Leveraging Eventuality-Centric Implicit Knowledge for Natural Language Understanding

Naoki Otani

CMU-LTI-23-001

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15123
www.lti.cs.cmu.edu

Thesis Committee:

Eduard Hovy **(Chair)**	Carnegie Mellon University & University of Melbourne
Lori Levin	Carnegie Mellon University
Yonatan Bisk	Carnegie Mellon University
Benjamin Van Durme	Johns Hopkins University & Microsoft Semantic Machines

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

Keywords: Natural language understanding, Implicit knowledge, Eventuality, Sentiment analysis, Representation learning, Response selection and generation, Procedural text understanding

To my wife, Yidi, and my daughter, Lina.

Abstract

Data-driven techniques have significantly advanced the state-of-the-art in a wide range of natural language processing (NLP) tasks. However, their primary strength lies in pattern discovery within observed signals. As a result, these techniques often struggle to capture implicit knowledge–information that is implicitly assumed in human communication and not explicitly present in textual data. Human communication is full of semantic gaps due to reliance on such implicit knowledge. Although the importance of implicit knowledge has been widely recognized in the research community, it still remains challenging to represent implicit knowledge for target applications, align it with language expressions, and incorporate it into NLP systems for high-level semantic analysis.

To achieve accurate and robust language understanding, this thesis proposes a series of techniques to identify and operationalize implicit knowledge, demonstrating that eventuality-centric knowledge is essential for understanding language phenomena across various real-world NLP application scenarios. The first part of the thesis focuses on the use of psychological and social implicit knowledge, showing that human basic motives underlie the sentiment expressed in customer reviews and that knowledge about human activities enhances computational representations of user-generated To-Do task descriptions. The thesis then argues that open-domain conversation systems necessitate an understanding of implicit situational context, encompassing psychological, social, and physical aspects. Empirical analyses reveal that implicit information can benefit NLP systems, but at the same time introduces the challenge of filtering out irrelevant information, for which potential solutions are explored. Finally, the thesis discusses implicit knowledge of the physical world, proposing frame-based representations of cooking recipes to capture unspoken knowledge about physical states, actions, and effects. Through these studies, this thesis introduces novel methods for defining tasks to achieve end goals, representing and acquiring implicit knowledge, and incorporating it into computational models as input signals and predictive targets.

Acknowledgments

I would like to express my gratitude to my advisor, Eduard Hovy, for his continuous guidance throughout my Ph.D. study. He taught me how to address significant research problems and create value for society. His kindness and forward-thinking nature have always been inspiring, and every duscussion with him encouraged me to take the next steps. I was very fortunate to advised by him to complete my Ph.D. study. Additionally, I would like to thank Lori Levin for her unwavering support. I have learned a great deal from her, particularly how linguistics can inform language technologies. I am also grateful to Yonatan Bisk and Benjamin Van Durme for being a member of my thesis committee and providing constructive feedback. Many of the additional experiments and discussions in this thesis are owed to their suggestions.

I would like to thank my collaborators and mentors. Specifically, the work presented in Chapters 5, 6, and 7 was conducted in collaboration with Jun Araki and HyeongSik Kim. Chapter 3 is based on my internship project at Microsoft Research Redmond with my great mentors Michael Gamon, Oriana Riva, Sujay Kumar Jauhar, Mei Yang, and Sri Malireddi. I have also been fortunate to work with many other exceptional collaborators, including Taiqi He, Leonie Weissweiler, Yue Yin, Satoru Ozaki, Xingyuan Zhao, Lily Li, Micaelah St Johns, and Florian Pfisterer.

Language Technologies Institute has been an excellent place for me to interact with many remarkable people, such as David R. Mortensen, Graham Neubig, Shinji Watanabe, Teruko Mitamura, Hiroaki Hayashi, Kimihiro Hasegawa, Dongyeop Kang, Brendon Boldt (and his family), and Aldrian Obaja Muis. I would especially like to thank my office mates, Billy Li and Xinjian Li, and LLab members for making it enjoyable and stimulating to study at LTI. I also want to express my appreciation for the administrative staff, who have consistently assisted me in a prompt and kind manner.

Over the past six years, my life in Pittsburgh has been enriched by the support of my wonderful friends. In particular, I would like to thank my first roommate, Hikaru Mamiya, for helping me adjust to life here in various ways. Without him, I would not have delved so deeply into Pittsburgh's craft beer culture, and my liver would have been intact.

I would like to express my infinite gratitude to my parents, Masahiro and Emiko, for allowing me to pursue my dream, and to my parents-in-law, Yingjun Zhao and Ji Liu, for always supporting my wife and me. My daughter, Lina, has been one of my greatest sources of energy since her birth in 2022. Lastly, I would like to express my deepest gratitude and love to my wife, Yidi, for her constant support. This Ph.D. thesis would not have been possible without you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

It has been a common perception in the natural language processing (NLP) community that the performance bottleneck of state-of-the-art NLP systems is, at least in part, due to a lack of comprehension of implicit knowledge. This thesis addresses the issue of identifying and operationalizing implicit knowledge in order to better understand and produce natural language.

Human communication is grounded in a variety of surrounding information, including physical and psychological states. In many situations, speakers rely on shared context and knowledge to convey meaning rather than explicitly detailing every aspect of their communicative intent. To fully understand human language, it is crucial to comprehend implicitly assumed facts. For example, consider the following interaction between a traveler and a station attendant at an information booth in a train station:[1]

(1) *Traveler*: Do you know when the Windsor train leaves?

(2) *Attendant*: *Yes.

While a simple "Yes" (2) may be a correct response to the question (1), a more appropriate response would include information about the time and location of the train to assist the traveler.

(3) *Attendant*: It's 3:15 at gate 7.

This is because certain facts, such as the understanding that people go to train stations to take trains and that being at the departure location at the departure time is necessary for doing so, are implicitly understood by both. Typically, people do not explicitly mention these pieces of knowledge to avoid redundancy, particularly if the information is already known or can be observed by the conversation participants. Similarly, customer reviews of restaurants, products, and other entities often merely describe what happened without explicitly detailing the writer's emotions or sentiments:

(4) *Restaurant guest*: Came for a lunch. It took 50 minutes for our food to all come out.

[1] An example from (Allen and Perrault, 1980)

Although the writer did not explain their mental states, we can infer that they wanted to have lunch without waiting too long, as this is a common basic motive shared by many people. Furthermore, when writing instructions such as cooking recipes, people often omit details even if they are crucial:

(5) Heat the pan on the stove and add butter.

This sentence assumes that the reader will understand that the butter is to be moved onto the pan, and that it will melt on the pan. Humans are able to obtain, infer, and use such implicit information in interactions to fill in semantic gaps. A lack of such knowledge can lead to errors and, at times, socially inappropriate output.

The significance of implicit knowledge has been widely recognized in the research community, and in the past few decades, numerous machine-readable data repositories of world knowledge have been created (Lenat, 1995; Speer et al., 2017; Tandon et al., 2014). These resources has facilitated knowledge-based NLP techniques that can produce relevant, informative language to the input (Yu et al., 2022). Knowledge-based methods have also been proven effective for detecting topical proximity between input and output candidates in tasks like question answering and information retrieval (Lan et al., 2021). Moreover, recent studies have shown that pre-training on a large-scale corpus enables language models to store various types of linguistic and world knowledge and utilize it in response to input texts (Brown et al., 2020; Chi et al., 2020; Petroni et al., 2019; Roberts et al., 2020).

Despite the advancements in knowledge-based techniques, it still remains challenging to build NLP systems that can perform complex language tasks in a robust, controllable and interpretable manner. Even advanced data-driven models can make simple mistakes and be easily misled by irrelevant noise in input (Pandia and Ettinger, 2021). In addition, it is not easy to induce required knowledge stored within a model for a target application while avoiding the generation of untruthful information (Lin et al., 2022). The blackbox nature of recent mainstream neural network models also makes it difficult to identify and rectify errors. These challenges impede the use of NLP techniques in potential real-world applications, and researchers are actively working to find solutions.

To achieve optimal results in applications, it is vital to ask the right question and provide sufficient information to systems. This is challenging as the interpretation and production of language often depend on various types of unmentioned knowledge, and the range of implicit knowledge is extensive, making it difficult to identify and define required knowledge. Different NLP tasks and domains require varying types of knowledge. For example, knowing that "an elephant cannot fit through the doorway in typical houses."[2] may be useful in some situations but not in others, such as sentiment analysis. In contrast, understanding underlying intent would be beneficial in comprehending human sentiment and could improve the performance and in-

[2]An example from the DARPA Machine Common Sense project (https://www.darpa.mil/program/machine-common-sense)

terpretability of automatic sentiment analysis. Thus, we need to address the following research questions: How can we define a target task? What type of knowledge is necessary? How can we represent it, and how can we inject it into systems?

This thesis explores several approaches to appropriately define tasks for target NLP applications, represent and acquire implicit knowledge, and equip computational models with knowledge. Specifically, this work focuses on the implicit knowledge centered on psychological and physical eventualities, which has been considered to be a fundamental component for natural language understanding (Rieger, 1976; Schank and Abelson, 1977; *inter alia*). The thesis presents a series of empirical analyses that demonstrate the potential of incorporating eventuality-centric implicit knowledge as training signals and predictive objectives to enhance NLP systems in real-world applications.

Chapter 2 provides an overview of the key theoretical and technical topics related to the presented work. The following chapters delve into the different aspects of implicit knowledge. Chapter 3 proposes a task of human motive detection, which aims to understand the reasons behind sentiments in customer reviews. This study demonstrates that connections between human motives and eventualities can be identified computationally and used for detailed sentiment analysis. Chapter 4 addresses the challenge of representation learning for under-specified to-do descriptions, proposing a multi-task learning approach that enables text encoders to associate short texts with eventuality knowledge. Chapters 5 and 6 discuss the importance of situational context in open-domain conversations. The empirical study shows that conversational systems can benefit from the understanding of implicit context, but irrelevant contextual information can also influence their behavior, indicating the need for filtering steps. Chapter 7 focuses on physical knowledge, which is crucial for reasoning about the physical world. This chapter highlights the challenges associated with procedural texts and proposes a novel computational representation that overcomes the limitations of prior work. The outcome of this study facilitates more accurate analysis of implicit knowledge with broader coverage. Overall, this thesis demonstrates the benefits of incorporating eventuality-centric implicit knowledge to improve the performance and interpretability of systems in various NLP applications.

Thesis Statement

The comprehension of implicit knowledge centered on psychological and physical eventuality is beneficial in NLP tasks that necessitate a thorough understanding of human activities and procedures from both semantic and pragmatic perspectives. This type of knowledge can be utilized to enhance the performance of machine learning models by incorporating it as training signals, constraints, and predictive targets in an interpretable manner.

Although it is necessary for language processing systems to possess knowledge, pre-trained language models consisting of billions of parameters can perform knowledge-intensive tasks to some degree even without external knowledge, suggesting these models already acquire required knowledge through pre-training. This thesis demonstrates that implicit knowledge inside the models can be induced for the application goal by appropriately defining tasks and providing external signals. Nevertheless, due to the opaque nature of these complex models, their internal logic is not clear, such as whether they ground language in world knowledge or simply reuse patterns found in training data. Furthermore, it remains unclear when and how these models use implicit knowledge to process language. To advance the state-of-the-art in language technologies, future research should establish systematic methods to evaluate the behavior of pre-trained language models.

Contributions

This thesis addresses the problem of implicit knowledge acquisition and reasoning for NLP with the following key contributions:

1. Identify the benefit of the implicit knowledge centered on psychological and physical eventualities in multiple real-world NLP tasks that require deep semantic analysis.
2. Provide implications for what kind of knowledge should be represented and acquired to advance the state-of-the-art of language technologies.
3. Introduce three new tasks along with manually-curated datasets to develop and evaluate knowledge-driven systems.
4. Propose techniques to improve system performance by incorporating implicit knowledge as training signals and predictive targets.
5. Perform empirical analysis of state-of-the-art models from the viewpoint of psychological and physical eventuality-centric knowledge.

Chapter 2

Background

This thesis is focused on knowledge that is required for understanding and generating natural language and can be represented symbolically in a textual format. This chapter discusses its key aspects, provides an overview of prior research, and contextualizes the chapters of this thesis within a broader context.

2.1 Facets of Knowledge

Knowledge can be categorized based on various factors such as the subject matter, universality, and prevalence (as illustrated in Figure 2.1). Implicit knowledge, the main focus of this thesis, manifests in various subjects and levels of universality and is closely linked to the prevalence of knowledge.

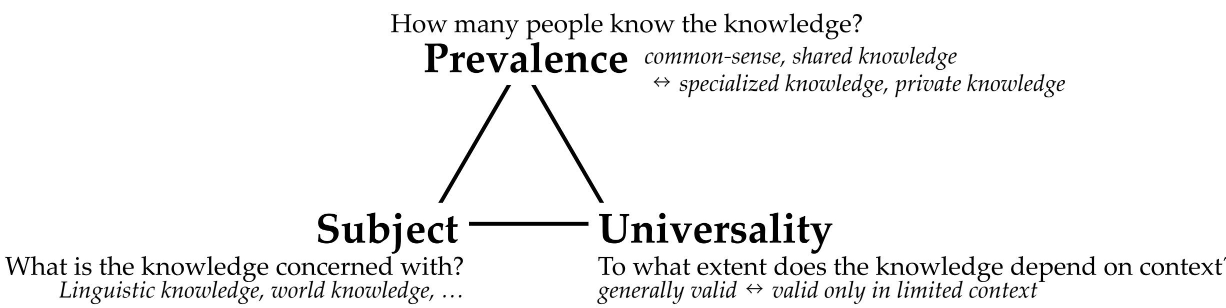


Figure 2.1: Three facets of knowledge discussed in this chapter: subject, universality, and prevalence.

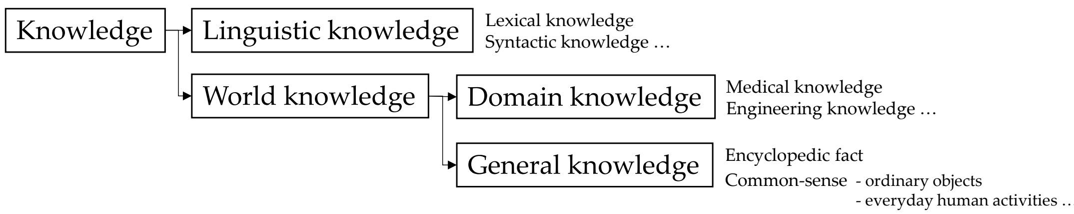


Figure 2.2: Categorization of knowledge based on subject.

2.1.1 Subject

Figure 2.2 depicts the categorization of knowledge based on subject matter. The top-level distinction can be made between linguistic knowledge and world knowledge. Linguistic knowledge is concerned with lexicon and grammar and is crucial for comprehending the literal meaning of language and generating well-formed language. Recent studies have shown that deep neural networks can capture and retain some degree of linguistic knowledge such as lexical semantics (Vulić et al., 2020), syntactic relations (Clark et al., 2019; Hewitt and Manning, 2019), and grammaticality (Linzen et al., 2016; Warstadt et al., 2020). However, possessing knowledge about lexicon and grammar does not necessarily imply that models can comprehend the meaning of language. A single language form can convey multiple distinct meanings, and the same meaning can be expressed in many different forms. The capacity of current state-of-the-art computational models to comprehend such form-meaning relationships is an active area of active exploration and not yet fully understood. Resolving this ambiguity often requires an understanding of the context or knowledge beyond just the surface forms. To illustrate this point, consider the following examples:

(6) Pat ate it with *a fork.*

(7) Pat ate it with *ketchup.*

(8) Pat ate it with *great relish.*

Interpreting the meaning of these sentences, despite sharing the same syntactic structure, requires relying on our world knowledge. Specifically, we must understand that "a fork" in the first sentence is a tool used for eating, "ketchup" in the third sentence is a condiment to be eaten together, and "great relish" in the second sentence expresses the manner of eating. Moreover, the communicative intention of language frequently depends on the understanding of the context, as exemplified by the various interpretations of the question "Can you drive a car?". This question could refer to the hearer's qualifications and willingness to drive the speaker somewhere, depending on the world state in which the conversation occurs.

World knowledge can be further divided into *domain knowledge* and *general knowledge*. Domain knowledge refers to specialized knowledge that individuals acquire through education or training in a particular subject area, such as medical science. This type of knowledge is typically shared only among a group of specialists who work in specific domains. In contrast, general knowledge is knowledge that is acquired by almost everyone in their daily lives, in many cases, unintentionally. For example, most people know the color of an apple from seeing it in places such as orchards, supermarkets, books, and television. The representation and acquisition of domain knowledge have significant practical implications (Roy and Pan, 2021; among others) as such knowledge plays an essential role in processing language in a specific application. On the other hand, general knowledge is important in many different domains and applications, spanning a wide range of categories and forms. Thus, incorporating general knowledge into computational models has been recognized as one of the major challenges in the research community (Jansen et al., 2016; Schubert, 2015; Weikum, 2017). In this thesis, we focus primarily on implicit knowledge under general knowledge. However, the methods proposed in this thesis will be applicable to implicit domain knowledge as well.

2.1.2 Universality

Some types of information are valid and shared only within a given context. I refer to this knowledge as "situational knowledge," which encompasses information that is dependent on a particular situation. For example, the condition of the following statement depends on the location and time in which it is spoken.

(9) It's a bit chilly today.

In other contexts, such as different cities or different dates, the statement may not hold true. Situational knowledge is acquired through interactions and observations and is usually not retained for an extended period. In contrast, certain statements, like the following example, are generally accepted as true by most individuals regardless of location or time:

(10) The outside temperature is low during winter.

People accumulate such knowledge (*universal knowledge*, henceforth) in their memory by experiencing and learning something. Effective semantic analyses of language require reasoning over both types of knowledge, situational and universal, to derive accurate meaning.

As the aforementioned example implies, universal knowledge can be acquired by finding general patterns in observations of the world. This idea has motivated researchers to develop resources of universal knowledge by aggregating observations from web texts and human annotators (Mitchell et al., 2015; Schubert, 2002; Speer et al., 2017; Tandon et al., 2014). These resources have facilitated NLP research that centers on universal knowledge. In contrast, situational knowledge has received less attention and is typically assumed to be implicitly provided within unstructured input texts, with some exceptions such as knowledge-grounded dialogue studies (Dinan et al., 2019; Moghe et al., 2018), where situational knowledge is allocated dedicated representations. Chapter 5 and Chapter 6 explore methods to connect situational and universal knowledge and demonstrate improved system performance in open-domain conversations.

2.1.3 Prevalence

The prevalence of knowledge—the degree of being shared by people is the most relevant dimension to this thesis. Information that is domain-specific or highly situational is generally shared among a limited number of people such as domain experts and conversation participants. In contrast, information that can be easily observed (by seeing, hearing, looking up, etc.) or can be obtained through daily activities is likely to be shared by a larger number of people. Commonsense, a type of knowledge that everyone knows, is recognized as a fundamental element of intelligent interactions and has attracted the attention of many researchers. (Ilievski et al., 2021; Lenat, 1995; Sap et al., 2019; Speer et al., 2017).

While the prevalence of knowledge is influenced by its subject matter and universality, domain-specific and situational knowledge can be considered common sense within certain groups of people and contexts. For instance, some cooking techniques, such as separating egg whites and yolks, constitute specialized knowledge but are widely known among those who frequently cook. It is crucial to note that common knowledge primarily pertains to stereotypical information about entities and events, which most people agree upon. Such knowledge is often *presupposed by certain language expressions and implicitly assumed* during communication to minimize redundancy (Grice, 1975; Levinson, 1983; Prince, 1978). This particular trait poses a distinct challenge for language technologies. This thesis will discuss methods for representing, acquiring, and reasoning over implicit knowledge to develop workable systems.

2.1.4 Eventuality-centric Knowledge

The main focus of this thesis is knowledge centered on eventualities (Bach, 1986), including the world state (e.g., "this room is hot") and dynamic change of states including unintentional process (e.g., "the snow on the street melted") and intentional action (e.g., "I cut carrots"). Eventualities have specifications such as participants, time, and location and are associated with each other semantically in temporal relations, causality, etc. The knowledge of such plays an important role in language understanding. Distinct from eventuality-centric knowledge is knowledge pertaining to objects and concepts. Examples of this type of knowledge include declarative statements, such as "beer is a type of alcoholic drink and can be purchased at liquor stores" and "Pittsburgh has many bridges." Although not the primary interest of this thesis, this type of knowledge is also crucial for many NLP tasks.

Numerous formal logic-based representations of events have been developed (Moore, 1984; *inter alia*), but it is often challenging to apply these approaches to real-world NLP problems that involve complex, open-ended language phenomena. In this thesis, we discuss representations that are more closely related to less formal data structures, such as a *semantic frame* and *script.* A semantic frame (Minsky, 1974) is a data structure that represents a stereotyped situation and its variable attributes. is a data structure that captures a stereotypical situation and its variable attributes. A collection of semantic frames is linked together to describe a scene from various viewpoints. The attributes (terminals) of each frame can be assigned default values if they are not explicitly declared:

> [...] if I say, "John kicked the ball", you probably cannot think of a purely abstract ball, but must imagine characteristics of a vaguely particular ball; it probably has a certain default size, default color, default weight. [...] In any case your image lacks the sharpness of presence because the processes that inspect and operate upon the weakly bound default features are very likely to change, adapt, or detach them. (Minsky, 1974)

This process of default assignments constitutes implicit knowledge. While semantic frames can represent a broad range of subjects not limited to eventualities, script knowledge (Schank and Abelson, 1977) is specifically focused on stereotypical event sequences that are associated with frequent scenarios and can be regarded as a subtype of semantic frame knowledge. Empirical studies by Bower et al. (1979) showed that people share some level of script knowledge. Later, researchers have shown that significant amounts of script knowledge can be extracted from large-scale web texts (Chambers and Jurafsky, 2008) and generated with machine learning models (Sakaguchi et al., 2021). Nonetheless, what is lacking is research on how to operationalize eventuality-centric knowledge in practical NLP tasks, particularly how to integrate such knowledge into cutting-edge data-driven NLP systems.

This thesis aims to discuss the relevance of eventuality-centric knowledge to real-world NLP tasks and show novel methods to represent, acquire, and incorporate it into NLP systems. The focus is on two main categories of events: psychological and physical. These categories will be discussed in more detail in the following sections.

2.1.4.1 Psychological Implicit Knowledge

Psychological knowledge pertains to *all of the aspects of the way that people think they think.* (Gordon and Hobbs, 2004) and includes both intra-personal (e.g. emotion, motive, and goal) and inter-personal ones (e.g. speech acts and social norms). Without a proper understanding of psychological knowledge, the behavior of NLP systems can be awkward, inefficient, and even impolite.

Research on dialogue systems is one area of NLP where psychological knowledge plays a crucial role. For instance, systems must capture emotions to engage in conversation and provide human-like responses (Rashkin et al., 2019), and desires/goals are essential to complete task-oriented dialogues, where users communicate with machines to carry out specific tasks, such as making a reservation (Budzianowski et al., 2018; Byrne et al., 2019). In a similar way, intelligent assistance technologies, such as information retrieval systems, must understand user intent to effectively satisfy their needs (Allen and Perrault, 1980). Basic human motives and needs have been studied outside the NLP community, and psychologists have devised various models (Reiss, 2004; Talevich et al., 2017). Maslow's hierarchy (Maslow, 1943) is arguably the most widely recognized theory in the NLP community, which conceptualizes a hierarchical view of needs that everyone seems to share. Some NLP studies discuss its implications for analyzing psychological factors in language expressions (Ding and Riloff, 2018; Li and Hovy, 2017; Rashkin et al., 2018). Recent studies have also constructed large-scale datasets for incorporating diverse types of psychological knowledge into machine learning systems (Asai et al., 2018; Chaturvedi et al., 2016; Goldberg et al., 2009; Rashkin et al., 2016; Sap et al., 2019).

