is



(a) Query Intent Distribution



(b) Error Source Distribution

Figure 4.1: Query log analysis in Semantic Scholar's search traffic in the first six months of 2016. Query intents are manually labeled on the 400 most frequent queries. Error sources are manually labeled on the 200 worst performing queries.

research concept related: searching for an academic concept (39%) and exploring research topics (15%). About one-third is searching an author (36%). Very little of the head queries are paper titles; most of such queries are in the long tail of the online traffic.

The large fraction of concept related queries shows that much of academic search is an ad-hoc search problem. The queries are usually short, on average about 2-3 terms, and their information needs are not as clear as the author, venue, or paper title queries. They are very typical ad-hoc search queries, in a specific domain – computer science.

To understand where `S2` is making mistakes, we studied the error sources of the failure queries in the query log. These failure queries were picked based on the average click depth in the query log: The lower the click, the worse `S2` might be performing. Among the queries with more than 10 clicks, we manually labeled the top 200 worst performing ones. The distribution of error types is shown in Figure 4.1b. The two major causes of failure are *author name not recognized* (22%) and *concept not understood* (20%).

`S2`'s strategy for author queries is to show the author's page. When the author name is not recognized, `S2` uses its normal ranking based on papers' textual similarity with the author name, which often results in unrelated papers.

A *concept not understood* error occurs when `S2` returns papers that do not correctly match the semantic meanings of concept queries. Since this is the biggest error source in S2's ad-hoc ranking part, we further analyzed what makes these queries difficult.

The first part of the difficulty is noise in the exact-match signal. Due to language variety, the query term may not appear frequently enough in the relevant documents (vocabulary mismatch), for example, ‘softmax categorization’ versus ‘softmax classification’. The segmentation of query concepts is also a problem. For example, the whole query ‘natural language interface’ should be considered as a whole because it is one informative unit, but the ranking model matches all words and n-grams of the query, and the result is dominated by the popular phrase ‘natural language’.

The second part is more subtle as it is about the meaning of the query concept. There can be multiple aspects of the query concept, and a paper that mentions it the most frequently may not be about the most important aspect. For example, ‘ontology construction’ is about how to construct ontologies in general, but may not be about how a specific ontology is constructed; ‘dynamic programming segmentation’ is about word segmentation in which dynamic programming is essential, but is not about image segmentation.

To conclude, our analysis finds that there is a gap between query-documents’ textual similarity and their semantic relatedness. Ranking systems that rely solely on term-level statistics have few principled ways to resolve this discrepancy. Our desire to handle this gap led to the development of ESR.

4.2.3 Our Method

We now describe the academic knowledge graph we automatically constructed and then introduce the Explicit Semantic Ranking (`ESR`) method that soft matches query and documents using the knowledge graph.

4.2.3.1 Automatically Constructed Academic Knowledge Graph

The prerequisite of semantic ranking systems is a knowledge graph that stores semantic information. Usually, an entity linking system that links natural language text with entities in the knowledge graph is also necessary [31, 61] (Section 3.2 and 4.1). In this work, we build a standard knowledge graph G with concept entities (E) and edges (predicates P and tails T) by harvesting `S2`’s corpus and Freebase, and use a popularity based entity linking system to link query and documents.

Concept Entities in our knowledge graph can be collected from two different sources: corpus-extracted (`Corpus`) and `Freebase`. `Corpus` entities are keyphrases automatically extracted from `S2`’s corpus. Key phrase extraction is a widely studied task that aims to find representative keyphrases for a document, for example, to approximate the manually assigned keywords of a paper. This work uses the keyphrases extracted by `S2`’s production system, which extracts noun phrases from a paper’s title, abstract, introduction, conclusion and citation contexts, and selects the top ranked ones in a keyphrase ranking model with typical features such as frequency and location [18].

The second source of entities is `Freebase`. Despite being a general domain knowledge base, our manual examination found that `Freebase` has rather good coverage on the computer science concept entities in `S2`’s head queries.

Entity Linking: We use `CMNS` which links surface forms (entity mentions) in a query or document to their most frequently linked entities in Google’s FACC1 annotation [39, 43] (Section 4.1). Although `CMNS` does not provide entity disambiguation, it has been shown to be effective for query entity linking [43], and the language in computer science papers is less ambiguous

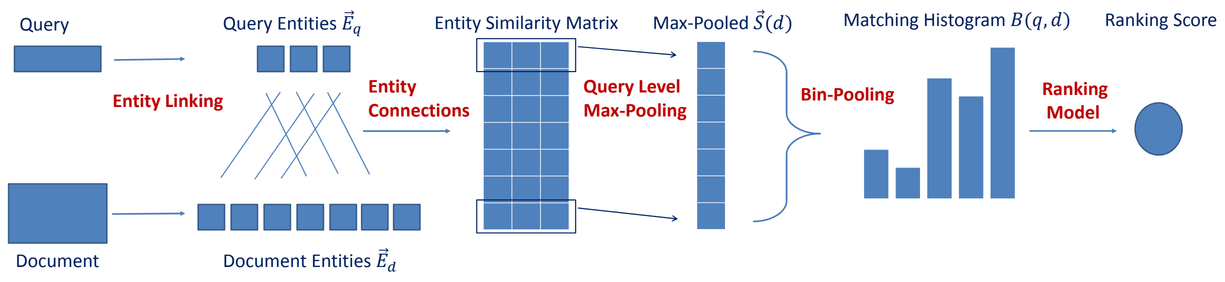


Figure 4.2: Explicit Semantic Ranking Pipeline

than in a more general domain.

Edges in our knowledge graph include three parts: Head H, Predicates P and tails T. From Freebase and the `CMNS` annotated `S2` corpus, the following four types of edges are harvested:

- `Author` edges link an entity to an author if the author has a paper which mentioned the entity in the title;
- `Context` edges link an entity to another entity if the two co-occur in a window of 20 words more than 5 times in the corpus;
- `Desc` edges link an entity to words in its Freebase description (if one exists); and
- `Venue` edges link an entity to a venue if the entity appears in the title of a paper published in the venue.

The knowledge graph G contains two types of entities: Corpus-extracted (`Corpus`) and `Freebase`, and four types of edges: `Author`, `Context`, `Desc` and `Venue`.

Embeddings of our entities are trained based on their neighbors in the knowledge graph. The graph structure around an entity conveys the semantics of this entity. Intuitively, entities with similar neighbors are usually related. We use entity embeddings to model such semantics.

We train a separate embedding model for each of {`Author`, `Context`, `Desc`, `Venue`} edges using the skip-gram model [73]:

$$l = \sum_{e \in E, t \in T} w(e,t)\big(\sigma(V(e)^T U(t)) - E_{\hat{t}\sim T}\sigma(-V(e)^T U(\hat{t}))\big).$$

The loss function l is optimized using a typical gradient method. V and U are the entity embedding matrices learned for entities and tails of this edge type. Each of their rows ($V(e)$ or $U(t)$) is the embedding of an entity e or a tail t, respectively. σ is the Sigmoid function. T is the collection of all tails for this edge type. $E_{\hat{t}\sim T}()$ samples negative instances based on the tails' frequency (negative sampling). $w(e,t)$ is the frequency of entity e and tail t being connected, for example, how many times an entity is used by an author.

4.2.3.2 Ranking Model

Given a query q, and a set of candidate documents $D = \{d_1, ..., d_n\}$, the goal of `ESR` is to find a ranking function $f(q, d|G)$, that better ranks D using the explicit semantics stored in the knowledge graph G. The explicit semantics include entities ($E = \{e_1, ..., e_{|E|}\}$) and edges (predicates P and tails T). The rest of this section describes the ESR framework, which is also shown in Figure 4.2.

Entity-based Representations

ESR represents query and documents by their bag-of-entities constructed using their entity annotations linked by CMNS [80] (Section 4.1). Each query or document is represented by a vector ($\vec{E}_q$ or $\vec{E}_d$). Each dimension in the vector corresponds to an entity e in the query or document's annotation, and the weight is the frequency of the entity being annotated to it.

Match Query and Documents in the Entity Space

Instead of being restricted to classic retrieval models [80] (Section 4.1), ESR matches query and documents' entity representations using the knowledge graph embedding.

ESR first calculates a query-document entity translation matrix. Each element in the matrix is the connection strength between a query entity e_i and a document entity e_j, calculated by their embeddings' cosine similarity:

$$s(e_i, e_j) = \text{cos}(V(e_i), V(e_j)). \tag{4.3}$$

A score of 1 in the entity matrix refers to an *exact match* in the entity space. It incorporates the semantics from entities and their surface forms: The entities in the text are recognized, different surface forms of an entity are aligned, and the exact match is done at the entity level. We call this effect 'smart phrasing'. Scores less than 1 identify related entities as a function of the knowledge graph structure and provide *soft match* signals.

Then ESR performs two pooling steps to generalize the exact matches and soft matches in the entity translation matrix to query-document ranking evidence.

The first step is a max-pooling along the query dimension:

$$\vec{S}(d) = \max_{e_i \in \vec{E}_q} s(e_i, \vec{E}_d). \tag{4.4}$$

$\vec{E}_q$ and $\vec{E}_d$ are the bag-of-entities of q and d. $\vec{S}(d)$ is a $|\vec{E}_d|$ dimensional vector. Its j^{th} dimension is the maximum similarity of the document entity e_j to any query entities.

The second step is a bin-pooling (histogram) to count the matches at different strengths [42]:

$$B_k(q, d) = \log \sum_j I(st_k \leq \vec{S}_j(d) < ed_k). \tag{4.5}$$

$[st_k, ed_k)$ is the range for the k^{th} bin. B_k is the number of document entities whose scores fall into this bin.

The max-pooling matches each document entity to its closest query entity using embeddings, which is the exact-match if one exists. Its score describes how closely related a document entity is to the query. The bin-pooling counts the number of document entities with different connection strengths to the query. The bin with range $[1, 1]$ counts the exact matches, and the other bins generate soft match signals [42]. The two pooling steps together summarize the entity matches to query-document ranking evidence.

Ranking with Semantic Evidence

The bin scores B are used as features for standard learning to rank models in ESR:

$$f(q, d|G) = w_0 f_{S2}(q, d) + W^T B(q, d) \tag{4.6}$$

where $f_{S2}(q, d)$ is the score from S2's production system, w_0 and W are the parameters to learn, and $f(q, d|G)$ is the final ranking score. Based on which edge type the entity embedding is trained, there are four variants of ESR: ESR-Author, ESR-Context, ESR-Desc, and ESR-Venue.

With entity-based representations, the exact matches (smart phrasing) allow ESR to consider multiple words as a single unit in a principled way, and the knowledge graph embeddings allow ESR to describe semantic relatedness via soft matches. The exact match and soft match signals are utilized by ESR's unified framework of embedding, pooling, and ranking.

4.2.4 Experimental Methodology

This section describes the ranking benchmark of S2 and the experimental settings.

4.2.4.1 Semantic Scholar Ranking Benchmark

The queries of the benchmark are sampled from S2's query log in the first six months of 2016. There are in total 100 queries, 20 uniformly sampled from the head traffic (100 most frequent queries), 30 from the median (queries that appear more than 10 times), and 50 hard queries from the 200 worst performing queries based on the average click depth. Author and venue queries are manually ignored as they are more about name recognition instead of ranking.

The candidate documents were generated by pooling from several variations of S2's ranking system. First, we labeled several of the top ranked results from S2. Then several variations of S2's ranking systems, with same features, but different learning to rank models, are trained and tested on these labels using cross-validation. The top 10 results from these rankers obtained from cross-validation were added to the document pool. They were then labeled for another iteration of training. The training-labeling process was repeated twice; after that, rankings had converged.

We also tried pooling with classic retrieval models such as BM25 and the query likelihood model, but their candidate documents were much worse than the production system. Our goal was to improve an already-good system, so we chose to use the higher quality pool produced by S2 variants.

We used the same five relevance categories used by the TREC Web Track. Judging the relevance of research papers to academic queries requires a good understanding of related research topics. It is hard to find crowd-sourcing workers with such research backgrounds. Thus we asked two researchers in the Allen Institute to label the query-document pairs. Label differences were resolved by discussion.

The distribution of relevance labels in our dataset is shown in Table 4.5. The same statistics from the TREC Web Track 2009-2012 are also listed for reference. Our relevance judgments share a similar distribution, although our data is cleaner, for example, due to a lack of spam.

The benchmark dataset is available at http://boston.lti.cs.cmu.edu/appendices/WWW2016/.

Table 4.5: Distribution of relevance labels in Semantic Scholar's benchmark dataset. **S2** shows the number and percentage of query-document pairs from the 100 testing queries that are labeled to the corresponding relevance level. **TREC** shows the statistics of the relevance labels from TREC Web Track 2009-2012's 200 queries.

Relevance Level	**S2**		**TREC**	
Off-Topic (0)	3080	65.24%	54660	77.45%
Related (1)	1060	22.45%	10778	15.27%
Relevant (2)	317	6.71%	3681	5.22%
Exactly-Right (3)	213	4.51%	598	0.85%
Navigational (4)	51	1.08%	858	1.22%

4.2.4.2 Experimental Settings

Data: Ranking performances are evaluated on the benchmark dataset discussed in Section 4.2.4.1. The entity linking performance is evaluated on the same queries with manual entity linking from the same annotators.

Baselines: The baseline is the Semantic Scholar (S2) production system on July 1st 2016, as described in Section 4.2.2. It is a strong baseline. An internal evaluation and a third-party evaluation indicate that its accuracy is at least as good as other academic search engines on the market. We also include BM25-F and tf.idf-F for reference. The BM25 and vector space model are applied to the paper's title, abstract, and body fields. Their parameters (field weights) are learned using the same learning to rank model as our method in the same cross-validation setting.

Evaluation Metrics: Query entity linking is evaluated by Precision and Recall at the query level (strict evaluation [20]) and the average of query level and entity level (lean evaluation [43]). Ranking is evaluated by NDCG@20, the main evaluation metric in the TREC Web Track. As an online production system, S2 especially cares about top positions, so NDCG@$\{1, 5, 10\}$ are also evaluated. Statistical significances are tested by permutation test with $p < 0.05$.

ESR Methods: Based on which edge type is used to obtain the entity embedding, there are four versions of ESR: ESR-Author, ESR-Context, ESR-Desc, and ESR-Venue. Embeddings for ESR-Author and ESR-Venue are trained with authors and venues with more than 1 publication. Description and context embeddings are trained with entities and terms with the minimum frequency of 5.

Entity linking is done by CMNS with all linked entities kept to ensure recall [43] (Section 4.1). Corpus entities do not have multiple surface forms so CMNS reduces to exact match. Freebase entities are linked using surface forms collected from Google's FACC1 annotation [39].

Entity connection scores are binned into five bins: $[1, 1]$, $[0.75, 1)$, $[0.5, 0.75)$, $[0.25, 0.5)$, $[0, 0.25)$ with the exact match bin as the first bin [42]. We discard the negative bins as negative cosine similarities between entities are not informative: most of them are not related at all.

All other settings follow previous standards. The embedding dimensionality is 300; The five bins and three paper fields (title, abstract, and body) generate 15 features, which are com-

Table 4.6: Entity linking evaluation results. Entities are linked by CMNS. Corpus shows the results when using automatically extracted keyphrases as the targets. Freebase shows the results when using Freebase entities as the targets. **Prec**ison and **Rec**all from lean evaluation and strict evaluation are displayed.

	Lean Evaluation		**Strict Evaluation**	
	Prec	**Rec**	**Prec**	**Rec**
Corpus	0.5817	0.5625	0.5400	0.5400
Freebase	0.6960	0.6958	0.6800	0.6800

Table 4.7: Overall accuracy of ESR compared to Semantic Scholar (S2). ESR-Author, ESR-Context, ESR-Desc and ESR-Venue are ESR with entity embedding trained from corresponding edges. Relative performances compared with S2 are in percentages. **W**in/**T**ie/**L**oss are the number of queries a method improves, does not change, or hurts, compared with S2. Best results in each metric are marked **Bold**. Statistically significant improvements (P>0.05) over S2 are marked by †.

Method	**NDCG@5**		**NDCG@20**		**W/T/L**
tf.idf-F	0.2254	−54.93%	0.3299	−39.91%	28/01/71
BM25-F	0.2890	−42.20%	0.3693	−32.75%	32/01/67
Semantic Scholar	0.5000	–	0.5491	–	–/–/–
ESR-Author	0.5501†	+10.02%	0.5935†	+8.08%	60/10/30
ESR-Context	0.5417†	+8.35%	0.5918†	+7.77%	58/04/38
ESR-Desc	0.5496†	+9.92%	0.5875†	+6.99%	55/11/34
ESR-Venue	**0.5700**†	+13.99%	**0.6090**†	+10.91%	59/11/30

bined with S2's original score using linear RankSVM [49]. All models and baselines' parameters are trained and evaluated using a 10-fold cross validation with 80% train, 10% development and 10% test in each fold. The hyper-parameter 'c' of RankSVM is selected from $\{0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1\}$ using the development part of each fold.

4.2.5 Evaluation Results

Five experiments investigated entity linking and document ranking accuracy, as well as the effects of three system components (entity based match, embedding, pooling).

4.2.5.1 Entity Linking Performance

The entity linking accuracy of CMNS on our queries is shown in Table 4.6. Corpus and Freebase refer to using entities from extracted keyphrases in S2's corpus or Freebase. **Prec**ison and **Rec**all are evaluated by lean evaluation (query and entity averaged) and strict evaluation (query only) metrics.

The results in Table 4.6 reveal a clear gap between the quality of automatically extracted entities and manually curated entities. Linking performance is 10-25% better with Freebase entities than Corpus entities on all evaluation metrics, demonstrating the significant differences in the quality of their entity sets. Further analysis finds that Freebase not only provides a larger set of surface forms, but also a cleaner and larger set of computer science concept entities. Indeed, our manual examination found that only the most frequent automatically extracted keyphrases (about ten thousand) are reliable. After that, there is much noise. Those most frequent keyphrases are almost all included in Freebase; little additional information is provided by the Corpus entities.

The absolute Precision and Recall are higher than those on the general domain (TREC Web Track) queries (evaluated in Section 4.1). Our manual examination finds that a possible reason is the lower ambiguity in academic queries than in TREC Web Track queries. Also, since our queries are from the head and middle of the online traffic, they are mostly about popular topics. Freebase's coverage on them is no worse than on general domain queries. Due to its dominating entity linking accuracy, the rest of our experiments used only Freebase.

4.2.5.2 Ranking Performance

The overall performance of ESR is shown in Table 4.7. The four versions of ESR only differ in the edges they used to obtain the entity embedding. Relative performances comparing with the production system Semantic Scholar (S2) are shown in percentages. **W**in/**T**ie/**L**oss are the number of queries outperformed, unchanged, and hurt by each method compared with S2. Statistically significant improvements over S2 are marked by †.

The production system S2 outperforms BM25-F by a large margin. NDCG@1 is almost doubled. This result confirms the quality of the current production system. Although half of the queries were deliberately picked to be difficult, at all depths S2 achieves absolute NDCG scores above 0.5.

ESR provides a further jump over the production system by at least 5% on all evaluation metrics and with all edge types. With Venue, the improvements are consistently higher than 10%. On the earlier positions which are more important for user satisfaction, ESR's performances are also stronger, with about $2 - 3\%$ more improvements on NDCG@1 and NDCG@5 than on NDCG@20. The improvements are statistically significant, with the only exception of ESR-Desc on NDCG@1. We think this behavior is due to the edge types Author, Context, and Venue being domain specific, because they are gathered from S2's corpus, whereas Desc is borrowed from Freebase and no specific attention is paid to the computer science domain.

ESR is designed to improve the queries that are hard for Semantic Scholar. To verify that ESR fulfills this requirement, we plot ESR's performance on individual queries with respect to S2's in Figure 4.3. Each point in the figure corresponds to a query. S2's NDCG@20 is shown in the x-axis. The relative performance of ESR compared with S2 is shown in the y-axis.

In all four sub-figures, ESR's main impact is on the hard queries (the left side). On the queries where S2 already performs well, ESR makes only small improvements: Most queries on the right side stay the same or are only changed a little bit. On the queries where S2's word-based ranking fails, for example, on those whose NDCG < 0.3, the semantics from the knowledge graph improve many of them with large margins. The improvements are also robust. More than half of the queries are improved with big wins. Fewer queries are hurt and the loss is usually small.

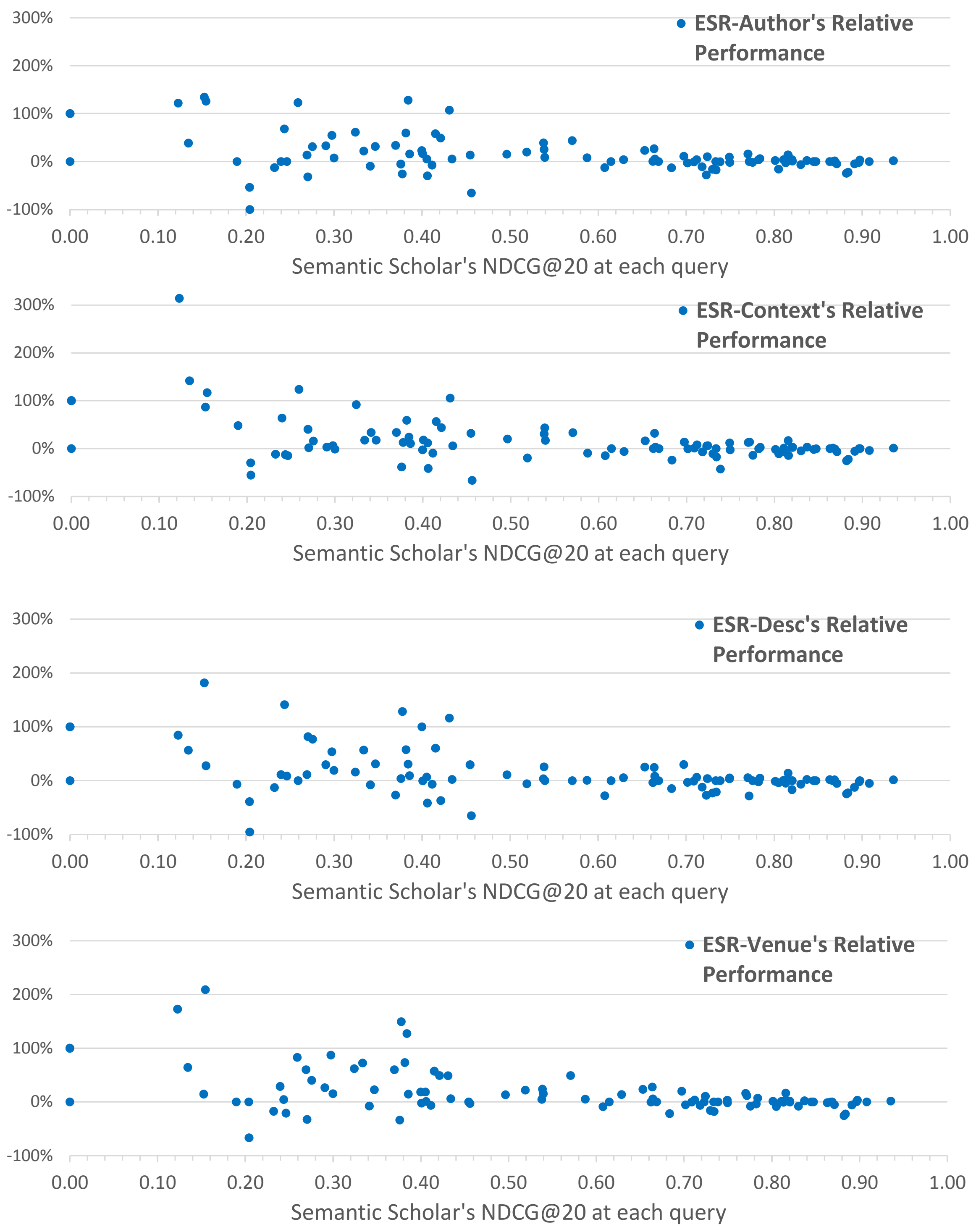


Figure 4.3: `ESR`'s relative NDCG@20 compared with `Semantic Scholar` (`S2`) on individual queries. Each point corresponds to a query. The value on the x-axis is `S2`'s NDCG@20 on each query. The y-axis shows `ESR`'s relative NDCG@20 (percentage) compared with `S2`.

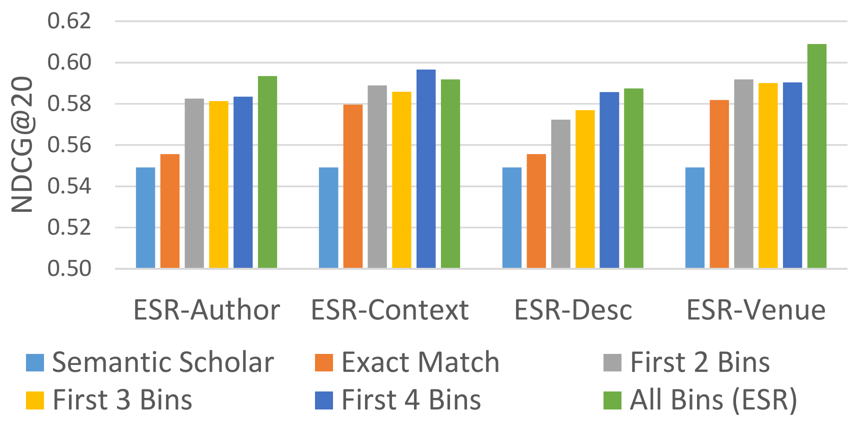


Figure 4.4: `ESR` accuracy using different numbers of matching bins. For each group, from left to right: `Semantic Scholar`, the baseline; `Exact match` which only uses the first bin; `First 2 Bins`, `First 3 Bins`, and `First 4 Bins` which refer to only using the first k bins with the highest matching scores; the last one `All Bins` is the original `ESR`.

This experiment demonstrates `ESR`'s ability to improve a very competitive online production system. `ESR`'s advantages also favor online search engines' user satisfaction: More improvements are at early ranking positions, the improvements are robust with fewer queries badly damaged, and most importantly, `ESR` fixes the hard queries that the production system finds difficult.

4.2.5.3 Effectiveness of Entity-Based Match

`ESR` matches query and documents on their entity representations using the semantics from the knowledge graph. The semantic contribution to `ESR`'s ranking can come from two aspects: exact match with smart phrasing and soft match with knowledge graph embedding. The third experiment investigated the effectiveness of `ESR`'s entity-based matching, both exact match and soft match.

Recall that `ESR` uses five bins to combine and weight matches of different strengths. This experiment used the same ranking model, but varies the bins used. We started with only using the first exact match bin $[1, 1]$; then we added in the second bin $[0.75, 1)$, the third $[0.5, 0.75)$, and the fourth $[0.25, 0.5)$, and evaluated effectiveness of exact match and soft matches at varying strength. Figure 4.4 displays the performance of `ESR` with different number of bins. For each `ESR` version, the bins correspond to, from left to right: `Semantic Scholar`, the baseline; `Exact match` which only uses the first bin; `First 2 Bins`, `First 3 Bins`, and `First 4 Bins` which refer to only using the first k bins with the highest matching scores; the last one `All Bins` is the original `ESR`. For brevity, only NDCG@20 is shown (the y-axis). The behavior is similar for other depths.

Entities' exact match information (the first bin) provides about 6% gains over `S2`, with `Context` and `Venue`. The exact match signals with `ESR-Author` and `ESR-Desc` are sparse, because some entities do not have `Author` or `Desc` edges. In that case, their rankings are reduced to `S2`, and their gains are smaller. Our manual examination found that this gain does come from the 'smart phrasing' effect: an entity's mention is treated as a whole unit and different phrases referring to the same entity are aligned. For example, the rankings of queries like 'natural language interface' and 'robust principle component analysis' are greatly improved, while in S2 their textual similarity signals are mixed by their individual terms and sub-phrases.

The soft match information (later bins) contributes approximately another 5% of improvement. The soft-match bins record how many document entities are related to the query entities at certain strengths. Intuitively, the soft match should contribute for all queries as knowing more about the document entities should always help. But our examination finds this information is more effective on some queries, for example, 'dynamic segmentation programming', 'ontology construction' and 'noun phrases'. The soft match helps ESR find the papers whose meanings (e.g. research area, topic, and task.) are semantically related to these queries' information needs, while S2's word-based match fails to do so.

This experiment helps us understand why the explicit semantics in knowledge graphs is useful for ranking. By knowing which entity a phrase refers to and whether different phrases are about the same thing, the *exact match* signal is polished. By knowing the relatedness between query entities and document entities through their embeddings, additional *soft match* connections that describe query-document relatedness with semantics are incorporated.

4.2.5.4 Effectiveness of Embedding

ESR captures the semantic relatedness between entities by using their distance in the embedding space. An advantage of using embeddings compared with the raw knowledge graph edges is their efficiency, which is tightly constrained in online search engines. By embedding the neighbors of an entity into a dense continuous vector, at run time, ESR avoids dealing with the graph structure of the knowledge graph, which is sparse, of varying length, and much more expensive to deal with than fixed length dense vectors.

However, does ESR sacrifice effectiveness for efficiency? The fourth experiment studied the influence of embeddings on ESR's effectiveness, by comparing it with the ranking performance of using raw knowledge graph edges. In this experiment, a discrete vector representation is formed for each entity for each edge type. Each of the vector's dimensions is a neighbor of the entity. Its weight is the frequency of them being connected together. The Raw variants of ESR are then formed with the knowledge graph embedding replaced by this discrete entity representation and everything else kept the same.

Table 4.8 shows the results of such 'Raw' methods. Different versions of Raw use different edges (Author, Context, Desc and Venue) to represent entities. The relative performance and **W**in/**T**ie/**L**oss are compared with the corresponding ESR version that uses exactly the same information but with knowledge graph embedding. The result shows that ESR actually benefits from the knowledge graph embedding; Raw methods almost always perform worse than the corresponding ESR version. We believe that the advantage of knowledge graph embedding is similar with word embedding [73]: the embedding better captures the semantics by factoring the raw sparse data into a smooth low-dimensional space.

Table 4.8: Performance of different strategies that make use of the knowledge graph in ranking. `Raw` directly calculates the entity similarities in the original discrete space. `Mean` uses mean-pooling when generalizing the entity translation matrix to query-document ranking evidence. `Max` uses max-pooling. `Mean&Bin` replaces the max-pooling in `ESR`'s first stage with mean-pooling. Relative performances (percentages), statistically significant differences (†), and **W**in/**T**ie/**L**oss are compared with the `ESR` version that uses the same edge type and embedding; for example, `Raw-Author` versus `ESR-Author`.

Method	**NDCG@20**		**W/T/L**
`ESR-Author`	0.5935	–	–/–/–
`ESR-Context`	0.5918	–	–/–/–
`ESR-Desc`	0.5875	–	–/–/–
`ESR-Venue`	**0.6090**	–	–/–/–
`Raw-Author`	0.5821	-1.91%	45/06/49
`Raw-Context`	0.5642†	-4.66%	38/06/56
`Raw-Desc`	0.5788	-1.48%	46/05/49
`Raw-Venue`	0.5576†	-8.43%	28/07/65
`Mean-Author`	0.5685†	-4.22%	33/09/58
`Mean-Context`	0.5676†	-4.08%	39/06/55
`Mean-Desc`	0.5660	-3.66%	45/12/43
`Mean-Venue`	0.5599†	-8.07%	32/10/58
`Max-Author`	0.5842	-1.56%	38/10/52
`Max-Context`	0.5861	-0.95%	52/07/41
`Max-Desc`	0.5659†	-3.67%	41/12/47
`Max-Venue`	0.5763†	-5.38%	32/12/56
`Mean&Bin-Author`	0.5823	-1.89%	41/06/53
`Mean&Bin-Context`	0.5808	-1.85%	41/08/51
`Mean&Bin-Desc`	0.5694	-3.07%	38/14/48
`Mean&Bin-Venue`	0.5639†	-7.41%	31/10/59

4.2.5.5 Effectiveness of Pooling

`ESR` applies a two stage pooling on the entity translation matrix to obtain query-document ranking evidence. The first, max-pooling, is used to match each document entity to its closest query entity. The second, bin-pooling, is used to summarize the matching scores of each document entity into match frequencies at different strengths. This experiment evaluates the effectiveness of `ESR`'s pooling strategy.

We compare `ESR`'s two-stage pooling with several common pooling strategies: mean-pooling that summarizes the translation matrix into one average score; max-pooling that only keeps the highest score in the matrix; and mean&bin-pooling which is the same as `ESR`'s max-pooling and bin-pooling, but in the first step the document entities are assigned with their average similarities to query entities. Except for the pooling strategy, all other settings of `ESR` are kept.

The results of different pooling strategies are shown in the second half of Table 4.8. Relative performances and **W**in/**T**ie/**L**oss are compared with the corresponding ESR version. The results demonstrate the effectiveness of ESR's two-stage pooling strategy: All other pooling strategies perform worse. Abstracting the entire entity translation matrix into one mean or max score may lose too much information. Mean&bin pooling also performs worse. Intuitively, we would prefer document entities that match one of the query entities very well, rather than entities that have mediocre matches with multiple query entities. Also, the exact match information is not preserved in the mean&bin-pooling when there are multiple query entities.

This result shows that generalizing the entity level evidence to the query-document level is not an easy task. One must find the right level of abstraction to obtain a good result. There are other abstraction strategies that are more complex and could be more powerful. For example, one can imagine building an RNN or CNN upon the translation matrix, but that would require more training labels. We believe ESR's two-stage pooling provides the right balance of model complexity and abstraction granularity, given the size of our training data.

4.2.6 Explicit Semantic Ranking Summary

This section presents Explicit Semantic Ranking, a new technique that utilizes the explicit semantics from a knowledge graph in academic search. In ESR, queries and documents are represented in the entity space using their annotations, and the ranking is defined by their semantic relatedness described by their entities' connections, in an embedding, pooling, and ranking framework.

Explicit Semantic Ranking is first applied on the academic search domain. This section first provides analysis on the query logs from Semantic Scholar and shows the crucial role entities played in the online production system. Then it presents a simple and effective knowledge graph construction method that automatically built an academic knowledge graph using Freebase and the Semantic Scholar corpus. ESR then utilizes this automatically constructed knowledge graph to improve the search accuracy of Semantic Scholar.

Experiments on a Semantic Scholar testbed demonstrate that ESR improves the production system by 6% to 14%. Additional analysis revealed the effectiveness of the explicit semantics: the entities and their surface forms help recognize the concepts in a query, and polish the *exact match* signal; the knowledge graph structure helps build additional connections between query and documents, and provides effective and novel *soft match* evidence. With the embedding, pooling, and ranking framework that successfully utilizes this semantics, ESR provides robust improvements, especially on the queries that are hard for word-based ranking models.

Perhaps surprisingly, we found that using Freebase entities was more effective than keyphrases automatically extracted from S2's corpus. Although Freebase is considered general-purpose, it provides surprisingly good coverage of the entities in our domain - computer science. The value of this knowledge graph can be further improved by adding domain specific semantics such as venues and authors. This result is encouraging because good domain-specific knowledge bases can be difficult to build from scratch. It shows that entities from a general-domain knowledge base can be a good start.

Different edge types in the knowledge graph convey rather different semantics. In our experiments, there is less than 15% overlap between the close neighbors of an entity in different embedding spaces. The different variants of ESR also perform rather differently on different queries. This work mainly focuses on how to make use of each edge type individually and specifically in the academic search domain. The following chapters will present more unified approaches to utilize knowledge graph semantics as well as their applications in general domain search environments.

4.3 Summary

This chapter presents the bag-of-entities representations and the Explicit Semantic Ranking approach that represents and matches query and documents in the entity space. Instead of restricting to the word-based retrieval framework, it provides a fully entity-driven approach that resembles the classic controlled vocabulary based retrieval but with the advantage of large scale knowledge graphs, automatic entity linking techniques, and new machine learning models. We present thorough analysis showing the applicability of automatic entity annotations in representing query and documents in both the general domain (web search) and a special domain (academic search). The search log analysis on Semantic Scholar further demonstrate the crucial role entities played in a real online production system and the needs of deeper text understanding about entities.

This chapter provides new ranking models that perform both exact match and soft match using the entity-based representations of query and documents. The exact match leverages the semantics from entity linking—disambiguation, synonym resolution, and chunking—and the soft match leverages the structural semantics associated with entities in the knowledge graph and is able to build connections between related entities in the query and documents. Various experiments have demonstrated the effectiveness of the entity-based text representations in search.

The bag-of-entities representation and the Explicit Semantic Ranking form the backbone of *entity-oriented search*, which is a new research topic in information retrieval that aims to leverage knowledge graphs in search systems. Next, Chapter 5 studies how to combine the entity-based text representations with the original word-based representations; it also develops new machine learning models that addresses the uncertainties brought in by the automatic construction of entity-based representations: entity linking errors.

Chapter 5

Combining Word-based and Entity-based Representations

This chapter studies how to combine entity-based text representations with the original word-based representations for search. In the previous chapter, we have shown the effectiveness of *bag-of-entities* representations and the advantage of matching query and documents in the entity space. Entity-oriented search operates in the automatically constructed *bag-of-entities* space and the ranking signals convey information from knowledge graphs that is external to the word-based search system. It is natural to study how the two search paradigms interact with each other and how to them to get the best of both representation spaces.

This chapter first proposes a *word-entity duet framework* that integrates bag-of-words and bag-of-entities. The framework not only combines the two but also leverages the interactions between them which leads to additional ranking signals. It also includes a hierarchical ranking model that handles the noise in the query annotations which is a major bottleneck in entity-oriented search. Then this chapter proposes the `JointSem` approach, which increases the scope of the hierarchical ranking model to include the entity linking step. In `JointSem`, the decision of entity linking is jointly made with entity-based document ranking. Thus the ranking model no longer treats the entity linker as a black box. Instead, it learns how to link entities and how to rank documents simultaneously from relevance judgments.

In the rest of this chapter, Section 5.1 presents the duet framework and Section 5.2 presents the `JointSem` approach.

5.1 Word-Entity Duet

This section presents the word-entity duet framework to utilize knowledge graphs in information retrieval[1] . Instead of centering around words or entities, this work treats them equally and represents the query and documents using both word-based and entity-based representations. Thus the interaction of query and document is no longer a 'solo' of their words or entities,

[1] Chenyan Xiong, Jamie Callan, and Tie-Yan Liu. *Word-Entity Duet Representations for Document Ranking.* In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2017) [100].

but a ‘duet’ of their words and entities. Working together, the word-based and entity-based representations form a four-way interaction: query words to document words (`Qw-Dw`), query entities to document words (`Qe-Dw`), query words to document entities (`Qw-De`), and query entities to document entities (`Qe-De`). This leads to a general methodology for incorporating knowledge graphs into text-centric search systems.

The rich and novel ranking evidence from the word-entity duet does come with a cost. Because it is created automatically, the entity-based representation also introduces uncertainties. For example, an entity can be mistakenly annotated to a query, and may mislead the search system. This sections develops an attention-based ranking model, `AttR-Duet`, that employs a simple attention mechanism to handle the noise in the entity representation. The matching component of `AttR-Duet` focuses on ranking with the word-entity duet, while its attention component focuses on steering the model away from noisy entities. Trained directly from relevance judgments, `AttR-Duet` learns how to demote noisy entities and how to rank documents with the word-entity duet simultaneously.

The effectiveness of `AttR-Duet` is demonstrated on the ClueWeb Category B corpora and TREC Web Track queries. On both ClueWeb09-B and ClueWeb12-B13 , `AttR-Duet` outperforms previous word-based and entity-based ranking systems by at least 14%. We demonstrate that the entities bring additional exact match and soft match ranking signals from the knowledge graph; all entity-based rankings perform similar or better compared to solely word-based rankings. We also find that, when the automatically-constructed entity representations are not as clean, the attention mechanism is necessary for the ranking model to utilize the ranking signals from the knowledge graph. Jointly learned, the attention mechanism is able to demote noisy entities and distill the ranking signals, while without such purification, ranking models become vulnerable to noisy entity representations, and the mixed evidence from the knowledge graph may be more of a distraction than an asset. In the rest of this section, we first present the word-entity duet framework for utilizing knowledge graphs in ranking and then the attention-based ranking model. Experimental settings and evaluations are described in Sections 5.1.3 and 5.1.4. This section wraps up with conclusions about the duet framework.

5.1.1 Duet Framework

We now present our word-entity duet framework for utilizing knowledge graphs in search. Given a query q and a set of candidate documents $D = \{d_1, ..., d_{|D|}\}$, our framework aims to provide a systematic approach to better rank D for q, with the help of a knowledge graph G. In the framework, query and documents are represented by two representations, one word-based and one entity-based (Section 5.1.1.1). The two representations' interactions create the word-entity duet and provide four matching components (Section 5.1.1.2).

5.1.1.1 Word and Entity Based Representations

Word-based representations of query and document are standard bag-of-words: $\text{Qw}(w) = \text{tf}(w, q)$, and $\text{Dw}(w) = \text{tf}(w, d)$. Each dimension in the bag-of-words Qw and Dw corresponds to a word w. Its weight is the word's frequency (tf) in the query or document.

A standard approach is to use multiple fields of a document, for example, title and body. Each document field is usually represented by a separate bag-of-words, for example, Dw_{title} and Dw_{body}, and the ranking scores from different fields are combined by ranking models. In this work, we assume that a document may have multiple fields. However, to make notation simpler, the field notation is omitted in the rest of this section unless necessary.

Entity-based representations are bag-of-entities constructed from entity annotations (Section 4.1): $\text{Qe}(e) = \text{tf}(e, q)$ and $\text{De}(e) = \text{tf}(e, d)$, where e is an entity linked to the query or the document. We use automatic entity annotations from an entity linking system to construct the bag-of-entities (Section 4.1).

An entity linking system finds the entity mentions (surface forms) in a text, and links each surface form to a corresponding entity. For example, the entity 'Barack Obama' can be linked to the query '<u>Obama</u> Family Tree'. 'Obama' is the surface form.

Entity linking systems usually contain two main steps [38]:

1. *Spotting*: To find surface forms in the text, for example, to identify the phrase 'Obama'; and
2. *Disambiguation*: To link the most probable entity from the candidates of each surface form, for example, choosing 'Barack Obama' from all possible Obama-related entities.

A commonly used information in *spotting* is the *linked probability* (lp), the probability of a surface form being annotated in a training corpus, such as Wikipedia. A higher lp means the surface form is more likely to be linked. For example, 'Obama' should have a higher lp than 'table'. The *disambiguation* usually considers two factors. The first is *commonness* (CMNS), the universal probability of the surface form being linked to the entity. The second is the context in the text, which provides additional evidence for disambiguation. A confidence score is usually assigned to each annotated entity by the entity linking system, based on spotting and disambiguation scores.

The bag-of-entities is not the set of surface forms that appear in the text (otherwise it is not much different from phrase-based representation). Instead, the entities are associated with rich semantics from the knowledge graph. For example, in Freebase, the information associated with each entity includes (but is not limited to) its name, alias, type, description, and relations with other entities. The entity-based representation makes these semantics available when matching query and documents.

5.1.1.2 Matching with the Word-Entity Duet

By adding the entity based representation into the search system, the ranking is no longer a solo match between words, but a word-entity duet that includes four different ways a query can interact with a document: query words to document words (`Qw-Dw`); query entities to document words (`Qe-Dw`); query words to document entities (`Qw-De`); and query entities to document entities (`Qe-De`). Each of them is a matching component and generates unique ranking features to be used in our ranking model.

Query Words to Document Words (`Qw-Dw`): This interaction has been widely studied in information retrieval. The matches of Qw and Dw generate term-level statistics such as term frequency and inverse document frequency. These statistics are combined in various ways by standard retrieval models, for example, BM25, language model (Lm), and vector space model.

Table 5.1: Ranking features from query words to document words (title and body) ($\Phi_{\texttt{Qw-Dw}}$).

Feature Description	Dimension
BM25	2
TF-IDF	2
Boolean OR	2
Boolean And	2
Coordinate Match	2
Language Model (Lm)	2
Lm with JM smoothing	2
Lm with Dirichlet smoothing	2
Lm with two-way smoothing	2
Total	18

Table 5.2: Ranking features from query entities (name and description) to document words (title and body) ($\Phi_{\texttt{Qe-Dw}}$).

Feature Description	Dimension
BM25	4
TF-IDF	4
Boolean Or	4
Boolean And	4
Coordinate Match	4
Lm with Dirichlet Smoothing	4
Total	24

This work applies these standard retrieval models on document title and body fields to extract the ranking features $\Phi_{\texttt{Qw-Dw}}$ in Table 5.1.

Query Entities to Document Words (Qe−Dw): knowledge graphs contain textual attributes about entities, such as names and descriptions. These textual attributes make it possible to build cross-space interactions between query entities and document words. Specifically, given a query entity e, we use its name and description as pseudo queries, and calculate their retrieval scores on a document's title and body bag-of-words, using standard retrieval models. The retrieval scores from query entities (name or description) to document's words (title or body) are used as ranking features $\Phi_{\texttt{Qe-Dw}}$. The detailed feature list is in Table 5.2.

Query Words to Document Entities (Qw−De): Intuitively, the texts from document entities should help the understanding of the document. For example, when reading a Wikipedia article, the description of a linked entity in the article is helpful for a reader who does not have the background knowledge about the entity.

The retrieval scores from the query words to document entities' name and descriptions are used to model this interaction. Different from Qe−Dw, in Qw−De, not all document entities are related to the query. To exclude unnecessary information, only the highest retrieval scores from

Table 5.3: Ranking features from query words to document entities (name and description) ($\Phi_{\texttt{Qw-De}}$).

Feature Description	Dimension
Top 3 Coordinate Match on Title Entities	6
Top 5 Coordinate Match on Body Entities	10
Top 3 TF-IDF on Title Entities	6
Top 5 TF-IDF on Body Entities	10
Top 3 Lm-Dirichlet on Title Entities	6
Top 5 Lm-Dirichlet on Body Entities	10
Total	48

Table 5.4: Ranking features from query entities to document's title and body entities ($\Phi_{\texttt{Qe-De}}$).

Feature Description	Dimension
Binned translation scores, 1 exact match bin, 5 soft match Bins in the range $[0, 1)$.	12

each retrieval model are included as features:

$$\Phi_{\texttt{Qw-De}} \supset \text{max-k}(\{\text{score}(q, e)|\forall e \in \text{De}\}),$$

where score(q, e) is the score of q and document entity e from a retrieval model, and max-k() takes the k biggest scores from the set. Applying retrieval models on title and body entities' names and descriptions, the ranking features $\Phi_{\texttt{Qw-De}}$ in Table 5.3 are extracted. We choose a smaller k for title entities as titles are short and rarely have more than three entities.

Query Entities to Document Entities (`Qe-De`): There are two ways the interactions in the entity space can be useful. The *exact match* signal addresses the vocabulary mismatch of surface forms [80] (Section 4.1). For example, two different surface forms, 'Obama' and 'US President', are linked to the same entity 'Barack Obama' and thus are matched. The *soft match* in the entity space is also useful. For example, a document that frequently mentions 'the white house' and 'executive order' may be relevant to the query 'US President'.

We apply the Explicit Semantic Ranking (ESR) technique, presented in Section 4.2, to obtain `Qe-De` features. ESR first calculates an entity translation matrix of the query and document using entity embeddings. Then it gathers ranking features from the matrix by histogram pooling.

ESR was originally applied in Semantic Scholar and its entity embeddings were trained using domain specific information like author and venue. In the general domain, there is much research that aims to learn entity embeddings from the knowledge graph [14, 58]. We choose the TransE model which is effective and efficient enough to be applied on large knowledge graphs [14].

Given the triples (edges) from the knowledge graph (e_h, p, e_t), including e_h and e_t the head entity and the tail entity, and p the edge type (predicate), TransE learns entity and relationship embeddings ($\vec{e}$ and $\vec{p}$) by optimizing the following pairwise loss:

$$\sum_{(e_h,p,e_t)\in G} \sum_{(e'_h,p,e'_t)\in G'} [1 + ||\vec{e_h} + \vec{p} - \vec{e_t}||_1 - ||\vec{e'_h} + \vec{p} - \vec{e'_t}||_1]_+,$$

where $[\cdot]_+$ is the hinge loss, G is the set of existing edges in the knowledge graph, and G' is the randomly sampled negative instances. The loss function ensures that entities in similar graph structures are mapped closely in the embedding space, using the compositional assumption along the edge: $\vec{e_h} + \vec{p} = \vec{e_t}$.

The distance between two entity embeddings describes their similarity in the knowledge graph [14]. Using L1 similarity, a translation matrix can be calculated:

$$T(e_i, e_j) = 1 - ||\vec{e_i} - \vec{e_j}||_1. \tag{5.1}$$

T is the $|\text{Qe}| \times |\text{De}|$ translation matrix. e_i and e_j are the entities in the query and the document respectively.

Then the histogram pooling technique is used to gather query-document matching signals from T [42] (Section 4.2):

$$\vec{S}(\text{De}) = \max_{e \in \text{Qe}} T(e, \text{De}),$$
$$B_k(\vec{S}(\text{De})) = \log \sum_j I(st_k \leq \vec{S}_j(\text{De}) < ed_k),$$

where $\vec{S}(d)$ is the max-pooled $|\text{De}|$ dimensional vector, whose j^{th} dimension is the maximum similarity of the j^{th} document entity to any query entities. $B_k()$ is the k^{th} bin that counts the number of translation scores in its range $[st_k, ed_k)$.

We use the same six bins as in Section 4.2: $[1, 1]$, $[0.8, 1)$, $[0.6, 0.8)$, $[0.4, 0.6)$, $[0, 2, 0.4)$, $[0, 0, 2)$. The first bin is the exact match bin and is equivalent to the entity frequency model (Section 4.1). The other bin scores capture the soft match signal between query and documents at different levels. These bin scores generated the ranking features $\Phi_{\texttt{Qe-De}}$ in Table 5.4.

5.1.1.3 Recap

The word-entity duet incorporates various semantics from the knowledge graph: The textual attributes of entities are used to model the cross-space interactions (`Qe-Dw` and `Qw-De`); the relations in the knowledge graphs are used to model the interactions in the entity space (`Qe-De`), through the knowledge graph embedding. The word-based retrieval models are also included (`Qw-Dw`).

Many prior methods are generalized by the duet framework. For example, the two query expansion methods using Wikipedia or Freebase represent the query using related entities, and then use these entities' texts to build additional connections with the document's text [31, 105] (Section 3.1); the latent entity space techniques first find a set of highly related query entities, and then rank documents using their connections with these entities [61] (Section 3.2); and the entity based ranking methods model the interactions between query and documents in the entity space using exact match [80] (Section 4.1) and soft match (Section 4.2).

Each of the four interactions generates a set of ranking signals. A straightforward way is to use them as features in learning to rank models. However, the entity representations may include noise and generate misleading ranking signals, which motivates our `AttR-Duet` ranking model in the next section.

5.1.2 Attention-based Ranking Model

Unlike bag-of-words, entity-based representations are constructed using automatic entity linking systems. It is inevitable that some entities are mistakenly annotated, especially in short queries where there is less context for disambiguation. If an unrelated entity is annotated to the query, it will introduce misleading ranking features; documents that match the unrelated entity might be promoted. Without additional information, ranking models have little leverage to distinguish the useful signals brought in by correct entities from those by the noisy ones, and their accuracies might be limited.

We address this problem with an attention-based ranking model `AttR-Duet`. It first extracts attention features to describe the quality of query entities. Then `AttR-Duet` builds a simple attention mechanism using these features to demote noisy entities. The attention and the matching of query-documents are trained together using back-propagation, enabling the model to learn simultaneously how to weight entities of varying quality and how to rank with the word-entity duet. The attention features are described in Section 5.1.2.1. The details of the ranking model are discussed in Section 5.1.2.2.

Table 5.5: Attention Features for Query Entities.

Feature Description	Dimension
Entropy of the Surface Form	1
Is the Most Popular Candidate Entity	1
Margin to the Next Candidate Entity	1
Embedding Similarity with Query	1
Total	4

5.1.2.1 Attention Features

Two groups of attention features are extracted for each query entity to model its annotation ambiguity and its closeness to the query.

Annotation Ambiguity features describe the *risk* of an entity annotation. There is a risk that the linker may fail to disambiguate the surface form to the correct entity, especially when the surface form is too ambiguous. For example, ‘Apple’ in a short query can be the fruit or the brand. It is risky to put high attention on it. There are three ambiguity features used in `AttR-Duet`.

The first feature is the entropy of the surface form. Given a training corpus, for example, Wikipedia, we gather the probability of a surface form being linked to different candidate entities, and calculate the entropy of this probability. The higher the entropy is, the more ambiguous the surface form is, and the less attention the model should put on the corresponding query entity. The second feature is whether the annotated entity is the most popular candidate of the surface form, i.e. has the highest commonness score (CMNS). The third feature is the difference between the linked entity’s CMNS to the next candidate entity’s.