While there are many practical NLP problems that could be better addressed by explicitly modeling psychological knowledge, machines' comprehension of this type of knowledge is still still far below a satisfactory level. Furthermore, different application areas are likely to require different types of psychological knowledge. This thesis discusses how psychological knowledge can be incorporated into NLP systems in three applications: human motives as predictive targets for sentiment analysis (Chapter 3), user intent as weak-supervision for to-do management assistance (Chapter 4).

2.1.4.2 Physical Implicit Knowledge

Physical knowledge plays a crucial role in the understanding of procedural texts, which are commonly used for conveying information about processes and actions in various domains, such as science (e.g., wet lab protocols), engineering (e.g., auto repair manuals), and daily human activities (e.g., cooking recipes). However, authors of such texts often assume that certain details are common-sense and avoid describing them. This makes it challenging for machines without the necessary knowledge to follow instructions and perform intelligent tasks.

To address this issue computationally, adequate and accurate semantic representations of physical phenomena must be defined. Formal physical-world representations, such as naïve physics (Hayes, 1985) and qualitative modeling (Forbus, 1984), are widely used in research on robotics and planning, where agents observe the world and perform actions via formal languages (Fikes and Nilsson, 1971; *inter alia*). However, the targets of these formal representations are typically too narrow to cover the long-tail language phenomena encountered in real-world NLP systems. Hence, recent applied NLP studies rely on less formal representations. Some studies capture procedural knowledge from web resources in a weakly canonicalized and structured textual form (Chu et al., 2017; Park and Motahari Nezhad, 2018; Zhang et al., 2012), which can be used for downstream applications such as how-to information retrieval (Yang and Nyberg, 2015) and automatic instruction generation (Chandu et al., 2019; Kiddon et al., 2016; Paris et al., 2002). Other studies proposed methods to parse instructions into meaning representations that organize predicates, arguments, and entity states structurally (Bosselut et al., 2018; Dalvi et al., 2018; Kulkarni et al., 2018; Mori et al., 2014; Tandon et al., 2020). Existing knowledge resources such as ConceptNet (Speer et al., 2017), WebChild (Tandon et al., 2014), and VerbNet (Schuler, 2005) provide some physical common-sense information, such as quality of objects and pre-conditions and effects of actions.

However, it is still relatively unexplored how to *leverage* implicit information in procedural texts. There is limited research on using physical implicit knowledge to parse real-world procedural texts at scale (Clark et al., 2018; Ribeiro et al., 2019; Tandon et al., 2018). Furthermore, the accuracy and coverage of physical knowledge depend heavily on how the task is framed and represented, which can consequently affect system performance. Chapter 7 of this thesis explores methods for representing and incorporating physical common-sense into machines to enhance procedural text understanding.

2.2 Representations of Language and Knowledge

This section provides an overview of the technical background relevant to the research presented in this thesis, specifically regarding how language and knowledge are represented computationally for NLP. In the early stages of AI and NLP research, the primary approach involved formal logic representations as we reviewed in the previous sections. Recent data-driven approaches typically represent the meaning of language using real-value vectors due to their easiness for optimization within a framework of machine learning. A large part of this thesis was conducted after 2019, when Transformer-based pre-trained language models were introduced. Below, I highlight some topics on pre-trained language models to provide technical backgrounds of this thesis.

2.2.1 Transformer-based Pre-trained Language Models (PLMs)

Pre-trained language models (PLMs) based on the Transformer architecture, a deep neural network introduced by (Vaswani et al., 2017), have demonstrated significant effectiveness in generating contextual representations of texts (Qiu et al., 2020), achieving improved performance across various NLP tasks. Generally, PLMs are built by optimizing parameters on large-scale general-domain corpora for language modeling, which involves predicting the next or masked token based on context. After pre-training, PLMs are fine-tuned with a task-specific module to maximize performance on the target task. This approach has become a common practice in NLP because of the ease of use and high representational power of the many publicly available pre-trained models.

Pre-training for representation learning was already recognized to be effective prior to the introduction of Transformer (Mikolov et al., 2013; Peters et al., 2018). However, compared with previous approaches, Transformer-based PLMs have considerably high model capacity with billions of parameters and are trained on gigabytes or even terabytes of text corpora. In addition, Transformer employs several advanced techniques such as the multi-head attention mechanism, which has proven effective in capturing contextual dependencies in language.

Examples: Ever since the first generation of Transformer-based PLMs, namely GPT (Radford et al., 2018) and BERT (Devlin et al., 2019), were presented, numerous extensions of these models have been developed for diverse languages, domains, and applications. GPT is optimized for the causal language modeling objective, which involves predicting the next token of an input sequence based on the past context, while BERT is optimized for predicting masked tokens and next sentences based on input token sequences (i.e., the masked language modeling objective). Notable among existing causal language models is GPT-3 (Brown et al., 2020), a PLM with 175 billion parameters trained on 45TB of text. It has been reported that GPT-3 can perform various tasks with a few-shot or even zero-shot manner. Masked language models have been extended with different learning objectives and data, resulting in improved models such as RoBERTa (Liu et al., 2019b) and DeBERTa (He et al., 2021a). In addition to causal and masked language models, pre-trained encoder-decoder models have also been developed, which are suitable for sequence-to-sequence NLP tasks such as machine translation and text summarization. Examples of such models include BART (Lewis et al., 2020a) and T5 (Raffel et al., 2020).

Prompt: Recent studies have revealed that task-specific instances can be converted into natural language expressions of instruction-answer pairs, and by training models on such data, PLMs can be trained to perform NLP tasks without task-specific fine-tuned modules (Ouyang et al., 2022; Raffel et al., 2020). This finding has led to a new technique used with PLMs, known as prompting. This approach provides a short prompt, optionally coupled with a few demonstrative examples, to PLMs to generate the desired task-specific output. The use of prompting with PLMs is gaining popularity as it is highly effective for a wide range of NLP tasks, including summarization, question answering, and text classification. (Liu et al., 2023).

Open challenges: (1) While PLMs are extremely effective at predicting/classifying textual language forms, the pre-training task does not involve symbol grounding and is unlikely to be sufficient for learning language meaning and for acting on language (Bender and Koller, 2020; Bisk et al., 2020). To address this problem, some sort of guidance is required to align form representation within PLMs and signals of meaning. Chapter 3 and 7 address this issue. (2) PLMs, particularly English models, are typically trained on well-formed sentences and are suboptimal for short or under-specified language. We review this problem in Chapter 4 and Chapter 7. (3) Large-scale PLMs are extremely capable of combining different pieces of information in input, but on the other hand, slight changes in input often lead to inaccurate output (Shi et al., 2023). Chapter 5 addresses this problem.

2.2.2 Knowledge in PLMs

It has been reported that corpora exhibit reporting bias (Gordon and Van Durme, 2013; Misra et al., 2016), wherein people tend to omit information that is too obvious for readers to deduce, while explicitly describing unlikely facts. This suggests that PLMs also reflect such biases in the data and may not learn universal or common knowledge comprehensively. However, recent

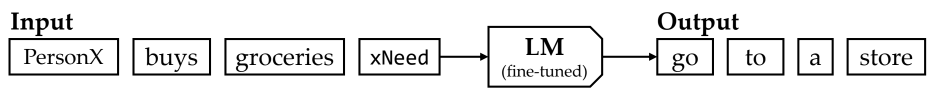


Figure 2.3: Illustration of COMET, a knowledge generation model. The LM component can be either a causal PLM or a pre-trained encoder-decoder model, which is fine-tuned to complete knowledge triples obtained from resources such as ATOMIC. Triples are converted (linearized) into a sequence of tokens that contain a special token representing the semantic relation of interest. In this figure, "xWant" is a relation type in ATOMIC that connects two consecutive events.

studies have shown that PLMs are capable of processing and generating implicit knowledge to some extent, without explicit training. The success of prompting indicates that PLMs already possess rich knowledge, although it may not necessarily be grounded in something external to language forms.

Knowledge Induction: A number of prior studies have demonstrated that various types of knowledge, including linguistic and general world knowledge, can be found in PLMs (Chi et al., 2020; Petroni et al., 2019; Roberts et al., 2020). This finding indicate that, given sufficiently large training corpora, even implicit knowledge can appear somewhere in the data and can be stored within the parameters of large-scale representation models. Furthermore, recent research has shown that fine-tuning PLMs for knowledge generation can significantly improve their ability to generate meaningful knowledge with higher accuracy and coverage. One such example is the Commonsense Transformers (COMET) model (Bosselut et al., 2019; Hwang et al., 2021; West et al., 2022), which is fine-tuned on linearized knowledge snippets to generate relevant concepts or events in response to an input prompt (Figure 2.3). Da et al. (2021) have demonstrated that even a few examples can substantially enhance the quality of the generated knowledge. This finding underscores the importance of providing appropriate signals to induce the knowledge that is already stored within the PLMs.

Knowledge Injection: While PLMs are capable of capturing various types of knowledge, their coverage is limited to the information present in their training corpora. Moreover, facts that are infrequent or require multi-hop reasoning are challenging to retrieve (Kandpal et al., 2022). To address these limitations, researchers have proposed a range of methods, including combining PLMs with information retrieval systems (Guu et al., 2020; Komeili et al., 2022; Lewis et al., 2020b), symbolic knowledge resources (Ma et al., 2021; Wang et al., 2021b; Yasunaga et al., 2021), and separate knowledge generation models (Bosselut et al., 2021; Liu et al., 2022b; Wang et al., 2020b; Zhou et al., 2022b). Knowledge can be represented in different forms, such as natural language expressions, knowledge graphs, and triples (Bauer and Bansal, 2021). Knowledge respresented in plain texts are often employed with prompting approaches (Liu et al., 2022b; Zhou et al., 2022b).

Open Challenges: (1) It is not easy to control the behavior of PLMs to ground them on specific types of knowledge, particularly when the target involves complex semantics such as psychological and physical reasoning. To mitigate this problem, tasks need to be designed carefully, with a clear definition of input and output requirements. We revisit this aspect in Chapter 3 and 6. (2) While the language generated by PLMs can be semantically rich and fluent, it is not always truthful (Lin et al., 2022). As discussed in Chapter 5, this is due, at least in part, to the underspecification of the problem space. When input information is insufficient, PLMs may resort to fabricating information.

Chapter 3

Human Motive Detection

Overview

In this chapter,[1] we delve into the task of sentiment analysis and discuss how the knowledge of underlying *human motives* can provide useful insights into the problem. Prior studies on sentiment analysis have primarily focused on identifying holders, facets, and valences of sentiment, but have overlooked *the reasons* behind sentiment decisions. Understanding the causes of sentiment has significant implications for product and service development. For instance, knowing that many customers are unsatisfied with the weight of their laptops is more informative for laptop manufacturers than simply observing numerical ratings that quantify their overall sentiment. In the work presented in this chapter, we consider human motives as the driving force behind sentiment and address the problem of motive detection as the first step. To characterize these underlying motives, we drew from psychological studies and defined six fundamental motive categories that encompass a broad range of topics found in review texts. We annotated 1,600 texts in the restaurant and laptop domains with these motives and evaluated data-driven classification methods on this new dataset. Our results demonstrate that cross-domain transfer learning boosts motive detection performance, indicating that these universal motives exist across various domains.

3.1 Introduction

Understanding the sentiment of a person based on text has practical implications for improving the quality of products and services, as well as scientific implications for psychology and other fields. Despite a rich body of sentiment analysis research, sentiment is often simply assumed to be expressed through uni-dimensional binary or ternary labels (positive, neutral, and negative). Relatively little attention has been paid to the reasons for holding a particular sentiment value.

[1] A large part of this chapter has been published in the proceedings of ACL 2019 (Otani and Hovy, 2019).

Figure 3.1: Restaurant review texts and human motives of interest (rectangles).

Aspect-based sentiment analysis (ABSA), which considers fine-grained categories (*aspects*) that may cause sentiment, partially addresses this problem. However, aspects are typically limited to properties of entities such as the price of food and design of a product (e.g., (Pontiki et al., 2016)). They do not really show *why* such aspects matter and *how* they cause sentiment. For example, some people desire cheap and quick meals to save time and money, while others desire high-quality food to enjoy the dining experience itself.

In this study, we follow Li and Hovy (2017) and view sentiment as a realization of an individual's mental state that relates to their satisfaction toward a specific event or entity. Sentiment can be driven by a sentiment holder's emotional and non-logical preference (such as "I just don't enjoy that kind of food") and can also be conditioned by long-term plans and resources that the holder has. However sentiment is largely triggered by whether one of the holder's goals is satisfied or not. As illustrated in Figure 3.1, one will have negative sentiment toward a restaurant if the service is terrible because one's basic motive for social behavior is not met.

The exploration of fundamental motives shared by humans has been the focus of research for decades in areas such as psychology (Maslow, 1943; Reiss, 2004). Talevich et al. (2017) recently proposed a taxonomy of motives based on a meta-review of prior work and human subject research. Their taxonomy includes Self-fulfillment, Appreciating Beauty, Social Relation, Health, Ambition&Ability, and Finance. We employ their definitions of motives for analyzing sentiment (§3.2).

Our work is in line with studies that aim to identify relevant motives in texts, aiming to enable machines to identify a more complete description of a situation and explain human decisions and actions. For instance, Ding and Riloff (2018) extracted predicate-argument tuples from web texts and classified them into motive categories. Rashkin et al. (2018) analyzed narratives authored by crowd workers with motive and emotion labels. In contrast, our work analyzes user-generated review texts at a sentence level.

As a first step towards deep understanding of sentiment, we conduct a task of human mo-

Self-fulfillment (Sf)	Finding meaning in life. Feeling satisfied with one's life.	"Ess-A-Bagel is by far the best bagel in NY."
*Embrace &Explore Life (Ee)	Being entertained. Exploring a new thing.	"The wine list is extensive."
Appreciating Beauty (AB)	Enjoying fine design/natural beauty. Being creative.	"A beautifully designed dreamy restaurant."
Social Relation (Sr)	Being treated well by others. Belonging to a social group.	"Everyone was cheerfully cooperative."
Health (H)	Being physically healthy.	"The fish was not fresh and the rice tasted old."
Ambition&Ability (Aa)	Being competent/knowledgeable. Keeping things in order. Being efficient.	"I've waited over one hour for food."
Finance (F)	Saving money Getting things worth the financial cost.	"The prices are high, but I felt it was worth it."

Table 3.1: Motive categories, definitions and examples sentences. *Embrace&Explore Life is merged to Self-fulfillment (Section 3.2).

tive detection. We manually annotated 1,600 review texts in restaurant and laptop domains from the SemEval ABSA datasets (Pontiki et al., 2016) with the six motives (§3.3). The annotation results reveal that sentiment is driven by different motives in different domains. We report the performance of machine learning models on this new dataset (§3.4). Based on research on human motivation, we hypothesize that underlying drivers of human behavior are universal across domains, although the distributions can vary. With this assumption, we leverage out-of-domain data to improve a human motive detector in the target domain. Our experiment shows that transfer learning across the restaurant and laptop domains is effective in motive detection. The best-performing model achieves an F1 score of 0.7, but there is still a gap between the system and human-level performance. Finally, we discuss how human motives can provide implications for sentiment analysis (§3.5).

3.2 Representation of Human Motives

Our aim is to anallyze sentiment using human motives. To this end, we require a taxonomy of human motives. Motives are defined as reasons people hold for initiating and performing voluntary behavior (Reiss, 2004). A study of human motives dates back to Aristotle (384—322BC), who

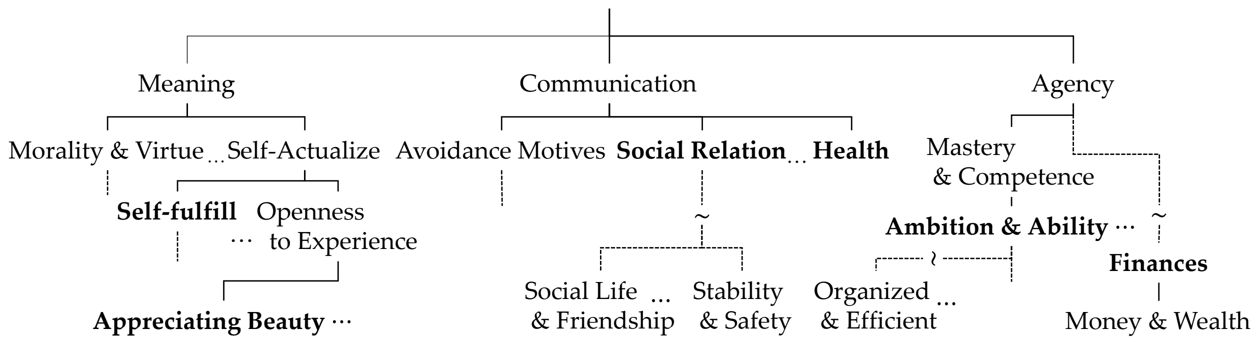


Figure 3.2: Illutration of the motive taxonomy of Talevich et al. (2017).

proposed a distinction between ends and means.[2] Means motives are desires for other motives, while end motives are ultimate goals of individuals. For example, eating fresh vegetables can be a mean of an end motive, becoming healthy. Ends, for which there are several theories, are believed to be a closed class (e.g. (Maslow, 1943; Reiss, 2004)).

The aforementioned motives are drawn from a taxonomy of 161 motives (Talevich et al., 2017). Talevich et al. derived basic motives based on an extensive literature survey and grouped them hierarchically based on similarity judgments collected from human subjects. The hierarchical structure of their taxonomy, as illustrated in Figure 3.2, embodies conceptual relationships between motives. Higher-level motives in the hierarchy are more abstract, and lower-level motives are more concrete. The motives we picked are intermediate categories in the taxonomy that cover a wide range of topics appearing in our review texts (Table 3.1). These intermediate categories represent 55% of the 161 bottom-level motives.

3.3 Annotation of Human Motives

To annotate review texts with human motives, we used Amazon Mechanical Turk (MTurk) and assigned three crowd annotators to each sentence. We then aggregated their responses to obtain the final results, where a single sentence may include multiple labels.

Data: We collected restaurant and laptop review sentences from the SemEval 2016 datasets (Pontiki et al., 2016). We discarded sentences with longer than 25 tokens,[3] and sample 800 sentences from each domain.

[2]In his book *"Nicomachean Ethics"*

[3]We use Stanford CoreNLP v.3.9.2 (Manning et al., 2014) to tokenize sentences.

In product reviews, people describe various positive/negative opinions. We are interested in what kind of human goals is behind those opinions.

Please select ALL human goals related to a given laptop computer review based on the following guideline.

Goal	Guidance
Finding meaning in life or feeling happy with one's life	Select this if the item embodies emotion/opinion related to a reviewer's desire for self-fulfillment. For example, "Having my own MacBook was one of my dreams!"
Appreciating fine design/arts/natural beauty or being creative	Select this if the item embodies emotion/opinion related to a reviewer's desire for enjoying the beauty. For example, "The design is awesome"
Exploring and enjoying life	Select this if the item embodies emotion/opinion related to a reviewer's desire for exploration or enjoyment. For example, "The HD display is sensational, and now watching movies is a very enjoyable experience for me."
Being treated well by others, being attractive, or belonging to a social group	Select this if the item embodies emotion/opinion related to a reviewer's desire for social relations. For example, "Their customer service is friendly, and I never had a problem."
Being physically healthy	Select this if the item embodies emotion/opinion related to a reviewer's desire for health. For example, "This large HD monitor helps to extend my screen and keep my eyes fresh!", "It is not ideal for children because of the temp."
Being competent/knowledgeable, keeping things in order, or being efficient	Select this if the item embodies emotion/opinion related to a reviewer's desire for competence or efficiency. For example, "This laptop is so slow and I hate it!.", "It's very user-friendly.", "It has a 5-6 hour battery life."
Saving money or getting things worth the financial cost	Select this if the item embodies emotion/opinion related to a reviewer's desire for money. For example, "I spent a lot of money on this product and it's been a nightmare."

One text is likely to be linked to one, two, or three goals. **If none of the listed goals fits the text, select NONE.** (See examples below.)

Correct Example 1:

This computer way overpriced.
→ "Saving money or getting things worth the financial cost "
The reviewer says this because getting a prodct worth the cost is one of his goals.

Correct Example 2: - One text may be related to two or more goals. Choose ALL goals that are applicable.

MacBook is always good-looking and fast.
→ "Appreciating fine design/arts/natural beauty or being creative" and "Being competent/knowledgeable, keeping things in order, or being efficient"

Both of nice design (good-looking) and efficiency (fast) are important for the reviewer.

Incorrect Example:
I'm so happy with this computer. It's absolutely AMAZING!!
→ "Finding meaning in life or feeling happy with one's life" (incorrect)
→ "NONE" (correct)
The reviewer expresses his/her emotion, but we cannot tell the reason from this sentence. So, choose "NONE" in this case.

Why did a reviewer say this?

Has a 5-6 hour battery life.

Because his/her goal was _____. (Please select ALL goals related to the text.)

- ☐ Finding meaning in life or feeling happy with his/her life
- ☐ Appreciating fine design/arts/natural beauty or being creative
- ☐ Exploring and enjoying life
- ☐ Being treated well by others, being attractive, or belonging to a social group
- ☐ Being physically healthy
- ☐ Being competent/knowledgeable, keeping things in order, or being efficient
- ☐ Saving money or getting things worth the financial cost
- ☐ NONE

Figure 3.3: MTurk interface for the annotation task of human motives

	SF	AB	SR	H	AA	F
Restaurant	348	79	137	31	95	109
Laptop	188	164	52	9	370	145

Table 3.2: Distribution of human motives. SF: Self-fulfillment, AB: Appreciating Beauty, SR: Social Relation, H: Health, AA: Ambition & Ability, F: Finance.

Annotation: We recruited crowd workers who reside in the US and completed 1,000 or more HITs at ≥ 95 approval rate with reward of $0.1 per sentence.[4] We presented the definitions of human motives along with a few examples to crowd workers and instructed them to assign all applicable motive labels to an input sentence. Figure 3.3 shows the annotation interface.

Quality Control: We first collect annotations on 200 sentences in each domain without any filtering of workers. We then evaluate the workers on the 400 sentences: one of the authors examine the responses and made the gold-standard label set, and we calculate the F1-score of each worker against the gold-standard. We only use the workers whose scores are ≥ 0.5 in the remaining annotation tasks.

Annotation Agreement: Our crowd workers agreed moderately on annotations: Krippendorff's α was 0.48 and 0.59 in the restaurant and laptop domains, respectively. We found that SELF-FULFILLMENT and EMBRACE & EXPLORE LIFE are often hard to distinguish. We, therefore, collapsed these categories, and Krippendorff's α increased to 0.51 and 0.61. For reference, three graduate students studying language technology annotated 150 sentences in the restaurant domain. Their Krippendorff's α was 0.72 on the original annotation scheme and 0.74 on the collapsed scheme.