A **Closeness** attention feature is extracted using the distance between the query entity and the query words in an embedding space. An entity and word joint embedding model are trained

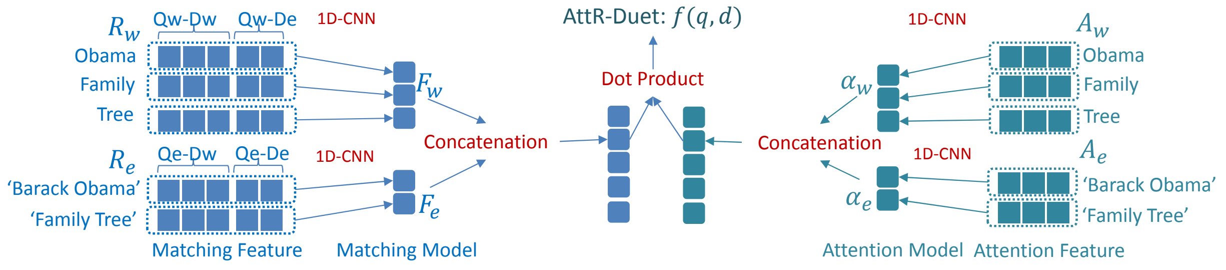


Figure 5.1: The Architecture of the Attention based Ranking Model for Word-Entity Duet (`AttR-Duet`). The left side models the query-document matching in the word-entity duet. The right side models the importances of query entities using attention features. They together produce the final ranking score.

on a corpus including the original documents and the documents with surface forms replaced by linked entities. The cosine similarity between the entity embedding to the query embedding (the average of its words' embeddings) is used as the feature. Intuitively, a higher similarity score should lead to more attention.

The full list of entity attention features, Att(e), is listed in Table 5.5.

5.1.2.2 Model

The architecture of `AttR-Duet` is illustrated in Figure 5.1. It produces a ranking function $f(q, d)$ that re-ranks candidate documents D for the query q, with the ranking features in Table 5.1-5.4 and attention features in Table 5.5. $f(q, d)$ is expected to weight query elements more properly and rank document more accurately.

Model inputs: Suppose the query contains words $\{w_1, ..., w_n\}$ and entities $\{e_1, \dots, e_m\}$, there are four input feature matrices: R_w, R_e, A_w, and A_e. R_w and R_e are the ranking feature matrices for query words and entities in the document. A_w and A_e are the attention feature matrices for words and entities. These matrices' rows are feature vectors previously described:

$$R_w(w_i, \cdot) = \Phi_{\texttt{Qw-Dw}}(w_i) \sqcup \Phi_{\texttt{Qw-De}}(w_i) \tag{5.2}$$
$$R_e(e_j, \cdot) = \Phi_{\texttt{Qe-Dw}}(e_j) \sqcup \Phi_{\texttt{Qe-De}}(e_j) \tag{5.3}$$
$$A_w(w_i, \cdot) = 1 \tag{5.4}$$
$$A_e(e_j, \cdot) = \text{Att}(e_j). \tag{5.5}$$

$\Phi_{\texttt{Qw-Dw}}$, $\Phi_{\texttt{Qw-De}}$, $\Phi_{\texttt{Qe-Dw}}$, and $\Phi_{\texttt{Qe-De}}$ are the ranking features from the word-entity duet, as described in Section 5.1.1. The two feature vectors of a query element are concatenated ($\sqcup$). Att(e_j) is the attention features for entity e_j (Table 5.5). In this work, we use uniform word attention ($A_w = 1$), because the main goal of the attention mechanism is to handle the uncertainty in the entity representations.

The **matching part** contains two Convolutional Neural Networks (CNN's). One matches query words to d (R_w); the other one matches query entities to d (R_e). The convolution is applied on the query element (word/entity) dimension, assuming that the ranking evidence from different query words or entities should be treated the same. The simplest setup with one 1d

CNN layer, 1 filter, and linear activation function can be considered as a linear model applied 'convolutionally' on each word or entity:

$$F_w(w_i) = W_w^m \cdot R_w(w_i, \cdot) + b_w^m \tag{5.6}$$

$$F_e(e_j) = W_e^m \cdot R_e(e_j, \cdot) + b_e^m. \tag{5.7}$$

$F_w(w_i)$ and $F_e(e_j)$ are the matching scores from query word w_i and query entity e_j, respectively. The matching scores from all query words form an n dimensional vector F_w, and those from entities form an m dimensional vector F_e. W_w^m, W_e^m, b_w^m, and b_e^m are the matching parameters to learn.

The **attention part** also contains two CNN's. One weights query words with A_w and the other one weights query entities with A_e. The same convolution idea is applied as the attention features on each query word/entity should be treated the same.

The simplest set-ups with one CNN layer are:

$$\alpha_w(w_i) = \text{ReLU}(W_w^a \cdot A_w(w_i, \cdot) + b_w^a) \tag{5.8}$$

$$\alpha_e(e_j) = \text{ReLU}(W_e^a \cdot A_e(e_j, \cdot) + b_e^a). \tag{5.9}$$

$\alpha_w(w_i)$ and $\alpha_e(e_j)$ are the attention weights on word w_i and entity e_j. $\{W_w^a, W_e^a, b_w^a, b_e^a\}$ are the attention parameters to learn. ReLU activation is used to ensure non-negative attention weights. The matching scores can be negative because only the differences between documents' matching scores matter.

The final ranking score combines the matching scores using the attention scores:

$$f(q, d) = F_w \cdot \alpha_w + F_e \cdot \alpha_e. \tag{5.10}$$

The **training** is done by optimizing the pairwise hinge loss:

$$l(q, D) = \sum_{d \in D^+} \sum_{d' \in D^-} [1 - f(q, d) + f(q, d')]_+. \tag{5.11}$$

D^+ and D^- are the set of relevant documents and the set of irrelevant documents. $[\cdot]_+$ is the hinge loss. The loss function can be optimized using back-propagation in the neural network, and the matching part and the attention part are learned simultaneously.

5.1.3 Experimental Methodology

This section describes the experiment methodology, including dataset, baselines, and the implementation details of our methods.

Dataset: Ranking performances were evaluated on the TREC Web Track ad-hoc task, the standard benchmark for web search. TREC 2009-2012 provided 200 queries for ClueWeb09, and TREC 2013-2014 provided 100 queries for ClueWeb12. The Category B of both corpora (ClueWeb09-B and ClueWeb12-B13) and corresponding TREC relevance judgments were used.

On ClueWeb09-B, the SDM runs provided by EQFE [31] are used as the base retrieval. It is a well-tuned Galago-based implementation and performs better than Indri's SDM. All their settings are inherited, including spam filtering using Waterloo spam score (with a threshold of 60), INQUERY plus web-specific stopwords removal, and KStemming. On ClueWeb12-B13, not all queries' rankings are available from prior work, and Indri's SDM performs similarly to its language model. For simplicity, the base retrieval on ClueWeb12-B13 used is Indri's default language model with KStemming, INQUERY stopword removal, and no spam filtering. All our methods and learning to rank baselines re-ranked the first 100 documents from the base retrieval.

The ClueWeb web pages were parsed using Boilerpipe[2]. The 'KeepEverythingExtractor' was used to keep as much text from the web page as possible, to minimize the parser's influence. The documents were parsed to two fields: title and body. All the baselines and methods implemented by ourselves were built upon the same parsed results for fair comparisons.

The **Knowledge Graph** used in this work is Freebase [13]. The query and document entities were both annotated by TagMe [38]. No filter was applied on TagMe's results; all annotation were kept. This is the most widely used setting of entity-based ranking methods on ClueWeb [80] (Section 3.2, 4.1).

Baselines included standard word-based baselines: Indri's language model (`Lm`), sequential dependency model (`SDM`), and two state-of-the-art learning to rank methods: `RankSVM`[3] [49] and coordinate ascent (`Coor-Ascent`[4]) [71]. `RankSVM` was trained and evaluated using a 10-fold cross-validation on each corpus. Each fold was split to train (80%), develop (10%), and test (10%). The develop part was used to tune the hyper-parameter c of the linear SVM from the set $\{1e-05, 0.0001, 0.001, 0.01, 0.03, 0.05, 0.07, 0.1, 0.5, 1\}$. `Coor-Ascent` was trained using RankLib's recommended settings, which worked well in our experiments. They used the same word based ranking features as in Table 5.1.

Entity-based ranking baselines were also compared. `EQFE` [31] runs are provided by its authors. Our own entity-oriented search methods, including `EsdRank` (Section 3.2), `BOE-TagMe` (Section 4.1) and `ESR` (Section 4.2) are also compared. The comparisons with `EQFE` and `EsdRank` are mainly done on ClueWeb09, because when they were developed not all ClueWeb12 TREC queries were released. `BOE-TagMe` is the best TagMe based runs in Section 4.1, which is TagMe-EF on ClueWeb09-B and TagMe-COOR on ClueWeb12-B13. We apply `ESR` on web search with the same set-up as it was applied on academic search (Section 4.2) except the entity embeddings are obtained by TransE [14] on Freebase instead of Skip-gram on the academic knowledge graph.

There are also other *unsupervised* entity-based systems [61, 80] (Section 3.1); it is unfair to compare them with our supervised methods.

Evaluation was done by NDCG@20 and ERR@20, the official TREC Web Track ad-hoc task evaluation metrics. Statistical significances were tested by permutation test with $p < 0.05$.

Feature Details: All parameters in the unsupervised retrieval model features were kept default. All texts were reduced to lower case, punctuation was discarded, and standard INQUERY stopwords were removed. Document fields included title and body, both parsed by Boilerpipe. Entity textual fields included name and description. When extracting `Qw-De` features, if a document did not have enough entities (3 in title or 5 in body), the feature values were set to -20. The

[2]https://github.com/kohlschutter/boilerpipe
[3]https://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html
[4]https://sourceforge.net/p/lemur/wiki/RankLib/

TransE embeddings were trained using Fast-TransX library[5]. The embedding dimension used is 50.

When extracting the attention features in Table 5.5, the word and entity joint embeddings were obtained by training a skip-gram model on the candidate documents using Google's word2vec [73] with 300 dimensions; the surface form's statistics were calculated from Google's FACC1 annotation [39].

Model Details: `AttR-Duet` was evaluated using 10-fold cross validation with the same partitions as `RankSVM`. Deeper neural networks were explored but did not provide much improvement so the simplest CNN setting was used: 1 layer, 1 filter, linear activation for the ranking part, and ReLU activation for the attention part. All CNN's weights were L2-regularized. Regularization weights were selected from the set $\{0, 0.001, 0.01, 0.1\}$ using the develop fold in the cross validation. Training loss was optimized using the Nadam algorithm [90]. Our implementation was based on Keras. To facilitate implementation, input feature matrices of query elements were padded to the maximum length with zeros. Batch training was used, given the small size of training data.

Using a common CPU, the training took 4-8 hours to converge on ClueWeb09-B and 2-4 hours on ClueWeb12-B13. The testing is efficient as the neural network is shallow. The document annotations, TransE embeddings, and surface form information can be obtained off line. Query entity linking is efficient given the short query length. If the embedding results and entities' texts are maintained in memory, the feature extraction is of the same complexity as typical learning to rank features.

The rankings, evaluation results, and the data used in our experiments are available online at `http://boston.lti.cs.cmu.edu/appendices/SIGIR2017_word_entity_duet/`.

5.1.4 Evaluation Results

This section first evaluates the overall ranking performances of the word-entity duet with attention based learning to rank. Then it analyzes the two parts of `AttR-Duet`: Matching with the word-entity duet and the attention mechanism.

5.1.4.1 Overall Performance

The overall accuracies of `AttR-Duet` and baselines are shown in Table 5.12. Relative performances over `RankSVM` are shown in percentages. **W**in/**T**ie/**L**oss are the number of queries a method improves, does not change, and hurts, compared with `RankSVM` on NDCG@20. Best results in each metric are marked **Bold**. § indicates statistically significant improvements over *all* available baselines.

`AttR-Duet` outperformed all baselines significantly by large margins. On ClueWeb09-B, a widely studied benchmark for web search, `AttR-Duet` improved `RankSVM`, a strong learning to rank baseline, by more than 20% at NDCG@20, and more than 30% at ERR@20, showing the advantage of the word-entity duet over bag-of-words. `ESR`, `EQFE` and `EsdRank`, previous

[5]https://github.com/thunlp/Fast-TransX

Table 5.6: Overall accuracies of AttR-Duet. (U) and (S) indicate unsupervised or supervised method. (E) indicates that information from entities is used. Relative performances compared with RankSVM are shown in percentages. **W**in/**T**ie/**L**oss are the number of queries a method improves, does not change, or hurts, compared with RankSVM on NDCG@20. Best results are marked **bold**. § marks statistically significant improvements over *all* baselines. The baseline scores are in general higher than in previous sections because the experimental setups are improved with more proper relevance judgments (Category-B), a better HTML parser (Boilerpipe), better tuned base retrieval models (from EQFE [31]), and revised ranking features.

ClueWeb09-B						
Method		**NDCG@20**		**ERR@20**		**W/T/L**
Lm	(U)	0.1757	−33.33%	0.1195	−22.63%	47/28/125
SDM	(U)	0.2496	−5.26%	0.1387	−10.20%	62/38/100
RankSVM	(S)	0.2635	–	0.1544	–	–/–/–
Coor-Ascent	(S)	0.2681	+1.75%	0.1617	+4.72%	71/47/82
BOE-TagMe	(UE)	0.2294	−12.94%	0.1488	−3.63%	74/25/101
ESR	(SE)	0.2695	+2.30%	0.1607	+4.06%	80/39/81
EQFE	(SE)	0.2448	−7.10%	0.1419	−8.10%	77/33/90
EsdRank	(SE)	0.2644	+0.33%	0.1756	+13.69%	88/28/84
AttR-Duet	(SE)	**0.3197**§	+21.32%	**0.2026**§	+31.21%	101/37/62

ClueWeb12-B13						
Method		**NDCG@20**		**ERR@20**		**W/T/L**
Lm	(U)	0.1060	−12.02%	0.0863	−6.67%	35/22/43
SDM	(U)	0.1083	−10.14%	0.0905	−2.08%	27/25/48
RankSVM	(S)	0.1205	–	0.0924	–	–/–/–
Coor-Ascent	(S)	0.1206	+0.08%	0.0947	+2.42%	36/32/32
BOE-TagMe	(UE)	0.1173	−2.64%	0.0950	+2.83%	44/19/37
ESR	(SE)	0.1166	−3.22%	0.0898	−2.81%	30/23/47
EQFE	(SE)	n/a	–	n/a	–	–/–/–
EsdRank	(SE)	n/a	–	n/a	–	–/–/–
AttR-Duet	(SE)	**0.1376**§	+14.22%	**0.1154**§	+24.92%	45/24/31

state-of-the-art entity-based ranking methods, were outperformed by at least 15%. It is not surprising because the duet framework includes all of their effects, as discussed in Section 5.1.1.3. ClueWeb12-B13 has been considered a hard dataset due to its noisy corpus and harder queries. The size of its training data is also smaller, which limits the strength of our neural network. However, AttR-Duet still significantly outperformed all available baselines by at least 14%. The information from entities is effective and also different with those from words: AttR-Duet influences more than 75% of the queries, and improves the majority of them.

Table 5.7: Ranking accuracy with each group of matching feature from the word-entity duet. `Base Retrieval` is `SDM` on ClueWeb09 and `Lm` on ClueWeb12. `LeToR-Qw-Dw` uses the query and document's BOW (Table 5.1). `LeToR-Qe-Dw` uses the query's BOE and document's BOW (Table 5.2), `LeToR-Qw-De` is the query BOW + document BOE (Table 5.3), and `LeToR-Qe-De` uses the query and document's BOE (Table 5.4). `LeToR-All` uses all groups. Relative performances in percentages, **W**in/**T**ie/**L**oss on NDCG@20, and statistically significant improvements (†) are all compared with `Base Retrieval`.

ClueWeb09-B					
Method	**NDCG@20**		**ERR@20**		**W/T/L**
`Base Retrieval`	0.2496	$--$	0.1387	$--$	–/–/–
`LeToR-Qw-Dw`	0.2635†	+5.55%	0.1544†	+11.36%	100/38/62
`LeToR-Qe-Dw`	0.2729	+9.33%	**0.1824**†	+31.51%	82/34/84
`LeToR-Qw-De`	**0.2867**†	+14.83%	0.1651†	+19.07%	91/39/70
`LeToR-Qe-De`	0.2695†	+7.97%	0.1607†	+15.88%	99/40/61
`LeToR-All`	**0.3099**†	+24.13%	**0.1955**†	+40.97%	103/38/59
ClueWeb12-B13					
Method	**NDCG@20**		**ERR@20**		**W/T/L**
`Base Retrieval`	0.1060	$--$	0.0863	$--$	–/–/–
`LeToR-Qw-Dw`	**0.1205**	+13.67%	0.0924	+7.14%	43/22/35
`LeToR-Qe-Dw`	0.1110	+4.66%	**0.0928**	+7.63%	40/20/40
`LeToR-Qw-De`	0.1146	+8.09%	0.0880	+1.96%	42/20/38
`LeToR-Qe-De`	0.1166	+10.01%	0.0898	+4.13%	38/20/42
`LeToR-All`	**0.1205**	+13.69%	**0.1000**	+15.93%	47/19/34

5.1.4.2 Matching with Word-Entity Duet

In a sophisticated system like `AttR-Duet`, it is hard to tell the contributions of different components. This experiment studies how each of the four-way interactions in the word-entity duet contributes to the ranking performance individually. For each group of the matching features in Table 5.1- 5.4, we train a RankSVM individually, which resulted in four ranking models: `LeToR-Qw-Dw`, `LeToR-Qe-Dw`, `LeToR-Qw-De`, and `LeToR-Qe-De`. `LeToR-All` which uses all ranking features is also evaluated. In `LeToR-Qw-De` and `LeToR-Qe-De`, the score of the base retrieval model is included as a feature, so that there is a feature to indicate the strength of the word-based match for the whole document. All these methods were trained and tested in the same setting as `RankSVM`. As a result, `LeToR-Qw-Dw` is equivalent to the `RankSVM` baseline, and `LeToR-Qe-De` is equivalent to the `ESR` baseline.

Their evaluation results are listed in Table 5.7. Relative performances (percentages), **W**in/**T**ie/**L**oss, and statistically significant improvements (†) are all compared with `Base Retrieval` (`SDM` on ClueWeb09 and `Lm` on ClueWeb12). All four groups of matching features were able to improve the ranking accuracy of `Base Retrieval` when used individually as ranking features, demonstrating the usefulness of all matching components in the duet. On ClueWeb09-B, all three entity related components, `LeToR-Qe-Dw`, `LeToR-Qw-De`, and

Table 5.8: Examples of entities used in Qw-De and Qe-De. The first half are examples of matched entities in relevant and irrelevant documents, which are used to extract Qw-De features. The second half are examples of entities falls into the exact match bin and the closest soft match bins, used to extract Qe-De features.

Examples of Most Similar Entities to the Query		
Query	**In Relevant Documents**	**In Irrelevant Documents**
Uss Yorktown Charleston SC	'USS Yorktown (CV-10)'	'Charles Cornwallis', 'USS Yorktown (CV-5)'
Flushing	'Roosevelt Avenue', 'Flushing, Queens'	'Flushing (physiology)', 'Flush (cards)'
Examples of Neighbors in Knowledge Graph Embedding		
Query	**In Exact Match Bin**	**In Soft Match Bins**
Uss Yorktown Charleston SC	'USS Yorktown (CV-10)', 'Charleston, SC'	'Empire of Japan', 'World War II'
Flushing	'Flushing, Queens'	'Brooklyn', 'Manhattan', 'New York'

LeToR-Qe-De, provided similar or better performances than the word-based RankSVM. When all features were used together, LeToR-All significantly improved RankSVM by 17% and 26% on NDCG@20 and ERR@20, showing that the ranking evidence from different parts of the duet can reinforce each other.

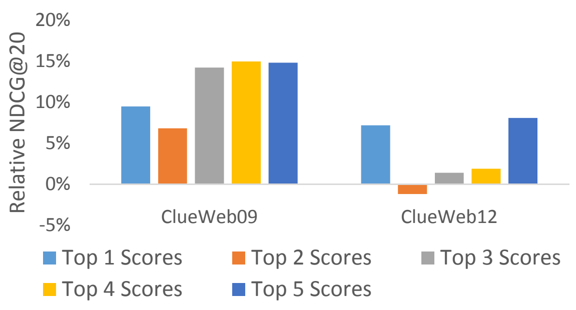
On ClueWeb12-B13, entity-based matching was less effective. LeToR-All's NDCG@20 was the same as RankSVM's, despite additional matching features. The difference is that the annotation quality of TagMe on ClueWeb12 queries is lower (Table 5.9). The noisy entity representation may mislead the ranking model, and prevent the effective usage of entities. To deal with this uncertainty is the motivation of the attention based ranking model, which is studied in Section 5.1.4.4.

5.1.4.3 Matching Feature Analysis

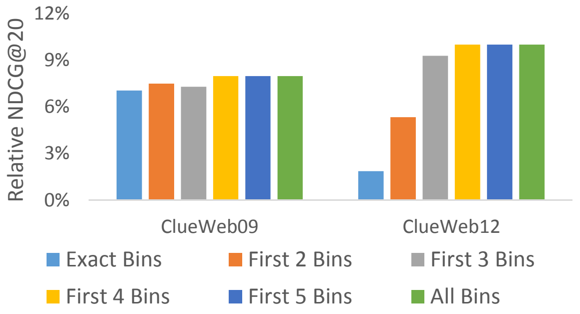
The features from the word space (Qw-Dw) are well understood, and the feature from the query entities to document words (Qe-Dw) have been studied in prior research [31, 61] (Section 3.2). This experiment analyzes the features from the two new components (Qw-De and Qe-De).

Qw-De features match the query words with the document entities. For each document, the query words are matched with the textual fields of document entities using retrieval models, and the highest scores are Qw-De features.

We performed an incremental feature analysis of LeToR-Qw-De. Starting with the highest scored entities from each group in Table 5.3, we incrementally added the next highest ones to the model and evaluated the ranking performance. The results are shown in Figure 5.2a. The y-axis is the relative NDCG@20 improvements over the base retrieval model. The x-axis is the used features. For example, 'Top 3 Scores' uses the top 3 entities' retrieval scores in each row of Table 5.3.



(a) Features from Query Words to Document Entities (`Qw-De`)



(b) Features from Query Entities to Document Entities (`Qe-De`)

Figure 5.2: Incremental analysis for duet ranking features. The y-axis is the relative NDCG@20 improvement over the base retrieval. The x-axis refers to the features from only top k (1-5) entity match scores (5.2a), and the features from only first k (1-6) bins in the `ESR` model (5.2b), both ordered incrementally from left to right.

The highest scores were very useful. Simply combining them with the base retrieval provided nearly 10% gain on ClueWeb09-B and about 7% on ClueWeb12-B13. Adding the following scores was not that stable, perhaps because the corresponding entities were rather noisy, given the simple retrieval models used to match query words with them. Nevertheless, the top 5 scores together further improve the ranking accuracy.

The first half of Table 5.8 shows examples of entities with highest matching scores. We found that such 'top' entities from relevant documents are frequently related to the query, for example, 'Roosevelt Avenue' is an avenue across Flushing, NY. In comparison, entities from irrelevant documents are much noisier. `Qw-De` features make use of this information and generate useful ranking evidence.

`Qe-De` features are extracted using the Explicit Semantic Ranking (ESR) method (Section 4.2). ESR is built upon the translation model. It operates in the entity space, and extracts the ranking features using histogram pooling. ESR was originally applied on academic search. This section introduces ESR to web search and uses the TransE model [14] to train general domain embeddings from the knowledge graph.

To study the effectiveness of ESR in our setting, we also performed an incremental feature analysis of `LeToR-Qe-De`. Starting from the first bin (exact match), the following bins (soft matches) were incrementally added into RankSVM, and their rankings were evaluated. The results are shown in Figure 5.2b. The y-axis is the relative NDCG@20 over the base retrieval model they re-rank. The x-axis is the features used. For example, First 3 Bins refers to using the first three bins: $[1, 1]$, $[0.8, 1)$, and $[0.6, 0.8)$.

The observation on Semantic Scholar (Section 4.2) holds on ClueWeb: Both exact match and soft match with entities are useful. The exact match bin provided a 7% improvement on ClueWeb09-B, while only 2% on ClueWeb12-B13. Similar exact match results were also observed in our prior study (Section 4.1). It is another reflection of the entity annotation quality differences on the two datasets. Adding the later bins almost always improves the ranking accuracy, especially on ClueWeb12.

Table 5.9: Query annotation accuracy and the gain of attention mechanism. TagMe Accuracy includes the precision and recall of TagMe on ClueWeb queries, evaluated in Section 4.1. Attention Gains are the relative improvements of `AttR-Duet` compared with `LeToR-All`. Statistical significant gains are marked by †.

	TagMe Accuracy		**Attention Gain**	
	Precision	**Recall**	**NDCG@20**	**ERR@20**
ClueWeb09	0.581	0.597	+3.16%†	+3.65%
ClueWeb12	0.460	0.555	+14.20%†	+15.45%†

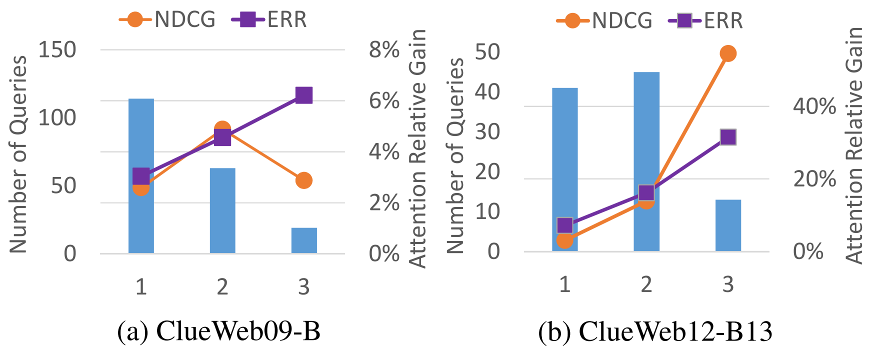


Figure 5.3: Attention mechanism's gain on queries that contain different number of entities. The x-axis is the number of entities in the queries. The y-axis is the number of queries in each group (histogram), and the gain from attention (plots).

The second half of Table 5.8 shows some examples of entities in the exact match bin and the nearest soft match bins. The exact match bin includes the query entities and is expected to help. The first soft match bin usually contains related entities. For example, the neighbors of 'USS Yorktown (CV-10)' include 'World War II' which is when the ship was built. The further bins are mostly background noise because they are too far away. The improvements are mostly from the first 3 bins.

5.1.4.4 Attention Mechanism Analysis

The last experiment studies the effect of the attention mechanism by comparing `AttR-Duet` with `LeToR-All`. If enforcing flat attention weights on all query words and entities, `AttR-Duet` is equivalent to `LeToR-All`: The matching features, model function, and loss function are all the same. The attention part is their only difference, whose effect is reflected in this comparison.

The gains from the attention mechanism are shown in Table 5.9. To better understand the attention mechanism's effectiveness in demoting noisy query entities, the query annotation's quality evaluated in Section 4.1 is also listed. The percentages in the Attention Gain columns are relative improvements of `AttR-Duet` compared with `LeToR-All`. † marks statistical significance. Figure 5.3 breaks down the relative gains to queries with different numbers of query entities. The x-axis is the number of query entities. The histograms are the number of queries in each group, marked by the left y-axis. The plots are the relative gains, marked by the right y-axis.

Table 5.10: Examples of learned attention weights. The entities in **bold blue** draw more attention; those in gray draw less attention.

Query	Entity Attention
Balding Cure	**'Cure'** 'Clare Balding'
Nicolas Cage Movies	**'Nicolas Cage'** 'Pokemon (Anime)'
Hawaiian Volcano Observatories	**'Volcano'**; **'Observatory'** 'Hawaiian Narrative';
Magnesium Rich Foods	**'Magnesium'**; **'Food'** 'First World'
Kids Earth Day Activities	**'Earth Day'** 'Youth Organizations in the USA'

The attention mechanism is essential to ClueWeb12-B13. Without the attention model, `LeToR-All` was confused by the noisy query entities and could not provide significant improvements over word-based models, as discussed in the last experiment. With the attention mechanism, `AttR-Duet` improved `LeToR-All` by about 15%, outperforming all baselines. On ClueWeb09 where TagMe's accuracy is better (Section 4.1), the ranking evidence from the word-entity duet was clean enough for `LeToR-All` to improve ranking, so the attention mechanism's effect was smaller. Also, in general, the attention mechanism is more effective when there are more query entities, while if there is only one entity there is not much to tweak.

The motivation for using attention is to handle the uncertainties in the query entities, a crucial challenge in utilizing knowledge bases in search. These results demonstrated its ability to do so. We also found many intuitive examples in the learned attention weights, some listed in Table 5.10. The **bold blue** entities on the first line of each block gain more attention (> 0.6 attention score). Those in gray on the second line draw less attention (< 0.4 score). The attention mechanism steers the model away from those mistakenly linked query entities, which makes it possible to utilize the correct entities' ranking evidence from a noisy representation.

5.1.5 Word-Entity Duet Summary

This section presents a word-entity duet framework for utilizing knowledge bases in document ranking. In the framework, the query and documents are represented by both word-based and entity-based representations. The four-way interactions between the two representation spaces form a word-entity duet that can systematically incorporate various semantics from the knowledge graph. From query words to document words (`Qw-Dw`), word-based ranking features are included. From query entities to document entities (`Qe-De`), entity-based exact match and soft match evidence from the knowledge graph structure are included. The entities' textual fields are used in the cross-space interactions `Qe-Dw`, which expands the query, and `Qw-De`, which enriches the document.

To handle the uncertainty introduced from the automatic-thus-noisy entity representations, a new ranking model `AttR-Duet` is developed. It employs a simple attention mechanism to demote the ambiguous or off-topic query entities, and learns simultaneously how to weight entities of varying quality and how to rank documents with the word-entity duet.

Experimental results on the TREC Web Track ad-hoc task demonstrate the effectiveness of proposed methods. `AttR-Duet` significantly outperformed all word-based and entity-based ranking baselines on both ClueWeb corpora and all evaluation metrics. Further experiments reveal that the strength of the method comes from both the advanced matching evidence from the word-entity duet, and the attention mechanism that successfully 'purifies' them. On ClueWeb09 where the query entities are cleaner, all the entity related matching components from the duet provide similar or better improvements compared with word-based features. On ClueWeb12 where the query entities are noisier, the attention mechanism steers the ranking model away from noisy entities and is necessary for stable improvements.

5.2 Joint Entity Linking and Entity-based Ranking

In entity-oriented search systems, a key step is entity linking, which aligns the texts in the query and document to the knowledge graph's semantics. Though significant progress has been made in both fronts, entity linking and entity-based ranking research were developed separately. Entity linking systems are mostly optimized for their own metrics, which may not suit the needs of entity-based ranking. For example, entity linking systems may prefer high accuracy on several named entity categories [20], while entity-based search systems need high recall on general domain entities to ensure coverage of the query traffic (Section 4.2). On the other hand, entity-based ranking systems merely treat the entity linker as a pre-processing step, and use the annotation as a black box. Even the state-of-the-art automatic entity linking systems still make mistakes, especially on short queries (Section 5.1). Frequently the noise introduced by the entity linker is the main error source of entity-based ranking systems [61] (Section 3.2). Without access to the detailed linking process, the best an entity-based search system can do is manual correction [31, 61] and post-pruning (Section 3.2, 5.1).

This section presents `JointSem`, a joint semantic ranking method that combines query entity linking and entity-based document ranking[6]. `JointSem` spots surface forms in the query, links multiple candidate entities to each spotted surface form to avoid over committing (soft-alignment), and ranks documents using the linked entities. The spotting and linking evidence widely used in the entity linking literature are incorporated to weight the entities, and standard entity-based ranking features are used to rank documents. The whole system – spotting, linking, and ranking – is trained jointly by a learning-to-rank model using document relevance labels.

[6] Chenyan Xiong, Zhengzhong Liu, Jamie Callan, and Eduard Hovy. *JointSem: Combining Query Entity Linking and Entity based Document Ranking.* In Proceedings of the 26th ACM International Conference on Information and Knowledge Management (CIKM 2017) [102]. The majority of this work was done as the course project for the class '11-727, Computational Semantics for NLP' at Carnegie Mellon University. Zhengzhong Liu was the TA and Eduard Hovy was a Instructor of the course. They provided valuable feedback and helped the paper writing.

Table 5.11: Spotting, linking, and ranking features. Surface Form Features are extracted for each spotted surface form. Entity Features are extracted for each candidate entity from each spot. Entity-Document Ranking Features are extracted for each entity-document pair. The number in brackets is the dimension of the corresponding feature group.

Surface Form Features (ϕ_s)	**Entity Features** (ϕ_e)
(1) Linked Probability	(1) Commonness
(1) Surface Form Entropy	(2) Max and Mean Similarity with Query Words
(1) Top Candidate Entities Margin	
(1) Surface Form Length and Coverage	(2) Max and Mean Similarity with Other Query Entities
(1) Surface Form Coverage	

Entity-Document Ranking Features (ϕ_r)
(16) BM25, Coordinate Match, TFIDF and language model with Dirichlet smoothing from entity's textual fields (name and description) to document's fields (title and body)

Our experiments on TREC Web Track datasets demonstrate that the joint model is more effective for ranking; significant improvements were found over both word-based and entity-based ranking systems. Our analysis further reveals that the advantage of `JointSem` comes from both the soft-alignment that passes more information to the ranking model, and the joint modeling of spotting, linking, and ranking signals.

5.2.1 Joint Semantic Ranking

This section first describes the spotting, linking, and ranking features used in `JointSem`, and then the joint learning-to-rank model.

5.2.1.1 Spotting, Linking, and Ranking

`JointSem` first aligns entities from the knowledge graph to the given query q in a two-step approach: spotting and linking. The spotting step detects surface forms (entity mentions) $S = \{s_1, ...s_i..., s_M\}$ that appear in the query. The linking step aligns each surface form s_i to some candidate entities: $E^i = \{e^i_1, ...e^i_j..., e^i_N\}$. `JointSem` uses a soft-alignment so that multiple candidate entities are kept. Typical spotting and linking features are extracted for the final ranking model to decide the importance of each soft-aligned entity.

Spotting: The spotting step is conducted by looking up the n-grams in the query in a surface form dictionary. The surface form dictionary contains all the possible surface forms (names and aliases) of entities, and is collected from a training corpus, for example, Wikipedia. Following prior convention [20], we start from the *first* word, spot the longest surface forms, and then move to the word after the surface form. No overlapped spots are allowed.

In spotting, the following features ($\phi_s(s_i)$) are extracted to describe the reliability of the surface form.

- *Linked Probability* is the probability of a surface form being linked to any entity in the training corpus.

- *Surface Form Entropy* is the entropy of the probabilities of a surface form being linked to different entities in the training corpus (Section 5.1).
- *Top Candidate Margin* is the difference between the probabilities of the surface form being linked to the most frequent entity and the second one.
- *Surface Form Length and Coverage* are the number of words the surface form contains and the fraction of the query words it covers.

Linking: The linking step aligns entities to each spotted surface form. The surface form dictionary contains the mapping from surface forms to its candidate entities, collected from the training corpus. If a surface form has multiple possible candidate entities, *all* of them are linked (soft-alignment).

The relevance of an entity (e) to the query is described by the following features ($\phi_e(e)$).

- *Commonness* is the probability of the surface form being linked to the entity among all its appearances in the training corpus (Section 5.1).
- *Similarity with Query Words:* The similarity between a query word and a candidate entity is calculated by the cosine of their pre-trained embeddings (see §5.2.2). The max and mean of the entity's embedding similarity to all query words are used as features.
- *Similarity with Other Query Entities:* The similarities between an entity and the top entity (the one with the highest Commonness) of the other spots are calculated using the pre-trained embeddings. The max and mean of these similarity scores are used as its features.

Ranking: The aligned query entities provide many new ranking features. For each linked entity, `JointSem` uses its textual fields as pseudo queries, and extracts entity-document ranking features using standard retrieval models. The ranking features ϕ_r used in this work are: BM25, TFIDF, Coordinate Match, and language model with Dirichlet smoothing, applied on the entity's name and description and the document's title and body [61] (Section 3.2, 5.1). The descriptions are lower-cased and the standard INQUERY stopwords are removed.

The full list of features is shown in Table 5.11. In a re-ranking setting, all these features are efficient enough to be extracted online if the surface form dictionary, embeddings, and the entity's textual fields are maintained in the memory.

5.2.1.2 Joint Learning to Rank

`JointSem` ranks the candidate document d for q using the entity-based ranking features $\phi_r(E, d)$. Additionally, `JointSem` aims to learn how to better utilize the entity-based ranking signals using the surface form features $\phi_s(S)$ and entity features $\phi_e(E)$. The joint ranking model contains three components.

The first part learns the importance of the surface form s_i:

$$f_s(s_i) = w_s^T \phi_s(s_i).$$

The second part learns the importance of the aligned entity e_j^i:

$$f_e(e_j^i) = w_e^T \phi_e(e_j^i).$$

The third part learns the ranking of document for the entity e^i_j:

$$f_r(e^i_j, d) = w_r^T \phi_r(e^i_j, d).$$

The three parts are combined to the final ranking score:

$$f(q, d|\theta, S, E) = \sum_{i=1}^{M} \sum_{j=1}^{N} f_s(s_i) \cdot f_e(e^i_j) \cdot f_r(e^i_j, d), \tag{5.12}$$

where M is the number of surface forms in the query, N is the number of candidate entities aligned per surface form ($N > 1$), $f(q, d|\theta, S, E)$ is the final ranking score produced by `JointSem`, and $\theta = \{w_s, w_e, w_r\}$ are the parameters to learn.

Training uses standard pairwise learning-to-rank with hinge loss:

$$\theta^* = \text{argmin}_\theta \sum_q \sum_{d^+, d^- \in D_q^{+,-}} [1 - f(q, d^+|S, E) + f(q, d^-|S, E)]_+.$$

$D_q^{+,-}$ is the pairwise document preferences ($d^+ > d^-$). The whole model is differentiable and is optimized by standard back-propagation.

In Equation 5.12, $f_s(s_i)$ and $f_e(e^i_j)$ together produce the weight, or attention, for the aligned entity e^i_j, which is used to weight the document ranking score $f_r(e^i_j, d)$ produced by the entity. As the whole model is trained jointly by learning-to-rank, the query entity linking is optimized together with the entity-based ranking for better end-to-end ranking performance.

5.2.2 Experiment Methodology

The experiments conducted in this section follows the same settings as in Section 5.1.

Dataset: Our experiments use two ClueWeb Category B corpora, TREC Web Track queries and corresponding relevance judgments. ClueWeb09-B has 200 queries from TREC Web Track 2009-2012; ClueWeb12-B13 has 100 queries from TREC Web Track 2013-2014.

All our methods re-rank the top 100 candidate documents from a base retrieval model. On ClueWeb09, the base retrieval is the SDM runs from the well-tuned and widely used EQFE [31]. On ClueWeb12, not all rankings are publicly available from EQFE, so the base retrieval is Indri's default language model, with KSteming, INQUERY stopword removal, and no spam filtering.

When extracting ranking features, the document's title and body are parsed by Boilerpipe with the 'KeepEverythingExtractor'; all parameters of the retrieval models used are kept default.
Baselines: Word-based baselines are unsupervised language model (`Lm`) and `SDM`, and supervised learning-to-rank models, including `RankSVM` and `Coordinate Ascent`. They are the same as in Section 5.1.

Entity-based ranking baselines include the widely used EQFE [31] and EsdRank (Section 3.2) on ClueWeb09-B. We also compare with `LeToR-Qe-Dw` with query entities from an off-the-shelf entity linker [38] and similar entity-based ranking features (Section 5.1).

The main goal of this experiment is to show the effectiveness of joint query entity linking and entity-based ranking; other entity-based ranking systems that use manual annotations or involve document entities are not fair comparisons.

Evaluation Metrics: The TREC Web Track's official evaluation metrics, NDCG@20 and ERR@20, are used. Statistical significance is tested by the permutation test with $p < 0.05$.

Implementation Details: All supervised methods implemented by us are evaluated using the same 10-fold cross validation, done separately on ClueWeb09 or ClueWeb12 queries. In each fold, the training of `JointSem` and its variants were repeated 20 times, and the one with the best *training* loss is used in testing.

The knowledge graph used is Freebase. The surface form dictionary, including the surface forms, their candidate entities, and corresponding commonness scores are obtained from Google's FACC1 annotation on the two ClueWeb corpora. The linked probabilities of the surface forms are calculated on a recent Wikipedia dump (20170420). The word and entity embeddings are trained with the skip-gram model in Google's word2vec toolkit, with 300 dimensions. The training corpus of embeddings is the Wikipedia dump mixed with its duplicate on which the manual entity annotations are replaced by their Freebase Ids. The base retrieval score is added as a ranking feature to `JointSem`.

The training uses batch training and ndam optimization. The maximum entities allowed per surface form (N) is 5; candidate entities not in the top 5 are extremely rare or noise.

5.2.3 Evaluation

This section presents the overall evaluation results and the analysis of `JointSem`'s source of effectiveness.

5.2.3.1 Overall Performance

The overall evaluation results in Table 5.12 demonstrate that jointly modeling the linking of entities and entity-based ranking helps. `JointSem` outperforms all entity-based baselines which also use query annotations and similar ranking evidence, but treat the entity linking step as a fixed pre-processing step. The performances of entity-based systems are also correlated with the entity linking difficulties on corresponding queries. On ClueWeb09 where the entity linking systems perform better (Section 5.1), directly using TagMe's results is already helpful (`LeToR-Qe-Dw`), and the improvement of joint semantic ranking is relatively smaller ($5 - 10\%$); on ClueWeb12 where query entity linking is harder (Section 5.1), fully trusting entity linking's results sometimes even fails to outperform word-based ranking, and `JointSem`'s improvements are bigger (15%).

Factoring in the entity linking influences most of the queries. `JointSem` acts rather differently than baselines, improving about half of the queries. The statistical significances are more frequently observed on ClueWeb09 but less on ClueWeb12, although the relative improvements on the latter are higher. Part of the reason is that ClueWeb12 has fewer queries (100), which also makes the learning of the query level models (spotting and linking) less stable.

`JointSem` differs from previous entity-based ranking systems in two aspects: the soft-alignment that introduces multiple entities per spot, and the joint modeling of the spotting, linking and ranking signals to optimize end-to-end ranking performance. The rest of the experiments study the effectiveness of these two factors.

Table 5.12: Overall accuracies of JointSem. Relative performances compared with LeToR-Qe-Dw are shown as percentages. **W**in/**T**ie/**L**oss are the number of queries a method improves, does not change, or hurts, compared with LeToR-Qe-Dw on NDCG@20. Best results in each metric are marked **bold**. †, ‡, §, and ¶ indicate statistically significant improvements over Coordinate Ascent†, EQFE‡, EsdRank§, and LeToR-Qe-Dw¶, respectively.

ClueWeb09-B					
Method	**NDCG@20**		**ERR@20**		**W/T/L**
Lm	0.1757	−35.63%	0.1195	−34.48%	70/25/99
SDM	0.2496	−8.54%	0.1387	−23.96%	84/28/82
RankSVM	0.2635	−3.46%	0.1544	−15.32%	90/29/75
Coordinate Ascent	0.2681	−1.77%	0.1617	−11.32%	91/28/75
EQFE	0.2448	−10.32%	0.1419	−22.18%	76/28/90
EsdRank	0.2644	−3.14%	0.1756	−3.73%	93/25/76
LeToR-Qe-Dw	0.2729	–	0.1824	–	–/–/–
JointSem	**0.3054**†‡§¶	+11.89%	**0.1926**†‡	+5.63%	99/30/65
ClueWeb12-B13					
Method	**NDCG@20**		**ERR@20**		**W/T/L**
Lm	0.1060	−4.45%	0.0863	−7.09%	40/20/40
SDM	0.1083	−2.41%	0.0905	−2.52%	43/20/37
RankSVM	0.1205	+8.61%	0.0924	−0.45%	39/22/39
Coordinate Ascent	0.1206	+8.70%	0.0947	+1.96%	44/23/33
EQFE	n/a	–	n/a	–	–/–/–
EsdRank	n/a	–	n/a	–	–/–/–
LeToR-Qe-Dw	0.1110	–	0.0928	–	–/–/–
JointSem	**0.1314**¶	+18.46%	**0.1076**	+15.93%	54/20/26

5.2.3.2 Effectiveness of Soft Alignment

This experiment studies the soft-alignment's influence by varying the number of entities allowed per spot in JointSem. The results are shown in Figure 5.4. The y-axis marks the relative improvements compared with LeToR-Qe-Dw which uses 'hard alignment' but similar ranking features. The 'k' in JointSem-Topk refers to the number of candidate entities considered per spot, selected by their commonness scores.

Although in most cases the Top1 entity is the right choice, in general, considering more candidate entities, especially when using all top 5, improves the ranking accuracy. Without many contexts in short queries, an entity linking system tends to merely choose the most popular candidate entity. Our manual examination found that improvements are often seen on ambiguous queries whose most popularly linked candidate entities are not the right choice. For example, the query 'bobcat' refers to bobcat the company, but bobcat the animal is the more popular choice for the entity linking system. JointSem's soft-alignment avoids such over-commitment, and lets the final ranking model select the most useful one(s).

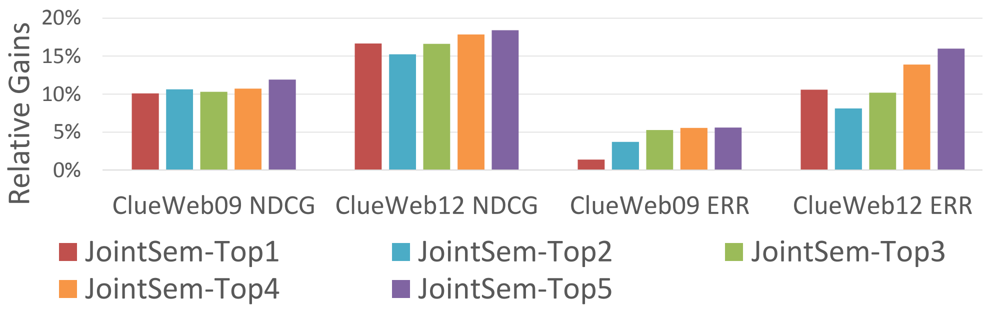


Figure 5.4: Relative improvements of `JointSem` with different numbers (TopK) of candidate entities per spot. The relative gains marked by the y-axis are compared with `LeToR-Qe-Dw`.

Table 5.13: Ablation study of `JointSem`. `JointSem-NoAtt` is the entity-based ranking without attention. `JointSem-SpotAtt` uses the spot attention with entity-based ranking. `JointSem-EntityAtt` uses the entity attention with entity-based ranking. `JointSem-All` is the full model. Relative performances and **W**in/**T**ie/**L**oss are compared with `JointSem-NoAtt`.

ClueWeb09-B					
Method	**NDCG@20**		**ERR@20**		**W/T/L**
`JointSem-NoAtt`	0.2919	–	0.1835	–	–/–/–
`JointSem-SpotAtt`	0.3005	$+2.95\%$	0.1882	$+2.61\%$	83/55/56
`JointSem-EntityAtt`	0.2999	$+2.74\%$	0.1872	$+2.06\%$	85/50/59
`JointSem-All`	**0.3054**	$+4.62\%$	**0.1926**	$+5.00\%$	88/49/57
ClueWeb12-B13					
Method	**NDCG@20**		**ERR@20**		**W/T/L**
`JointSem-NoAtt`	0.1258	–	0.1012	–	–/–/–
`JointSem-SpotAtt`	0.1247	-0.88%	0.1010	-0.27%	31/32/37
`JointSem-EntityAtt`	0.1240	-1.48%	0.1058	$+4.52\%$	36/34/30
`JointSem-All`	**0.1314**	$+4.44\%$	**0.1076**	$+6.31\%$	46/27/27

5.2.3.3 Effectiveness of Joint Modeling

This experiment studies the effectiveness of joint modeling by comparing `JointSem` and its sub-models. The results are listed in Table 5.13. `JointSem-NoAtt` uses the Top1 entity per spot as fixed and only includes the ranking part ($f_r(E, d)$). `JointSem-SpotAtt` includes the surface form attention part (f_s), but only the Top1 entities are included with uniform weights; it is similar to the recent attention-based ranking model with word-entity duet (Section 5.1), but without document entities. `JointSem-EntityAtt` includes the soft-alignment and entity weighting (f_e), but without surface form weighting. `JointSem-All` is the full model.

The ranking part alone provides better or comparable performance with baselines. Adding in the spotting or the linking part individually helps on ClueWeb09 but has mixed effects on ClueWeb12. Only `JointSem-All` provides stable 5% improvements, confirming the importance of jointly modeling the linking and the utilization of entities for document ranking.

5.2.4 JointSem Summary

This section addresses the discrepancy between entity linking and entity-based ranking systems by performing the two tasks jointly. Our method, `JointSem`, spots and links entities in the query, and then uses the linked entities to rank documents. The signals from spotting and linking are incorporated as entity importance features, and the similarities between entities' texts and the document are used as ranking features. `JointSem` uses a joint learning-to-rank model that combines all three components together, and directly optimizes them towards the end-to-end ranking performance.

Experiments on two TREC Web Track datasets demonstrated the effectiveness of `JointSem`, and the influences of the two novelties: the soft-alignment includes multiple entities per spot thus is more robust to ambiguous queries; and the joint modeling stably combines the features from spotting, linking, and ranking together. These results demonstrate that entity linking, a widely studied natural language processing task, and document ranking, a core information retrieval task, can be, and should be developed together.

5.3 Summary

This chapter first presents the *word-entity duet framework* that integrates the word-based and entity-based representations. The duet representation naturally leads to a four-way match scheme that provide a systematic way to incorporate various information from the knowledge graph. To handle the noise from automatically linked query entities, we develop an attention-based learning to rank model that utilizes the attention mechanism on query entities. The four-way matches and the hierarchical ranking model together improve the word-based learning to rank systems by large margins. The same attention-based ranking model is then generalized in Section 5.2 to include entity linking. It leads to a soft-linking approach that enables the ranking model to 'fix' some linking errors instead of 'pruning' them.

The research presented in this chapter wraps up our exploration of knowledge graphs in search within the feature-based framework. With a series of progress, we successfully developed the entity-oriented search paradigm that effectively and systematically integrates entities and their structured semantics from knowledge graphs to search engines. The next chapter goes beyond the 'bag-of-terms' representations and aims to improve text understanding with more structure and neural networks.

Chapter 6

From Representation to Understanding through Entity Salience Estimation

Understanding the meaning of text has been a long desired goal in information retrieval. In search engines, the processing of texts begins with the representations of query and documents. The representations can be bag-of-words, bag-of-entities (Chapter 4), or the combination of them (Chapter 5). After representation, the next step is to estimate the term (word or entity) importance in the text, which is also called term *salience* estimation [33, 36]. The ability to know which terms are important and central (salient) to the meaning of a text is crucial to many text processing tasks. In ad hoc search, the ranking of documents is often determined by the salience of query terms in the documents, which is typically estimated by combining frequency-based signals such as term frequency and inverse document frequency [29].

Effective as it is, frequency is not equal to salience. For example, a Wikipedia article about an entity may not repeat the entity the most times; a person's homepage may only mention her name once; a frequently mentioned term may be a stopword. In word-based retrieval, many approaches have been developed to better estimate term importance and improve retrieval models [12]. However, in entity-based representations (Sections 4.2 and 5.1), although richer semantics are conveyed by entities, entity salience estimation is a rather immature task [33, 36] and its effectiveness in search has not yet been explored.

This chapter focuses on improving text understanding and retrieval by better estimating the *entity salience* in documents[1]. We present a Kernel Entity Salience Model (`KESM`) that models the entity salience end-to-end using neural networks. Given annotated entities in a document, `KESM` represents them using Knowledge Enriched Embeddings and models the interactions of an entity with other entities and words using a Kernel Interaction Model [101]. In the entity salience task [36], the kernel scores from the interaction model are combined by `KESM` to estimate entity salience, and the whole model, including the Knowledge Enriched Embeddings and Kernel Interaction Model, is learned end-to-end using a large number of salience labels.