Post-processing: We next aggregated crowd workers' responses using MACE (Hovy et al., 2013), where a response was regarded as a binary value of a combination of a text and a human motive. We set the prior probability of a positive class to 1/6 (i.e., one text is likely to have one of the six motives). This prior fits the responses better than a uniform prior.

Analysis: Table 3.2 shows the distributions of human motive labels. There is a clear difference between domains: the restaurant domain has a variety of motives relevant to hedonic motives (i.e. pleasure seeking) like SELF-FULFILLMENT (SF) and SOCIAL RELATION (SR), while the laptop domain tends to have utilitarian motives (i.e. practical needs) such as AMBITION&ABILITY (AA) and FINANCE (F).

[4] We set the reward based pilot annotation studies so that workers can earn at least $6/hour.

3.4 Human Motive Detection

We propose the task of motive detection, where a system detects human motives relevant to a given input sentence. This is essentially a multi-label sentence classification task. Note that a single text can have multiple labels, as it may contain more than one motive.

3.4.1 Baseline Systems

We evaluated four groups of systems, namely: (1) support vector machines, (2) shallow neural networks (specifically, multi-layer perceptron), (3) Transformer models that were fine-tuned for this task, and (4) a Transformer model (specifically, GPT-3) that performed the task using a prompt-based approach without fine-tuning.

3.4.1.1 Support Vector Machines (SVM)

We built a linear SVM classifier on bag of n-grams (BoNG) of sentences. We counted 1-, 2-, and 3-grams of words in each sentence to construct a BoNG vector. To avoid overfitting to rare words, we discarded n-grams that occurred only once in a training set. We also applied TF-IDF scaling to BoNG vectors to emphasize topic words ($\text{BoNG}_{\text{tfidf}}$).

3.4.1.2 Multi-layer Perceptron (MLP)

We built an MLP classifier with one hidden layer on top of word embedding-based sentence representations. We employed three methods to convert off-the-shelf word embeddings into fixed-sized sentence embeddings.

Simple word-embeddings model (SWEM): We calculated element-wise average and max-pooling of word embeddings in a sequence and concatenate them (Shen et al., 2018).

Convolutional neural network (CNN): We used a CNN to aggregate adjacent word embeddings in a hierarchical manner. We followd Kim (2014) and used filter windows of 3, 4, and 5.

Bidirectional LSTM (BiLSTM): The aforementioned methods are not sensitive to word order. To consider word order, to which the aforementioned two methods are insensitive, we encoded a sequence of word embeddings by a bidirectional LSTM. We concatenated hidden states at the final time steps from both directions to obtain a sentence vector. We set the number of layers to two.

3.4.1.3 Pre-trained Language Models (PLMs)

Transformers: We encoded input tokens by PLMs based on the Transformer architecture (Vaswani et al., 2017). We employed three PLMs: (1) BERT (Devlin et al., 2019), (2) RoBERTa (Liu et al., 2019b) and (3) DeBERTaV3 (He et al., 2021a). Following the standard practice, we wrapped input tokens with special tokens that denotes the sentence beginning (BOS) and ending (EOS). After encoding the input tokens, we fed the last hidden state of the BOS token to an MLP classifier with one hidden layer. We set the hidden units of the MLP classifier to the hidden dimension of the PLM models (=768).

3.4.1.4 Few-shot In-context Learning with a Language Model

GPT-3: We fed to GPT-3 (text-davinci-003; 175B parameters) (Brown et al., 2020) a prompt that contains instructions, definition of human motives, and six examples in addition to an input sentence. For each motive category, we picked one example from a training split that has similar sentence embedding to that of the input text. The sentence embeddings were computed through Sentence Transformers (Reimers and Gurevych, 2019) with MPNet (Song et al., 2020) as a backbone model. We devised the prompt based on the performance on validation splits.

3.4.2 Training

We simply treated our multi-label classification task as a set of binary classification tasks, where MLP classifiers shared parameters except for those of an output layer over motive categories. To handle highly skewed class distributions, we weighted a loss function to train a model (Morik et al., 1999). For example, MLP classifiers were trained to minimize a weighted cross-entropy loss:

$$\mathcal{L} = - \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} \sum_{c\in\mathcal{C}} \left[w_c y_c \log \mathrm{MLP}_c(\boldsymbol{x}) + (1-y_c) \log(1-\mathrm{MLP}_c(\boldsymbol{x})) \right], \tag{3.1}$$

where $(\boldsymbol{x}, \boldsymbol{y})$ is a pair of a sentence and a label in dataset $\mathcal{D}$, $\mathcal{C}$ is a set of categories, and MLP_c is an output function w.r.t. category c. We computed the class weight as follows.

$$w_c = \frac{\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} (1-y_c)}{\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} y_c} \quad (c \in \mathcal{C}) \tag{3.2}$$

Transfer Learning Across Domains: Our hypothesis is that, in contrast to entity aspects that must be defined for each domain, underlying human motives are universal across domains while their distributions can be different. If this hypothesis holds true, then it would be possible to

	Restaurant			Laptop		
Method	Precision	Recall	F1	Precision	Recall	F1
SVM-BoNG	.565 (±.028)	.394 (±.052)	.451 (±.046)	.480 (±.043)	.358 (±.018)	.397 (±.006)
+ Transfer	.518 (±.042)	.422 (±.055)	.459 (±.050)	.555 (±.102)	.396 (±.023)	.453 (±.039)
SVM-BoNG$_{\text{tfidf}}$	.544 (±.018)	.482 (±.032)	.492 (±.015)	.577 (±.106)	.449 (±.014)	.477 (±.038)
+ Transfer	.542 (±.060)	.477 (±.041)	.475 (±.012)	.566 (±.071)	.461 (±.012)	.489 (±.029)
MLP-SWEM	.376 (±.027)	.783 (±.002)	.478 (±.026)	.359 (±.007)	.592 (±.022)	.416 (±.001)
+ Transfer	.462 (±.040)	.662 (±.025)	.516 (±.013)	.393 (±.011)	.534 (±.047)	.436 (±.025)
MLP-CNN	.565 (±.032)	.499 (±.045)	.524 (±.032)	.468 (±.011)	.410 (±.014)	.423 (±.007)
+ Transfer	.700 (±.059)	.541 (±.017)	.583 (±.020)	.519 (±.033)	.432 (±.021)	.456 (±.006)
MLP-LSTM	.447 (±.007)	.631 (±.008)	.511 (±.007)	.419 (±.007)	.568 (±.001)	.473 (±.005)
+ Transfer	.475 (±.017)	.618 (±.011)	.531 (±.005)	.500 (±.031)	.572 (±.003)	.518 (±.006)
PLM-BERT	.569 (±.032)	.627 (±.077)	.569 (±.032)	.596 (±.079)	.544 (±.045)	.596 (±.079)
+ Transfer	.614 (±.019)	.656 (±.086)	.614 (±.019)	.645 (±.018)	.562 (±.030)	.645 (±.018)
PLM-RoBERTa	.542 (±.025)	.743 (±.094)	.542 (±.025)	.530 (±.080)	.710 (±.058)	.530 (±.080)
+ Transfer	.615 (±.032)	.742 (±.056)	.615 (±.032)	.627 (±.011)	.694 (±.052)	.627 (±.011)
PLM-DeBERTa	.533 (±.016)	.685 (±.048)	.533 (±.016)	.489 (±.060)	.577 (±.063)	.489 (±.060)
+ Transfer	.621 (±.015)	.673 (±.063)	.621 (±.015)	.590 (±.059)	.644 (±.107)	.590 (±.059)
GPT-3	**.709** (±.008)	**.804** (±.007)	**.709** (±.004)	**.719** (±.002)	**.829** (±.002)	**.732** (±.003)
(Ref.) Human	.724 (±.014)	.859 (±.014)	.781 (±.012)	.766 (±.021)	.855 (±.019)	.806 (±.017)

Table 3.3: **Results of human motive detection.** Macro-precision, recall, and F1-measure scores are averaged over three folds in cross-validation (except for the performance of crowd workers in row Human). The higher numbers in each metric are denoted in **bold face**.

leverage out-of-domain data to enhance the performance of motive detectors. Motived by this, we implemented transfer learning across domains to minimize the loss function below.

$$\mathcal{L}' = \mathcal{L}_{\text{in}} + \lambda\mathcal{L}_{\text{out}}, \tag{3.3}$$

where $\mathcal{L}_{\text{in}}$ and $\mathcal{L}_{\text{out}}$ are loss functions defined on in-domain and out-of-domain data, and λ is a hyperparameter to discount the out-of-domain loss.

3.4.3 Experimental Setup

Our primary evaluation metrics is macro-average F1 measures across motive categories. We performed three-fold cross-validation, where the dataset is evenly divided into training, validation, and test sets. In each fold, we conducted a grid search of hyperparameters based on the validation set, and subsequently trained a model on the training and validation sets before testing it on a test set. The average scores over test splits were then reported as the final score.

We used pretrained 100-D GloVe embeddings that were trained on 6 billion tokens from the Wikipedia and Gigaword corpora (Pennington et al., 2014)[5] for the MLP classifiers. For the Transformer classifiers, we used the pre-trained parameters through the `transformers` library and OpenAI's API (for GPT-3). Further implementation details can be found in the appendix A.1.

3.4.4 Results

The results of the baseline systems are presented in Table 3.3. The performance of GPT-3 was the best, even with only six training examples. GPT-3 was superior especially in terms of recall. The other supervised models, however, produced fewer motive labels and resulted in lower recall, which might be due to the imbalance in class distribution in the training data. Furthermore, the poor performance of SVM classifiers suggest that surface-level features are not enough to detect the various realizations of human motives.

For estimating human performance, we compared individual responses from crowd workers against aggregated, gold-standard labels. We generated 100 sets of human responses by repeatedly sampling one of three workers for each sentence. The performance of GPT-3 (0.7 F1) was very close to human performance, but there is still room for improvement.

Interestingly, adding out-of-domain data (+Transfer) improved the performance of the SVM, MLP, and PLM classifiers except SVM-BoNG$_{\text{tfidf}}$. The improvement is particularly evident in precision scores. The transfer learning was most effective for PLM, which likely benefited from the increased training data size. This transferability of inductive bias demonstrates the universality of underlying motives across domains.

3.5 Connection to Sentiment Analysis

This chapter so far has highlighted the significance and potential for automatically detecting underlying human motives across domains. In this section, we once again turn our focus to sentiment analysis.

Firstly, as previously discussed, we believe that the detected motives themselves offer valuable insights into services or products of interest. For instance, the distribution of motive labels can reveal the types of human motives that are important in a particular domain, thereby providing guidance for future decision-making.

Moreover, we can approach sentiment analysis based on motives as proposed by Li and Hovy (2017). Once the relevant motives are identified, we can analyze sentiment by determining the

[5]We also tried other word embeddings like word2vec (Mikolov et al., 2013), but they yielded similar results.

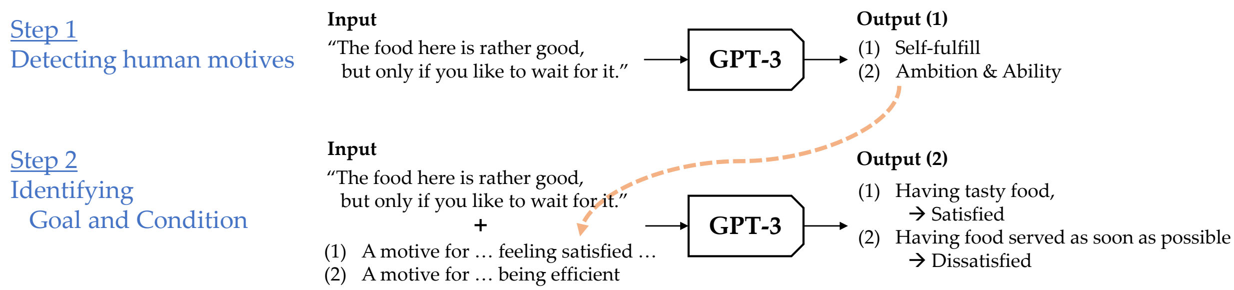
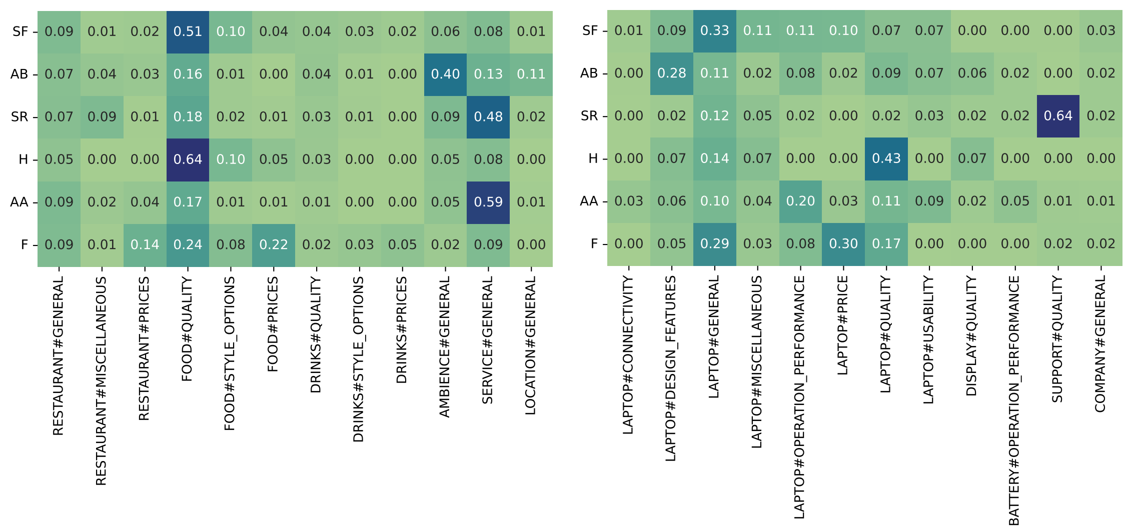


Figure 3.4: Illustration motive-based sentiment analysis using GPT-3. The process involves two steps: First, GPT-3 identifies relevant motive categories for the given text, and in the second step, it uses the predicted motive labels to generate relevant goals and judge if each goal was satisfied.



(a) Restaurant domain

(b) Laptop domain

Figure 3.5: Co-occurence of the gold-standard motive and aspect labels. The co-occurence counts are normaliezd by the total occurence of each human motive. SF: Self-fulfillment, AB: Appreciating Beauty, SR: Social Relation, H: Health, AA: Ambition & Ability, F: Finance.

goals, plans, and actions that were activated to fulfill these motives and whether they were successfully completed. To evaluate the feasibility of this approach using state-of-the-art language models, we tested ten random samples from the restaurant domain with GPT-3 (text-davinci-003) in few-shot in-context learning in two step depicted (Figure 3.4). First, we performed the human motive task with GPT-3 to obtain relevant motive categories. In the second step, we added the predicted motive labels to the input and generated goals and their conditions (satisfied or not). As demonstrated by the examples presented in Table 3.4, GPT-3 provided reasonable analyses, indicating the potential of this approach for future research.

Input text	"The food here is rather good, but only if you like to wait for it."
Motives	(1) Self-fulfill, (2) Ambition & Ability
Goals	(1) Having tasty food. (2) Having food served as soon as possible.
Conditions	(1) Satisfied. (2) Dissatisfied.

Input text	"This place is worth a one-hour drive."
Motives	(1) Self-fulfill
Goals	(1) Having a meal that is worth the effort of driving for an hour.
Conditions	(1) Satisfied.

Table 3.4: Examples of motive-based sentiment analysis using GPT-3. We provided GPT-3 with a prompt containing six examples and the input text. The model generated relevant motives, goals, and conditions (satisfied or dissatisfied) step-by-step (Figure 3.4). The motive labels were detected by GPT-3 in a separate process and then converted into language expressions through templates before being used for generating goals and conditions.

Lastly, aspect categories in aspect-based sentiment analysis can be seen as the intermediate realization of human motives within specific domains. The co-occurrence of human motive and aspect labels, as shown in Figure 3.5, indicates a connection between human motives and particular groups of domain-specific aspect categories. For example, the Social Relation (Sr) motive is often present in sentences that discuss service (restaurant) and customer support (laptop). The same motive can be manifested as different aspects for different services and products as exemplified by the Appreciating Beauty (Ab) motive. Typically, defining domain-specific aspects typically requires domain knowledge and is done manually by experts. Human motives can potentially facilitate this process. By leveraging the universality of basic motives, researchers can detect the most relevant motives in the target domain and subsequently define domain-specific aspects through an analysis of how these motives are manifested.

3.6 Conclusion

We aimed at to understand why a writer of a text holds a particular sentiment and proposed a task of human motive detection as an essential building block for achieving this goal. We presented a taxonomy of motives derived from a psychology study and annotated 1,600 restaurant and laptop reviews with six motives in order to facilitate the development and evaluation of computational models.

We evaluated the performance of baseline predictive models on this dataset and found that automatic identification of implicit human motives is feasible.[6] The best-performing model, GPT-3, achieved an F1 score of 0.7 in the restaurant and laptop domains. However, there is still a performance gap between automatic detectors and humans.

One interesting finding is that the same underlying motives can appear in different domains, although their distribution may differ. We empirically verified this by transferring learned parameters across domains, which showed that predictive models can strongly benefit from out-of-domain instances.

Future Work: We analyzed two domains, restaurants and laptops, and found that people are driven by hedonic motives in many cases and utilitarian motives in some cases in the restaurant domain. The laptop domain is mostly driven by utilitarian motives. There are many other domains that could be explored, such as the movie domain, where people watch movies primarily for enjoyment. Exploring other domains would be an interesting direction for future research. Although our focus was on sentiment analysis in this study, detection of motives can benefit other NLP applications, such as in-depth machine reading. For example, underlying motives could provide a strong clue for modeling a sequence of actions that share the same actor (a.k.a *narrative chains* (Chambers and Jurafsky, 2008)).

[6] Code and data used in this study are available online:
https://github.com/notani/acl2019-human-motive-identification

Chapter 4

Representation of To-Do Texts

Overview

In recent years, an increasing number of people have started managing their daily and work-related tasks with digital to-do management software. Within these software applications, users create personal notes to help them remember and organize the various things they need to complete. However, the brevity and lack of specificity in these to-do descriptions present a challenge for text representation models, making it difficult for NLP systems to provide effective assistance. In this chapter, we discuss a solution to this problem based on the implicit knowledge about user intent.[1] This work presents a novel neural multi-task learning framework that extracts representations of to-do texts using a multi-head attention mechanism on top of a pre-trained text encoder. To adapt the representation model to to-do texts, weak-supervision labels are collected from semantically rich external resources, such as a common-sense knowledge base, based on the principle that to-do tasks with similar intents have similar labels. The model is then trained on multiple generative and predictive training objectives jointly. The trained representation model is evaluated on five downstream tasks, and the results show that the proposed approach consistently outperforms baseline Transformer models, achieving an error reduction of up to 38.7%. This demostrates the effectiveness of the approach in improving the performance of NLP systems on to-do descriptions with limited contextual information.

4.1 Introduction

Task management tools are widely used to organize tasks and keep track of progress in work and daily lives. Examples include Microsoft To-do, Todoist, Trello, and digital assistants such as Amazon Alexa and Google Assistant. Machine learning techniques can automate various aspects of task management such as task creation (Mukherjee et al., 2020), organization (Landes and Di

[1]This chapter is based on the work conducted at Microsoft Research Redmond (Otani et al., 2022).

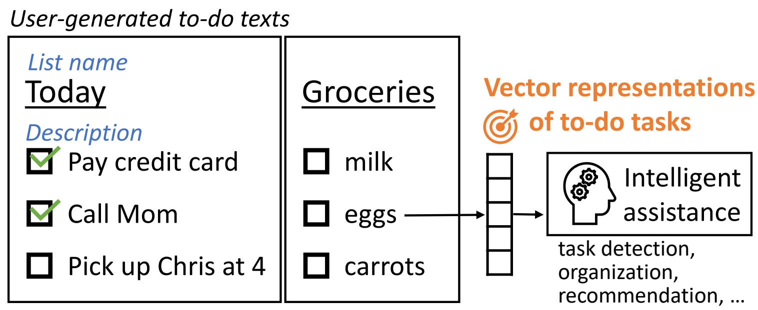


Figure 4.1: Our aim is to encode user-generated to-do texts (list names and descriptions) into vector representations so that machine learning systems can provide various kinds of intelligent assistance.

Eugenio, 2018), prioritization, and decomposition of complex tasks (Nouri et al., 2020; Zhang et al., 2021a).

The goal of this work is to develop a *single*, general-purpose encoding system that converts to-do task texts into real-valued vector representations (Figure 4.1). Using one encoding system for multiple task functions (task detection, organization, recommendation, etc.) as opposed to having multiple dedicated encoders saves the computational costs of updating models regularly and encoding texts from millions of users.

Representation learning has been extensively studied in natural language processing (Camacho-Collados and Pilehvar, 2018). Adapting models pre-trained on massive amounts of raw texts to a target domain or a task has become common practice (Qiu et al., 2020), with many publicly available pre-trained models such as BERT (Devlin et al., 2019), GPT-2 (Radford et al., 2018), and sentence encoders (Cer et al., 2018; Reimers and Gurevych, 2019). Leveraging word context is one of the key strengths of these pre-trained models. However, to-do texts exhibit unique characteristics that make this context-based modeling less effective (§4.2).

Our analysis on a dataset of 6.5 million entries shows that to-do texts are short and often lack an action verb. While similar to web search queries, they are not written to be understood by a search engine but instead are personal notes to the users themselves and assume rich personal context. On the other hand, some task management applications allow users to organize their to-dos under user-defined lists, which, our analysis shows, can *sometimes* convey important information about their meaning. For instance, a "grocery" list indicates that the listed items are things to *buy*, but some lists are not indicative of actions like "today".

Our hypothesis is that we can effectively fine-tune contextualized representation models for under-specified texts using multiple weakly-supervised prediction/generation tasks that focus on knowledge about to-do tasks. We induce supervision signals semi-automatically from existing resources so that to-do tasks that have similar intents share similar target labels. To this end, we propose *LITE*,[2] a framework for training to-do task representation models using the following auxiliary tasks: (1) autocompletion of to-do descriptions, (2) pre-action and goal generation based on COMET (Bosselut et al., 2019; Hwang et al., 2021), and (3) action attribute predictions based on FrameNet (Ruppenhofer et al., 2016). We implement LITE on top of existing pre-trained language models and evaluate its performance through downstream tasks on two proprietary and two publicly available datasets (Jauhar et al., 2022; Landes, 2018): urgent and important to-do detection, actionable to-do classification, co-located and co-timed to-do pair detection, and intent detection.

Overall, we make the following contributions: (1) A neural multi-task learning framework to fine-tune embeddings of to-do texts based on intents.[3] (2) A methodology to collect weak supervision signals from various resources without costly manual annotations. (3) An empirical comparison of contextual embeddings models on real to-do texts, where LITE outperforms various baseline models including BERT, RoBERTa, Sentence-BERT/RoBERTa, achieving error reduction of 4.8-38.7%.

4.2 User-generated To-do Data

In this section, we introduce our primary to-do dataset and discuss its characteristics: (1) to-do descriptions are generally short, (2) most of to-do descriptions do not have action verbs, and (3) some list names are less informative about task intents than others.

4.2.1 Data Collection

For training and data analysis, we use a dataset based on the now-retired *Wunderlist* task management app. The app was available on multiple platforms and had more than 13 million users in 2015. The dataset (henceforth *WL*) contains 6.5 million English to-do texts. Each *to-do text* includes a *description* (e.g., "call mom") and associated *list name* (e.g, "today"). See Appendix B.1 for more details on how the dataset was anonymized.

We performed a basic linguistic analysis on the WL data. As observed by Landes and Di Eugenio (2018), general-purpose analyzers often fail to analyze to-do texts correctly due to the writing style and the lack of context words. To alleviate this problem, we use frequency information obtained from a large corpus to correct automatically assigned POS tags, through the following 3-step process. First, we run the spaCy tagger (Honnibal et al., 2020)[4] to assign POS tags. Then, motivated by Keyaki and Miyazaki (2017), we correct the POS tags based on frequency information derived from 3 billion sentences from the DepCC corpus (Panchenko et al., 2018).[5] Finally,

[2] Short for Latent Intent Task Embedding

[3] The code is available at github.com/microsoft/Intent-based-Task-Representation-Learning

[4] We use the English model `en_core_web_lg` v3.0.0

[5] We extracted the first 100 files from DepCC and re-tagged the sentences using spaCy. We counted the fre-

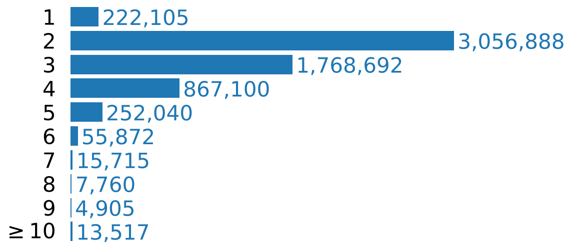


Figure 4.2: Distribution of the lengths (number of tokens) of to-do descriptions. The average length is 2.4.

we apply the spaCy dependency parser to the texts with the corrected POS tags and identified main verbs and arguments.

4.2.2 Observations

To-do descriptions are very short: The mean number of tokens per to-do description is 2.4 (Figure 4.2), which is similar to that of search engine queries (Taghavi et al., 2012), but with two key differences: (1) many search queries are intended for information seeking (Broder, 2002), while to-dos typically express things to perform or to remember, and (2) people write search queries with the capabilities of a search engine in mind, but to-do descriptions are personal notes to the users themselves.

Most to-do descriptions have no action verb: We observed that 87.8% of to-do descriptions do not have action verbs. If an action verb is present, 75.1% and 12.7% have a direct object and a prepositional phrase, respectively. The degree of under-specification depends on a to-do's list name. An action verb is more frequently used in to-do descriptions that appear in *generic* lists, such as "inbox"[6] (29.7%), "to do" (28.4%), and "today" (22.1%). When list names already imply a *specific* action, the action verb is more likely to be omitted such as in the "shopping" (3.3%) and "movies to watch" (4.7%) lists.

List names can be indicative of task intents but not always: For example, a to-do text (*description* = "avocados", *list name* = "to buy") signifies the intent "to buy avocados", but the same description can appear also in generic lists, such as "to do" or "reminders". When a list name is generic, a task description needs to be weighted more to accurately capture the intent of a task. Figure 4.3 shows that this is a non-trivial problem for pre-trained language models like BERT. The

quencies of 1-3 grams of token-XPOS pairs and replaced tokens that appeared fewer than 100 times with an out-of-vocabulary token. The frequencies were used to score the sequences of the POS tokens obtained in the previous step, and replace them with more frequent ones, if found. One of the authors manually evaluated the 100 frequent to-do descriptions with tags changed by post-processing and found 17/57 errors were corrected.

[6] "Inbox" was the default list name in the Wunderlist app.

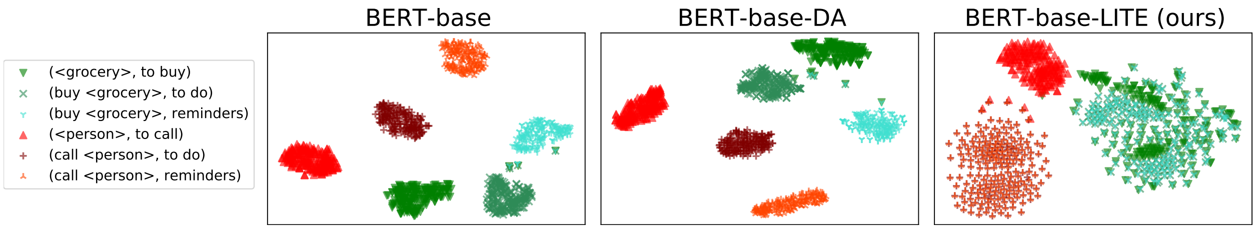


Figure 4.3: t-SNE (van der Maaten and Hinton, 2008) visualization of the embeddings of to-do texts generated by (1) BERT-base, (2) BERT-base domain-adapted by masked language modeling on WL, and (3) LITE.

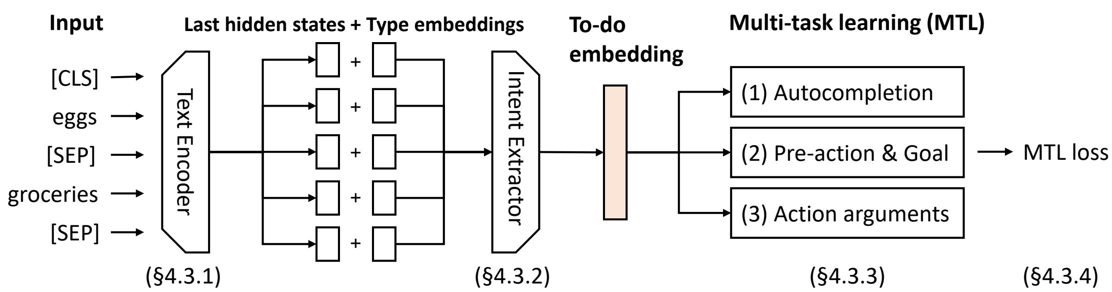


Figure 4.4: LITE model overview. We encode input tokens with an off-the-shelf text encoder and feed the hidden states and type embeddings to an intent extractor to obtain the representation of the to-do task. We train LITE over three training objectives (1-3) jointly.

figure visualizes the distribution of the embeddings of the "buy <grocery>" and "call <person>" to-do texts[7] expressed in two ways: (1) the descriptions "buy <grocery>" and "call <person>" are paired with generic list names ("to do" and "reminders"); or (2) the descriptions "<grocery>" and "<person>" are paired with specific list names (indicating the actions "to buy" and "to call"). To produce embeddings, we concatenated descriptions and list names in the input and extracted their pooled output from the encoders (§4.3.1). We can see that a BERT model cannot capture the similarity within the *buy* nor *call* intent groups even after domain adaptation (DA) to to-do texts (see §4.4.3 for more details on DA). Our model, LITE, can successfully *ignore* the generic lists and group similar tasks together.

4.3 Method: Multi-task Learning (MTL)

We propose a multi-task learning (MTL) framework to represent to-do descriptions along with their list names (Figure 4.4). Our model first encodes text using off-the-shelf encoders (§4.3.1). The token representations along with information about their types are merged by an intent

[7] <grocery> stands for grocery items, and <person> stands for person names taken from the following web pages: vegetablesfruitsgrains.com and ssa.gov/oact/babynames

extractor with multi-head attention (§4.3.2). We train the encoder and extractor on three auxiliary tasks (§4.3.3,4.3.4).

4.3.1 Off-the-shelf Text Encoder

We encode input texts using off-the-shelf pre-trained transformer-based language models. Our model takes as input the concatenation of two types of texts, descriptions and list names, separated by the token

$$\texttt{[SEP]} : \texttt{<s>}\ \text{description}\ \texttt{[SEP]}\ \text{list name}\ \texttt{</s>}, \tag{4.1}$$

where <s> and </s> are beginning-of-sentence and end-of-sentence tokens pre-defined for the encoder. The encoder converts a sequence of N input tokens $w_1, w_2, \cdots w_N$ into real-valued vector representations using multiple layers of attention mechanism and fully-connected networks. We use the last hidden states $H = \{\boldsymbol{h}_i\}_{i=1,2,\cdots,N}$ as the contextual token representations of the input.

4.3.2 Intent Extraction with Attention

List names are often—but not always—indicative of task intents (§4.2). For example, a "shopping" list tends to have items that a user wishes to purchase and is useful for identifying intents, but some list names merely express time (e.g., "today"), topics/targets (e.g., "family"), or nothing specific (e.g., "things to do"). In these cases, the model should "pay more attention" to the to-do description. To handle this, we use a multi-head attention mechanism (Chaudhari et al., 2021; Vaswani et al., 2017) to extract a vector representing the intent of a to-do task and introduce token type embeddings to explicitly inform a model of text types.

Multi-head attention: An attention mechanism is suitable to model the variable nature of token importance. We use a multi-head, scaled dot-product attention mechanism (Vaswani et al., 2017) and aggregate H based on token importance into the intent embedding $\boldsymbol{z}$.

Suppose we have T attention heads. For each head, we convert a token representation $\boldsymbol{h}_i \in H$ into vectors $\boldsymbol{u}_i^t, \boldsymbol{v}_i^t \in \mathbf{R}^{d'}$ by trainable transformation matrices, $W_u^t, W_v^t \in \mathbf{R}^{d' \times d}$. We set d' to d/T.

$$\boldsymbol{u}_i^t = W_u^t \tanh(\boldsymbol{h}_i) \tag{4.2}$$

$$\boldsymbol{v}_i^t = W_v^t \tanh(\boldsymbol{h}_i) \tag{4.3}$$

We then compute attention scores $\alpha^t \in (0, 1)$ and an output vector $\boldsymbol{o}^t \in \mathbf{R}^{d'}$:

$$\alpha_i^t = \frac{\exp(\boldsymbol{q}^{t^T} \boldsymbol{u}_i^t / \sqrt{d'})}{\sum_{j=1}^{N} \exp(\boldsymbol{q}^{t^T} \boldsymbol{u}_j^t / \sqrt{d'})} \tag{4.4}$$

$$\tag{4.5}$$

$$\boldsymbol{o}^t = \sum_{i=1}^{N} \alpha_i^t \boldsymbol{v}_i^t, \tag{4.6}$$

where $\boldsymbol{q}^t \in \mathbf{R}^{d'}$ is a trainable vector. Finally, we obtain an intent vector $\boldsymbol{z}$ by concatenating the output vectors of the T attention heads:

$$\boldsymbol{z} = \text{Concatenate}(\boldsymbol{o}^1, \boldsymbol{o}^2, \cdots, \boldsymbol{o}^T) \tag{4.7}$$

Token type embedding: We introduce token type embeddings, $\boldsymbol{e}_{\text{task}}, \boldsymbol{e}_{\text{list}}, \boldsymbol{e}_{\text{other}} \in \mathbf{R}^d$, to inform a model of the source of each token. BERT injects token type embeddings in the lowest layer, the embedding layer, and we train them during pre-training (Devlin et al., 2019), but other models do not (Liu et al., 2019b; Radford et al., 2018; Raffel et al., 2020). To avoid breaking the pre-trained parameters of those models, we add type embeddings to H and feed it to the multi-head attention module:

$$\boldsymbol{h}_i' = \tanh(\boldsymbol{h}_i) + \tanh(\boldsymbol{e}_{\text{type}(i)}) \tag{4.8}$$

where type(i) is the type of the i-th token.

4.3.3 Auxiliary Tasks for MTL

One straightforward way to train the extractor is to directly optimize it to predict the intent of a given to-do task. However, task intents are often obscure and hard to discretize into a fixed number of categories. As a result, manual collection of such categories can be costly and subjective. For example, "buy milk" and "buy a car" are both purchase action, but they differ in many aspects: different locations, different prerequisite events, and different motives.

Instead, we propose to train the extractor on multiple auxiliary tasks with weak supervision that provide semantic augmentation to under-specified to-do texts. The underlying assumption is that tasks with similar intents have similar target labels/texts in the auxiliary tasks. Below, we present our three auxiliary training tasks.

4.3.3.1 Autocompletion

Motivation: Inspired by the success of reconstruction learning tasks (e.g., Lewis et al. (2020a)), our first task focuses on surface-level information of to-do texts, namely prediction of missing

Input$_{\text{(desc., list)}}$	Output
(milk, groceries)	buy milk
(buy milk, things to do)	buy milk
(eggs, costco)	buy eggs at costco
(Chris, today)	call Chris today

Table 4.1: Examples of texts for the autocompletion objective (§4.3.3.1). Suppose to-do descriptions "buy milk", "buy eggs" and "call Chris" exist in the WL dataset. The phrases in the last two examples, which are underlined, were generated by our list-based templates.

tokens based on context tokens. Specifically, we feed a to-do text (the combination of a description and a list name) to a model, convert it into an intent embedding, and generate the maximal form of a to-do description that is inferable from the input. We call this auxiliary task *autocompletion objective*. We automatically collect such forms for under-specified to-do descriptions from the WL dataset.

Data collection: As previously observed, to-do descriptions under generic lists (e.g., "today") tend to be more specified than those under lists whose names imply specific action. For each to-do description in our WL dataset, we collect their longer descriptions (i.e., super-strings) up to five. We also use several templates for lists that represent locations and times to further expand descriptions (see details in Appendix B.2). Table 4.1 shows examples, two of which were generated with templates. The resulting dataset contains 1,487,161 pairs of short and long to-do descriptions. We combine them with specified to-do descriptions, which already have action verbs and do not have longer counterparts, and sample 2M examples (50% of examples are under-specified.) During training, one generation target is picked at random for each instance.

4.3.3.2 Pre-action and Goal Generation

Motivation: This task aims to represent to-do tasks based on their prerequisite actions (what we must do beforehand) and goal events (what we want to achieve), assuming that tasks with similar intents have similar prerequisites and goals. Here, a model is trained to generate prerequisite and goal actions for a given to-do item (a task description and a list name). We call this objective *pre-action and goal generation objective.*

Data collection: We leverage COMET (Hwang et al., 2021), a BART model (Lewis et al., 2020a) fine-tuned on ATOMIC$^{20}_{20}$, to collect weak supervision signals about to-do tasks' prerequisites and goals.[8] Specifically, we feed a long description of a to-do task generated in the previous

[8]We can retrieve prerequisites and goals of some to-do tasks from knowledge bases such as ATOMIC$^{20}_{20}$ (Hwang et al., 2021) and ConceptNet (Speer et al., 2017) without relying on language generation, but it is not always the case

Input$_{\text{(desc.)}}$	Output	
buy milk	go to store	get milk for breakfast
call Chris	find number	talk to someone
subscribe Netflix	go to website	watch a movie
Relation used:	`xNeed`	`xIntent`

Table 4.2: Texts generated for the pre-action and goal objective (§4.3.3.2) by COMET.

Input$_{\text{(desc.)}}$	Output	
buy milk	Buyer, Goods	Money, $\cdots$
call Chris	Addressee, Topic, $\cdots$	Medium, $\cdots$
FEs used:	`Core`	`Non-core`

Table 4.3: Labels collected from FrameNet for the action arguments prediction task (§4.3.3.3).

step (§4.3.3.1) to the BART model as a prompt followed by a relation token: (1) `xNeed` (prerequisite) token to generate the task's prerequisite or (2) `xIntent` (goal) token to generate the task's goal. We use beam search with width of 3 and collect the top-3 results for each relation. Table 4.2 shows generation results for three example to-dos.

4.3.3.3 Action Arguments Prediction

Motivation: Different to-do tasks have different domain-specific arguments. For example, a *purchase* task must have a *purchase target*, and possibly a *price* argument. In contrast, *contact* tasks usually have a *receiver* and a *topic of communication* argument. We design a multi-label training task called *action arguments prediction*, where, given a description and a list name, a model predicts all the action arguments associated with the to-do task.

Data collection: We use FrameNet (Ruppenhofer et al., 2016), a manually-created database on the meaning and usage of English words/phrases. Semantic representations are defined for concepts and events (called *frames*) and for their semantic elements (called *frame elements, FEs*); example texts that trigger frames and FEs are also provided. FEs can be *core* FEs (essential information for a frame), or *non-core* (optional). Table 4.3 shows examples.

Using the "long" to-do descriptions collected for the autocompletion task (§4.3.3.1), we identify frames in them using an off-the-shelf frame identifier (Swayamdipta et al., 2017). As our focus

that we can find the action of interest in the existing resources. The use of COMET is advantageous in handling unseen actions as shown by several studies (Bosselut et al., 2019; Hwang et al., 2021).

is on to-do tasks, we discard frames whose root frame is not `Event`. We then collect FEs for each frame from FrameNet. If a to-do description has two or more frames, we take the union of their FEs. For non-core FEs, we calculate importance weights by TF-IDF over the whole FrameNet repository so that common FEs appearing in many frames (e.g., `Manner`) have low weight. We normalize the weights into $(0, 1]$ by dividing them by the maximum weight.

4.3.4 Optimization

For the autocompletion as well as the pre-action and goal generation tasks, we employ a two-layer GRU (Cho et al., 2014) decoder with a cross-attention mechanism (Luong et al., 2015). We use the embedding layer of the encoder also in the decoder. We train the model to minimize the following cross-entropy loss for each instance:

$$\mathcal{L}_{\text{gen}} = -\sum_{j=1}^{M} \log P(y_j | y_{<j}, \boldsymbol{z}, H), \tag{4.9}$$

where M is the length of the output text. We apply label smoothing with a smoothing factor of 0.1 (Pereyra et al., 2017).

For the action arguments prediction task (multi-label classification), we use GILE as a label-embedding approach (Pappas and Henderson, 2019). Given an intent embedding and label embedding, GILE projects them into a joint vector space and computes an association score from their element-wise product. Concretely, for each label l, we calculate its score $P(l) \in (0, 1)$ as follows:

$$\boldsymbol{e}_{\text{in}} = \text{Act}(W_{\text{in}}\boldsymbol{z}) \tag{4.10}$$

$$\boldsymbol{e}^{(l)}_{\text{label}} = \text{Act}(W_{\text{label}}\boldsymbol{v}^{(l)}) \tag{4.11}$$

$$P(l) = \text{Sigmoid}\left(W_{\text{out}}(\boldsymbol{e}_{\text{in}} \odot \boldsymbol{e}^{(l)}_{\text{label}})\right), \tag{4.12}$$

where $\boldsymbol{v}^{(l)} \in \mathbf{R}^d$ is a pre-computed label embedding (constant), Act is an activation function and $W_{\text{in}}, W_{\text{label}} \in \mathbf{R}^{d\times d}$ and $W_{\text{out}} \in \mathbf{R}^{1\times d}$ are model parameters. To compute the label embeddings for FEs (Eq(4.11)), we encode the definitions of FEs in FrameNet with pre-trained transformer models.

We define the loss function to be:

$$\mathcal{L}_{\text{clf}} = \frac{1}{C}\sum_{c=1}^{C} \left(c \log P(c) + (1-c)\log\left(1 - P(c)\right)\right), \tag{4.13}$$

where C is the number of classes. We optimize a model to minimize the following weighted loss across three MTL objectives:

$$\mathcal{L} = \sum_{\text{task}} \frac{\mathcal{L}_{\text{task}}}{\log N_{\text{task}}}, \tag{4.14}$$

Task	Size	Example: (description, list name) [class]
Urgent and Important To-Do Detection (UIT)	2,254	(pick up packages at FedEx, n/a) [urgent], (sign up for HBO, n/a) [non-urgent]
Actionable To-Do Classification (AT)	12,189	(Sign up for dance class, inbox) [Actionable], (tomatoes, groceries) [ActionableCollection], (Alex, baby names) [Notes]
Co-located To-Do Pair Detection (CoLoc)	25,000	(fix tv, inbox)-(clean sink, today) [+], (fix tv, inbox)-(refill medicines, today) [-]
Co-timed To-Do Pair Detection (CoTim)	25,000	(get breakfast, daily)-(check news, inbox) [+], (get breakfast, daily)-(pickup drycleaner, inbox) [-]
Intent Detection (ID)	253	(schedule appointments with site managers, n/a) [calendar], (fix the CD ROM drive on my computer, n/a) [find-service]

Table 4.4: Evaluation tasks. Note that the UIT and ID datasets do not have list names.

where $\mathcal{L}_{\text{task}}$ is the loss function of each task, which is either $\mathcal{L}_{\text{gen}}$ or $\mathcal{L}_{\text{clf}}$. N_{task} is the number of target labels in a subtask (Aghajanyan et al., 2021). For the generation tasks, it is equivalent to the vocabulary size.

4.4 Experiments

Our aim is to obtain a single, general-purpose representation model that is effective on various downstream applications. We run LITE on top of $\text{BERT}_{\text{base}}$, $\text{BERT}_{\text{large}}$, and RoBERTa and evaluate its performance.

4.4.1 Evaluation Tasks

We evaluate LITE on four downstream tasks (Table 4.4): (1) urgent and important to-do detection (UIT), (2) actionable to-do classification (AT), (3) co-located and co-timed to-do pair detection (CoLoc and CoTim), and (4) intent detection (ID).

Urgent and Important To-do Detection (UIT): The goal of this task is to detect urgent or important tasks, an essential step for to-do prioritization in real applications. To evaluate this task we use a proprietary dataset (derived from WL) containing 2,254 human-labeled to-do descriptions. Each description is categorized into urgent and not-urgent classes based on the majority vote of 3 annotators. This dataset does not provide list names, hence we use a dummy list name "inbox" for LITE.

Actionable To-do Classification (AT): This task aims to identify to-do tasks that require a concrete, individual action to accomplish (*ActionableTask*) (e.g., “Sign up for dance class”). We evaluate this task using a proprietary dataset (derived from WL) containing 12,189 to-dos. Each instance consists of a description and a list name, and is manually categorized into *ActionableTask*, *Note*, and *ActionableCollection*. A *Note* is a list item that users add for future use, without the need for immediate action (e.g., “baby names”). Tasks that are labeled as *ActionableCollection* are not performed individually but rather as part of a collection of items in a larger task: “tomatoes” in the “groceries” list, for example, are part of the larger task "do groceries" where all the individual to-dos are addressed at the same time and location. Each example was annotated by 3 annotators, the majority label is the gold label. Tasks where one or more annotators were unsure about the correct label were eliminated.

Co-located and Co-timed To-do pairs Detection (CoLoc/CoTim): his task focuses on the location and time where to-do tasks are accomplished. Time and location are particularly powerful cues for task recommendations and reminders (Graus et al., 2016). In this task, given a pair of to-do items, the model predicts whether the two to-do tasks can be completed in the same location (CoLoc) or at the same time (CoTime). To evaluate this task we use the MS-LaTTE (Jauhar et al., 2022) dataset (derived from WL), which contains 25,000 pairs of to-do tasks (description + list name), of which 17,778 (71%) are labeled as CoLoc and 9,469 (38%) as CoTime.

Intent Detection (ID): This task focuses on predicting the intent associated with a to-do description. We use Landes and Di Eugenio (2018)’s dataset, which contains 253 to-do instances, each one labeled with one of nine intent classes (“calendar”, “find-service”, “buy”, etc.). No list name is provided in this dataset, so we use the generic list name “inbox” for LITE.

4.4.2 Setup

In all tasks, we first generate vector representations of instances in the dataset with a pre-trained encoder and train a simple classifier on them. The quality of the embeddings is measured by the performance of the classifier. We use a logistic regression classifier implemented in scikit-learn (Pedregosa et al., 2011), with or without a penalty term. To train a classifier for CoLoc and CoTim, which provide two to-do descriptions as input (see section 4.1), we concatenate the vector representations of the two items along with their element-wise product and difference vectors (Mou et al., 2016).