Our experiments on a news corpus [36] and a scientific proceedings corpus (Section 4.2)

[1] Chenyan Xiong, Zhengzhong Liu, Jamie Callan, and Tie-Yan Liu. *Towards Better Text Understanding and Retrieval through Kernel Entity Salience Modeling.* In Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2018) [104]. Zhengzhong Liu pointed me to the salience task, pre-processed the Annotated NYT data and conducted entity salience baselines.

demonstrate `KESM`'s effectiveness in the entity salience task. It outperforms previous frequency-based and feature-based models by large margins; it also requires less linguistic pre-processing than the feature-based model. Our analyses found that `KESM` has a better balance on popular (head) entities and rare (tail) entities when predicting salience, while frequency-based or feature-based methods are heavily biased towards the most popular entities—less attractive to users as they are more expected. Also, `KESM` is more reliable on documents with different lengths while frequency-based methods are not as effective on shorter documents.

`KESM` also improves ad hoc search by modeling the salience of query entities in candidate documents. Given a query-document pair and their entities, `KESM` uses its kernels to model the interactions of *query entities* with the entities and words in the document. It then merges the kernel scores to ranking features and combines these features to rank documents. In ad hoc search, `KESM` can either be trained end-to-end when sufficient ranking labels are available, or use its pre-trained distributed representations and kernels from the entity salience task to provide salience ranking features for standard learning to rank systems.

Our experiments on TREC Web Track search tasks show that `KESM`'s text understanding ability in estimating entity salience also improves search accuracy. The salience ranking features extracted by `KESM`'s pre-trained embeddings and kernels on the news corpus outperform both word-based and entity-based features in learning to rank, despite various differences in the salience and search tasks. Our case study found interesting examples showing that `KESM` distinguishes documents that are more focused on the query entities from those just mentioning the query terms. We found it encouraging that the fine-grained text understanding ability of `KESM`—the ability to model the consistency and interactions between entities and words in texts—is indeed able to improve the accuracy of ad hoc search.

In the rest of this chapter, Section 6.1 describes `KESM`, the Kernel Entity Salience Model, and its application in entity salience estimation; Section 6.2 discusses the application of `KESM` in ad hoc search; Section 6.3 concludes this chapter.

6.1 Entity Salience Modeling

This section first discusses the related work in term importance estimation. Then it provides the background information about the kernel technique which was originally developed for neural information retrieval. After that, this section presents the Kernel Entity Salience Model (`KESM`), the experimental methodology, and the evaluation results in entity salience modeling.

6.1.1 Related Work in Term Salience Estimation

Representing and understanding texts is a key challenge in information retrieval. The standard approaches in modern information retrieval represent a text by a bag-of-words and model a term's importance using frequency-based signals such as term frequency (TF), inverse document frequency (IDF), and document length [29]. The bag-of-words representation and frequency-based signals are the backbone of modern information retrieval and have been used by many unsupervised and supervised retrieval models [29, 59].

Nevertheless, bag-of-words and frequency-based statistics only provide shallow text understanding. One way to improve the text understanding is to use more meaningful language units than words in the text representations. These approaches include the first generation of search engines that were based on controlled vocabularies [29] and also the recent entity-oriented search systems which utilize knowledge graphs in search [31, 61, 80] (Chapter 4-5). In these approaches, the texts are often represented by entities, which introduces information from taxonomies and knowledge graphs to improve various components of search engines.

In both word-based and entity-based text representations, frequency signals such as TF and IDF provide good approximations for the importance or salience of terms (words or entities). However, solely relying on frequency signals limits the search engine's text understanding capability. Many approaches have been developed to improve *term importance estimation*.

In the word space, the query term weighting research focuses on modeling the importance of words or phrases in the query. For example, Bendersky et al. use a supervised model to combine the signals from Wikipedia, search log, and external collections to better estimate term importance in verbose queries [9]; Zhao and Callan predict the necessity of query terms using evidence from pseudo relevance feedback [110]; word embeddings have also been used as features in supervised query term importance prediction [111]. These are just a few examples from that broad literature. In general, these methods leverage extra signals to model how important a term is to the search intent.

Many graph-based approaches have been developed to better model the word importance in documents, for example, TextRank [12] and TW-IDF [82]. Instead of using isolated words, the graph-based approaches connect words by co-occurrence or proximity. Then graph ranking algorithms, for example, PageRank and in-degree, are used to estimate the term importance in the document. The graph ranking scores reflect the centrality and connectivity of words and better estimate the word importance [12, 82].

In the entity space, modeling the term importance is even more crucial. Unlike word-based representations, the entity-based representations are automatically constructed and inevitably include noisy entities. The noisy query entities have been a major bottleneck for entity-oriented search and often required manual cleaning [31, 37, 61]. In this dissertation, we have presented a series of approaches to model the importance of entities in a query, including the latent-space learning to rank (Section 3.2) and the attention-based hierarchical ranking network (Section 5.1). These approaches learn the importance of query entities and the ranking of documents jointly using ranking labels. The features used to describe the entity importance include IR-style features (Section 3.2) and NLP-style features from entity linking (Section 5.1).

Previous research on modeling entity importance mainly focused on query representations, while the entities in document representations are still weighted by frequencies, i.e. in the bag-of-entities model (Section 4.1) and the word-entity duet (Section 5.1). Recently, Dunietz and Grllick [36] proposed the entity salience task using the New York Times corpus [84]; they consider the entities annotated in the expert-written summary to be salient to the article, enabling them to automatically construct millions of training data. Dojchinovski et al. constructed a deeper study and found that crowdsource workers consider entity salience an intuitive task [33]. They demonstrated that the frequency of an entity is not equal to its salience in the document; a supervised model with linguistic and semantic features is able to outperform frequency, though mixed findings have been found with graph-based methods such as PageRank.

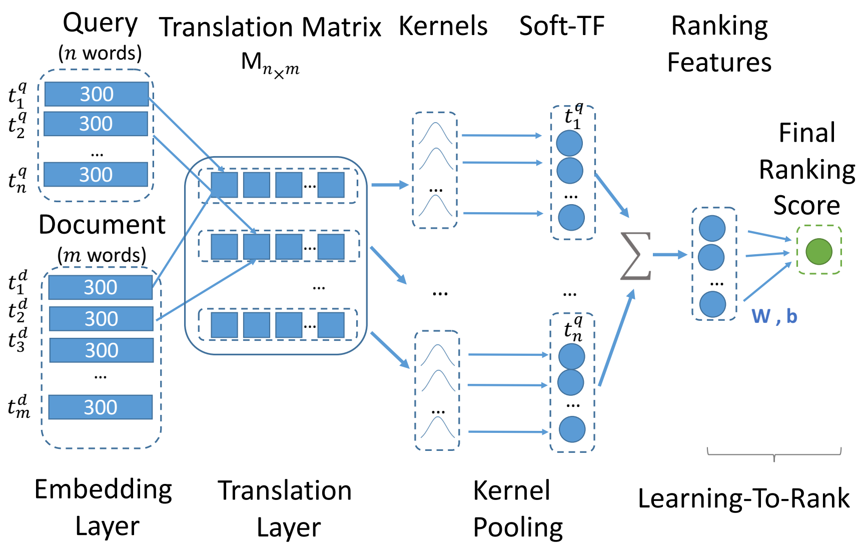


Figure 6.1: The Architecture of K−NRM. Given input query words and document words, the embedding layer maps them into distributed representations, the translation layer calculates the word-word similarities and forms the translation matrix, the kernel pooling layer generate soft-TF counts as ranking features, and the learning to rank layer combines the soft-TF to the final ranking score.

6.1.2 Background: Kernel-based Neural Ranking Model

An important component of our Kernel Entity salience Model is the kernels that effectively model the interactions between terms in the embedding space. The kernel technique was originally developed in our kernel-based neural ranking model (K−NRM), which uses kernels to capture the soft match patterns between query and document words. To better understand KESM, this section first provide the necessary background about K−NRM, including its architecture, learning, and effectiveness in neural information retrieval.[2]

6.1.2.1 K-NRM Architecture

K−NRM was first developed for ad hoc search. It takes a query-document pair (q, d) and generates a ranking score $f(q, d)$ using query words $q = \{t_1^q, ...t_i^q..., t_n^q\}$ and document words $d = \{t_1^d, ...t_j^d..., t_m^d\}$. As shown in Figure 6.1, K−NRM includes three components: translation model, kernel-pooling, and learning to rank.

Translation Model: K−NRM first uses an `embedding layer` to map each word t to an

[2]Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. *End-to-End Neural Adhoc Ranking with Kernel Pooling*. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2017) [101]. Zhuyun and I contributed equally to this work. Zhiyuan and Russell helped with the paper writing.

L-dimension embedding $\vec{v}_t$:

$$t \Rightarrow \vec{v}_t.$$

Then a `translation layer` constructs a translation matrix M. Each element in M is the embedding similarity between a query word and a document word:

$$M_{ij} = \cos(\vec{v}_{t_i^q}, \vec{v}_{t_j^d}).$$

The translation model in `K-NRM` uses word embeddings to recover the word similarities instead of trying to learn one for each word pair. Doing so requires much fewer parameters to learn. For a vocabulary of size $|V|$ and the embedding dimension L, `K-NRM`'s translation model includes $|V| \times L$ embedding parameters, much fewer than learning all pairwise similarities ($|V|^2$).

Kernel-Pooling: `K-NRM` then uses kernels to convert word-word interactions in the translation matrix M to query-document ranking features $\rho(M)$:

$$\rho(M) = \sum_{i=1}^{n} \log \Phi(M_i)$$
$$\Phi(M_i) = \{\phi_1(M_i), ..., \phi_K(M_i)\}$$

$\Phi(M_i)$ applies K kernels to the i-th query word's row of the translation matrix, summarizing (pooling) it into a K-dimensional feature vector. The log-sum of each query word's feature vector forms the query-document ranking feature vector ρ.

The effect of Φ depends on the kernel used. This work uses the RBF kernel[3]:

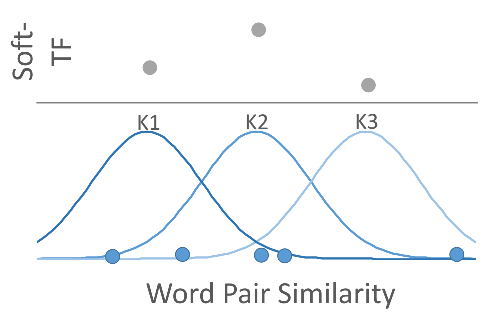
$$\phi_k(M_i) = \sum_j \exp(-\frac{(M_{ij} - \mu_k)^2}{2\sigma_k^2}).$$

As illustrated in Figure 6.2a, the RBF kernel ϕ_k calculates how word pair similarities are distributed around it: the more word pairs with similarities closer to its mean μ_k, the higher its value. Kernel pooling with RBF kernels is a generalization of existing pooling techniques. As $\sigma \to \infty$, the kernel pooling function devolves to the mean pooling. $\mu = 1$ and $\sigma \to 0$ results in a kernel that only responds to exact matches, equivalent to the TF value from sparse models. Otherwise, the kernel functions as 'soft-TF'. μ defines the similarity level that 'soft-TF' focuses on; for example, a kernel with $\mu = 0.5$ calculates the number of document words whose similarities to the query word are close to 0.5. σ defines the kernel width, or the range of its 'soft-TF' count.

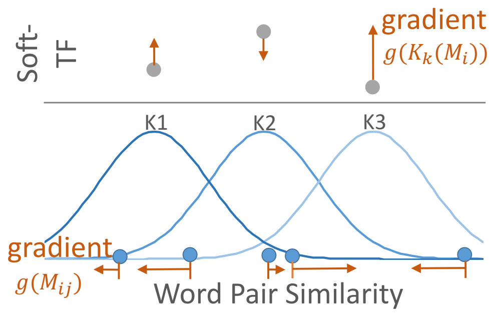
Learning to Rank: The ranking features $\rho(M)$ are combined by a ranking layer to produce the final ranking score:

$$f(q, d) = \tanh(w^T \rho(M) + b).$$

[3]The RBF kernel is one of the most popular choices. Other kernels with similar density estimation effects can also be used, as long as they are differentiable. For example, polynomial kernel can be used, but histograms [42] cannot as they are not differentiable.



(a) Ranking



(b) Learning

Figure 6.2: Effect of Kernels in Ranking and Learning. (a) illustrates how kernels convert the five word pair similarities into three kernel scores in the ranking process. (b) shows how the kernels use gradients to adjust the word pair similarities during training.

w and b are the ranking parameters to learn. tanh() is the activation function. It controls the range of ranking score to facilitate the learning process. It is rank-equivalent to a typical linear learning to rank model.

Putting every together, `K-NRM` is defined as:

$$f(q,d) = \tanh(w^T \rho(M) + b) \qquad \text{Learning to Rank} \qquad (6.1)$$

$$\rho(M) = \sum_{i=1}^{n} \log \Phi(M_i) \qquad \text{Soft-TF Features} \qquad (6.2)$$

$$\Phi(M_i) = \{\phi_1(M_i), ..., \phi_K(M_i)\} \qquad \text{Kernel Pooling} \qquad (6.3)$$

$$\phi_k(M_i) = \sum_j \exp(-\frac{(M_{ij} - \mu_k)^2}{2\sigma_k^2}) \qquad \text{RBF Kernel} \qquad (6.4)$$

$$M_{ij} = \cos(\vec{v}_{t_i^q}, \vec{v}_{t_j^d}) \qquad \text{Translation Matrix} \qquad (6.5)$$

$$t \Rightarrow \vec{v}_t. \qquad \text{Word Embedding} \qquad (6.6)$$

Eq. 6.5-6.6 embed query words and document words, and calculate the translation matrix. The kernels (Eq. 6.4) count the soft matches between query and document's word pairs at multiple levels, and generate K soft-TF ranking features (Eq. 6.2-6.3). Eq. 6.1 is the learning to rank model. The ranking of `K-NRM` requires no manual features. The only input used is the query and document words. The kernels extract soft-TF ranking features from word-word interactions automatically.

6.1.2.2 K-NRM Learning

The **training** of `K-NRM` uses the pairwise learning to rank loss:

$$l(w, b, \mathcal{V}) = \sum_q \sum_{d^+, d^- \in D_q^{+,-}} max(0, 1 - f(q, d^+) + f(q, d^-)). \qquad (6.7)$$

$D_q^{+,-}$ are the pairwise preferences from the ground truth: d^+ ranks higher than d^-. The parameters to learn include the ranking parameters w, b, and the word embeddings $\mathcal{V}$.

The parameters are optimized using back propagation through the neural network. Starting from the ranking loss, the gradients are first propagated to the learning-to-rank part (Eq. 6.1) and update the ranking parameters (w, b), the kernels pass the gradients to the word similarities (Eq. 6.2-6.4), and then to the embeddings (Eq. 6.5).

Back propagations through the kernels: The embeddings contain millions of parameters $\mathcal{V}$ and are the main capacity of the model. The learning of the embeddings is guided by the kernels.

The back propagation first applies gradients from the loss function (Eq. 6.7) to the ranking score $f(q, d)$, to increase it (for d^+) or decrease it (for d^-); the gradients are propagated through Eq. 6.18 to the feature vector $\rho(M)$, and then through Eq. 6.2 to the the kernel scores $\Phi(M_i)$. The resulted $g(\Phi(M_i))$ is a K dimensional vector:

$$g(\Phi(M_i)) = \{g(\phi_1(M_i)), ..., g(\phi_K(M_i)\}.$$

Its each dimension $g(\phi_k(M_i))$ is jointly defined by the ranking score's gradients and the ranking parameters. It adjusts the corresponding kernel's score up or down to better separate the relevant document (d^+) from the irrelevant one (d^-).

The kernels spread the gradient to word similarities in the translation matrix M_{ij}, through Eq. 6.4:

$$g(M_{ij}) = \sum_{k=1}^{K} \frac{g(\phi_k(M_i)) \times \sigma_k^2}{(\mu_k - M_{ij}) \exp(\frac{(M_{ij}-\mu_k)^2}{-2\sigma_k^2})}. \tag{6.8}$$

The kernel-guided embedding learning process is illustrated in Figure 6.2b. A kernel pulls the word similarities closer to its μ to increase its soft-TF count, or pushes the word pairs away to reduce it, based on the gradients received in the back-propagation. The *strength* of the force also depends on the the kernel's width σ_k and the word pair's distance to μ_k: approximately, the wider the kernel is (bigger σ_k), and the closer the word pair's similarity to μ_k, the stronger the force is (Eq. 6.8). The gradient a word pair's similarity received, $g(M_{ij})$, is the combination of the forces from all K kernels.

The word embedding model receives $g(M_{ij})$ and updates the embeddings accordingly. Intuitively, the learned word embeddings are aligned to form multi-level soft-TF patterns that can separate the relevant documents from the irrelevant ones in training, and the learned embedding parameters $\mathcal{V}$ memorize this information. When testing, `K-NRM` extracts soft-TF features from the learned word embeddings using the kernels and produces the final ranking score using the ranking layer.

6.1.2.3 K-NRM Effectiveness in Ranking

`K-NRM` is effective for various search scenarios. Using commercial search logs where large scale user feedback signals are available for end-to-end training, `K-NRM` outperforms feature-based learning to rank models and other previous neural models by large margins, as was shown by our experiments on a Chinese search log dataset from Sogou.com [101] and a English search log dataset from Bing.com [30]. Its effectiveness can also be transferred to public search benchmarks (TREC Web Track) by adapting the model trained from the Bing search log to TREC queries [30].

The effectiveness of K-NRM comes from the kernels; they provide effective multi-level soft match signals by locating terms in the embedding space to capture the relevance match pattern reflected from data [101]. Building from K-NRM's success, the rest of this chapter develops an entity salience estimation model, which uses the kernels to capture the interaction patterns between entities and words in text for salience estimation (Section 6.1.3-6.1.5). Then the learned entity salience model is also adapted to improve the ranking accuracy using our entity-oriented search framework (Section 6.2).

6.1.3 Kernel Entity Salience Model

We now present our Kernel Entity Salience Model (KESM), which leverages the knowledge graph embeddings (Section 4.2) and the kernel technique (Section 6.1.2) to estimate entity salience. As shown in Figure 6.3, the two techniques form KESM's two main components: the *Knowledge Enriched Embedding* (Figure 6.3a) and the *Kernel Interaction Model* (Figure 6.3b).

Knowledge Enriched Embedding (KEE) encodes each entity e into its distributed representation $\vec{v}_e$. It is achieved by first using an embedding layer that maps the entity to an embedding:

$$e \xrightarrow{V} \vec{e}. \qquad \text{Entity Embedding}$$

V is the parameters of the embedding layer to be learned from data.

An advantage of entities is that they are associated with external semantics in the knowledge graph, for example, synonyms, descriptions, types, and relations. Those external semantics provide prior knowledge about entities which can help improve the generalization ability of their distributed representations [63]. Thus, instead of only using the "data-driven" embedding $\vec{e}$, KEE also enriches the entity representation with its description in the knowledge graph.

Specifically, given the description $\mathbb{D}$ of the entity e, KEE uses a Convolutional Neural Network (CNN) to compose the words in $\mathbb{D}$: $\{w_1, ..., w_p, ..., w_l\}$, into one embedding:

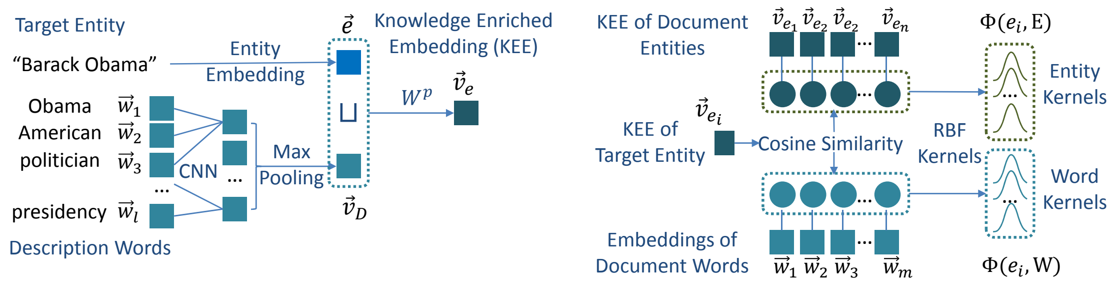
$$\begin{aligned} &w_p \xrightarrow{V} \vec{w}_p, && \text{Word Embedding} \\ &C_p = W^c \cdot \vec{w}_{p:p+h}, && \text{CNN Filter} \\ &\vec{v}_{\mathbb{D}} = \max(C_1, ..., C_p, ..., C_{l-h}). && \text{Description Embedding} \end{aligned}$$

It embeds the words into $\vec{w}$ using the embedding layer, composes the word embeddings using CNN filters, and generates the description embeddings $\vec{v}_{\mathbb{D}}$ using max-pooling. W^c and h are the weights and length of the CNN.

$\vec{v}_D$ is then combined with the entity embedding $\vec{e}$ by projection:

$$\vec{v}_e = W^p \cdot (\vec{e} \sqcup \vec{v}_{\mathbb{D}}). \qquad \text{KEE Embedding}$$

$\sqcup$ is the concatenation operator and W^p is the projection weights. $\vec{v}_e$ is the KEE vector for e. It incorporates the external information from the knowledge graph and is to be learned as part of KESM.



(a) Knowledge Enriched Embedding (KEE) (b) Kernel Interaction Model (KIM)

Figure 6.3: KESM Architecture. (a): Entities are represented using embeddings enriched by their descriptions. (b): The salience of an entity in a document is estimated by kernels that model its interactions with entities and words in the document. Squares are continuous vectors (embeddings) and circles are scalars (cosine similarities).

Kernel Interaction Model (KIM) models the interactions of a target entity, e.g. an entity in the document (for the salience task) or in the query (for entity-oriented search), with entities and words in the document using their distributed representations.

Given a document d, its annotated entities $\mathbb{E} = \{e_1, ...e_i..., e_n\}$, and its words $\mathbb{W} = \{w_1, ...w_j..., w_m\}$, KIM models the interactions of a target entity e_i with $\mathbb{E}$ and $\mathbb{W}$ using kernels:

$$KIM(e_i, d) = \Phi(e_i, \mathbb{E}) \sqcup \Phi(e_i, \mathbb{W}). \tag{6.9}$$

The entity kernels $\Phi(e_i, \mathbb{E})$ model the interaction between e_i and document entities $\mathbb{E}$:

$$\Phi(e_i, \mathbb{E}) = \{\phi_1(e_i, \mathbb{E}), ...\phi_k(e_i, \mathbb{E})..., \phi_K(e_i, \mathbb{E})\}, \tag{6.10}$$

$$\phi_k(e_i, \mathbb{E}) = \sum_{e_j \in \mathbb{E}} \exp\left(-\frac{\left(cos(\vec{v}_{e_i}, \vec{v}_{e_j}) - \mu_k\right)^2}{2\sigma_k^2}\right). \tag{6.11}$$

$\vec{v}_{e_i}$ and $\vec{v}_{e_j}$ are the KEE embeddings of e_i and e_j. $\phi_k(e_i, \mathbb{E})$ is the k-th RBF kernel with mean μ_k and variance σ_k^2. If $(\mu_k = 1, \sigma_k \to \infty)$, ϕ_k counts the entity frequency. Otherwise, it models the interactions between the target entity e_i and other entities in the KEE representation space. The role of kernels in salience estimation is to count the number of entities whose similarities with e_i are in its region (μ_k, σ_k^2); it can also be viewed as collecting votes from other entities in a certain neighborhood (kernel region) of the current entity.

The word kernels $\Phi(e_i, \mathbb{W})$ model the interactions between e_i and document words $\mathbb{W}$:

$$\Phi(e_i, \mathbb{W}) = \{\phi_1(e_i, \mathbb{W}), ...\phi_k(e_i, \mathbb{W})..., \phi_K(e_i, \mathbb{W})\}, \tag{6.12}$$

$$\phi_k(e_i, \mathbb{W}) = \sum_{w_j \in \mathbb{W}} \exp\left(-\frac{\left(cos(\vec{v}_{e_i}, \vec{w}_j) - \mu_k\right)^2}{2\sigma_k^2}\right). \tag{6.13}$$

$\vec{w}_j$ is the word embedding of w_j, mapped by the same embedding parameters (V). The word kernel $\phi_k(e_i, \mathbb{W})$ models the interactions between e_i and document words, gathering 'votes' from words to e_i in its kernel region. The distributed representations make it convenient to model the interactions between a word and a entity as they are mapped to the same continuous embedding space.

For each entity e_i, `KEE` encodes it to $\vec{v}_{e_i}$ and `KIM` models its interactions with entities and words in the document. The kernel scores $KIM(e_i, d)$ include signals from three sources: the description of the entity in the knowledge graph, its interaction with the document entities, and its interactions with the document words.

Estimating Entity Salience. Combining the `KIM` kernel scores yields the salience score of the entity:

$$f(e_i, d) = W^s \cdot KIM(e_i, d) + b^s. \tag{6.14}$$

$f(e_i, d)$ is the salience score of e_i in d. W^s and b^s are parameters to learn.

Learning: The entity salience training data are labels on document-entity pairs that indicate whether the entity is salient to the document. For example, the salience label of entity e_i to document d is:

$$y(e_i, d) = \begin{cases} +1, & \text{if } e_i \text{ is a salient entity in } d, \\ -1, & \text{otherwise.} \end{cases}$$

We use pairwise learning to rank [59] to train `KESM`:

$$\sum_{e^+, e^- \in d} \max(0, 1 - f(e^+, d) + f(e^-, d)), \tag{6.15}$$
$$\text{w.r.t. } y(e^+, d) = +1 \ \& \ y(e^-, d) = -1.$$

The loss function enforces `KESM` to rank the salient entities e^+ ahead of the non-salient ones e^- within the same document.

The learning of `KESM` is end-to-end using back-propagation. During training, the gradients from the labels are first propagated to the Kernel Interaction Model (`KIM`) and then the Knowledge Enriched Embedding (`KEE`). `KESM` updates the kernel weights; `KIM` converts the gradients from kernels to 'expectations' on the distributed representations—how the entities and words should be located in the space to better reflect salience; `KEE` updates its embeddings and parameters according to these 'expectations'. The knowledge learned from the training labels is encoded and stored in the model parameters, mainly the embeddings (Section 6.1.2).

6.1.4 Experimental Methodology

This section presents the experimental methodology for the entity salience task. It mainly follows the setup by Dunietz and Gillick [36] with some revisions to facilitate the applications in search. An additional dataset is also introduced.

Datasets used include *New York Times* and *Semantic Scholar*.

The *New York Times* corpus has been used in previous work [36]. It includes more than half million news articles and their expert-written summaries [84]. Among all entities annotated to a

Table 6.1: Statistics of entity salience estimation datasets. New York Times are news articles and salient entities are those in the expert-written news summaries. Semantic Scholar are paper abstracts and salient entities are those in the titles.