We generate 20 sets of training, validation, and test splits at random (Gorman and Bedrick, 2019)[9], and, in each trial, we use a validation split to tune hyperparameters by grid search (a regularization ∈ {None, L1, L2} and a regularization coefficient $\in \{2^{-5}, 2^{-4}, 2^{-3}, 2^{-2}, 2^{-1}, 1\}$).

[9]We split data into 6:2:2 for UIT, AT, and CoLoc/CoTim, and 8:1:1 for ID.

4.4.2.1 Implementation Details

We implemented our MTL framework using PyTorch v1.10.0 (Paszke et al., 2019) and ran experiments on NVIDIA GeForce GTX TITAN X and RTX A6000 (for $\text{BERT}_{\text{large}}$). We use uncased $\text{BERT}_{\text{base}}$, uncased $\text{BERT}_{\text{large}}$, and cased $\text{RoBERTa}_{\text{base}}$, in the `transformers` library v4.6.1 (Wolf et al., 2020) with the default parameters for dimensions, activation functions, and dropout. We set the number of attention heads in the extractor and the dimension of hidden states based on the choice of a text encoder, namely $(T, d) = (12, 768)$ for $\text{BERT}_{\text{base}}$-LITE and RoBERTa-LITE, and $(T, d) = (16, 1024)$ for $\text{BERT}_{\text{large}}$-LITE. We applied dropout of 0.1 to our modules except for the output layers. We optimized the model parameters using AdamW (Loshchilov and Hutter, 2019) with batch size of 2,400, learning rate of 5e-5, L2 weight decay of 0.01, and linear learning rate decay with warm-up steps of 2% of the total steps. We also apply gradient norm clipping of 1. We train our models for 15 epochs, and freeze the transformer encoder for the first 5 epochs. We sampled 3,459 examples as validation data, on which we evaluate a model every epoch, and terminate training if the validation loss does not improve for three consecutive epochs. We tuned hyperparameters and architectural choices (§4.3.2) based on the average validation scores over 20 random trials on all the datasets (more details in Appendix B.3).

4.4.3 Baselines

We compare the following encoders as baselines.[10]

BERT (Devlin et al., 2019): We take the embedding of the first token, `[CLS]`, to represent a to-do text. `[CLS]` embeddings are trained to represent the whole input sequence by next sentence prediction (NSP). We compare the base (12 layers, 768D) and large (24 layers, 1024D) models.

RoBERTa (Liu et al., 2019b): We take the average of the last hidden states to represent an input sequence as RoBERTa is not trained with NSP. We use $\text{RoBERTa}_{\text{base}}$(12 layers, 768D).

Motivated by Gururangan et al. (2020), we also compare the domain-adapted ("DA") version of BERT and RoBERTa. We perform additional pre-training to $\text{BERT}_{\text{base}}$ and RoBERTa on the 6M raw to-do texts (`<s>` description `[SEP]` list name `</s>`) from WL.

Sentence-Transformer: We also test off-the-shelf *general-purpose* sentence encoders based on Transformers. These encoders are pre-trained to induce sentence embeddings with siamese and triplet network on top of pre-trained Transformer models (Reimers and Gurevych, 2019). We use the pre-trained encoder based on $\text{BERT}_{\text{base}}$ and $\text{RoBERTa}_{\text{base}}$. The encoders are trained on about 286k of natural language inference and textual similarity instances.

[10] We evaluate additional baselines in Appendix B.4. The implementation details can be found in Appendix B.5.

	UIT			AT	CoLoc			CoTim			ID
	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Prec.	Rec.	F1	Acc.
BERT	.826	.798	.811	.906	.800	.917	.855	**.511**	.362	.423	.628
BERT-DA	.862	.821	.840	.928	.801	**.921**	.857	.510	.386	.439	.614
Sentence-BERT	.821	.787	.803	.901	.817	.892	.853	.499	.396	.442	.542
BERT-LITE	**.871**	**.855**	**.863**	**.932**	**.826**	.901	**.862**	**.511**	**.409**	**.454**	**.670**
RoBERTa	.805	.763	.783	.868	.777	**.923**	.844	.492	.335	.398	.506
RoBERTa-DA	.819	.745	.779	.913	.787	.922	.849	.488	.360	.414	.500
Sentence-RoBERTa	.831	.789	.809	.897	.820	.893	.855	.493	.386	.433	.572
RoBERTa-LITE	**.871**	**.847**	**.859**	**.919**	**.826**	.905	**.864**	**.509**	**.402**	**.449**	**.674**
$\text{BERT}_{\text{large}}$	.817	.795	.805	.896	.805	**.910**	.854	.488	.404	.442	.636
$\text{BERT}_{\text{large}}$-LITE	**.863**	**.849**	**.855**	**.936**	**.830**	.907	**.867**	**.516**	**.441**	**.475**	**.718**

Table 4.5: Results on downstream applications. The best scores in each text encoder are denoted in **bold,** and the overall best scores are underlined. The results of statistical significance tests can be found in Table 4.6.

4.4.4 Results

Table 4.5 shows the main results. LITE demonstrates superior performance across all tasks and for all three encoders, indicating the generality of the learned representations. DA resulted in performance improvements in most cases but only marginally.[11] This is probably because to-do texts are too short to perform effective language model training.

Despite their effectiveness in various sentence-level tasks (Reimers and Gurevych, 2019), the Sentence-Transformer models did not perform as well as the vanilla BERT and RoBERTa encoders in this experiment. We speculate that these sentence encoders struggle to leverage contextual information effectively, as they are pre-trained on sentences that differ significantly from to-do texts.

Following Gorman and Bedrick (2019), we conducted a permutation test with 5,000 trials, comparing the scores of vanilla Transformer vs. DA, vanilla Transformer vs. LITE method, and DA vs. LITE in each of twenty trials. We used Bonferroni correction to cariburate p-values obtained from the tests (Dror et al., 2017) to avoid over-estimate statistical significance. Table 4.6 reports the number of trials where one model's score is significantly higher than that of the other ($\alpha = 0.05$). The results indicate that LITE performs significantly better than the vanilla counterpart more frequently than DA. In some tasks, the score of RoBERTa-LITE is even significantly higher than that of RoBERTa-DA. (UIT, CoLoc and CoTim).

Our training framework also allows for fine-tuning Sentence Transformers to adapt them to

[11] It is also possible to combine domain adaptation by language modeling and LITE, however, it underperformed LITE overall. With $\text{BERT}_{\text{base}}$, the performance were UIT 0.873, AT 0.931, CoLoc 0.863, CoTim 0.447, and ID 0.656.

	UIT	AT	CoL	CoT	ID
BERT$_{base}$					
vanilla < DA	0	20	0	1	0
vanilla < LITE	7	20	4	8	0
DA < LITE	1	0	1	2	0
Sent. < LITE	14	20	5	0	0
RoBERTa					
vanilla < DA	0	20	3	2	0
vanilla < LITE	20	20	20	19	1
DA < LITE	13	1	20	10	1
Sent. < LITE	6	20	6	0	0
BERT$_{large}$					
vanilla < LITE	6	20	3	9	1

Table 4.6: The number of random trials (out of 20) where the test score of the model on the right side is significantly better than the model on the left side after Bonferroni correction ($\alpha = 0.05$).

	UIT$_{\mathrm{F1}}$	AT$_{\mathrm{Acc}}$	CoLoc$_{\mathrm{F1}}$	CoTim$_{\mathrm{F1}}$	ID$_{\mathrm{Acc}}$
Full	**.863**	**.932**	**.862**	**.454**	.670
-Ac	.855	.931	.861	.448	.656
-PG	.859	.923	.860	.449	**.726**
-Aa	.857	.928	.860	.440	.702

Table 4.7: Ablation study on BERT-LITE demonstrating the effect on F1 and accuracy scores of removing (A)uto(c)ompletion, (P)re-action and (G)oal generation, or (A)ction (a)rguments prediction.

short and under-specified to-do texts, which we plan to explore in future work. Although our goal is to train a general-purpose encoder, readers interested in *task-specific* fine-tuning can refer to Appendix B.6 for evaluation results.

4.4.5 Analysis

In contrast to DA, the proposed method consistently resulted in improved performance. We speculate that the auxiliary tasks of LITE collectively aided in training text encoders to extract richer semantic information from under-specified texts. Table 4.7 shows the contribution of our auxiliary tasks to overall performance. The full model performed the best in all tasks except ID.

As discussed in Section 4.2, text encoders need to combine information from both To-Do de-

List	Attn. × Norm
*errands	.036±.004
to do list	.040±.003
things to do	.040±.003
movies to watch	.040±.004
house to do	.041±.004
⋮	
trip	.093±.005
target	.094±.006
cleaning	.097±.005
bring	.097±.006
movies	.098±.007

Table 4.8: To-do lists that are assigned low and high attention weight × vector norm scores by LITE. Generic lists are denoted in **bold,** and specific lists in *italic.* *"errands" is an interesting case which implies actions (go to somewhere to do something) but is a general list at the same time as it has many kinds of different tasks like "(buy) water filter" and "(get) an emission test".

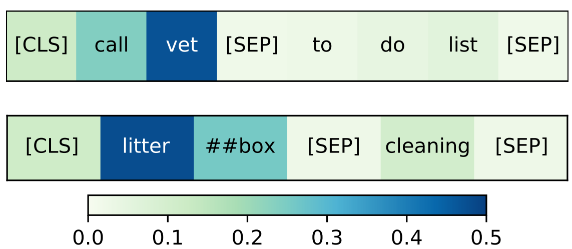


Figure 4.5: Comparison of attention weight × vector norm scores between "to do list" (general list) and "cleaning" (specific list).

scriptions and list names to infer the meaning of To-Do tasks. To evaluate LITE in this regard, we analyzed the product of attention weights and vector norms (Kobayashi et al., 2020) of encoded tokens for list names from the AT dataset that appear with more than 17 different to-do descriptions (90th percentile). Table 4.8 displays list names with the highest and lowest average scores assigned by BERT-LITE. We can see that generic lists (such as "to do list" and "house to do") have low scores, while specific, action-related lists (such as "bring" and "cleaning") have much higher scores. Figure 4.5 visualizes the distributions of attention scores on two examples. Although generic lists are unlikely to indicate the core meaning of intents, we believe it would not be wise to ignore them, as they can still provide semantic and pragmatic clues. For example, a list named "wishlist" typically contains things and activities that a user does not need to act on immediately, and can thus be a strong indicator of a non-actionable task in AT.

Desc. (In)	List (In)	Pre-action (Out)
18 min cardio	inbox	go to gym
1 swim suits	packing list	go to store
ride bikes	kids	go to bike shop
fertilize lawns	home	go to garden center
cell bio book	inbox	go to library

Table 4.9: Generation examples of "go to (location)" pre-action. The pre-action text was generated from BERT-LITE with a prompt "go to".

LITE leverages implicit knowledge related to To-Do tasks such as pre-actions, goals, and action arguments, which can be advantageous for relevant downstream tasks. For instance, we hypothesized that encoding pre-actions such as "go to a store" would be useful in identifying co-located To-Do pairs. However, we did not observe a clear benefit from the PG objective in the CoLoc task. Examining the top-1 greedy generation result of "go to (location)" pre-action from BERT-LITE (examples provided in Table 4.9), we found that the model tends to over-generate "go to store," which was generated for 32% of the examples in CoLoc. Consequently, the generated pre-action is not a helpful clue. When two tasks share the same location in the generated "go to" pre-action, 70% of them are labeled as co-located, which is close to the overall class distribution. The over-generation of "go to store" is understandable as "grocery" is one of the most frequent list names in the WL data, and the model encountered many shopping tasks during training. If we exclude examples with "go to store" predicted, then the ratio of co-located labels increases to 84%, and the model's precision, recall, F1 scores also rise to 0.875 ($\pm$0.027), 0.966 ($\pm$0.021), 0.918 ($\pm$0.016), respectively. This result is promising, indicating that more precise and richer supervision could help a model encode more accurate knowledge and improve performance.

4.5 Related Work

Intelligent systems can assist users with task management in many ways. To-do tasks can be inferred automatically from emails (Mukherjee et al., 2020). Systems can detect types of to-do items and suggest relevant applications or resources to users (Gil et al., 2012; Landes and Di Eugenio, 2018; Shah and White, 2021). Once to-do tasks have been created, a system can help users manage the completion progress, e.g., by sending reminders (Graus et al., 2016). Complex tasks can be decomposed automatically into more manageable sub-steps (Nouri et al., 2020; Zhang et al., 2021a). In all these use cases, a common step is to represent the input language as computational vector representations, but none of the existing studies has produced general-purpose representations of to-do tasks.

Short-text Representations: Multiple NLP areas involve very short texts with some unique characteristics. Several methods have been developed for tweets (e.g., Nguyen et al., 2020). Tweets pose the added challenge of containing many non-standard colloquial expressions and contain non-language text like URLs. Still, Wang et al. (2020a) present a similar finding to ours: massively pre-trained encoders do not always perform well. Search queries are also short, with an average of three terms (Taghavi et al., 2012). Unlike to-dos, information such as click logs (Zhang et al., 2019) can be used as an indicator of user intent. Another key difference is that search queries are written with the goal of having a machine interpret them.

Multi-task Learning: Multi-task learning improves the performance of pre-trained language models in various NLP tasks (Aghajanyan et al., 2021; Liu et al., 2019a; Shuster et al., 2020). The common perception in the research community is that auxiliary training tasks are effective when they are similar to the target domain/task (Shui et al., 2019). However, there are few relevant tasks and datasets for the to-do domain. Our study is the first work to propose a time- and cost-efficient way to harvest weak-supervision for MTL in that domain.

4.6 Conclusion

This work addresses the challenge of computing general-purpose representations of short and under-specified to-do texts for intelligent task assistance. As our analysis of user-generated data has revealed, To-Do texts are often extremely short and lack in context that context-based representation models can leverage. To encoding such texts, representation models need to effectively combine task descriptions and list names and infer latent intents.

Our method, LITE, employs a multi-head attention mechanism with token type embeddings on top of an off-the-shelf contextual text encoder for effectively inducing semantic information from the combination of to-do descriptions and list names. The model is trained using three auxiliary tasks: autocompletion, pre-action and goal generation, and action arguments prediction. We applied LITE to $\text{BERT}_{\text{base}}$, $\text{BERT}_{\text{large}}$, and RoBERTa and compared them with various baseline models on five downstream tasks. LITE consistently outperformed the baselines, demonstrating its effectiveness in generating semantic representations of to-do texts.

Chapter 5

A Textual Dataset of Situated, Goal-aware Responses

Overview

Recent data-driven conversational models have made significant progress in producing fluent, consistent, and informative responses to a variety of requests and utterances in help-seeking scenarios. However, such responses often lack broader contextual understanding and proactive engagement with the interlocutor, such as offering relevant suggestions to help customers achieve their goals. This limitation arises due to the models' inability to fully comprehend the interlocutor's situation and intent, which are typically implicit in conversations. To address this issue, this chapter proposes a textual conversational task that is anchored by situational context. A manually-curated dataset of 1.7k English conversation examples is presented, which includes situational background information and a set of response options. A well-designed conversation system should be able to provide suitable responses while avoiding inappropriate ones. Achieving this objective demonstrates the system's capacity to comprehend the request and contextual cues. As a proof-of-concept, several response selection systems were developed and evaluated on the new dataset. The benchmark experiments showed that this task is challenging, even for strong neural models, indicating potential opportunities for future research. The dataset can be used to advance the development of conversationally-informed and proactive dialogue engines.[1]

5.1 Introduction

Conversational assistant systems have recently shown significant improvements for understanding users' inquiries along with background knowledge, conducting requested operations, and

[1] This chapter is based on Otani et al. (2023a).

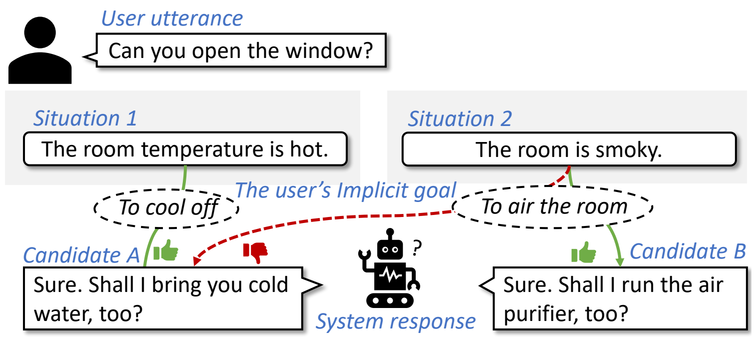


Figure 5.1: An example of situated goal-aware proactive response selection. The response candidate A is appropriate in Situation 1 but not in Situation 2.

returning natural language responses. Yet, typical systems are likely to be *passive* and only process user-initiated requests or merely ask values for domain-specific slots (Ammari et al., 2019; Williams et al., 2013). In contrast, human assistants like hotel concierges are more *proactive*, acting to address unmentioned needs and expected future events (Bellini and Convert, 2016; Cho et al., 1996). They do not only make a direct response or a clarification question to their interlocutors but also provide personalized information/assistance based on context and knowledge.

To push the frontier of task-oriented conversation technologies, we propose a task of *proactive* response selection for single-turn help-seeking conversations in English. We mean by proactive that a system engages in an interaction in a cooperative manner (Grice, 1975) and suggests something helpful to a user. The proposed task touches upon two crucial aspects of help-seeking conversations: situation-awareness and goal-awareness.

Situation: Situational information plays an important role in conversations as we illustrate in Figure 5.1. The example shows a user utterance "Can you open the window for me?" (top) and two response candidates (bottom). Although both candidates here sound helpful, their appropriateness varies depending on context: When the room is hot, suggesting a cold drink is appropriate assistance (left), but on the other hand, if the room is smoky, then running an air purifier is more helpful (right). Likewise, different situations make different responses more appropriate. A fair amount of situational information can be perceived as visual image, sound, and other kinds of sensory signals, and some of those are effectively incorporated into multi-modal conversational systems (Crook et al., 2019; Kottur et al., 2019). Yet, there are many other types of information that modern conversation assistance systems have access to, for example, via external APIs such as calendars and maps. In this study, we represent situation statements of six semantic categories (location, possession, etc.) in free English texts, which are more explicit as a semantic representation than just maintaining conversation histories (Henderson et al., 2019; Li et al., 2017; Lowe et al., 2015) and more flexible than structured representations of limited vocabulary (Budzianowski et al., 2018; Williams et al., 2013).

Goal: In the aforementioned example, the two actions address two different goals associated with opening a window, namely, *to cool off* and *to air the room*. While often being unspoken, underlying goals provide important semantic connections among context and utterances on many occasions (Allen and Perrault, 1980) particularly when language is indirect (Perrault, 1980; Stevens et al., 2015; Walker et al., 2011). We use goal information as a stimulus for soliciting naturalistic and proactive responses from human annotators in data collection.

We introduce a new dataset of **S**it**U**atated, **G**oal-**A**ware, and proactive **R**esponses (SUGAR; §5.3), which contains 1,761 examples of single-turn English conversations. Each conversation includes a user request anchored by an implicit goal, a reference response, and 12 sentences of situational information. As a proof of concept, we perform the task of *situated* response selection on SUGAR by adding two extra response candidates to each example. All responses are annotated with three-point appropriateness ratings. This dataset can also be used to develop and evaluate response generation systems as discussed in the next chapter.

To create SUGAR, we extracted user utterances and goals from common-sense knowledge bases, namely ATOMIC (Sap et al., 2019) and ConceptNet (Speer et al., 2017), and collected proactive responses with supporting situational context by crowdsourcing. We then used a language generation model, COMET (Bosselut et al., 2019; Hwang et al., 2021), to generate additional situational statements. Finally, we selected two more response options for each reference response using an adversarial method to form examples of three-choice response selection. To ensure data quality, we performed multiple manual validation steps during data collection. In our experiments on SUGAR (§5.4), Transformer-based rankers achieved over 80% precision@1 when when only the relevant situational statements were presented. However, precision decreased when distractors were included in the input, and this trend further continued as more distractors were added in our controlled experiments. These results suggest potential opportunities for future research.

5.2 Related Work

5.2.1 Conversational Dataset

Acquisition and annotation of real(istic) conversational data has been an essential step for developing conversation engines that imitate human communication (Serban et al., 2018). Various datasets have been constructed with a focus on different aspects of communication.

With regard to target communicative aspects, the most relevant to our work is SIMMC (Moon et al., 2020). SIMMC encompasses surrounding situational information that gives a basis for verbal interactions in task-oriented scenarios in the shopping domain. Moon et al. collected visually-grounded conversation examples from pairs of human annotators interacting with each other in a virtual environment (Crook et al., 2019), where one annotator seeks help for shopping, and the other provides assistance. SUGAR is also concerned with how human interlocutors perform situated conversations in a help-seeking setting. Our work extends this direction to scenarios other than shopping and includes more diverse types of information that modern conversational assistants could access via sensors or external APIs (e.g., temperature and schedule) by representing situational information in a textual form as opposed to visual images.

The choice of modality is motivated by existing conversational datasets that express various kinds of background information in plain text: the persona of an interlocutor (Dinan et al., 2020; Zhang et al., 2018), emotional states (Rashkin et al., 2019), and related documents (Dinan et al., 2019). These examples demonstrate the utility of textual forms for representing both explicit and implicit information of various kinds.

Some existing datasets are concerned with information-seeking conversations like restaurant recommendation where suggestions by assistants naturally occur (e.g., "If you like French cuisine, how about RestaurantX?", "I can find transportation for you."). However, it is not trivial to solicit such naturalistic proactive utterances in more diverse help-seeking scenarios. In many cases, the minimum objective of a conversation can be achieved by responding to user-initiated inquiries, and such kinds of responses are relatively easy to collect from non-expert annotators (Budzianowski et al., 2018; Byrne et al., 2019; Eric et al., 2020). We address this problem by leveraging implicit goals behind user requests. The comprehension of goals in conversations has been recognized to be important not only in task-oriented dialog research but also in a broad range of research areas such as linguistics, psychology, and artificial intelligence. (Clark and Schaefer, 1989; Gordon and Hobbs, 2004; Rahimtoroghi et al., 2017; Schank and Abelson, 1977). Human interactions often involve indirect speech acts (Gibbs and Bryant, 2008; Perrault, 1980) and indirect responses like non-yes/no answers to polar questions (de Marneffe et al., 2009; Hockey et al., 1997; Louis et al., 2020; Stevens et al., 2015). These studies motivate our strategy for soliciting natural-sounding proactive responses from crowd workers.

In contrast to most datasets we introduced here, SUGAR only contains single-turn conversation examples due to the ease of data collection and quality control. Yet, our primary focus is on conversational assistance, which is likely to be completed within a few turns (Völkel et al., 2021). Thus, we believe that single-turn examples are still useful for system development. It is possible to extend our problem setting and data collection approach to a multi-tern setting, which we leave as future work.

5.2.2 Response Selection

Automatic response models can be divided into two approaches: response generation and response selection. Response generation directly generates natural language response text from scratch, and response selection selects a response from a candidate pool built by humans, templates, or language generation systems. The latter approach is widely used in many real-world applications cases because of the controllability of responses and the easiness of evaluation (Deriu et al., 2020). In this study, we focus on the task of response selection as a proof of concept. We

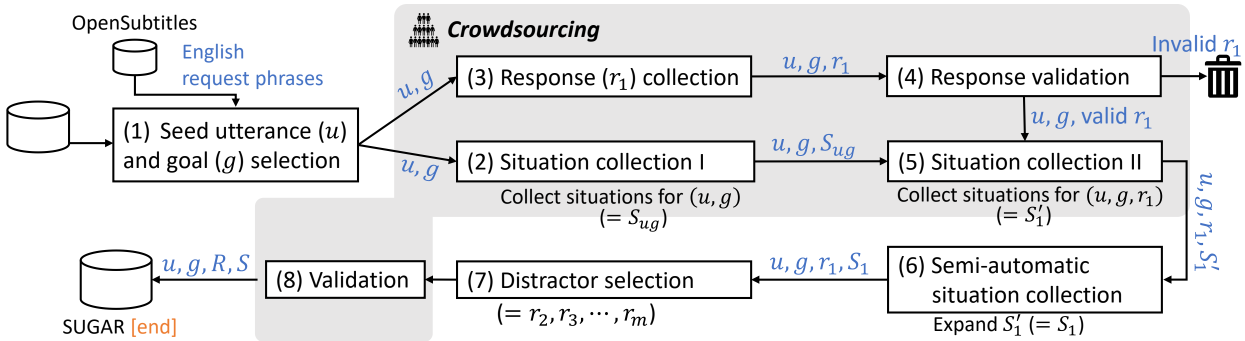


Figure 5.2: Pipeline for data collection. We start with existing common-sense knowledge bases (ATOMIC and ConceptNet) and extract utterance and goal events as a seed (1). We collect responses and situation statements for each seed by crowdsourcing (2-5), acquire more situation statements semi-automatically (6), and select distractor responses and situations to form response selection examples (7). We finally validate the examples manually (8). Steps (2) and (3-4) are executed in parallel.

assume that an external response generation system generates candidates based on the system's functionality and focus on picking the appropriate ones. SUGAR can also be used for training and testing generation systems on SUGAR, which is the focus of next chapter.

To train and evaluate a response selection system, each example must have distractors (negative responses), but typically, conversational datasets only contain ground truth responses. Thus, it has been commonly practiced to pick negative responses by random sampling (Henderson et al., 2019; Lowe et al., 2015). This approach is practically advantageous but may introduce negative responses that are clearly off-topic or false negatives (Akama et al., 2020; Hedayatnia et al., 2022). To alleviate this problem, we use an adversarial filtering algorithm (Bhagavatula et al., 2020; Sakaguchi et al., 2019; Zellers et al., 2018) to select competitive distractors and recruit crowd workers to rate candidates, allowing each example to have multiple acceptable responses.

5.3 Task and Data

The goal of this study is to provide a resource for developing a system that can observe situational information and return a proactive response to a user. We consider six categories of observable *situational information* (Table 5.1): location (where the user is), possession (what the user has), time, date, behavior (what the user is/was doing), and environment (temperature, etc.) We define a *proactive* response to be a response that provides *suggestions* to help users achieve their goals.

Category	Definition	Example
Location	Information about [user]'s current location.	[user] is home. / [user] is at the entrance of a house.
Possession	Information about what [user] possesses.	[user] owns a car. / There are apples in the kitchen.
Time	Information about time.	It's midnight. / It's morning.
Date	Information about date and season.	It's [user]'s birthday. / It's summer.
Behavior	Information about [user]'s behavior.	[user] just woke up. / [user] came back from jogging.
Environment	Information about non-user entities (person, objects, etc.).	The room is hot. / [user]'s car has a flat tire.

Table 5.1: Definitions of the situation categories.

5.3.1 Problem Formulation

Our task has five components: (1) a user utterance u, (2) situation statements $S = \{s_i\}_{i=1,\cdots,l}$, where l is the number of statements, (3) responses $R = \{r_i\}_{i=1,\cdots,m}$, where m is the number of response candidates[2], (4) their appropriateness ratings $Y = \{y_i\}_{i=1,\cdots,m}$, where y_i is a three-point Likert scale, and (5) an implicit goal g. S can include distractors that are not directly relevant to the conversation. u, S, and R are given as input, and the task is to re-rank R. Response selection systems are trained and evaluated by Y. In this study, we set $l = 12$ and $m = 3$. Table 5.2 shows an example.

5.3.2 Data

SUGAR contains 1,760 high-quality examples, each of which has three response candidates and 12 sentences of situational information. Table 5.3 shows the dataset statistics. We constructed the dataset with the eight steps shown in Figure 5.2. We describe them below.[3]

(1) Seed Utterance & Goal Selection: We harvested action and goal events from two commonsense knowledge bases, ATOMIC (Sap et al., 2019) and ConceptNet (Speer et al., 2017), where knowledge is represented as nodes representing events or concepts and edges connecting them with semantic relations. The collected action-goal node pairs served as the seed utterance-goal for soliciting responses and situational statements in the following data collection steps. First,

[2]We pick $m-1$ responses automatically such that they are less appropriate than the reference response in a given context (See Step 7). Nevertheless, there usually exist one or more acceptable responses to a given user utterance. We thus annotate all acceptable responses manually (Step 8).

[3]See also Appendix C.1 for technical details.

Utterance	Please turn on the TV.
Situations	It is evening now. [user] is home. [user] is in the living room. [user] is sitting on the couch. [user] has a TV in the house. [user] has an outfit on the bed. [user] has drinks and snacks in the kitchen. [user] has game cards on the shelf. The TV is off. [someone]'s birthday is today. There are several sports games available to watch. There is a basketball game scheduled.
Responses	Sure. Would you like me to check today's sports listings? *(Best)* Sure. Shall I pour a drink and bring some snacks for the game? *(Acceptable)* Sure, shall I select an outfit for you? *(Bad)*

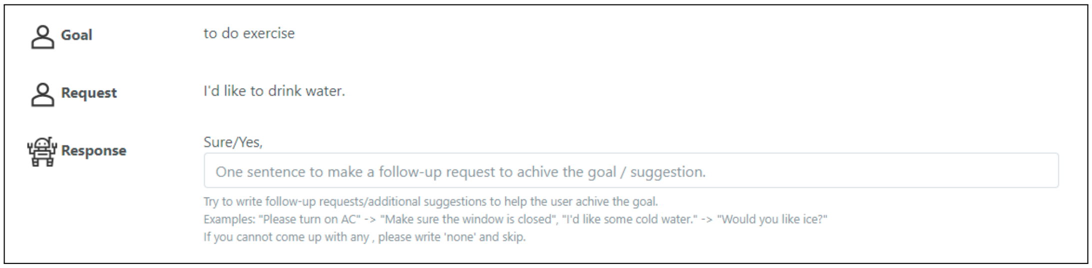
Table 5.2: Response selection example in SUGAR. Each example has 12 situational statements, some of which are distractors. [user] and [someone] are placeholders to denote person names.

we extracted nodes consisting of verb phrases (VPs) that appear at least five times within English request phrases (e.g., Please VP, Could you VP?, etc.) in the OpenSubtitles corpus (Henderson et al., 2019). These request expressions were also used as the surface form of u. Two of the authors then selected 563 events that can be achieved within a reasonable time span, can be assisted by someone else, and can be triggered by a goal. We retrieved their implicit goals g by goal-related edges in ATOMIC and ConceptNet. Specifically, we used `xNeed` in the reverse direction and `xIntent` in ATOMIC and `HasPrerequisite` in the reverse direction and `MotivatedByGoal` in ConceptNet. Finally, two of the authors evaluated the node pairs and picked 501 (u, g) pairs for which we can naturally say "I do u to achieve g." We also merged synonymous expressions (e.g., *go to a market* and *go to a supermarket*) into a single entry and corrected grammatical errors and unnatural phrases.

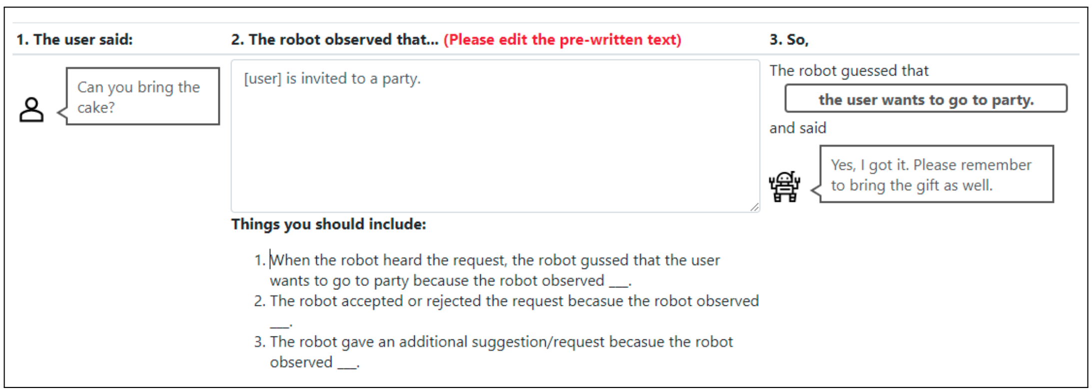
(2) Situation Collection I: We collected situation statements in two phases to simplify annotation work. The first phase focuses on u and g, and the second phase considers r in addition to u and g. In this step, we presented a pair of u and g texts to crowd workers and instructed them to specify situational information that is required to guess the goal based on the utterance. For example, an implicit goal "to cool off" can be naturally inferred by situations like "The user is home. The room temperature is hot." We asked workers to write *observable* facts in the six

	u	r	g	s
Unique sentences	380	1,738	431	4,450
Tokens	14,458	28,694	7,499	147,710
Tokens per example (mean average)	8.2	16.3	4.3	83.9

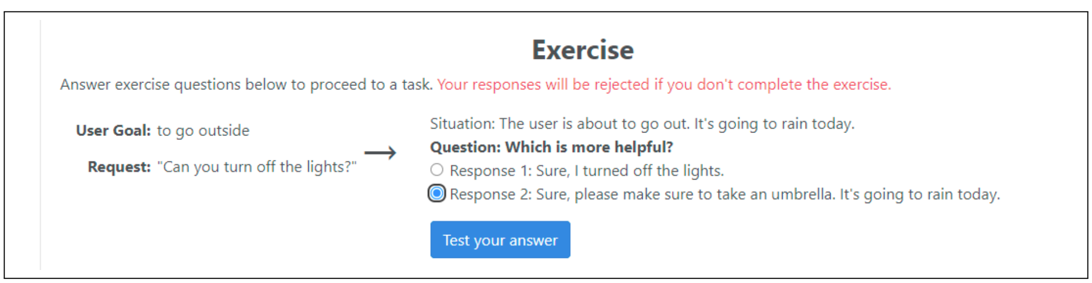
Table 5.3: Dataset statistics. The dataset contains 1,760 examples (33,794 sentences).



(a) Step 3 (Response Collection)



(b) Step 5 (Situation Collection II). The output of Step 2 is provided as an initial value.



(c) Exercise question. (This figure is for Step 3.)

Figure 5.3: Annotation interface for data creation. In addition to annotation guidelines, we provide one exercise question per task to train crowd workers. We used exercise questions in all the crowdsourced annotation tasks in our pipeline (c).

semantic categories (Table 5.1). For example, "The room temperature is hot." is valid, but "The user feels hot." is invalid as assistance systems cannot *observe* the user's feeling. We recruited one worker for each (u, g) pair and paid $0.12 per HIT[4] (one ($u$, g) pair/HIT).

(3) Response Collection: In parallel to Step (2), we recruited two crowd workers for each (u, g) pair to collect responses. The workers created at least two responses: one of the responses accepts and the other rejects the request. We asked the workers to write a *proactive* response, a response providing suggestions for goal fulfilment.[5] To solicit responses closely connected to implicit goals rather than to domain knowledge, we instructed the workers to avoid posing a clarification question like "Sure, I'll turn on the air conditioner for you. *Would you like it on a high or low setting?* (= clarification)" The workers were presented one u-g pair in each HIT and were paid $0.30/HIT. Figure 5.3a shows the annotation interface for this step.

(4) Response Validation: We present the utterances, goals, and collected responses to crowd workers and evaluated the helpfulness of the response. A response is considered to be valid if it satisfies the following criteria: (1) the response suggests or requests something new, and (2) the suggestion or request is helpful for achieving the goal. Each response was evaluated by three workers. We then picked the responses that were approved by two or three workers. We call a verified response *a reference response* r_1 hereafter. Each HIT contains up to seven responses, and one of them is a dummy question for evaluating crowd workers. For quality control, we filtered out crowd workers who participated in the task twice or more and did not reach 0.75% accuracy for the dummy questions. The workers were paid $0.18 for this task. Krippendorff's α was 0.547.

(5) Situation Collection II: We collected situation statements from crowd workers with the following two goals: (1) to collect situation statements that cover the reference response r_1 and (2) to verify the situation statements collected in Step (2). We presented (u, g, r_1) with the statements obtained in Step (2) and again instructed crowd workers to write observable facts. The results of Step (2) were provided as editable initial values, and we encouraged workers to update the texts when it is necessary. We recruited one crowd worker for each (u, g, r_1) with the reward of $0.42/HIT. Figure 5.3b shows the annotation interface for this step.

(6) Semi-automatic Situation Collection: We found that the collected situations were often under- or over-specified. We addressed this by automatic situation generation and manual verification.

One of the authors examined all the situation statements, discarded/modified inappropriate situations, and categorized them into six categories. We then used the cleaned and labeled texts to

[4] Human Intelligence Task, a unit of task in MTurk.

[5] For a response that rejects a user's request, we instructed the workers to provide a reason for rejection (*e.g.,* we cannot brew coffee *because we are out of coffee filters*) in addition to a suggestion.

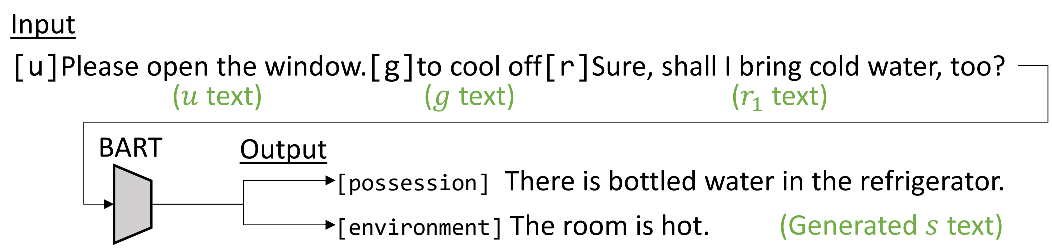


Figure 5.4: Example of automatic situation generation by BART (Step 6). [u], [g], and [r] are special symbols to denote the types of the following texts. The first output token is given as a prompt to control the semantic category of output.

Location	Possession	Time	Date	Behavior	Environment
1990	3546	1083	152	1699	2793

Table 5.4: Number of situation statements ($\in S_1$).

fine-tune a neural sequence-to-sequence to generate more situations. Specifically, we fine-tuned BART (Lewis et al., 2020a) trained on ATOMIC$^{20}_{20}$ (Hwang et al., 2021)[6] to take a concatenation of u, g, and r_1 as input and generate a text for a given situation category as illustrated in Figure 5.4. We performed a beam search of width 3 and took top-3 generation results for each input and relation. Finally, we manually verified the generated situations, resulting in 4,375 unique situations (6.4 ± 1.3 statements per example). We denote the situation statements attached to (u, g, r_1) by S_1. Table 5.4 shows the distribution of situation categories in SUGAR. Statements about possession and environment appear most frequently, which is reasonable because such situational information often decides actions that can be carried out (e.g., to drink coffee, coffee must be available). The other categories are less frequent, but 64% of examples have at least one time or date information, and 69% have a statement about behavior.

(7) Distractor Selection: The examples collected in the previous steps only contain reference responses r_1 and supporting situation statements S_1. We added $m - 1$ response candidates along with their relevant situational information as distractors so that all examples have m response candidates and l situation statements. We set $m = 3$ and $l = 12$. In this section, we describe the high-level idea of our algorithm. Appendix C.2 presents technical details.

Distractors can be obtained by random sampling as practiced in many studies (Henderson et al., 2019) or by advanced methods such as adversarial filtering (Gupta et al., 2021; Li et al., 2019). However, such approaches may introduce off-topic responses that are easy to rule out and false

[6]Note that the framework of pre-training Transformer models on common-sense knowledge bases was originally proposed by Bosselut et al. (2019).

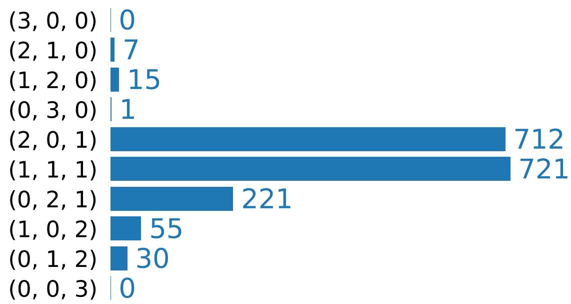


Figure 5.5: Result of rating annotations (Step 8). The labels denote (the number of *Bad* options, the number of *Acceptable* options, number of *Best* options). We removed one example with three *Acceptable* responses (0, 3, 0) from the dataset.

negatives — acceptable responses treated as negative examples, degrading system performance as well as reliability of evaluation (Akama et al., 2020; Hedayatnia et al., 2022).

To alleviate this problem, we combine lexical matching and adversarial filtering (Bhagavatula et al., 2020; Sakaguchi et al., 2019; Zellers et al., 2018) to construct distractors and validate them manually (see Step 8). We first created an initial dataset by a lightweight method based on sentence embeddings and lexical matching. We then performed $J = 3$ rounds of adversarial filtering. In each round, we split the dataset into $K = 10$ folds, and for each split, we trained a binary logistic regression classifier that takes sentence embeddings of u, S_1, and a response candidate. We computed sentence embeddings by SentenceTransformers (Reimers and Gurevych, 2019) with MPNet (Song et al., 2020). We used the trained classifier to identify easy distractors and replace them with more confusing ones with respect to the score function. We sampled two responses for each example. All response candidates in the same example have the same polarity. Finally, we expanded S_1, which only contains relevant information to u and r_1, to obtain a set of $l = 12$ situations S such that some of them are related to distractors but do not disqualify r_1, and statements do not contradict with each other. We again used sentence embeddings to find topically related situational information and avoid contradiction with keyword-based heuristics.

(8) Validation: There are usually multiple appropriate responses in one conversational context, and therefore, some of the challenging "distractors" picked in the previous step can be acceptable or even more appropriate than the reference r_1. To avoid introducing false negatives, we rated all response candidates on a three-point Likert scale (*Bad, Acceptable,* or *Best*) by crowdsourcing. We recruited three crowd workers per example with the reward of \$0.25/each and asked them to pick an appropriate response candidate (Krippendorff's α (Krippendorff, 2006) of 0.484). We then aggregated ratings by the statistical model proposed by Zhou et al. (2014) to obtain the final rating Y.[7] We discarded one example in this validation step and obtained 1861 examples with all responses rated. Figure 5.5 shows the annotation result. As we expected, a fair number of examples (56%) have more than one *Best* or *Acceptable* responses. One of the authors reviewed 61 examples (3.5%) where r_1 was rated as *Bad* and fixed contradicting situational statements. Examples without *Best* responses were also reviewed and revised if necessary.

5.4 Experiments

We evaluate several baseline models on SUGAR to explore two questions concerned with the nature of the proposed task and dataset: (1) Is understanding of situational information required to identify proactive responses in SUGAR? (2) Can standard matching-based systems capture relevant situational information and solve the task?

5.4.1 Baselines

We evaluate a lexical-matching approach and several Transformer-based response selection systems. A variety of neural networks have been proposed for the task of response selection Tao et al. (2021), but we opted to focus on the direct application of pre-trained Transformers rather than equipping them with extra modules/resources. Pre-trained models have proven effective in conversation tasks with minimal adaptation (Budzianowski and Vulić, 2019) and even achieves the best performance in a response selection task (Han et al., 2021).

TF-IDF ranker: As the simplest approach in our evaluation, we used a lexical-matching baseline system that ranks response candidates by cosine similarity of TF-IDF vectors of context and a response candidate (Lowe et al., 2015). We calculated TF-IDF weights on a training split with `scikit-learn` library.

Transformer ranker: We fine-tuned and evaluated four variants of Transformer-based rankers:

1. **BERT-FP** (Han et al., 2021): This model is an uncased $\text{BERT}_{\text{base}}$ that underwent additional training on the Ubuntu Dialogue Corpus (Lowe et al., 2015). The training process includes unsupervised post-training and supervised fine-tuning. As of 2023, this model is one of the leading systems on the Ubuntu dataset.
2. **BERT** (Devlin et al., 2019): We also tested an uncased $\text{BERT}_{\text{base}}$ without the additional training of Han et al. to analyze its benefits in our task. In the experiments of Hedayatnia et al. (2022), the BERT ranker performed similarly to BERT-FP.

[7] In the first run, all candidates were rated as equally good or bad in 18 examples. We updated and re-annotated 17 examples.

3. **RoBERTa** (Liu et al., 2019b): RoBERTa has the same architecture as BERT as a backbone but was trained using improved training configurations, resulting in better performance across multiple tasks and datasets. We used the pre-trained base model (12 layers $\approx$ 125M parameters)
4. **DeBERTa** (He et al., 2021a,b): DeBERTa is a model that improves upon BERT and RoBERTa by using disentangled attention mechanisms. In our experiments, we used the base DeBERTa v3 model (12 layers $\approx$ 86M parameters).

Following Han et al., we encoded a concatenation of input tokens, which will be explained in the next section, and a response option using these Transformer encoders. We then roduced a score of the option by a logistic regression classifier that takes the last hidden state of a special token, [CLS], at the first position in the input. Model parameters were optimized using Adam (Kingma and Ba, 2015) to minimize the max-margin loss.

5.4.2 Experimental Setup

Input format: We concatenated context and a response candidate for the Transformer rankers. To address our questions, we experimented with three variants of context:

1. u: Utterance (u)-only
2. $u + S_1$: Utterance (u) plus relevant situation (S_1)
3. $u + S$: Utterance (u) plus relevant and irrelevant situation (S)

Training and Test: We performed five-fold cross-validation (training:validation:test=6:2:2).[8] For each round, we trained a Transformer ranker for 10 epochs with a batch size of 32 and evaluated the model by nDCG@3 on the validation split every epoch. We then selected the best checkpoint for evaluation. To stabilize training, we applied weight decay of 0.05, set the maximum gradient norm to 5.0, and used a linear learning rate scheduler with 5% (≈ 20) warm-up steps. We further performed light-weight grid-search for hyperparameter tuning based on an average nDCG@3 score on validation splits, with learning rate $\in \{5e-5, 1e-5\}$, and margin for the max-margin loss $\in \{1.0, 0.5, 0.1\}$. One epoch of training took 1-2m on GeForce GTX TITAN X. We report the average Precision@1 and nDCG@3 on the test splits.