	New York Times			**Semantic Scholar**		
	Train	Dev	Test	Train	Dev	Test
# of Documents	526k	64k	64k	800k	100k	100k
Entities Per Doc	198	197	198	66	66	66
Salience Per Doc	27.8	27.8	28.2	7.3	7.3	7.3
Unique Word	609k	278k	281k	921k	300k	301k
Unique Entity	622k	319k	317k	331k	162k	162k

news article, those that also appear in the summary of the article are considered as salient entities and others are not [36].

The *Semantic Scholar* corpus is a one million papers random sampled from the index of SemanticScholar.org, the academic search engine we experimented with in Section 4.2. The full texts of the papers are not publicly available. We treat the entities annotated in the abstract as the candidate entities of a paper and those also annotated in the title as salient.

The entity annotations on both corpora are Freebase entities linked by TagMe [38], a setup widely used in entity-oriented search. Instead of only keeping named entities in several categories [36], we keep all annotated entities in all categories to ensure coverage, which has been found very crucial in providing effective text representations (Section 4.1).

The statistics of the two corpora are listed in Table 6.1. The Semantic Scholar corpus has shorter documents (paper abstracts) and a smaller entity vocabulary because its papers are mostly in the computer science and medical science domains.

Baselines: Three baselines from previous research are compared: `Frequency`, `PageRank`, and `LeToR`.

`Frequency` [36] estimates the salience of an entity by its frequency. It is a straightforward but effective baseline in many related tasks. IDF is not as effective in entity-based text representations [80] (Chapter 4), so we used only frequency counts.

`PageRank` [36] estimates the salience score of an entity using its PageRank score. It is also a widely used baseline in graph-based text representations [12]. We conduct a supervised PageRank on a fully connected graph whose nodes are the entities in the document and edges are the embedding similarities of corresponding entity pairs. The entity embeddings are configured and learned using the same pairwise loss as used in `KESM`.

Similar to previous work [36], we found `PageRank` not very effective in the entity salience task. It is hard for the model to learn anything; many times the training loss did not drop at all. The best setup we were able to find and presented in the evaluation results is the one-step random walk linearly combined with `Frequency`. It is essentially an embedding-based translation model [101].

`LeToR` [36] is a feature-based learning to rank (entity) model. It is trained using the same pairwise loss with `KESM`, which we found more effective than the pointwise loss used in prior research [36].

Table 6.2: Entity salience features used by the LeToR baseline [36]. The features are extracted via various natural language processing techniques.

Name	Description
Frequency	The frequency of the entity being annotated to the document
First Location	The first location of the entity annotation
Head Word Count	The frequency of the entity's first head word in dependency parsing
Is Named Entity	Whether the entity is recognized by named entity recognition
Coreference Count	The frequency of the entity mentions after entity coreference resolution
Embedding Vote	Cosine similarities with other entities in skip-gram embedding

We tried our best to reproduce the features used by Dunietz and Gillick [36]. As listed in Table 6.2, the features are extracted by various linguistic and semantic techniques including entity linking, dependency parsing, named entity recognition, and entity coreference resolution. The entity embeddings are trained on the same corpus using Google's Word2vec toolkit [73]. Entity linking is done by TagMe; all entities (named entities or general entities) are kept (Section 4). Other linguistic and semantic preprocessing use the Stanford CoreNLP toolkit [67].

Compared to Dunietz and Gillick [36], we do not include the headline feature because it uses information from the expert-written summary and does not improve the performance much anyway; we also replace the head-lex feature with Embedding Vote which has similar effectiveness but is more efficient.

Evaluation Metrics: We use the ranking-focused evaluation metrics: Precision@$\{1, 5\}$ and Recall@$\{1, 5\}$. These ranking evaluation metrics avoid the requirement to select cutoff thresholds in the classification evaluation metrics [36], which may vary across documents and are tricky to choose properly. Statistical significances are tested by permutation (randomization) test with $p < 0.05$.

Implementation Details: The hyper-parameters of `KESM` are configured following popular choices or previous research. The dimension of entity embeddings, word embeddings, and CNN filters are all set to 128. The kernel pooling layers use one exact match kernel ($\mu = 1, \sigma = 1e - 3$) and ten soft match kernels with μ equally splitting the cosine similarity range $[-1, 1]$, i.e. $\mu \in \{-0.9, -0.7, ..., 0.9\}$ and $\sigma = 0.1$ [101]. The length of the CNN used to encode entity description is set to 3 which is tri-gram. The entity descriptions are fetched from Freebase. The first 20 words of the description are used which defines and explains the entity. The words or entities that appear less than 2 times in the training corpus are replaced with "Unk_word" or "Unk_entity".

The parameters include the embeddings V, the CNN weights W^c, the projection weights W^p, and the kernel weights W^s, b^s. They are learned end-to-end using Adam optimizer, size 64 mini-batching, and early-stopping on the development split. V is initialized by the word2vec of words and entities jointly trained on the training corpora, which takes several hours (Section 5.1). With our PyTorch implementation, `KESM` usually only needs one pass on the training data and converges within several hours on a typical GPU. In comparison, `LeToR` takes days to extract its features since parsing and coreference are costly.

6.1.5 Evaluation Results

This section first presents the overall evaluation results for the entity salience task. Then it analyzes the advantages of modeling salience over counting frequency.

6.1.5.1 Entity Salience Performance

Table 6.3 shows the experimental results for the entity salience task. `Frequency` provides reasonable estimates of entity salience. The most frequent entity is often salient to the document; the Precision@1 is rather high, especially for the New York Times. `PageRank` barely improves `Frequency`, although its embeddings are trained by the salience labels. `LeToR`, on the other hand, significantly improves both Precision and Recall of `Frequency` [36], which is expected as it has much richer features from various sources.

`KESM` outperforms all baselines significantly. Its improvements over `LeToR` are more than 10% on both datasets with only one exception: Precision@1 on New York Times. The improvements are also robust: About twice as many documents are improved (Win) than hurt (Loss).

We also conducted ablation studies on the individual source of evidence in `KESM`. Those marked with (E) include only the entity kernels; those with (W) also include word kernels; those with (K) enrich the entity embeddings with description embeddings. All variants include the entity kernels (E), which is the essential evidence.

`KESM` performs better than all of its variants, showing that all three sources contributed. Individually, `KESM (E)` outperforms all baselines. Compared to `PageRank`, the only difference is that `KESM (E)` uses kernels to model the interactions which are much more powerful than the raw embedding similarities used in `PageRank` [101]. `KESM (EW)` always significantly outperforms `KESM (E)`. The interaction between an entity and document words conveys useful information, the distributed representations make them easily comparable, and the kernels model the word-entity interactions effectively. Knowledge enrichment (K) provides mixed results. A possible reason is that the training data is large enough to train good entity embeddings. Nevertheless, we find that adding the external knowledge makes the training converge faster and more stable.

6.1.5.2 Modeling Salience VS. Counting Frequency

This experiment provides two analyses that study the advantage of `KESM` over counting frequency.

Ability to Model Tail Entities. The first advantage of `KESM` is that it is able to model the salience on less frequent (tail) entities. To demonstrate this effect, Figure 6.4 illustrates the distribution of predicted-salient entities in different frequency ranges. The entities with top k highest predicted scores are predicted-salient. k is the number of salient entities in the ground truth. The frequency percentiles of these entities in the corpus are labeled on the x-axes. $< 0.1\%$ refers to the top 0.1% most frequent entities. Their distributions are marked by the y-axes. The distribution of `Ground Truth` is also displayed.

In both datasets, the frequency-based methods are highly biased towards the head entities: The top 0.1% most popular entities receive almost two-times more salience predictions from

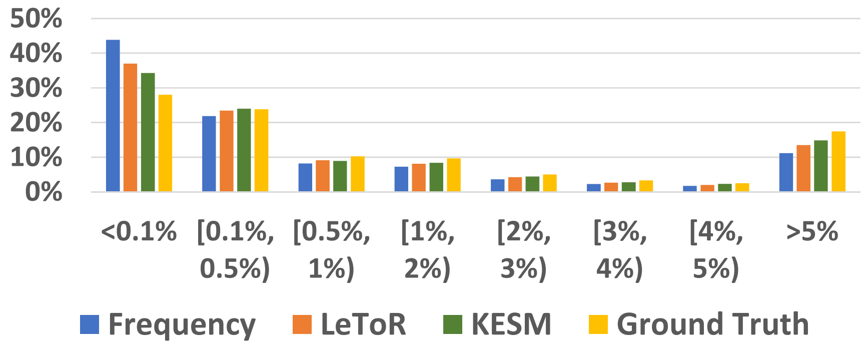
Table 6.3: Entity salience performances on New York Times and Semantic Scholar. (E), (W), and (K) mark the resources used by `KESM`: Entity kernels, Word kernels, and Knowledge enrichment. `KESM` is the full model. Relative performances over `LeToR` are shown in the percentages. W/T/L are the number of documents a method improves, does not change, and hurts, compared to `LeToR`. $\dagger$, $\ddagger$, $\S$, and $\P$ mark the statistically significant improvements over `Frequency`†, `PageRank`‡, `LeToR`§, and `KESM (E)`¶.

New York Times

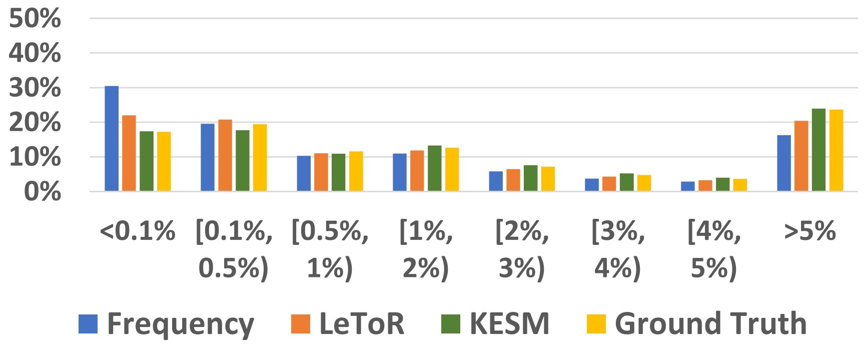
Method	**Precision@1**		**Precision@5**		**Recall@1**		**Recall@5**		**W/T/L**
`Frequency`	0.5840	−8.5%	0.4065	−11.8%	0.0781	−11.9%	0.2436	−14.4%	6k/39k/19k
`PageRank`	$0.5845^{\dagger}$	−8.5%	$0.4069^{\dagger}$	−11.7%	$0.0782^{\dagger}$	−11.8%	$0.2440^{\dagger}$	−14.3%	6k/39k/19k
`LeToR`	0.6385	–	0.4610	–	0.0886	–	0.2848	–	–/–/–
`KESM (E)`	$0.6470^{\dagger\ddagger\S}$	+1.3%	$0.4782^{\dagger\ddagger\S}$	+3.7%	$0.0922^{\dagger\ddagger\S}$	+4.0%	$0.3049^{\dagger\ddagger\S}$	+7.1%	20k/28k/16k
`KESM (EK)`	$0.6528^{\dagger\ddagger\S\P}$	+2.2%	$0.4769^{\dagger\ddagger\S}$	+3.5%	$0.0920^{\dagger\ddagger\S}$	+3.8%	$0.3026^{\dagger\ddagger\S}$	+6.3%	19k/30k/15k
`KESM (EW)`	$0.6767^{\dagger\ddagger\S\P}$	+6.0%	$0.5018^{\dagger\ddagger\S\P}$	+8.9%	$0.0989^{\dagger\ddagger\S\P}$	+11.6%	$0.3277^{\dagger\ddagger\S\P}$	+15.1%	23k/26k/14k
`KESM`	$\mathbf{0.6866}^{\dagger\ddagger\S\P}$	+7.5%	$\mathbf{0.5080}^{\dagger\ddagger\S\P}$	+10.2%	$\mathbf{0.1010}^{\dagger\ddagger\S\P}$	+13.9%	$\mathbf{0.3335}^{\dagger\ddagger\S\P}$	+17.1%	23k/27k/13k

Semantic Scholar

Method	**Precision@1**		**Precision@5**		**Recall@1**		**Recall@5**		**W/T/L**
`Frequency`	0.3944	−10.0%	0.2560	−11.4%	0.1140	−12.2%	0.3462	−13.7%	11k/65k/24k
`PageRank`	$0.3946^{\dagger}$	−9.9%	$0.2561^{\dagger}$	−11.3%	$0.1141^{\dagger}$	−12.1%	$0.3466^{\dagger}$	−13.6%	11k/64k/24k
`LeToR`	0.4382	–	0.2889	–	0.1299	–	0.4010	–	–/–/–
`KESM (E)`	$0.4793^{\dagger\ddagger\S}$	+9.4%	$0.3192^{\dagger\ddagger\S}$	+10.5%	$0.1432^{\dagger\ddagger\S}$	+10.3%	$0.4462^{\dagger\ddagger\S}$	+11.3%	28k/56k/16k
`KESM (EK)`	$0.4901^{\dagger\ddagger\S\P}$	+11.9%	$0.3161^{\dagger\ddagger\S}$	+9.4%	$0.1492^{\dagger\ddagger\S\P}$	+14.9%	$0.4449^{\dagger\ddagger\S}$	+11.0%	28k/54k/18k
`KESM (EW)`	$0.5097^{\dagger\ddagger\S\P}$	+16.3%	$0.3311^{\dagger\ddagger\S\P}$	+14.6%	$0.1555^{\dagger\ddagger\S\P}$	+19.8%	$0.4671^{\dagger\ddagger\S\P}$	+16.5%	33k/50k/17k
`KESM`	$\mathbf{0.5169}^{\dagger\ddagger\S\P}$	+18.0%	$\mathbf{0.3336}^{\dagger\ddagger\S\P}$	+15.5%	$\mathbf{0.1585}^{\dagger\ddagger\S\P}$	+22.1%	$\mathbf{0.4713}^{\dagger\ddagger\S\P}$	+17.5%	32k/52k/16k



(a) New York Times



(b) Semantic Scholar

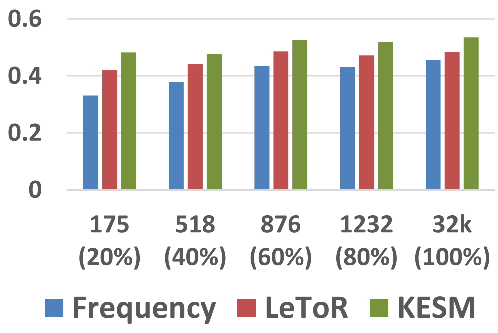
Figure 6.4: The distribution of salient entities predicted by different models. The entities are binned by their frequencies in testing data. The bins are ordered from most frequent (Top 0.1%) to less frequent (right). The x-axes mark the percentile range of each group. The y-axes are the fraction of salient entities in each bin. The histograms are ordered the same as the legends.

`Frequency` than in ground truth. This is an intrinsic bias of frequency-based methods which not only limits their effectiveness but also attractiveness—less unexpected entities are selected.

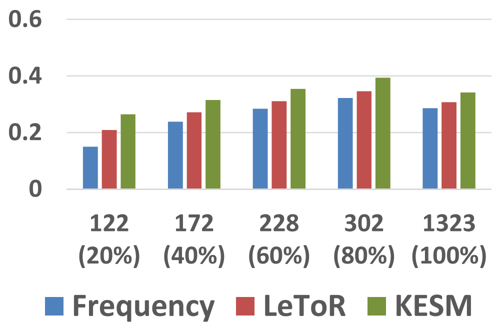
In comparison, the distributions of `KESM` are much closer to the ground truth. `KESM` does a better job in modeling tail entities because it estimates salience not only by frequency but also by modeling the *interactions* between entities and words. A tail entity can be estimated salient if many other entities and words in the document are closely related to it. For example, there are many entities and words describing various aspects of an entity in its Wikipedia page; the entities and words on a personal homepage are probably related to the person; these entities and words can 'vote up' the title entity or the person because they are strongly connected to it/her. The ability to model such interactions with distributed representations and kernels is the main source of `KESM`'s stronger text understanding capability.

Reliable on Short Documents. The second advantage of `KESM` is its reliability on short texts. To demonstrate it, we analyzed the performances of models on documents with varying lengths. Figure 6.5 groups the testing documents into five bins by their lengths (number of words), ordered from short (left) to long (right). Their upper bounds and percentiles are marked on the x-axes. The Precision@5 of corresponding methods are marked on the y-axes.

Both `Frequency` and `LeToR` (whose features are also mostly frequency-based) are less



(a) New York Times



(b) Semantic Scholar

Figure 6.5: Performances on documents with varying lengths (number of words). The x-axes are the maximum length of the documents and the percentile of each group. The y-axes mark the performances on Precision@5. The histograms are ordered the same as the legends.

reliable on shorter documents. The advantages of KESM are more significant when documents are shorter, while even in the longest bins where documents have thousands of words, KESM still outperforms Frequency and LeToR. Solely counting frequency is not sufficient to understand documents. The interactions between words and entities provide richer evidence and help KESM perform more reliably on shorter documents.

6.2 Ranking with Entity Salience

This section presents the application of KESM in ad hoc search. It first show how to adapt KESM for the ad hoc search task. Then it describes the experimental methodologies and evaluation results in improving search accuracy.

6.2.1 Ranking Approach

Ranking: Knowing which entities are salient in a document indicates a stronger text understanding ability [33, 36]. The improved text understanding should also improve search accuracy: the salience of query entities in a document reflects how focused the document is on the query, which is a strong indicator of relevancy. For example, a web page that exclusively discusses Barack Obama's family is more relevant to the query "Obama Family Tree" than those that just mention his family members.

The ranking process of KESM following this intuition is illustrated in Figure 6.6. It first calculates the kernel scores of the query entities in the document using KEE and KIM. Then it merges the kernel scores from multiple query entities to ranking features and uses a ranking model to combine these features.

Specifically, given query q, query entities $\mathbb{E}^q$, candidate document d, document entities $\mathbb{E}^d$,

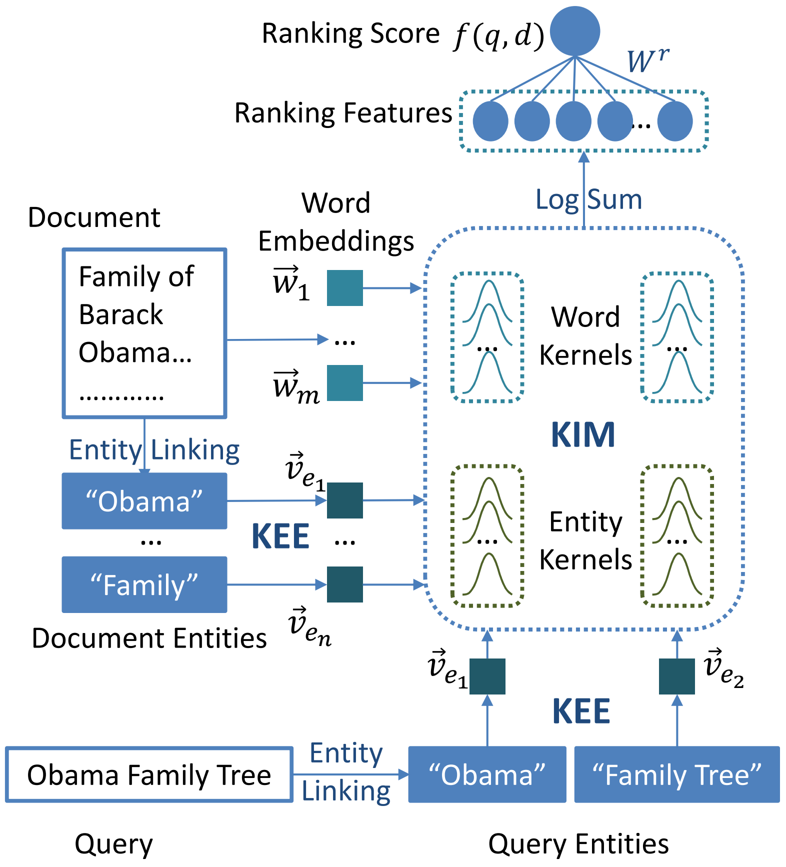


Figure 6.6: Ranking with KESM. KEE embeds the entities. KIM calculates the kernel scores of query entities vs. document entities and words. The kernel scores are combined to ranking features and then to the ranking score.

and document words $\mathbb{W}^d$, the ranking score is calculated as:

$$f(q,d) = W^r \cdot \Psi(q,d), \tag{6.16}$$

$$\Psi(q,d) = \sum_{e_i \in \mathbb{E}^q} \log \left(\frac{KIM(e_i, d)}{|\mathbb{E}^d|} \right), \tag{6.17}$$

$KIM(e_i, d)$ are the kernels scores of the query entity e_i in document d, calculated by the KIM and KEE modules described in last section. $|\mathbb{E}^d|$ is the number of entities in d. W^r is the ranking parameters and $\Psi(q,d)$ are the salience ranking features.

To apply KESM in ad hoc search, several adaptations have been made compared to Section 4.2. First, Equation (6.17) normalizes the kernel scores by the number of entities in the document ($|\mathbb{E}^d|$), making them more comparable across *different* documents. In the entity salience task, this is not required because the goal is to distinguish salient entities from non-salient ones in the *same* document. Second, there can be multiple entities in the query and their kernel scores need to be combined to model query-document relevance. We use log-sum for the combination, following language modeling approaches [29].

Learning: In the search task, `KESM` is trained using standard pairwise learning to rank and relevance labels:

$$\sum_{d^+ \in D^+, d^- \in D^-} \max(0, 1 - f(q, d^+) + f(q, d^-)). \tag{6.18}$$

D^+ and D^- are the relevant and irrelevant documents. $f(q, d^+)$ and $f(q, d^-)$ are the ranking scores calculated by Equation (6.16).

There are two ways to train `KESM` for ad hoc search. First, when sufficient ranking labels are available, for example, in commercial search engines, the whole `KESM` model can be learned end-to-end by back-propagation from Equation (6.18). On the other hand, when ranking labels are not available for end-to-end learning such as in TREC benchmarks, the `KEE` and `KIM` can be first trained using the labels from the entity salience task, and only the ranking parameters W^r need to be learned from relevance labels. As a result, the knowledge learned by `KIM` and `KEE` from the salience labels is conveyed to ad hoc search through the ranking features, which can be used in any learning to rank system.

6.2.2 Experimental Methodology

This section presents the experimental methodology for the ad hoc search task, which is kept the same as in Section 5.1.

Datasets are again the ClueWeb09-B and ClueWeb12-B13 corpora with TREC queries. The pre-processing and re-ranking setups used in Section 5.1 are kept: The ClueWeb09-B rankings re-ranked the top 100 documents provided by the Galogo sequential dependency model (SDM) with standard post-retrieval spam filtering [31]; the ClueWeb12-B13 rankings re-ranked the vanilla Indri language model with no spam filtering (Section 5.1); all documents were parsed by Boilerpipe [50] to title and body fields.

Evaluation Metrics are NDCG@20 and ERR@20. Statistical significances are tested by permutation test (randomization test) with $p < 0.05$.

Baselines: The goal of our experiments is to explore the usage of entity salience modeling in ad hoc search. To this purpose, our experiments focus on evaluating the effectiveness of `KESM`'s entity salience features in standard learning to rank; the proper baselines are the ranking features from word-based matches (`IRFusion`) and entity-based matches (`ESR`). Unsupervised retrieval with words (`BOW`) and entities (`BOE`) are also included.

`BOW` is the base retrieval model, which is SDM on ClueWeb09-B and Indri language model on ClueWeb12-B.

`BOE` is the frequency-based retrieval with bag-of-entities (Section 4.1). It uses TagMe annotations and exact-matches query and documents in the entity space. It performs similarly to the entity language model [80] as they use the same information.

`IRFusion` is the learning to rank system in Section 5.1 which uses standard word-based IR features such as language model, BM25, and TFIDF applied to body and title fields.

`ESR` is the Explicit Semantic Ranking implementation for web search in Section 5.1. Compared to `KESM`, it also matches query and documents in the entity space, but represents the query and documents by frequency-based bag-of-entities. Its entity embeddings are also from the TransE knowledge graph embedding [14] instead of from the entity salience task.

Implementation Details: As discussed in Section 6.2.1, the TREC benchmarks do not have sufficient relevance labels for effective end-to-end learning; we pre-trained the KEE and KIM of KESM using the New York Time corpus and used them to extract salience ranking features. The entity salience features are combined by the same learning to rank model (RankSVM [49]) as used by IRFusion and ESR, with the same cross validation setup (Section 5.1). Similar to ESR, the base retrieval score is included as a feature in KESM. In addition, we also concatenate the features of ESR or KESM to IRFusion to evaluate their effectiveness when combined with word-based features. The resulting feature sets ESR+IRFusion and KESM+IRFusion were evaluated exactly the same as they were individually.

As a result, the comparisons of KESM with LeToR and ESR hold out all other factors and directly investigate the effectiveness of the salience ranking features in a widely used learning to rank model (RankSVM). Given the current exploration stage of entity salience in information retrieval, we believe this is more informative than mixing entity salience signals into more sophisticated ranking systems such as latent space models [61] (Section 3.2) and attention-based models (Section 5.1), where many other factors come into play.

6.2.3 Evaluation Results

This section presents the evaluation results and a case study in the ad hoc search task.

6.2.3.1 Overall Results

Table 6.4 lists the ranking evaluation results. The three supervised methods, IRFusion, ESR, and KESM, all use the exact same learning to rank model (RankSVM) and only differ in their features. ESR+IRFusion and KESM+IRFusion concatenate the two feature groups and use RankSVM to combine them.

On both ClueWeb09-B and ClueWeb12-B13, KESM features are more effective than IRFusion and ESR features. On ClueWeb12-B13, KESM individually outperforms other features significantly by $8-20\%$. On ClueWeb09-B, KESM provides more novel ranking signals and KESM+IRFusion significantly outperforms ESR+IRFusion. The fusion on ClueWeb12-B13 (KESM+LeToR) is not as successful. A possible reason is that there are not enough ranking labels (training data) in ClueWeb12-B13 to learn a stable combination of the features.

To better investigate the effectiveness of entity salience in search, we evaluated the features on individual document fields. Table 6.5 shows the ranking accuracies of the three feature groups when only the title field (Title) or the body field (Body) is used. As expected, KESM is more effective on the body field than on the title field: Titles are less noisy and perhaps all title entities are salient—not much new information is provided by salience modeling; on the other hand, body texts are longer and more complicated, providing more opportunities for better text understanding.

The salience ranking features also behave differently than ESR and IRFusion. As shown by the W/T/L ratios in Table 6.4 and Table 6.5, more than 70% query rankings are changed by KESM. The ranking evidence provided by KESM features is from the interactions of query entities with the entities and words in the candidate documents. This evidence is learned from the entity salience corpus and is hard to be described by traditional frequency-based features.