5.4.3 Results

The results in Table 5.5 shows the average test scores over a five-fold cross-validation. Two general patterns can be observed: (1) the Transformer-based models, except for BERT-FP, outperformed the TF-IDF baseline, and (2) the systems that were provided with the request utterance

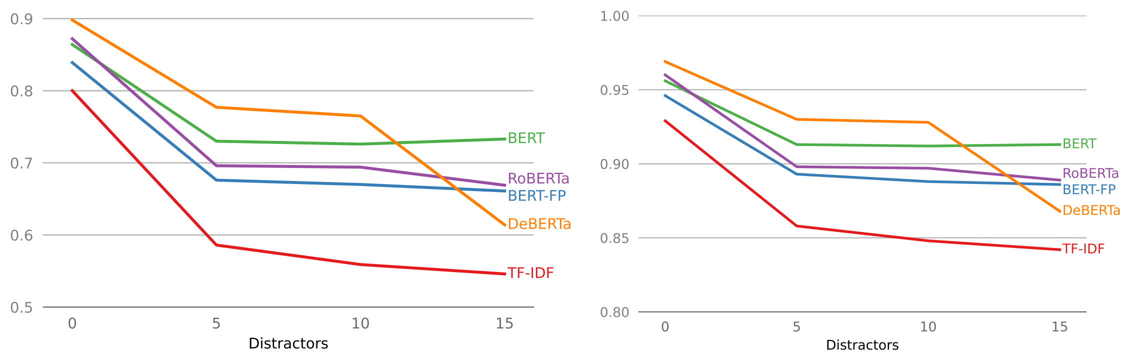
[8] We removed examples without *Bad* response options from the validation and test splits

System	Input	Precision@1	nDCG@3
TF-IDF	u	$.5993_{\pm.0223}$	$.8377_{\pm.0042}$
	$u+S_1$	$.7995_{\pm.0119}$	$.9289_{\pm.0042}$
	$u+S$	$.5683_{\pm.0121}$	$.8499_{\pm.0035}$
BERT-FP	u	$.6455_{\pm.0254}$	$.8799_{\pm.0076}$
	$u+S_1$	$.8386_{\pm.0280}$	$.9461_{\pm.0084}$
	$u+S$	$.6631_{\pm.0273}$	$.8869_{\pm.0094}$
BERT	u	$.7292_{\pm.0256}$	$.9102_{\pm.0071}$
	$u+S_1$	$.8637_{\pm.0109}$	$.9563_{\pm.0030}$
	$u+S$	$.7266_{\pm.0158}$	$.9110_{\pm.0038}$
RoBERTa	u	$.7178_{\pm.0273}$	$.9055_{\pm.0097}$
	$u+S_1$	$.8723_{\pm.0173}$	$.9596_{\pm.0059}$
	$u+S$	$.6992_{\pm.0230}$	$.9039_{\pm.0040}$
DeBERTa	u	$.7787_{\pm.0265}$	$.9305_{\pm.0074}$
	$u+S_1$	$.8981_{\pm.0112}$	$.9686_{\pm.0041}$
	$u+S$	$.7850_{\pm.0286}$	$.9314_{\pm.0084}$

Table 5.5: Average test scores over five-fold cross-validation.

u and relevant statements S_1 outperformed their counterparts with different input settings. In regard to the key questions, the results reveal several interesting findings:

1. Comparison of two input settings u and $u + S_1$ demonstrates that relevant situational information leads to a clear performance boost as expected (e.g., +0.13 in Precision@1 and +0.05 in nDCG@3 with BERT).
2. The performance gain in $u+S_1$ can be attributed to the increased word overlaps between the context and the correct responses, as indicated by the performance of the TF-IDF baseline. However, with the addition of distractors in the $u + S$ setting, the performance of the TF-IDF baseline dropped substantially (-0.20 in Precision@1 and -0.09 in nDCG@3). This result suggests that our dataset effectively avoids superficial clues, highlighting the importance of a higher-level understanding of situational context.
3. Interestingly, in the $u + S$ setting, the performance of Transformer rankers also decreased significantly to the same level as their corresponding systems without situational statements in the input (the u setting).
4. Additional pre-training of BERT-FP was not effective in our task, which is consistent with the observation of Hedayatnia et al. (2022). We speculate that this is due to a domain mismatch of training corpora. BERT-FP is pre-trained on technical topics related to Ubuntu,



(a) Precision@1 (b) nDCG@3

Figure 5.6: Average test scores over five-fold cross-validation with different numbers of distractors

whereas SUGAR concerns a wider range of topics in daily life.

These findings provide valuable insights into our research questions. First, the understanding of relevant situational context helps systems select proactive responses accurately, indicating that SUGAR is an effective resource for the development and evaluation of situated conversation systems. Secondly, it is challenging for Transformer rankers to identify useful clues from a mixture of relevant and irrelevant situational context.

5.4.4 Robustness to Distractors

The results presented in the previous section indicate that Transformer rankers have difficulty distinguishing relevant and irrelevant information. To explore this further, we evaluated these rankers with varying numbers of distractors to examine if their performance continues to decline as more distractors are added.

In this experiment, we controlled the number of distractors by creating instances with 5, 10, and 15 distractors. Situational statements were randomly added as necessary. We trained and tested the same response rankers following the same setup, with the exception that we fixed the learning rate to 5e-5, which generally produced better results than 1e-5 in the main experiments. It is important to note that the first 1-7 distractors were adversarially selected (§5.3), while the remaining distractors were added at random.

Figure 5.6 displays the precision@1 and nDCG@3 scores of the response rankers. The performance of TFIDF indicated that the addition of random distractors slightly increased the word overlap rates between input and distractor responses, but not substantially. However, as hypothesized, all systems demonstrated decreasing scores as more distractors were included. Interestingly, the performance of the advanced models, RoBERTa and DeBERTa, decreased drastically as more distractors were added ($0.87 \rightarrow 0.67$ for RoBERTa and $0.90 \rightarrow 0.61$ for DeBERTa in Precision@1). We speculate that these models are powerful but also susceptible to overfitting spurious patterns between situational statements and response options, resulting in low test scores. In contrast, the BERT-based rankers were more robust to distractors, but their absolute performance remained low (Precision@1 of 0.73 and nDCG@3 of 0.91 for BERT). This finding highlights the need for future work to develop models that are more robust to the inclusion of irrelevant situational context.

5.5 Limitations

Data size: SUGAR is relatively small compared to recently published datasets. This is due to the complexity of our problem setting and annotation pipeline. We prioritized quality over quantity and performed multiple steps of manual intervention to reduce errors, false negatives, and annotation artifacts. These problems have been reported in various NLP tasks not limited to conversational tasks (Akama et al., 2020; Elazar et al., 2020; Gururangan et al., 2018). Nonetheless, our experiment has shown that pre-trained Transformer models can be trained to outperform a TF-IDF ranker by a clear margin, which is encouraging. In addition, we could automatically induce noisy but large-scale training instances from existing resources, for example, by harvesting event pairs that can be used as u and r from event knowledge bases such as ATOMIC$^{20}_{20}$and generating situation statements using our generator (§5.3).

Representation of situation information: In SUGAR, situation information is represented in textual expressions. In real-world applications, such information could be collected via external APIs (e.g., calendar and map) and sensors (e.g., camera) and stored in non-textual forms. Our study is a proof-of-concept that shows the understanding of situational information is very important for response selection. Future research should explore ways to process situation information that is expressed in other forms of data (e.g., structured texts, numbers, images). Even if the value is structured or images, we could transform them into textual forms as done in data-to-text research (Miura et al., 2021; Shen et al., 2020). Besides, we acknowledge that situational information is often under-specified in SUGAR because some information is considered to be common-sense (e.g., a room has a door) or presupposed (e.g., "Please open the door" presupposes that the door is closed.), and such information was not explicitly stated by human annotators during data collection. Therefore, response selection systems should be equipped with a mechanism to handle implicit knowledge to solve the task.

5.6 Conclusion and Future Work

We proposed a task of situated proactive response selection for developing and evaluating conversational assistants that can help users proactively in various help-seeking scenarios. We constructed a dataset of 1.7k examples by crowdsourcing and semi-automatic generation.

There are several interesting directions for future research. First, as shown in our experiments, it is challenging to pick up relevant situational information and use it to reason about user requests and potential assistance. To achieve this, conversational systems will need to be equipped with world knowledge to effectively align situation information with an interaction. One promising approach is knowledge-based response models such as graph neural networks, which recently has shown to be effective in various NLP tasks (Zhang et al., 2020; Zhou et al., 2022b; *inter alia*). Second, although we leveraged implicit goals only for soliciting proactive responses in data collection in this study, understanding of goals should be necessary for building better conversation engines as claimed in early studies (Allen and Perrault, 1980; *inter alia*). We believe SUGAR can facilitate future research in this direction.

Chapter 6

Towards Situated Response Generation

Overview

The development of open-domain conversation systems has been advanced by a plethora of automatically or manually constructed datasets of conversations. However, most of these datasets lack situational context information, such as time and location, that plays a crucial role in human communication. This absence of situational context can lead to an under-specification of the problem space, resulting in undesired behavior by the system. This chapter explores the benefits and challenges associated with processing contextual information. An analysis of response generation using SUGAR and two external datasets shows that explicitly providing situational context information enhances the coherence and specificity of generated responses. However, further experiments reveal that generation systems struggle to distinguish relevant contextual information from irrelevant ones, which aligns with the findings in the previous chapter. To address this challenge, automatic filtering methods that leverage topical similarity and implicit knowledge are evaluated. This evaluation provides valuable insights into the problem and directions for future research.[1].

6.1 Introduction

Human communication heavily relies on information about a situation–a set of observable information about relevant objects, their properties, and relationships in the world at a specific time and place. Crucial information can often be implicitly assumed during a conversation, particularly when it can be inferred from co-observed facts (e.g., "They are talking at the front desk in a hotel.") or shared a priori (e.g., "the person was staying in the hotel").

Incorporating such information is essential for both the development and evaluation of conversational technologies. To illustrate this point, we show a conversation example from Daily-

[1]This chapter is based on Otani et al. (2023b)

A:	Hello. I'm leaving. Here is my key.
B:	[Sytem output]
BlenderBot2: Thank you. It is very kind of you. Do you want me to lock the door?	
GPT-3: Thanks. Have a safe trip.	
ChatGPT: Thank you for leaving the key. Have a good day.	
+They are in the hotel. A has a car and is carrying a suitcase. It is raining outside.	
BlenderBot2: Sure. Do you want me to load your luggage in the car? It's raining.	
GPT-3: Thank you for staying with us. Do you need help with your luggage?	
ChatGPT: Can I help you with anything, sir? Is there something you need assistance with in your room or with your car?	

Table 6.1: Responses from three systems with and without situational information as input. When the context is unknown, systems produced responses based on their made-up assumptions (top). However, when a few statements of context are given, all the systems recognized the same scenario *(A is checking out of a hotel)* and generated richer and more cooperative responses (bottom). See Appendix D.1.2 for the generation setup.

Dialog (Li et al., 2017), along with responses generated by three systems: BlenderBot2 (Komeili et al., 2022; Xu et al., 2022), GPT-3 (Brown et al., 2020), and ChatGPT (GPT-3.5) in Table 6.1.[2] DailyDialog is a widely used[3] dataset of multi-turn conversations in English. The original example does not describe a surrounding environment explicitly, resulting in ambiguity regarding the situation. Person A could be a traveler leaving a hotel or someone handing over their house key, among other possibilities. The response generated by BlenderBot2 is somewhat relevant to the latter situation but clearly inappropriate in the former. In contrast, the response generated by GPT-3 is appropriate in the latter situation but not in other contexts. ChatGPT's response is neutral, though less engaging. This ambiguity underscores the fundamental problem caused by the *underspecification* of the situation. The provision of situational information, such as "they are in a hotel," narrows down the range of ideal behaviors, which helps generation systems produce context-specific responses and establishes a more solid standard for judging quality. This issue is not limited to this particular dataset. Many common open-domain conversational datasets contain little or no additional information besides conversation history (the Twitter dataset (Ritter et al., 2011); DREAM (Sun et al., 2019); MuTual (Cui et al., 2020); *inter alia*). This task setting, which requires systems to infer almost all information solely from previous utterances, poses unnecessary challenges and may lead to undesired system behavior.

This chapter discusses the current state of open-domain conversational datasets concerning

[2]See Appendix D.1.2 for the generation setup.

[3]Based on Semantic Scholar, the dataset paper (Li et al., 2017) is cited by over 700 papers as of April 2023.

how situations are represented (§6.2). Specifically, we consider situational statements[4] that provide partial information about immediately observable (e.g., today's weather), commonly known (e.g., umbrellas are often used on rainy days), or directly derivable facts related to the task, speaker, and goals (e.g., the hotel's check-out and a guest's required action). Some of these elements have already been effectively integrated into modern conversational systems, particularly for closed-domain, task-oriented dialogues. Situational information can be represented in multiple ways, including formal logic and natural language statements. We argue that open-domain conversational tasks and datasets should be equipped with some form of situational information. Additionally, we conducted case studies on several datasets to explore the potential benefits and challenges associated with situational information (§6.3). Our analysis indicates that distinguishing between relevant and irrelevant situational information can be challenging for data-driven response engines. To address this challenge, we evaluate automatic filtering methods that use topical similarity and knowledge-based relevance. We demonstrate that the combination of these two approaches achieves a ROC-AUC score of 0.82 in the situation filtering task on SUGAR, which can lead to better response generation results.

6.2 Status Quo

In open-domain response generation tasks, systems generate responses in natural language based on input dialog history (a list of utterances from previous turns). Dialog history often serves as the primary, and sometimes sole, source of context information in many datasets. In this section, we discuss how conventional task design can be improved through the explicit inclusion of situational information.

6.2.1 Open-domain Conversational Datasets

The recent advancement of open-domain conversational technologies can be largely attributed to the development of large-scale conversation datasets, which facilitate the training of data-driven language generation models. However, many commonly used datasets lack crucial situational information. Below, we provide a brief overview of representative datasets in the field.[5]

Collection of naturally occuering conversation data can be costly (Godfrey et al., 1992). This bottleneck was greatly alleviated by websites that contain naturalistic textual conversations. For instance, millions of conversations can be scraped automatically from Twitter (Ritter et al., 2010). Likewise, many large-scale datasets were produced from social media (Henderson et al., 2019; Shang et al., 2015; Sordoni et al., 2015; Wang et al., 2013). While conversations on social media are

[4]The situation of a conversation consists of numerous predicates that describe various aspects of surroundings. By *a situational statement*, we mean a single predicate that describes part of a situation.

[5]For a more comprehensive literature review, refer to survey papers on available resources (Kann et al., 2022; Serban et al., 2017).

essentially text chat and do not cover many of the dailylife interactions, online language learning coursewares contain conversation examples in diverse scenarios (Cui et al., 2020; Li et al., 2017; Sun et al., 2019). DailyDialog (Li et al., 2017) is one of the datasets built from English learning materials and 13k multi-turn conversationswe spanning various topics and scenarios. Although these datasets are generally large and effectively used for pre-training language models (Humeau et al., 2020; Shuster et al., 2020), they contain only conversation history.

Some prior studies have created conversational datasets enriched with various semantic and pragmatic features. Notably, multi-modal and task-oriented datasets generally allocate dedicated representations for essential situational information such as physical signals (Haber et al., 2019; Moon et al., 2020) and task-specific information or domain knowledge (Budzianowski et al., 2018), but their coverage is limited to one or a few specialized domains. For open-domain conversation systems, the use of focused information has been explored for improving response quality, such as related documents (Dinan et al., 2019; Zhou et al., 2018) and user-based features such as persona (Dinan et al., 2020; Majumder et al., 2020; Zhang et al., 2018), emotion (Rashkin et al., 2019), social norms (Kim et al., 2022), and behavior (Ghosal et al., 2022; Zhou et al., 2022a). Sato et al. (2017) explored the utilization of time information as well as user types for analyzing conversations on Twitter. Though these studies demonstrate that integrating surrounding information improves response quality in aspects such as informativeness and engagement, the scope has been limited to specific modalities, domains, and semantic categories. Moreover, detecting certain features, like internal emotion and plans, can be non-trivial in practice. Observable situational information has received little attention. While SUGAR (Chapter 5) aims to represent such information in free-form English texts, the available resources are limited, and it remains unclear whether existing datasets can be extended to include situational information.

6.2.2 Necessity of Situational Information

Most importantly, the absence of situational information leads to the underspecification of the problem space. Without knowing the situation in which an utterance is expressed, its interpretation cannot always be determined. For instance, the request "please call Pat" could mean at least two actions: speaking to Pat in person or making a phone call.

Additionally, without sufficient knowledge of the world state, systems may produce meaningless or contradictive responses even if they appear natural. In the research community, the inconsistency within generated responses is recognized to be one of the unsolved problems (Nie et al., 2021; Shuster et al., 2022). This problem may be attributed to the underspecified task setting. As previous examples suggest, the interpretation of human communication often relies on unspoken information. When situational information is absent, systems must assume implicit parameters of the world state on their own, which may not always be correct. Furthermore, training on this problem formulation may force systems to learn superficial patterns.

The challenge of evaluating conversation systems is also compounded by the broadness of

	Training	Validation	Test	Avg. turn
SUGAR	1,214	102	25	1.0
CICERO	15,171	5,325	25	3.0
ConvAI2	16,878	1,000	25	4.7

Table 6.2: Datasets used in this study. For manual evaluation, we sampled 25 examples from the test split of each dataset (not presented in this table).

the problem space. Previous studies have discredited the use of automatic evaluation methods in response generation tasks (Liu et al., 2016). Although techniques such as considering multiple reference responses may alleviate this problem to some extent (Sai et al., 2020), it remains a significant challenge. Furthermore, even in the task of response selection, reliably evaluating system output is non-trivial due to the potential for false negatives when confusing distractor statements are included in the pool of candidate responses (Hedayatnia et al., 2022).

6.3 Situated Response Generation

In order to analyze the impact of incorporating situational information into response generation, we conducted an empirical analysis using two neural generation models and three English datasets.[6]

6.3.1 Datasets

We used the following English datasets.

1. SUGAR This dataset consists of single-turn conversations in different help-seeking scenarios. Each example includes 12 sentences that describe situational information across six categories, including date, time, location, speaker's behavior, environment, and speaker's possession (§5.3). Some of the statements are irrelevant and serve as *distractors.* SUGAR represents datasets that provide rich situational information.
2. CICERO (Ghosal et al., 2022): This dataset is a compilation of three datasets, including DailyDialog (Li et al., 2017), MuTual (Cui et al., 2020), and DREAM (Sun et al., 2019). CICERO is an example of conversational datasets that do not explicitly present situational information.[7]
3. ConvAI2 (Dinan et al., 2020; Zhang et al., 2018): This dataset is designed for persona chats, with each conversation featuring the speaker's persona information in 4-5 sentences.[8] ConvAI2 is

[6] The purpose of this analysis is to find out if there are any notable patterns associated with the inclusion of situational statements rather than benchmarking response generation systems.

[7] Although CICERO includes annotations of common-sense reasoning about target utterances, we did not use them as they include unobservable facts. We only used CICERO for the pre-filtering it underwent.

[8] We used `revised` persona statements.

A	Hi, Mike! how are you feeling now?
B	How did you know I was here? is it Tom?
A	I was talking with Bob yesterday and I learnt your right leg had been injured. How did it happen?
B	[System output]
Generated situational statements	
Person B's leg had a surgery last night. It is afternoon now. Person A and Person B are in the hospital. Person B injured his right leg when he was playing baseball. Person A has been informed. Person A has a phone. Person B has a leg brace on. Person B's leg is injured. Person B's leg is getting better. Person A's car is in the parking lot.	

Table 6.3: An example of generated situational statements. This conversation is taken from the CICERO dataset. These statements represent *an assumption* about the situation. In practice, situational information is *perceived* in some way rather than generated.

a dataset with user-based features.

We selected 25 test instances for manual evaluation from the test split of each dataset. For CICERO and ConvAI2, which consist of multi-turn conversations, we randomly selected one target turn from each dialogue. We chose targets of test instances the second to the fourth turn to reduce the cognitive load during evaluation. As the test split for ConvAI2 is not publicly available, we used its validation split as our test data and selected 1,000 examples for validation from the training split. Table 6.2 shows the dataset sizes after our filtering process.

6.3.2 Generating Situational Statements

CICERO and ConvAI2 do not contain descriptions of situational information. We utilized a Transformer-based generation model to automatically generate situational statements for these datasets, which allowed us to analyze how systems could generate situated responses within a specific context (See Appendix D.1.1 for details). Table 6.3 shows an example of generated situational statements.

To generate the situational information descriptions, we used the SUGAR dataset to fine-tune $\text{Comet}^{\text{DIS}}_{\text{TIL}}$ (West et al., 2022), which is a GPT-2-XL model (Radford et al., 2018) trained on commonsense knowledge data. We concatenated a previous utterance, a response, and a reference situational statement into one sequence and trained the model to minimize a cross-entropy loss over the situation part. We also fine-tuned another $\text{Comet}^{\text{DIS}}_{\text{TIL}}$ (West et al., 2022) model without reference responses in input to avoid including the gold-standard information in testing instances. In input sequences, each text was headed by special symbols indicating the text type: <utterance> for an utterance, <response> for a response, and <situation category> for a situational statement. The <situation category> symbol is one of date, time, location, behavior, environment, and possession.

Using the fine-tuned model, we added 10 situational statements to each example, including one each for date, time, location, and behavior, and three each for environment and possession. Finally, for quality control, one of the authors manually checked the test samples from CICERO and ConvAI2 (25 for each) and corrected context statements when required (e.g., conflicting facts). The reference responses were hidden during the manual verification to avoid bias. This manual verification process ensures the quality of the test dataset in order to minimize the confusion of annotators in the following manual evaluation of responses.

6.3.3 Setup

Systems:

1. BlenderBot2: A Transformer-based response generation model that is pre-trained on multiple conversational datasets. We used a distilled 400M-parameters model in the ParlAI library (Miller et al., 2017).
2. GPT-3: A Transformer-based causal language model that is pre-trained on a massive collection of documents. We used GPT-3-DaVinci (175B parameters) through OpenAI API. For each dataset, we manually selected four high-quality training examples and embeded them in a prompt.

We fine-tuned BlenderBot2 on the mixture of the aforementioned datasets in a multi-task learning setting. We up-sampled SUGAR and CICERO to balance the data sizes. To alleviate the randomness of system output, we trained two BlenderBot2 models with different random seeds, and for each model, we generated one response by beam search with width 2. We obtained top-2 generations from GPT-3 with a beam width of 4. Appendix D.1.2 describes implementation details.

Evaluation: We recruited three annotators on Amazon Mechanical Turk to evaluate each response.[9] We employed three criteria: (1) grammaticality (whether the response is grammatically correct), (2) Coherence (whether the response is coherent and contextually appropriate), and (3) context-specificality (whether the response is specifically relevant to the given context.) The latter two criteria were defined based on prior work (Thoppilan et al., 2022; Zhou et al., 2022a).[10] Table 6.4 shows some examples. We collected a total of 1,800 binary judgments for each criterion in our evaluation. The inter-annotator agreement was relatively low, with a Fleiss' kappa of 0.38, likely due to the subjective nature of the quality assessment. The agreement for evaluating BB2 was notably low, possibly because its responses are often not fluent, making it difficult for annotators to rate individual criteria. In contrast, GPT-3's responses were fluent, resulting in a Fleiss'

[9]Prior work provides multiple empirical results showing that automatic evaluation metrics are not strongly correlated with human judgments (Deriu et al., 2022; Liu et al., 2016). Therefore, we solely rely on manual evaluation in this study. See Appendix D.2 for details.

[10]We adopted the definitions of Zhou et al. with modified terminology.