Table 6.4: Ad hoc search accuracy of KESM when used as ranking features in learning to rank. Relative performances over IRFusion are shown in the percentages. W/T/L are the number of queries a method improves, does not change, or hurts, compared with IRFusion. †, ‡, §, and ¶ mark the statistically significant improvements over BOE†, IRFusion‡, ESR§, and ESR+IRFusion¶. BOW is the base retrieval model, which is SDM in ClueWeb09-B and language model in ClueWeb12-B13.

ClueWeb09-B					
Method	**NDCG@20**		**ERR@20**		**W/T/L**
BOW	0.2496	−5.26%	0.1387	−10.20%	62/38/100
BOE	0.2294	−12.94%	0.1488	−3.63%	74/25/101
IRFusion	0.2635	–	0.1544	–	–/–/–
ESR	0.2695†	+2.30%	0.1607	+4.06%	80/39/81
KESM	0.2799†	+6.24%	0.1663	+7.68%	85/35/80
ESR+IRFusion	0.2791†‡	+5.92%	0.1613	+4.46%	91/34/75
KESM+IRFusion	**0.2993**†‡§¶	+13.58%	**0.1797**†‡§¶	+16.38%	98/35/67

ClueWeb12-B13					
Method	**NDCG@20**		**ERR@20**		**W/T/L**
BOW	0.1060	−12.02%	0.0863	−6.67%	35/22/43
BOE	0.1173	−2.64%	0.0950	+2.83%	44/19/37
IRFusion	0.1205	–	0.0924	–	–/–/–
ESR	0.1166	−3.22%	0.0898	−2.81%	30/23/47
KESM	0.1301†§	+7.92%	**0.1103**‡§¶	+19.35%	43/25/32
ESR+IRFusion	0.1281	+6.30%	0.0951	+2.87%	45/24/31
KESM+IRFusion	**0.1308**†§	+8.52%	0.1079‡§¶	+16.77%	43/23/34

6.2.3.2 Case Study

The last experiment provides case studies about how KESM transfers its text understanding ability to search, by comparing the rankings of KESM-Body with ESR-Body. Both ESR and KESM match query and documents in the entity space, but ESR uses frequency-based bag-of-entities to represent documents while KESM uses entity salience. We picked the queries where KESM-Body improved or hurt compared to ESR-Body and manually examined the documents on which they disagreed. The examples are listed in Table 6.6.

The improvements from KESM are mainly from its ability to determine whether a candidate document emphasizes the query entities or just mentions the query terms. As shown in the top half of Table 6.6, KESM promotes documents where the query entities are more salient: the Wikipedia page about the ER TV show, a homepage about wind power, and a news article about the hurricane. On the other hand, ESR's frequency-based ranking might be confused by web pages that only partially talk about the query topic. It is hard for ESR to exclude those web pages because they also mention the query entities multiple times.

Many errors KESM made are due to the lack of text understanding on the query side. KESM focuses on modeling the salience of entities *in the candidate documents* and its ranking model

Table 6.5: Ranking performances of IRFusion, ESR, and KESM with title or body field individually. Relative performances (percentages) and Win/Tie/Loss are calculated by comparing with IRFusion on the same field. † and ‡ mark the statistically significant improvements over IRFusion† and ESR‡, also on the same field.

ClueWeb09-B					
Method	**NDCG@20**		**ERR@20**		**W/T/L**
IRFusion-Title	0.2584	−3.51%	0.1460	−5.16%	83/48/69
ESR-Title	0.2678	–	0.1540	–	–/–/–
KESM-Title	**0.2780**†	+3.81%	**0.1719**†‡	+11.64%	91/46/63
IRFusion-Body	0.2550	+0.48%	0.1427	−3.44%	80/46/74
ESR-Body	0.2538	–	0.1478	–	–/–/–
KESM-Body	**0.2795**†‡	+10.13%	**0.1661**†‡	+12.37%	96/39/65
ClueWeb12-B13					
Method	**NDCG@20**		**ERR@20**		**W/T/L**
IRFusion-Title	0.1187	+6.23%	0.0894	+3.14%	41/23/36
ESR-Title	0.1117	–	0.0867	–	–/–/–
KESM-Title	**0.1199**	+7.36%	**0.0923**	+6.42%	35/28/37
IRFusion-Body	0.1115	+4.61%	0.0892	−3.51%	36/30/34
ESR-Body	0.1066	–	0.0924	–	–/–/–
KESM-Body	**0.1207**‡	+13.25%	**0.1057**†‡	+14.44%	43/24/33

treats all query entities equally. As shown in the lower half of Table 6.6, the query entities may contain errors, for example, "Malindi Fickle", or general entities that blur the (perhaps implied) query intent, for example "Civil War", "State government", and "US Tax'. These query entities do not align well with the information needs and thus mislead KESM. Modeling the entity salience in *queries* is a different task which is more about understanding search intents. To address these error cases may require a deeper fusion of KESM in more sophisticated entity-oriented search systems that also focus on handling noisy query entities (Section 3.2, 5.1). However, that may require more training labels than those currently available in public benchmarks.

Table 6.6: Examples from queries that KESM improved or hurt, compared to ESR. Documents are selected from those where ESR and KESM disagreed. Their doc id's in ClueWeb and descriptions are listed. Their descriptions are manually written to describe their main topics. Documents in blue are those KESM promoted. Those in gray are those KESM demoted. Both compared to the original ranking of ESR.

Cases KESM Improved		
Query	**Query Entities**	**Promoted/Demoted Document**
"ER TV Show"	"ER (TV Series)"	clueweb09-enwp00-55-07707 The Wikipedia page of "ER (TV series)"
	"TV Program"	clueweb09-enwp02-22-20096 A list of films in Wikipedia
"Wind Power"	"Wind Power"	clueweb12-0009wb-54-01932 A homepage solely about wind energy
		clueweb12-0200wb-66-32730 Mainly about home solar power
"Hurricane Irene Flooding in Manville NJ"	"Hurricane Irene" "Flood"	clueweb12-0715wb-81-29281 Videos and news about Hurricane Irene
	"Manville, NJ"	clueweb12-0705wb-49-04059 Discusses Hurricane Irene disaster funding
Cases KESM Hurt		
Query	**Query Entities**	**Promoted/Demoted Document**
"Fickle Creek Farm"	"Malindi Fickle" "Stream"	clueweb09-en0005-66-00576 A list of hotels near Fickle Creak
	"Farm"	clueweb09-en0003-97-27345 A list of breeding farms
"Illinois State Tax"	"Illinois" "State Government"	clueweb09-en0011-23-05274 Discusses retirement-related state taxes
	"US Tax"	clueweb09-enwp01-67-20725 Discusses sales taxes in the US
"Battles in the Civil War"	"Battles"	clueweb09-enwp01-30-04139 A list of wars in the Muslim world
	"Civil War"	clueweb09-enwp03-20-07742 A list of American Civil War battles

6.3 Summary

This chapter presents KESM, the Kernel Entity Salience Model that estimates the salience of entities in documents. KESM represents entities and words with distributed representations, models their interactions using kernels, and combines the kernel scores to estimate entity salience. The semantics of entities in the knowledge graph—their descriptions—are also incorporated to enrich entity embeddings. In the entity salience task, the whole model is trained end-to-end using automatically generated salience labels.

In addition to the entity salience task, KESM is also applied to ad hoc search and ranks documents by the salience of query entities in them. KESM calculates the kernel scores of query entities in the document, combines them to salience ranking features, and uses a ranking model to predict the query-document ranking score. When ranking labels are scarce, the ranking features can be extracted by pre-trained distributed representations and kernels from the entity salience task and then used by standard learning to rank. These ranking features convey KESM's text understanding ability learned from entity salience labels to search.

Our experiments on two entity salience corpora, a news corpus (New York Times) and a scientific publication corpus (Semantic Scholar), demonstrate the effectiveness of KESM in the entity salience task. Significant and robust improvements are observed over frequency and feature-based methods. Compared to those baselines, KESM performs significantly better on rare entities and short documents where frequency signals are limited; its Kernel Interaction Model is also more powerful than the raw similarities used by PageRank. Overall, KESM is a stronger model with richer sources of evidence and a more powerful architecture.

In our ad hoc search experiments, the salience features provided by KESM trained on the New York Times corpus outperform both word-based ranking features and frequency-based entity-oriented ranking features, despite differences between the salience task and the ranking task. The advantages of the salience features are more observed on the document bodies on which more sophisticated text understanding is required.

Our case studies on the winning and losing queries of KESM demonstrate the influence of entity salience in search. By considering the salience of query entities, KESM favors the candidate documents where the query entities are the core topics (salient) over those only mentioning the query entities. For example, many candidate documents demoted by KESM contain lists of entities; the query entities are only partial to their topics. With only frequency signals, it is hard to distinguish such documents from more relevant ones.

We find it very encouraging that KESM successfully conveys the text understanding ability from entity salience estimation to search. Estimating entity salience is a fine-grained text understanding task that focuses on the detailed interactions between entities and words. Previously it was uncommon for text processing techniques at this granularity to be as effective in information retrieval. Often shallower methods worked better for search. However, the fine-grained text understanding provided by KESM—the interaction and consistency between query entities with the document entities and words—actually improves the ranking accuracy. We view this work as an encouraging step from "search by matching" to "search with meanings" [7] and hope it will motivate more future explorations towards this direction.

Chapter 7

Conclusion

The final chapter concludes this dissertation with its summary and contributions. It also discusses some best practices found in this thesis research and its potential impacts.

7.1 Thesis Summary

This thesis research improves the representation, retrieval, and understanding of texts using knowledge graphs. It progresses through four stages: enriching word-based query representation with knowledge graphs (Chapter 3), building and ranking with entity-based text representations (Chapter 4), the word-entity duet framework that integrates word-based and entity-based representations (Chapter 5), and the entity salience model which improves the understanding of documents (Chapter 6).

Chapter 3 presents a series of techniques that integrate the information from knowledge graphs into word-based retrieval systems through *query representations*. Section 3.1 improves *query expansion* with entities from knowledge graphs. It develops a two-step query expansion method. The first step finds relevant entities from entity search results and top retrieved documents' annotations. The second step selects expansion words from the textual descriptions of related entities by frequency and their ontology similarities with the query. This two-step process introduces new signals to the search systems: the alignment from query to related entities, the words from entity descriptions, and the ontology from the knowledge graph. These signals are external to the original query and candidate documents, while also often at a higher quality: Entity search and entity linking are usually more accurate than ad hoc retrieval; entity descriptions are often a cleaner source of expansion terms than pseudo relevance feedback documents; the ontology is carefully designed and incorporates expert knowledge. These methods improve both the effectiveness and the robustness of previous query expansion methods on TREC datasets, with about 30% better accuracy and 50% less damaged queries. This performance shows the great potential of knowledge graphs in information retrieval and motivated this thesis research.

Section 3.2 presents `EsdRank` which improves supervised ranking by connecting query and documents through related entities. It proposes and experimented with three sources of related entities: query annotations, entity search, and entities annotated to top retrieved documents. These related entities serve as a latent layer and introduce additional connections between query and documents. They introduce information from knowledge graphs as features between query, related entities, and candidate documents. The ranking process with the latent entity space is formulated as a generative process and modeled by a novel latent space learning to rank model, Latent-ListMLE, which learns the latent entity weights and the document rankings jointly from ranking labels. `EsdRank` outperforms word-based learning to rank systems on both the general domain using Freebase, and the medical domain with the medical controlled vocabulary. As shown in our analyses, the improvements come from both the effective ranking signals from the knowledge and the latent ranking model that effective uses them. Our experiments also find that query entity linking is a more effective way to find related entities. These conclusions are widely adopted in the rest of this thesis research.

Section 3.3 provides a new entity search method using learning to rank. When using knowledges in search, a crucial step is to find related entities, which can be done be entity search. In the same time, entity search itself is a real-world task that has a wide range of applications. This work represents our initial attempt to build our own related entity finding system in the format of entity search. It uses learning to rank to combine the text similarities features between query and entity's predefined fields. The feature-based ranking model significantly outperforms previous entity retrieval systems in queries with variant search intent. Interestingly, even for queries that intended to search for entity attributes, the multi-field text representations and textual ranking features still perform the best in our analyses. These observations further confirms the effectiveness of entity texts in information retrieval tasks.

Chapter 4 presents *entity-based text representations*. Rather than relying on the word-based representations and trying to enrich them with knowledge graphs, Chapter 4 directly represents text in the entity space using automatic entity annotation, which is the most reliable source of related entities found in previous chapters.

Section 4.1 conducts the studies of the feasibility of pure entity-based representations in the environment of general domain web search. It first studies the accuracy and coverage of three popular entity linking systems. The results show that though the annotation accuracy is far from perfect (about 60%), if configured properly, the coverage on general domain texts is promising. Then it builds a simple bag-of-entities model that represent query and documents by their entity annotations. In our experiments on TREC data, simple frequency-based unsupervised retrieval models built upon bag-of-entities are able to outperform their bag-of-words versions, even with imperfect automatic entity annotations. Those results confirmed the feasibility of automatic entity-based representations.

Section 4.2 presents *Explicit Semantic Ranking* (`ESR`), a new ranking technique that integrates knowledge graph semantics in search in the format of entity embeddings. It builds upon the bag-of-entities representations and matches query and document in the entity space. Compared to the exact match based models in Section 4.1, `ESR` leverages knowledge graph edges—relations and attributes—to *soft match* entities in the query and documents. An obstacle found in utilizing knowledge graphs is the sparsity of their edges: many related entities are not connected in the graph or even share no neighbors. To handle this sparsity, `ESR` converts the edges around an entity into its distributed representations using embeddings, which are dense, continuous, and easy to handle. The ranking in `ESR` is performed by a translation model and a pyramid pooling layer. The translation model calculates the embedding similarities between all query-document entity pairs; the pyramid pooling layer convert the translation scores into multi-level soft match features to be used in learning to rank. This section also introduces our techniques into a new search domain, Academic Search, by using the search logs and benchmarks from SemanticScholar.org. ESR improves the word-based online production systems of Semantic Scholar, with the majority of improvements on the queries that are hard for word-based search. The knowledge graph embeddings and multi-level soft matches are crucial for these improvements. They made ESR one of the first to successfully utilize knowledge graph relations in ad hoc search.

Chapter 5 presents the *duet framework* that combines the word-based and entity-based text representations. The effectiveness of bag-of-entities makes combining it with bag-of-words a natural step to improve search accuracy. The research presented in Chapter 5 provides a systematic approach to perform this combination.

Section 5.1 develops the general *word-entity duet* framework to combine the two representation spaces. It represents query and documents in the duet space of bag-of-words and bag-of-entities. The duet provides four different ways to match query-document: matching within the word space, within the entity space, and two cross-space matches from query words to document entities and the other way around. The in-space matches cover the ranking features previously studied in word-based retrieval and entity-based retrieval; the cross-space matches incorporate signals from entity texts, for example, features in EsdRank and entity search (Section 3.3). This section then provides a novel hierarchical learning to rank model to handle the noise in the query annotations: Queries are short and often ambiguous; linking them inevitably introduces noisy entities. The hierarchical ranking model builds an attention mechanism on the query entities, which utilizes signals from entity linking and embeddings to model their reliability. The model is jointly trained using relevance labels without requiring ground truth labels at the entity level. Our experiments in the TREC benchmarks demonstrate the effectiveness of each of the four-way matches, as well as the necessity of the hierarchical ranking model on noisy queries.

Section 5.2 presents JointSem, which jointly conducts query entity linking and document ranking using the hierarchical ranking model.Its linking part keeps multiple possible candidate entities and provides entity relatedness and ambiguity features to the attention mechanism. The latter learns the weights on the candidate entities simultaneously with the ranking part. This leads to a soft-linking scheme where the decision of entity linking is not to pick one from possible candidate entities but to learn the weights on multiple choices. As shown in our experiments, this "soft-linking" is very useful on ambiguous queries; it diversifies the linking results and allows the ranking model to "fix" some linking errors.

Chapter 6 advances this thesis research from frequency-based text representations to salience-based. In bag-of-words and bag-of-entities, the texts are broken down into isolated terms, and the term importances are based on frequency signals. Effective as it is, frequency is only shallow text understanding. The research presented in Chapter 6 improves text understanding by better estimating the importance of entities in texts with neural networks and knowledge graph semantics.

Section 6.1 developed a new entity salience estimation model, KESM, that better estimate term salience by modeling their *interactions*. Given entities and words in the text, KESM represents them using distributed representations and uses a novel kernel technique to model the term interactions in the embedding space. The kernels convert the embedding similarities into multi-level interaction strengths which also are smooth and trainable. The interaction patterns modeled by the kernels are then combined by KESM to estimate entity salience. The knowledge graph semantics can also be integrated as external memories (extra embeddings) of the corresponding entities. In the entity salience task, KESM is learned end-to-end and encodes the entity salience signals into its embeddings, which are optimized by the kernels. KESM provides more accurate and balanced entity salience estimations on both a new corpus and a scientific publication corpus. The kernels are crucial to its effectiveness as shown in our experiments.

Section 6.2 adapts KESM to ad hoc search using our entity-oriented search framework. It uses KESM to estimate the salience of query entities in candidate documents, assuming that documents centered around query entities are more likely to be relevant than those just mention the query terms. In search scenarios where not sufficient relevance labels are available to train large neural models, we train KESM in the entity salience estimation task, and use the pre-trained model to produce kernel interaction scores between query entities and document terms. The kernel scores can be used in standard learning to rank systems as salience ranking features. In our experiments on the TREC ad hoc search benchmarks, the KESM pre-trained in the Those adapted salience ranking features perform better than word-based features and frequency-based features from Explicit Semantic Ranking. They model the interaction between query terms and document terms and help promote documents whose main topics are about the query entities. These results demonstrate that KESM transfers its text understanding capability learned from the salience task to improve the search task. Previously it is uncommon for such fine-grained text understanding techniques to be effective at ad hoc search tasks.

7.2 Thesis Contributions

This dissertation improves the representation, retrieval, and understanding of texts in search engines using entities and their semantics from knowledge graphs. The idea of integrating human knowledge in search dates back to the classic controlled vocabulary based retrieval systems. However, until recently, structured semantics were mainly applicable to specifically designed task and a few particular domains. This thesis research revisits this classic idea with the aids of modern knowledge graphs, automatic grounding techniques, and novel machine learning models. It contributes a new entity-oriented search paradigm, which not only successfully utilizes knowledge graphs to improve search, but also achieves this success robustly in various search domains, different types of knowledge graphs, and multiple search tasks.

Throughout this thesis research, three main ideas lead to the success of entity-oriented search. The rest of this section describes each of them in detail.

Semantic Grounding for Search

A prerequisite to the utilization of knowledge graph in search is the alignment of knowledge graphs to the search process. More specifically, it needs to find related entities for the query and documents. This dissertation explored the effectiveness of two ground techniques, entity linking and entity search, and found their insufficiencies in finding related entities for search: Entity linking focuses more on precision and named entities; entity search focuses more on satisfying the information needs of a few query types; they both only cover a small fraction of the general search traffic.

This dissertation developed a series of semantic grounding techniques that *emphasize general domain coverage to meet the needs of general search tasks*. We examined the entity linking technique and discovered the necessity of recall in finding related entities for search. Then by relaxing the precision constraint, we developed a recall-oriented entity linking system that has sufficient coverage in both special domain and general domain search. It becomes the standard way to find related entities in entity-oriented search. Later, this dissertation developed the JointSem method that collectively conducts the entity alignment step and the ranking step. It directly tailors the alignment step towards to needs of the ranking model and thus reduces the discrepancy between the two parties.

These approaches resemble the information retrieval tradition of "noisy information is better than no information". They provide simple and practical solutions for aligning knowledge graphs to query and documents, which are crucial to the advancement of entity-oriented search. These techniques are also not only limited to retrieval tasks and could offer the benefits of knowledge graphs to a broad range of text applications.

Integrating Knowledge Graph Semantics in Search

The structures in knowledge graphs follow the conventions in symbolic systems: attributes, ontologies, and relations organized in RDF triples. These precise structures make the corresponding semantic systems precise but fragile. From the perspective of search engines, it was unclear how to utilize these 'symbolic' evidence to improve search in a general and robust way.

This dissertation invented a series of approaches to integrate the knowledge graph semantics to search. Instead of restricting to the symbolic style, our approaches find a middle-ground between their precise structures and IR-style text processing. We convert the RDF triples into bag-of-words of entities and extracts IR-style ranking features, which are effective and easy-to-use in existing search systems. We encode the sparse relations into embeddings, which effectively soft match query and documents in the bag-of-entities space. We also develop a domain adaptation approach that transfers the text understanding capability learned from entity salience to improve ad hoc search.

These approaches provide robust, flexible, and effective integration of knowledge graph semantics in search. They do not require 100% accuracy from the knowledge graph—errors are mitigated as the ranking features and embeddings are expected to be non-exact. The inclusion of unstructured elements from knowledge graphs, e.g., entity descriptions, also leads to significant improvements. These properties also make our approaches more compatible with automatically constructed semantic resources. For example, our academic knowledge graph, with just four types of automatically extracted edges, improves academic search accuracy significantly.

Joint Learning

The automatic and coverage-oriented entity annotations inevitably include errors. This type of errors is a significant bottleneck for entity-oriented search and often required manual filtering. To address this problem, this dissertation developed several machine learning models that learn how to handle the errors in entity annotations jointly with the ranking process. The first such joint learning model is the latent-space learning to rank model. It uses the probabilistic graphic model approach, treats the entities as a latent space between query and documents, and leverages the EM method to learn the entity weights. Then we built an attention-based neural ranking model, which applies an attention mechanism to learn to weight entities directly from ranking labels. Our final model, JointSem, directly combines the query entity linking step with the entity-based ranking step in a neural network.

These joint learning models are applicable to any supervised ranking scenarios, as document relevance labels are already available. They effectively leverage ranking labels to handle the annotation noisy, which dramatically relaxes the needs of entity linking quality. Their plug-in-and-use flexibility and effectiveness significantly broadened the influence of entity-oriented search, especially to search scenarios where the entity annotations are noisy.

7.3 Best Practices

This thesis research explored and experimented with variants design choices in multiple domains. This section suggests some of the best practices and requirements for building knowledge-enhanced information retrieval (KG4IR) systems .

High Coverage Entity Set

A preliminary requirement when building a KG4IR system is a high-coverage entity set in the target domain. It determines the potential impacts of this dissertation's techniques on the search system—If there is no entity available in the query and documents, there is little knowledge graphs can help.

In the general domain, Wikipedia and Freebase often provide the entity set with sufficient coverage. In specific domains, one should start with the entities from domain-specific resources, for example, medical controlled vocabularies for the medical domain. If such domain-specific resources are not available, we suggest to start with inheriting entities from general domain knowledge graphs. Usually Wikipedia contain many in-domain entities, which are a good starting point. More in-domain entities can be added to the set later (Section 4.2). We also recommend include entities from all possible categories. Entities only from handful of named entity categories are unlike to provide sufficient coverage on the search traffic.

There are many approaches that automatically recognize and extract entities. In our experiences, automatically extracted entities do not outperform Wikipedia entities in precision or coverage. Even in some special domains, the automatically extracted ones are often very noisy and the correct ones often also exist in Wikipedia. At the current stage, the best practice is to leverage entities from existing resources (and add high-confident extracted ones to them), not to extract everything automatically.

In-Domain Knowledge Graph Semantics

The most important factor of useful knowledge graph semantics is that they convey in-domain information for the target application. For example, academic search requires semantics about the academic domain (Section 4.2); medical search prefers the medical controlled vocabulary

(Section 3.2); general domain web search work well with Freebase. Semantics about another domain is not as helpful as those in-domain; those from the general domain might be to general for the target domain. When using the knowledge graph semantics, the coverage is also crucial; overly emphasizing the structure preciseness often results in limited coverage and restricts the potential of knowledge graph semantics in real-world applications.

On the other hand, the formats and precision of the knowledge graph semantics are rather flexible. The formats can vary from textual attributes, RDF triples, closed form schema relations, or just entity contexts. They can be manually curated as in Wikipedia or automatically extracted. In fact, the two main approaches utilizing knowledge graph semantics in this dissertation are by bag-of-words and embeddings. Both are flexible—other semantic formats can be easily converted to them—and also robust—they are used to draw connections between related texts and do not have to be as exact as when used in reasoning.

Recall-Oriented and Efficient Entity Alignment

In the entity alignment stage (entity linking or related entity finding), the two most important properties required for the alignment system is its recall and efficiency. Recall is required to ensure the coverage of aligned entities. Efficiency is required when aligning entities to the query and the documents. The alignment on queries is often done online during search; it must satisfy the latency requirements of the search engine. The alignment on documents cannot be too complicated when the corpus is large. Some exiting entity linking systems require too many computing resource to process large corpora such as ClueWeb.

On the other hand, the entity alignment does not have to be perfect. In many cases, this thesis research finds simple entity linking systems work reasonably well, and their precise is not far from more complex ones in the open domain. In our experience, the best practice is to start with simple entity linkers such as TagMe [38] and CMNS [43], with 'keep everything setup' to ensure coverage. The linking accuracy can be fixed by integrating the entity linking stage into the search system and joint learning the two together (Section 5.2).

Choice of Ranking Model

The choice of ranking model depends on the quality of aligned entities and the training data size.

If the quality of introduced entities is not as good, the ranking model needs to handle their noises. This is more common in the general domain whose texts are more ambiguous than special domains. In this case, hierarchical ranking models that also learn to weight entities are preferred. If the quality of related entities is decent, for example, with more than 70% accuracy, adding the entity information as features to existing learning to rank systems can already be effective.

The size of training data determines the complexity of the ranking model. With limited training data, feature-based systems are often easier to work with. Sometimes the feature set may needs pruning to prevent overfitting, especially when using hierarchical ranking models. For example, in a typical TREC benchmark with one hundred labeled queries, more than one hundred feature dimensions may make the learning unstable. On the other hand, if sufficient training data is available, end-to-end learned neural models might be more effective.

7.4 Thesis Impact

This last section discusses potential research directions motivated by this dissertation.

Building Knowledge Graphs for Information Retrieval Systems

The current knowledge graphs follow the traditions of symbolic systems, which do not fully align with the needs of search engines. The semantics in existing knowledge graphs are precise but may lack the coverage on search tasks. The structures of knowledge graph semantics are also strict and complex, but the most effective way to use them now is through shallow formats such as bag-of-words and embeddings.

One potential future direction is to build knowledge graphs towards the needs of real-world information retrieval systems. Different applications have different needs: Web search benefits more from entity texts; recommendation systems prefer relations about user shopping behaviors; dialog systems require task slots and action states. Instead of building a knowledge graph for all possible applications, constructing a knowledge graph specifically for the target application may be more doable. Towards this line of research, a potential research topic is to automatically identify the needs of semantic information for downstream tasks. Another possible topic is to build end-to-end knowledge-enhanced search systems that actively extract knowledge to integrate to the search process.

The format of knowledge graph semantics can also be further studied. RDF triples work well with tasks such as logic reasoning, but their exact structures might be too fragile. Bag-of-words and embeddings are more robust, but might be too shallow. Identifying the right granularity of structures for different scenarios is an interesting future research topic. Perhaps the solution is to maintain flexible structures with variant complexities.

Combining Neural Networks and Knowledge Graphs

Variant parts of this dissertation have demonstrate the advantages of combining neural techniques and knowledge graphs. This combination leads to many promising future research directions.

One direction is in improving the utilization of knowledge graphs with neural methods. For example, neural techniques may better select which part of the knowledge graph to use for the current task, perhaps with more sophisticated neural architectures. They can also improve the ranking of documents in the entity space, perhaps by learning better entity representations and more fine-grained soft matches.

Another direction is to bring the advantage of knowledge graphs to neural systems. The incorporation of prior knowledge can improve the generalization ability of neural networks, as the information from knowledge graphs is more generally applicable and shared across domains [63]. Entities and explicit structures may also make the neural systems more explainable.

We believe future intelligent systems will benefit from the joint effort of knowledge graphs and neural networks. This dissertation provides some initial evidence for it; we hope more will come in the near future.

Deeper Language Understanding in Information Systems

Previously it was hard for fine-grained text processing techniques to improve information retrieval tasks. In many cases, search engines use user clicks to by-pass the needs of understanding language. However, as information systems try to serve more and more sophisticated information needs, deeper language understanding becomes more and more necessary.

One research direction in this area is to build more advanced text representations. The text representations built in this dissertation include weak structures around entities to integrate knowledge graph semantics. A possible next step is to learn more structures within the document. The structures can be formed by entities, embeddings, and text fragments, with the goal to better reflect text meanings, for example, the sub topics, salience sentences, or discourse transactions in the document. Research questions to study include how to learn these structures, how to find proper training data, and how to utilize the learned structures in downstream applications.

Deeper language understanding will change the way search engines interact with users. It may enable search engines to present users more distilled information, e.g. those abstracted from the search result page through information extraction and summarization. It can also help user modeling if search engines better understand user's text data. For example, it might be possible to maintain a personalized knowledge graph for each user, which contains her points of interests, past actions, level of expertises, and potential information needs. Such a personal knowledge graph can help a wild range of applications, for example, web search engines, conversational search systems and personal assistants.

Language understanding is one of the most critical milestones in Artificial Intelligence. Achieving it will require efforts from multiple research communities. We hope this dissertation has illustrated a promising picture and will lead to much more in the future.

Bibliography

[1] Medical Subject Headings - Home Page. `http://www.nlm.nih.gov/mesh/meshhome.html`. Accessed: 2018-06-01. 2.1, 3.2

[2] Andrew Arnold and William W Cohen. Information extraction as link prediction: Using curated citation networks to improve gene detection. In *International Conference on Wireless Algorithms, Systems, and Applications (WASA 2009)*, pages 541–550. Springer, 2009. 4.2.1

[3] Krisztian Balog and Robert Neumayer. A test collection for entity search in DBpedia. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2013)*, pages 737–740. ACM, 2013. 2.3, 3.3, 3.3.1, 3.3.2, 3.3.3

[4] Krisztian Balog, Leif Azzopardi, and Maarten De Rijke. Formal models for expert finding in enterprise corpora. In *Proceedings of the 29th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2006)*, pages 43–50. ACM, 2006. 4.2.1

[5] Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. Open information extraction for the web. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 2670–2676. IJCAI, 2007. 2.2

[6] Hannah Bast and Elmar Haussmann. More accurate question answering on Freebase. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management (CIKM 2015)*, pages 1431–1440. ACM, 2015. 3.3.1

[7] Hannah Bast, Björn Buchhold, Elmar Haussmann, et al. Semantic search on text and knowledge bases. *Foundations and Trends in Information Retrieval*, 10(2-3):119–271, 2016. 6.3

[8] Michael Bendersky, David Fisher, and W Bruce Croft. UMass at TREC 2010 Web Track: Term dependence, spam filtering and quality bias. In *Proceedings of The 19th Text Retrieval Conference, (TREC 2010)*. NIST, 2010. 3.1.1, 3.1.2.1, 3.1.3, 3.1.3

[9] Michael Bendersky, Donald Metzler, and W. Bruce Croft. Parameterized concept weighting in verbose queries. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2011)*, 2011. 6.1.1

[10] Michael Bendersky, Donald Metzler, and W Bruce Croft. Effective query formulation with multiple information sources. In *Proceedings of the Fifth ACM International Conference*

on Web Search and Data Mining (WSDM 2012), pages 443–452. ACM, 2012. 3.2.2

[11] Adam Berger and John Lafferty. Information retrieval as statistical translation. In *Proceedings of the 22nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1999)*, pages 222–229. ACM, 1999. 4.2.1

[12] Roi Blanco and Christina Lioma. Graph-based term weighting for information retrieval. *Information Retrieval*, 15(1):54–92, 2012. 6, 6.1.1, 6.1.4

[13] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data (SIGMOD 2008)*, pages 1247–1250. ACM, 2008. 2.2, 3.2, 5.1.3

[14] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems (NIPS 2013)*, pages 2787–2795, 2013. 5.1.1.2, 5.1.3, 5.1.4.3, 6.2.2

[15] Wladmir C Brandão, Rodrygo LT Santos, Nivio Ziviani, Edleno S Moura, and Altigran S Silva. Learning to expand queries using entities. *Journal of the Association for Information Science and Technology (JASIST)*, 65(9):1870–1883, 2014. 3.2.2

[16] Andrei Z Broder, Marcus Fontoura, Evgeniy Gabrilovich, Amruta Joshi, Vanja Josifovski, and Tong Zhang. Robust classification of rare queries using web knowledge. In *Proceedings of the 30th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2007)*, pages 231–238. ACM, 2007. 3.2.2

[17] Guihong Cao, Jian-Yun Nie, Jianfeng Gao, and Stephen Robertson. Selecting good expansion terms for pseudo-relevance feedback. In *Proceedings of the 31st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2008)*, pages 243–250. ACM, 2008. 3.1, 3.1.1, 3.1.2.2, 3.1.3

[18] Cornelia Caragea, Florin Adrian Bulgarov, Andreea Godea, and Sujatha Das Gollapalli. Citation-enhanced keyphrase extraction from research papers: A supervised approach. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1435–1446. ACL, 2014. 4.2.3.1

[19] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, volume 5, page 3. AAAI, 2010. 1.2, 2.2

[20] David Carmel, Ming-Wei Chang, Evgeniy Gabrilovich, Bo-June (Paul) Hsu, and Kuansan Wang. ERD'14: Entity recognition and disambiguation challenge. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2014)*. ACM, 2014. 1.2, 2.3, 3.2.1, 3.2.1.2, 3.2.2, 3.2.3.1, 3.2.3.3, 4.1, 4.1.1, 4.1.2, 4.2.4.2, 5.2, 5.2.1.1

[21] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software

available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`. 3.1.2.2

[22] Jing Chen, Chenyan Xiong, and Jamie Callan. An empirical study of learning to rank for entity search. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2016)*, pages 737–740. ACM, 2016. 9

[23] Kevyn Collins-Thompson. *Robust Model Estimation Methods for Information Retrieval*. PhD thesis, Carnegie Mellon University, December 2008. 3.1.1, 3.1.4.3

[24] Kevyn Collins-Thompson. Estimating robust query models with convex optimization. In *Proceedings of the 21st Advances in Neural Information Processing Systems (NIPS 2009)*, pages 329–336. NIPS, 2009. 3.1.1

[25] Kevyn Collins-Thompson. Reducing the risk of query expansion via robust constrained optimization. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM 2009)*, pages 837–846. ACM, 2009. 3.1.1

[26] Kevyn Collins-Thompson and Jamie Callan. Estimation and use of uncertainty in pseudo-relevance feedback. In *Proceedings of the 30th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2007)*, pages 303–310. ACM, 2007. 3.1.1

[27] Gordon V Cormack, Mark D Smucker, and Charles LA Clarke. Efficient and effective spam filtering and re-ranking for large web datasets. *Information Retrieval*, 14(5):441–465, 2011. 3.1.3, 3.2.2

[28] Nick Craswell, Arjen P. de Vries, and Ian Soboroff. Overview of the TREC 2005 Enterprise Track. In *Proceedings of The 14th Text Retrieval Conference (TREC 2005)*, volume 5, pages 199–205, 2005. 4.2.1

[29] W Bruce Croft, Donald Metzler, and Trevor Strohman. *Search Engines: Information Retrieval in Practice*. Addison-Wesley Reading, 2010. 1.1, 6, 6.1.1, 6.2.1

[30] Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM 2018)*, pages 126–134. ACM, 2018. 6.1.2.3

[31] Jeffrey Dalton, Laura Dietz, and James Allan. Entity query feature expansion using knowledge base links. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2014)*, pages 365–374. ACM, 2014. 2.4, 3.1.3, 3.1.3, 3.2, 3.2.1, 3.2.1.2, 3.2.2, 3.10, 3.2.3.1, 3.2.3.2, 4.1.2, 4.1.3.2, 4.1.3.2, 4.2.3.1, 5.1.1.3, 5.1.3, 5.6, 5.1.4.3, 5.2, 5.2.2, 6.1.1, 6.2.2

[32] Laura Dietz and Patrick Verga. UMass at TREC 2014: Entity query feature expansion using knowledge base links. In *Proceedings of The 23st Text Retrieval Conference (TREC 2014)*. NIST, 2014. 3.2.3.1, 3.2.3.3

[33] Milan Dojchinovski, Dinesh Reddy, Tomás Kliegr, Tomas Vitvar, and Harald Sack. Crowdsourced corpus with entity salience annotations. In *Proceedings of the 10th Edition of the Languge Resources and Evaluation Conference (LREC 2016)*, 2016. 6, 6.1.1, 6.2.1

[34] Metzler Jr Donald A. *Beyond Bags of Words: Effectively Modeling Dependence and Features in Information Retrieval.* PhD thesis, University of Massachusetts Amherst, September 2007. 3.1.1, 3.1.3

[35] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2014)*, pages 601–610. ACM, 2014. 1.2

[36] Jesse Dunietz and Daniel Gillick. A new entity salience task with millions of training examples. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2014)*, pages 205–209, 2014. 6, 6.1.1, 6.1.4, 6.1.4, 6.1.4, 6.2, 6.1.5.1, 6.2.1

[37] Faezeh Ensan and Ebrahim Bagheri. Document retrieval model through semantic linking. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (WSDM 2017)*, pages 181–190. ACM, 2017. 2.4, 6.1.1

[38] Paolo Ferragina and Ugo Scaiella. Fast and accurate annotation of short texts with Wikipedia pages. *arXiv preprint arXiv:1006.3498*, 2010. 2.3, 2.4, 3.2.2, 4.1, 4.1.1, 4.1.2, 5.1.1.1, 5.1.3, 5.2.2, 6.1.4, 7.3

[39] Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. FACC1: Freebase annotation of ClueWeb corpora, Version 1 (Release date 2013-06-26, Format version 1, Correction level 0). `http://lemurproject.org/clueweb09/FACC1/`, 2013. Accessed: 2018-06-01. 1.2, 2.3, 3.1.2.1, 3.1.3, 3.2.2, 4.1, 4.1.1, 4.1.2, 4.2.3.1, 4.2.4.2, 5.1.3

[40] Siddharth Gopal and Yiming Yang. Recursive regularization for large-scale classification with hierarchical and graphical dependencies. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2009)*, pages 257–265. ACM, 2013. 2.1

[41] Gregory Grefenstette and Laura Wilber. *Search-Based Applications: At the Confluence of Search and Database Technologies.* Morgan & Claypool Publishers, 2010. 2.1

[42] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W.Bruce Croft. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM 2016)*, pages 55–64. ACM, 2016. 4.2.1, 4.2.3.2, 4.2.3.2, 4.2.4.2, 5.1.1.2, 3

[43] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. Entity linking in queries: Tasks and evaluation. In *Proceedings of the Fifth ACM International Conference on The Theory of Information Retrieval (ICTIR 2015)*, pages 171–180. ACM, 2015. 2.3, 4.1, 4.1.1, 4.1.2, 4.2.3.1, 4.2.4.2, 7.3

[44] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. Exploiting entity linking in queries for entity retrieval. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval (SIGIR 2016)*, pages 209–218. ACM, 2016. 2.3

[45] Faegheh Hasibi, Fedor Nikolaev, Chenyan Xiong, Krisztian Balog, Svein Erik Bratsberg, Alexander Kotov, and Jamie Callan. DBpedia-Entity v2: A test collection for entity search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2017)*, pages 1265–1268. ACM, 2017. 2.3

[46] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management (CIKM 2013)*, pages 2333–2338. ACM, 2013. 4.2.1

[47] Heng Ji, Joel Nothman, Ben Hachey, and Radu Florian. Overview of TAC-KBP 2015 trilingual entity discovery and linking. In *Proceedings of the 2015 Text Analysis Conference (TAC2015)*. NIST, 2015. 1.2, 2.3

[48] T. Joachims. Making large-scale SVM learning practical. LS8-Report 24, Universität Dortmund, LS VIII-Report, 1998. 3.2.2

[49] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2002)*, pages 133–142. ACM, 2002. 3.2.2, 4.2.4.2, 5.1.3, 6.2.2

[50] Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. Boilerplate detection using shallow text features. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining (WSDM 2010)*, pages 441–450. ACM, 2010. 6.2.2

[51] Bevan Koopman, Guido Zuccon, Peter Bruza, Laurianne Sitbon, and Michael Lawley. Information retrieval as semantic inference: A graph inference model applied to medical search. *Information Retrieval*, 19:6–37, 2016. 2.1

[52] Alexander Kotov and ChengXiang Zhai. Tapping into knowledge base for concept feedback: Leveraging ConceptNet to improve search results for difficult queries. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining (WSDM 2012)*, pages 403–412. ACM, 2012. 3.1.1

[53] Robert Krovetz. Viewing morphology as an inference process. In *Proceedings of the 16th International ACM SIGIR Conference on Research and Development in Information Retrieval, (SIGIR 1993)*, pages 191–202. ACM, 1993. 3.1.3

[54] Victor Lavrenko and W Bruce Croft. Relevance based language models. In *Proceedings of the 24th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2001)*, pages 120–127. ACM, 2001. 3.1.1, 3.1.2.3, 3.1.3, 3.2.1.2, 3.2.2

[55] Kyung Soon Lee, W Bruce Croft, and James Allan. A cluster-based resampling method for pseudo-relevance feedback. In *Proceedings of the 31st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2008)*, pages 235–242. ACM, 2008. 3.1.1

[56] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and

Christian Bizer. DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web Journal*, 2014. 2.2

[57] Hang Li and Jun Xu. Semantic matching in search. *Foundations and Trends in Information Retrieval*, 8:89, 2014. 2.1

[58] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI 2015)*, pages 2181–2187, 2015. 5.1.1.2

[59] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009. 3.2.1.3, 3.2.3.1, 4.2.2, 6.1.1, 6.1.3

[60] Tie-Yan Liu, Jun Xu, Tao Qin, Wenying Xiong, and Hang Li. LETOR: Benchmark dataset for research on learning to rank for information retrieval. In *Proceedings of SIGIR 2007 Workshop on Learning to Rank for Information Retrieval*, pages 3–10, 2007. 1.1

[61] Xitong Liu and Hui Fang. Latent entity space: A novel retrieval approach for entity-bearing queries. *Information Retrieval*, 18(6):473–503, 2015. 2.4, 4.1.2, 4.1.3.2, 4.2.3.1, 5.1.1.3, 5.1.3, 5.1.4.3, 5.2, 5.2.1.1, 6.1.1, 6.2.2

[62] Xitong Liu, Peilin Yang, and Hui Fang. Entity came to rescue - Leveraging entities to minimize risks in web search. In *Proceedings of the 23st Text Retrieval Conference, (TREC 2014)*. NIST, 2014. 2.4, 3.2.1.2, 3.2.3.1, 3.2.3.3

[63] Zhenghao Liu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. Entity-duet neural ranking: Understanding the role of knowledge graph semantics in neural information retrieval. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, page To Appear. ACL, 2018. 6.1.3, 7.4

[64] Chunliang Lu, Wai Lam, and Yi Liao. Entity retrieval via entity factoid hierarchy. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015)*, pages 514–523. ACL, 2015. 3.3, 3.3.1, 3.3.3, 3.3.4.1

[65] Yue Lu, Hui Fang, and Chengxiang Zhai. An empirical study of gene synonym query expansion in biomedical information retrieval. *Information Retrieval*, 12(1):51–68, 2009. 2.1

[66] Zhiyong Lu, Won Kim, and W John Wilbur. Evaluation of query expansion using MeSH in PubMed. *Information Retrieval*, 12(1):69–80, 2009. 1.2, 2.1, 3.2.2

[67] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL 2014) System Demonstrations*, pages 55–60. ACL, 2014. 6.1.4

[68] Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. DBpedia spotlight: Shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems*, pages 1–8. ACM, 2011. 2.3

[69] Donald Metzler and W Bruce Croft. A Markov random field model for term dependencies. In *Proceedings of the 28th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2005)*, pages 472–479. ACM, 2005. 3.1.3

[70] Donald Metzler and W Bruce Croft. Latent concept expansion using Markov random fields. In *Proceedings of the 30th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2007)*, pages 311–318. ACM, 2007. 3.1.1, 3.2.2

[71] Donald Metzler and W Bruce Croft. Linear feature-based models for information retrieval. *Information Retrieval*, 10(3):257–274, 2007. 5.1.3

[72] Rada Mihalcea and Andras Csomai. Wikify!: Linking documents to encyclopedic knowledge. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management (CIKM 2007)*, pages 233–242. ACM, 2007. 2.3

[73] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 27th Advances in Neural Information Processing Systems 2013 (NIPS 2013)*, pages 3111–3119, 2013. 4.2.3.1, 4.2.5.4, 5.1.3, 6.1.4

[74] Robert Neumayer, Krisztian Balog, and Kjetil Nørvåg. When simple is (more than) good enough: Effective semantic search with (almost) no semantics. In *Proceedings of the 34th European Conference on Information Retrieval (ECIR 2012)*, pages 540–543. Springer, 2012. 3.3.1

[75] Dong Nguyen and Jamie Calan. Combination of evidence for effective web search. In *Proceedings of The 19th Text Retrieval Conference (TREC 2010)*. NIST, 2010. 3.1.1, 3.1.2.1, 3.1.3, 3.1.3

[76] Fedor Nikolaev, Alexander Kotov, and Nikita Zhiltsov. Parameterized fielded term dependence models for ad-hoc entity retrieval from knowledge graph. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR 2016)*, pages 435–444. ACM, 2016. 2.3

[77] Dazhao Pan, Peng Zhang, Jingfei Li, Dawei Song, Ji-Rong Wen, Yuexian Hou, Bin Hu, Yuan Jia, and Anne De Roeck. Using Dempster-Shafer's evidence theory for query expansion based on freebase knowledge. In *Information Retrieval Technology*, pages 121–132. Springer, 2013. 3.2.1.2

[78] TB Rajashekar and Bruce W Croft. Combining automatic and manual index representations in probabilistic retrieval. *Journal of the American Society for Information Science*, 46(4):272–283, 1995. 2.1

[79] Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. Local and global algorithms for disambiguation to Wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT 2011)*, pages 1375–1384. ACL, 2011. 2.4

[80] Hadas Raviv, Oren Kurland, and David Carmel. Document retrieval using entity-based language models. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2016)*, pages 65–74. ACM, 2016. 2.4, 4.2.3.2, 4.2.3.2, 5.1.1.2, 5.1.1.3, 5.1.3, 6.1.1, 6.1.4, 6.2.2

[81] Stephen E Robertson and Steve Walker. Okapi/Keenbow at TREC-8. In *Proceedings of*

The 8th Text Retrieval Conference (TREC 1999), pages 151–162. NIST, 1999. 3.1.1

[82] François Rousseau and Michalis Vazirgiannis. Graph-of-word and TW-IDF: New approach to ad hoc IR. In *Proceedings of the 22nd ACM International Conference on Conference on Information & Knowledge Management (CIKM 2013)*, pages 59–68. ACM, 2013. 6.1.1

[83] Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1986. 1.2, 4.1, 4.1.1

[84] Evan Sandhaus. The New York Times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752, 2008. 6.1.1, 6.1.4

[85] Michael Schuhmacher, Laura Dietz, and Simone Paolo Ponzetto. Ranking entities for web queries through text and knowledge. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM 2015)*, pages 1461–1470. ACM, 2015. 3.3.1

[86] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-june Paul Hsu, and Kuansan Wang. An overview of Microsoft academic service (MAS) and applications. In *Proceedings of the 24th International Conference on World Wide Web (WWW 2015)*, pages 243–246. ACM, 2015. 4.2.1

[87] Nicola Stokes, Yi Li, Lawrence Cavedon, and Justin Zobel. Exploring criteria for successful query expansion in the genomic domain. *Information Retrieval*, 12(1):17–50, 2009. 2.1

[88] Trevor Strohman, Donald Metzler, Howard Turtle, and W Bruce Croft. Indri: A language model-based search engine for complex queries. In *Proceedings of the International Conference on Intelligent Analysis*, volume 2, pages 2–6. Central Intelligence for Analysis and Production, 2005. 3.1.3

[89] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web (WWW 2007)*, pages 697–706. ACM, 2007. 2.2

[90] Ilya Sutskever, James Martens, George E Dahl, and Geoffrey E Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 29th International Conference on Machine Learning (ICML 2013)*, pages 1139–1147, 2013. 5.1.3

[91] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. ArnetMiner: Extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2008)*, pages 990–998. ACM, 2008. 4.2.1

[92] Tao Tao and ChengXiang Zhai. Regularized estimation of mixture models for robust pseudo-relevance feedback. In *Proceedings of the 29th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2006)*, pages 162–169. ACM, 2006. 3.1.1, 3.1.3

[93] Ivan Vulić and Marie-Francine Moens. Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings. In *Proceedings of the 38th Inter-*

national ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2015), pages 363–372. ACM, 2015. 4.2.1

[94] Alex D Wade, Kuansan Wang, Yizhou Sun, and Antonio Gulli. WSDM Cup 2016: Entity ranking challenge. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining (WSDM 2016)*, pages 593–594. ACM, 2016. 4.2.1

[95] Xing Wei and W Bruce Croft. LDA-based document models for ad-hoc retrieval. In *Proceedings of the 29th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2006)*, pages 178–185. ACM, 2006. 4.2.1

[96] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: Theory and algorithm. In *Proceedings of the 25th International Conference on Machine Learning (ICML 2008)*, pages 1192–1199. ACM, 2008. 3.2.1.1, 3.2.2

[97] Chenyan Xiong and Jamie Callan. EsdRank: Connecting query and documents through external semi-structured data. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management (CIKM 2015)*, pages 951–960. ACM, 2015. 4

[98] Chenyan Xiong and Jamie Callan. Query expansion with Freebase. In *Proceedings of the Fifth ACM International Conference on the Theory of Information Retrieval (ICTIR 2015)*, pages 111–120. ACM, 2015. 1

[99] Chenyan Xiong, Jamie Callan, and Tie-Yan Liu. Bag-of-entities representation for ranking. In *Proceedings of the sixth ACM International Conference on the Theory of Information Retrieval (ICTIR 2016)*, pages 181–184. ACM, 2016. 1

[100] Chenyan Xiong, Jamie Callan, and Tie-Yan Liu. Word-entity duet representations for document ranking. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2017)*, pages 763–772. ACM, 2017. 1

[101] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2017)*, pages 55–64. ACM, 2017. 6, 2, 6.1.2.3, 6.1.4, 6.1.4, 6.1.5.1

[102] Chenyan Xiong, Zhengzhong Liu, Jamie Callan, and Eduard H. Hovy. JointSem: Combining query entity linking and entity based document ranking. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM 2017)*, pages 2391–2394. ACM, 2017. 6

[103] Chenyan Xiong, Russell Power, and Jamie Callan. Explicit semantic ranking for academic search via knowledge graph embedding. In *Proceedings of the 25th International Conference on World Wide Web (WWW 2017)*, pages 1271–1279. ACM, 2017. 4

[104] Chenyan Xiong, Zhengzhong Liu, Jamie Callan, and Tie-Yan Liu. Towards better text understanding and retrieval through kernel entity salience modeling. In *Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2018)*, page To Appear. ACM, 2018. 1

[105] Yang Xu, Gareth JF Jones, and Bin Wang. Query dependent pseudo-relevance feedback based on Wikipedia. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2009)*, pages 59–66. ACM, 2009. 3.1, 3.1.1, 3.1.2.1, 3.1.3, 3.2, 3.2.1, 3.2.1.2, 3.2.2, 5.1.1.3

[106] Yi Yang and Ming-Wei Chang. S-MART: Novel tree-based structured learning algorithms applied to tweet entity linking. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL2015)*, pages 504–513. ACL, 2015. 2.3

[107] Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL 2015)*, pages 1321–1331. ACL, 2015. 3.3.1

[108] Chengxiang Zhai and John Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of the 10th ACM Conference on Information and Knowledge Management (CIKM 2001)*, pages 403–410. ACM, 2001. 3.1.1

[109] Jing Zhang, Jie Tang, and Juanzi Li. Expert finding in a social network. In *International Conference on Database Systems for Advanced Applications*, pages 1066–1069. Springer, 2007. 4.2.1

[110] Le Zhao and Jamie Callan. Term necessity prediction. In *Proceedings of the 19th ACM International on Conference on Information and Knowledge Management (CIKM 2010)*, pages 259–268, 2010. 6.1.1

[111] Guoqing Zheng and Jamie Callan. Learning to reweight terms with distributed representations. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2015)*, pages 575–584. ACM, 2015. 6.1.1

[112] Nikita Zhiltsov, Alexander Kotov, and Fedor Nikolaev. Fielded sequential dependence model for ad-hoc entity retrieval in the web of data. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2015)*, pages 253–262. ACM, 2015. 2.3, 3.3, 3.3.1, 3.3.2, 3.3.3