	Input	Situation	Response	Coh.	Spe.
1	Please start my car.	The speaker's car is functioning.	Sure, I'll do that right now.	✓	
2	Please start my car.	The speaker's car is out of gas.	Sure, I'll do that right now.		
3	Please start my car.	The speaker's car is out of gas.	Sorry, but I'm afraid your car is out of fuel. Would you like me to call a taxi for you instead?	✓	✓
4	Please start my car.	The speaker's car is out of gas.	Sorry, but I can't. Would you like me to call a taxi for you instead?	(*)	✓
5	What is the weather like today?	It's currently raining outside.	It's sunny and warm.		✓
6	Can I get some beer, please?	No beer is left, but wines are available.	Yes, I'll get it right now for you.		
7	Can I get some beer, please?	There are beer and some snacks in the kitchen.	Yes, I'll get it right now for you. Would you also like me to bring some food?	✓	✓

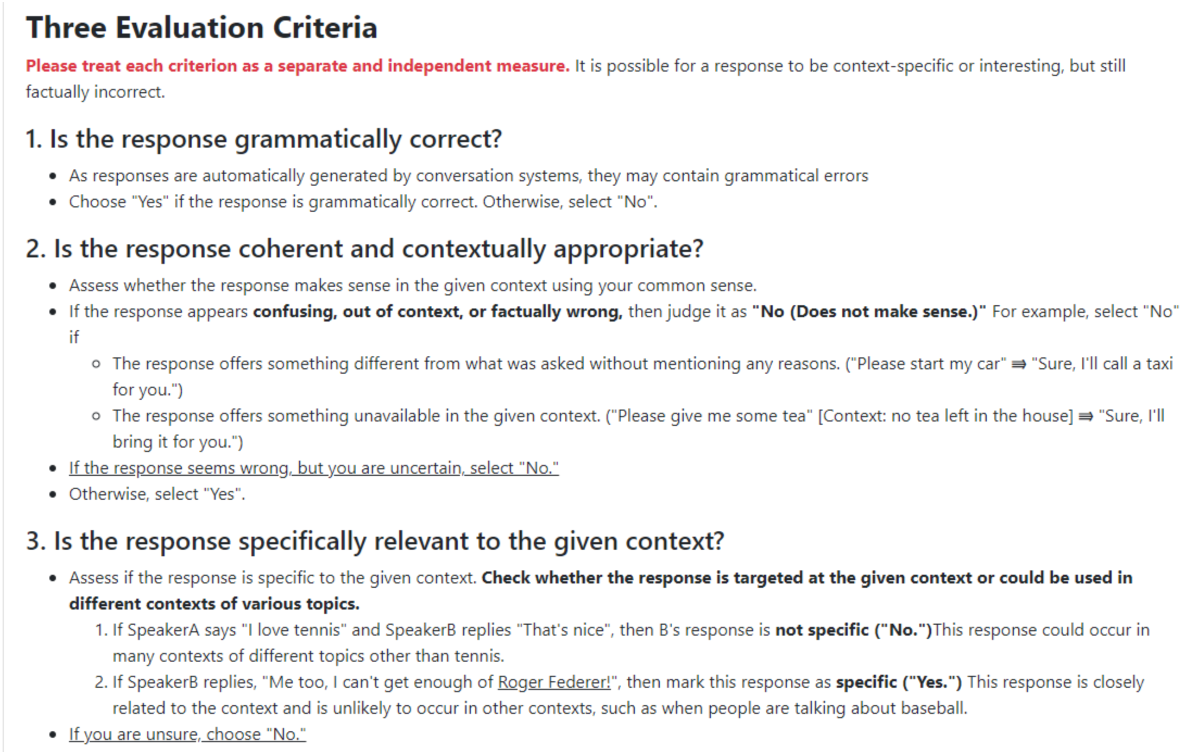
Table 6.4: Examples for the coherence (denoted as Coh) and context-specificity (denoted as Spe) ratings. ✓ means the criterion is satisfied. (*) The response is not coherent because it does not explain why the car is unavailable.

kappa of 0.48. To minimize the impact of this variance, we calculated the final score of each response by averaging the ratings of three annotators and manually reviewed responses with high variance.

Evaluation: We recruited three annotators on Amazon Mechanical Turk to evaluate each response[11]. We evaluated if the responses are grammatically correct, coherent, and context-specific. The latter two criteria were defined based on prior work (Thoppilan et al., 2022; Zhou et al., 2022a).[12] We collected a total of 1,800 binary judgments for each criterion in our evaluation. The inter-annotator agreement was relatively low, with a Fleiss' kappa of 0.38, likely due to the

[11]Prior work provides multiple empirical results showing that automatic evaluation metrics are not strongly correlated with human judgments (Deriu et al., 2022; Liu et al., 2016). Therefore, we solely rely on manual evaluation in this study. We recruited human evaluators in Amazon Mechanical Turk. Our evaluation task does not collect any personal information other than anonymized worker IDs and country of residence (due to our location-based worker qualification). We decided the task reward (= \$0.30/HIT) based on trial studies so that the estimated hourly rate would reach at least \$9.00.

[12]We adopted the definitions of Zhou et al. with modified terminology (sensible $\rightarrow$ coherent and specific $\rightarrow$ context-specific).


Three Evaluation Criteria

Please treat each criterion as a separate and independent measure. It is possible for a response to be context-specific or interesting, but still factually incorrect.

1. Is the response grammatically correct?

- As responses are automatically generated by conversation systems, they may contain grammatical errors
- Choose "Yes" if the response is grammatically correct. Otherwise, select "No".

2. Is the response coherent and contextually appropriate?

- Assess whether the response makes sense in the given context using your common sense.
- If the response appears **confusing, out of context, or factually wrong,** then judge it as **"No (Does not make sense.)"** For example, select "No" if
  - The response offers something different from what was asked without mentioning any reasons. ("Please start my car" ⇛ "Sure, I'll call a taxi for you.")
  - The response offers something unavailable in the given context. ("Please give me some tea" [Context: no tea left in the house] ⇛ "Sure, I'll bring it for you.")
- If the response seems wrong, but you are uncertain, select "No."
- Otherwise, select "Yes".

3. Is the response specifically relevant to the given context?

- Assess if the response is specific to the given context. **Check whether the response is targeted at the given context or could be used in different contexts of various topics.**
  1. If SpeakerA says "I love tennis" and SpeakerB replies "That's nice", then B's response is **not specific ("No.")**This response could occur in many contexts of different topics other than tennis.
  2. If SpeakerB replies, "Me too, I can't get enough of Roger Federer!", then mark this response as **specific ("Yes.")** This response is closely related to the context and is unlikely to occur in other contexts, such as when people are talking about baseball.
- If you are unsure, choose "No."

Figure 6.1: Evaluation criteria presented to crowd workers.

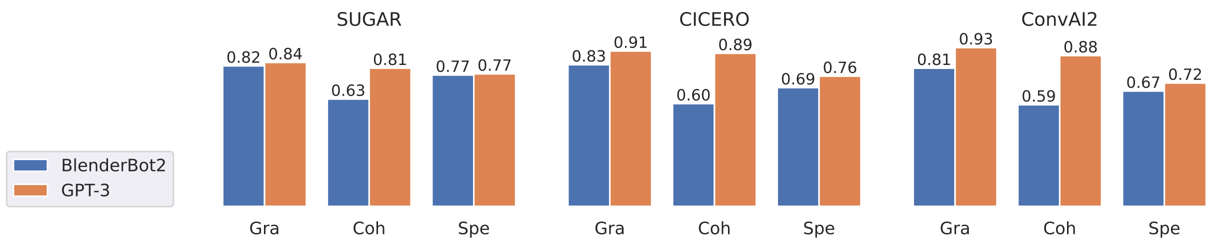
subjective nature of the quality assessment. The agreement for evaluating BlenderBot2 was notably low, possibly because its responses are often not fluent, making it difficult for annotators to rate individual criteria. In contrast, GPT-3's responses were fluent, resulting in a Fleiss' kappa of 0.48. To minimize the impact of this variance, we calculated the final score of each response by averaging the ratings of three annotators.

6.3.4 Results

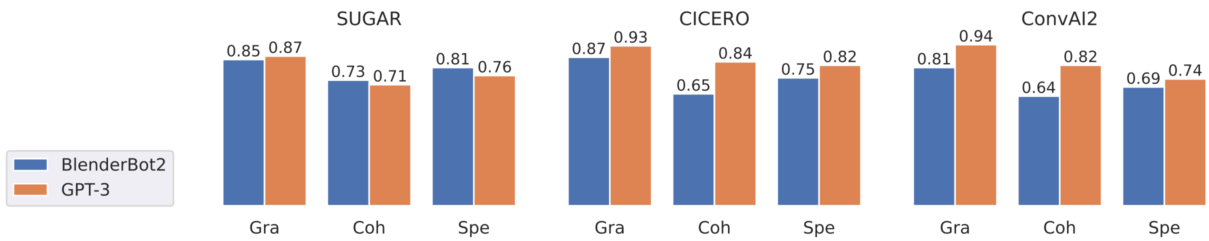
Figure 6.2 shows the average scores of responses. shows the average scores of responses. GPT-3 was highly capable of generating fluent responses based on its internal assumptions about the situations. The annotators was likely inferred the assumed situations to interpret the GPT-3's responses, resulting in higher ratings. In the example below, GPT-3 produced appropriate responses regardless of the presence of situational information.

Example 1

A: "Hi, Mike! how are you feeling now?"
B: "How did you know I was here? is it Tom?"
A: "I was talking with Bob yesterday and I learnt your right leg had been injured. How did it happen?"



(a) Without situational context in input



(b) With situational context in input

Figure 6.2: Manual evaluation results. The values displayed represent the average scores of responses in three criteria: (1) grammatical correctness, (2) coherence, and (3) specificity to the context.

GPT-3 (Without situational information in input)
(a) "Yeah, I slipped on the stairs and I sprained my ankle."
(b) "It was a sports accident. I was playing basketball and I sprained my ankle."

GPT-3 (+Situation: "Person A and Person B are in the hospital.", ..., "Person B injured his right leg when he was playing baseball.")
(c) "Yeah, I was playing baseball and I twisted my ankle."
(d) "Yeah, I had a surgery last night. It happened when I was playing baseball. I'm feeling much better now, thanks."

On the other hand, BB2 struggled to identify generalizable patterns through fine-tuning when situational information was not provided. This was evident in SUGAR, where only a single utterance was provided as input to the system. The following example shows responses generated by BB2.

Example 2 *A:* "Could you help me take a bus?"

BB2 (Without situational information in input)
(a) *"Sure. shall I book a taxi as well? I can also book a ride for you to and from work."
(b) "Sure. would you like me to bring your bags, too? they are heavy."

BB2 (+Situation: "The speaker has a bus schedule." ... "There is a backpack near the desk.")
(c) "Sure. shall I pull up your bus schedule? it will show you when the bus is coming."

	[· · ·]
A	I just have my dog, trooper.
B	My wife is thinking about getting our kids a dog. What kind is yours?
A	[System output]
Without context	
1st: I have **a border collie.** He's a great companion.	
2nd: He's **a lab mix.** He's a great companion.	
+A has a pet bulldog named Trooper.	
1st: Trooper is **a bulldog.** He's the best!	
2nd: He's **a bulldog.** He's a great companion!	

Table 6.5: Top-2 response generation results from GPT-3. The example was taken from CICERO.

(d) "Sure. would you like me to get your bag ready? it will go in your backpack."

BB2 produced more coherent and context-specific responses when situational information was explicitly provided. However, as the examples above indicate, BB2's responses were not as fluent as GPT-3's, resulting in lower overall performance.

We observed that in some examples, GPT-3 selected information that was not directly relevant to the immediate previous utterance and returned unnatural responses like the example below.

Example 3 *A:* "Could you help me stretch?"

GPT-3 (Without situational information in input)
"Absolutely. Let me know which muscles you want to focus on and I can show you some stretches."

GPT-3 (+Situation: "The speaker is wearing a jogging suit and running shoes.", ..., "There is a beach ball at home." ... "The nearest beach is open.")
*"Of course! Let's go to the beach and play with the beach ball while we stretch."

Furthermore, as shown in Example 1 and Table 6.5, semantic consistency within the top-k responses improved when more situational information is given, although this was not evaluated in our experiments. This behavior is fundamentally rooted in the design of the task. Having inconsistent and mutually-exclusive top-k candidates is not a problem when systems need to ground them on nothing.

6.3.5 GPT-3's Robustness to Distractors

GPT-3 demonstrated superior performance in response generation, but our observations suggest that it can be misled by irrelevant situational statements (distractors) and return suboptimal responses. To further investigate the model's robustness to distractors, we evaluated GPT-3 on SUGAR with varying numbers of situational statements.

Experimental Setup: We controlled the number of distractors by extending the SUGAR dataset, whose examples have about six distractors. We created instances with 5, 10, and 15 distractors by sampling statements from the whole dataset as necessary. We selected 30 examples at random for this experiment and manually fixed contradictions within situational statements. We generated two responses for each example using GPT-3 through OpenAI's API, employing the same hyper-parameter setting used in the previous experiment.

Evaluation: We recruited two annotators on Amazon Mechanical Turk to evaluate each response. Based on previous annotation results, we updated the annotation guidelines and qualification criteria[13], resulting in substantial agreement (Fleiss' kappa of 0.76). We calculated average scores for three dimensions: (1) grammaticality, (2) coherence, and (3) context-specificity.

Results: The evaluation scores of GPT-3 are presented in Figure 6.3. The grammaticality and context-specificity of responses remained at the same level regardless of the number of distractors. On the other hand, the coherence scores exhibit a different trend. The scores are high when no or five distractors were presented (0.86 and 0.90 for 0 and 5 distractors, respectively). As more distractors were added, the coherence scores decreased (0.81 and 0.77 for 10 and 15 distractors, respectively). This is exemplified by the following model outputs.

Example 4 *A:* "Please lock the door."

No distractors ("The speaker is dressed in nice clothing." ..., "The door is unlocked.", ..., "The windows are open.")
"Sure, and do you want me to close the windows too?"

5 distractors (+ "The speaker has a printout titled, Interior design for beginners.", "Person A's doctor told him he should lose weight.", ...)
? "Sure, and do you need any help with the interior design project?"[14]

10 distractors (+ "The speaker has laundry piled up in the room.", "Person A has scissors in the house.", ...)
? "Sure, and should I turn off the windows too?"

15 distractors (+ "Person A has a telescope in the house.", "Person A owns an exercise bike.", ...)
*"Sure, and would you like me to help you with the laundry too?"

The coherence score was relatively low (0.86) when no distractors were presented. We conjecture the model may have learned to copy phrases to generate responses instead of reasoning over input texts. We found that the model's responses tended to sound abrupt, as shown in the following example:

Example 5 *Person A:* "Can you assist me with brushing my teeth?"

[13]See Appendix D.2.

[14]This response might be acceptable given that the speaker has a printout about interior design.

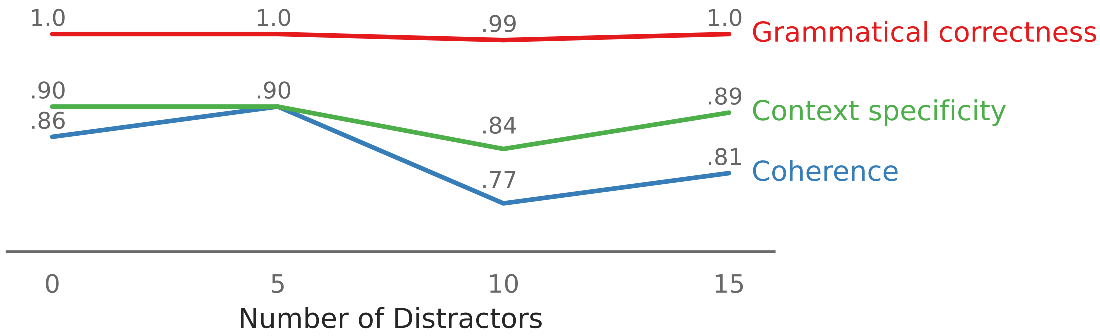


Figure 6.3: Manual evaluation results with varying number of distractors.

No distractors ("The speaker has a dental cleaning scheduled.", ..., "There is no more toothpaste in the house.")
*"Sure, I can help you get some toothpaste at the store. Would you like me to come with you?"

6.4 Situation Filtering

The empirical analysis presented so far shows that providing situational information can enhance the quality of responses generated by response generation systems. However, it is difficult for these systems to identify relevant information when just a small number of irrelevant statements are provided in input. The experimental results presented in Sections 5.4.4 and 6.3.5 also indicate that large, high-performing neural models are more likely to be misled by distractors. Additionally, there is a technical limitation on the length of input that a model can handle. Situational information can typically be obtained from various sources, and often, an excessive amount of information is present. Humans can quickly focus on crucial information and discard the rest, otherwise, it would take forever to read, process, and reason over surrounding information.[15]

To overcome these challenges, we need to develop methods to identify the information that is most relevant. In this section, we explore computational methods to filter out irrelevant situational statements before generating responses. We compare two distinct methods: a method based on topical similarity and a method based on implicit knowledge. Our experiments using SUGAR demonstrate that these methods have their own strengths, and their combination provides accurate results.

[15]Researchers have identified the Frame Problem (McCarthy and Hayes, 1969) that describes the dilemma of a reasoning system in determining which aspects of a situation change and which remain constant after an action.

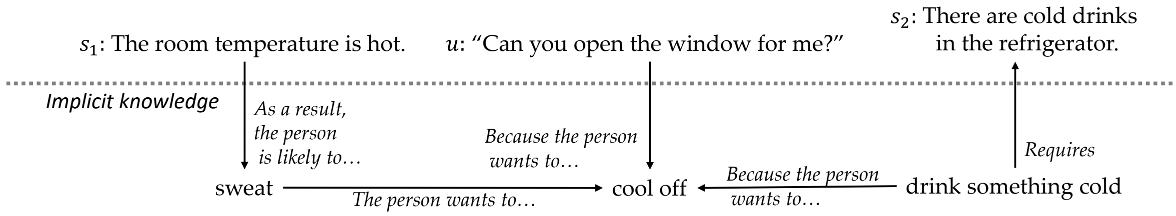


Figure 6.4: To capture the relevance of situational context, we often require the comprehension of implicit eventuality knowledge.

Problem Setting: The task involves an input utterance u[16] and L situational statements $S = s_1, \cdots, s_L$. Each situational statement is a single sentence that is assigned a binary relevance judgment $y_i \in \{\text{relevant}, \text{irrelevant}\}(i = 1, \cdots, L)$. The goal is to rerank S so that relevant statements are ranked higher in the output.

6.4.1 Method

We employed two distinct approaches to filter out irrelevant situational context. The first method is based on topical similarity between the input utterance and situational statements, and the second method relies on implicit eventuality knowledge that connects input texts.

During the development of our dataset, we found that topical similarity is highly correlated with the relevance of situational context. For instance, if an utterance is requesting drinks (e.g., "Can I have some water, please?"), context about drink and food is likely relevant. Our first approach uses sentence embeddings for detecting such similarity. We encoded the utterance u and situational statements S by Sentence Transformers using MPNet and computed the cosine similarity between u and each s_i as the score of the statement. We refer to this method as TSim henceforth.

Topical similarity is not always sufficient for detecting relevant information. It is also important to consider other factors such as temporal/causal associations and interactions among situational statements. For instance, consider the example utterance "Can you open the room window for me?" and two situational statements: (s_1) "The room temperature is hot" and (s_2) "There are cold drinks available in the refrigerator." While the relevance of u and s_1 may be explained by topical similarity, s_2's relevance involves more complex multi-hop reasoning (as illustrated in Figure 6.4), and the three pieces of information are mutually dependent. That is, the relevance of u and s_2 may not be obvious if s_1 is absent.

To address these problems, we developed an alternative approach, KRel (a knowledge-based relevance measure). KRel estimates the relevance of situational context based on implicit eventuality knowledge represented as a graph. We built the graph using the $\text{Comet}^{\text{DIS}}_{\text{TIL}}$ (West et al.,

[16]In our study, we follow the configuration of SUGAR and regard u as a single sentence. However, utterances may span multiple sentnces and may consist of multiple turns of conversations.

2022) model by generating top-K knowledge triples for input texts ($u + S$) up to 3 hops. We post-processed the graph to remove language variation and errors in the generation, and discarded knowledge nodes connected only with one input text.[17] Personalized PageRank (PPR) algorithm was used to estimate the relevance of situational context, with the interactions among knowledge nodes taken into account. Let us suppose we have N nodes, u corresponds to the first node, and $s_i \in S$ corresponds to $i+1$-th node. We set the initial score and the personalization score of node j to the following values.

$$\mathrm{PPR}_j^{(0)} = \begin{cases} \frac{W-1}{W} & (j = 1; \text{utterance node}) \\ \frac{1}{W} \times \frac{c_{j-1}}{\sum_{i=1}^{L} c_i} & (j \in \{2, \cdots, L+1\}; \text{situation node}) \\ 0 & (\text{Otherwise}), \end{cases} \tag{6.1}$$

where $W > 0$ is a hyperparameter to represent the relative importance of u, and $c_i (\geq 0)$ denotes the initial weight of i-th situational context. We set $c_j = 1/L$. We then updated the scores until they converge using the equation below.

$$\mathrm{PPR}_j^{(t)} = \gamma \sum_{k \in \mathcal{N}_j} w_{k,j} \mathrm{PPR}_j^{(t-1)} + (1-\gamma)\mathrm{PPR}_j^{(0)} (t = 1, 2, \cdots) \tag{6.2}$$

$\gamma \in [0, 1]$ is a damping factor, $\mathcal{N}_j$ is the set of nodes connected to node j, and $w_{k,j} \in [0, 1]$ is the weight of the edge from node k to node j. We computed edge weights by a pre-trained knowledge scorer released with $\text{Comet}^{\text{DIS}}_{\text{TIL}}$.[18] Intuitively, the PPR algorithm simulates a random walk over the graph, starting at the node determined by $\mathrm{PPR}^{(0)}$ and moving to another node using the graph edges or the personalization scores. Nodes that are more likely to be visited during the walk are assigned higher scores, which, in our case, indicates their relevance to the input texts.

Finally, we also experimented with the combination of the two methods. We first computed scores by TSim and used them as c_j in the initial and personalization scores (Equation (6.1)). TSim's scores were truncated into $[0, 1]$ before being used with KRel.

6.4.2 Experiments

We evaluated TSim, KRel, and their combination on the SUGAR dataset, which contains manually annotated relevance labels for situational statements. Each example contains six relevant statements on average and 15 distractors (irrelevant statements). We randomly split the dataset into 10% for tuning hyperparameters and 90% for evaluation. We set $W = 10$ and $\gamma = 0.85$.

Our results (Table 6.6) show that TSim achieved a high ROC-AUC score of 0.79, demonstrating the effectiveness of sentence embeddings in detecting the semantic proximity of situational context to the given input utterance. KRel performed slightly better than TSim when a larger knowledge graph ($K = 5$) was used but performed worse with a smaller knowledge graph ($K = 3$).

[17] Appendix D.3 provides technical details of this post-processing.

[18] https://github.com/peterwestai2/symbolic-knowledge-distillation

	TSim	KRel ($K = 3$)	KRel ($K = 5$)	TSim + KRel ($K = 3$)	TSim + KRel ($K = 5$)
ROC-AUC	0.792	0.763	0.798	0.811	0.820

Table 6.6: ROC-AUC scores of situation selection.

This is because a smaller knowledge graph results in fewer or no connections between the input utterance and situational context, making PPR less effective. Finally, combining TSim and KRel with PPR resulted in better performance than either method alone, indicating that the two approaches have complementary strengths.

To further analyze this complementary nature of TSim and KRel, we reviewed examples with relevance score estimated differently by the two methods. Overall, TSim tended to successfully identify the relevance of statements with overlapping words (Table 6.7a: "Please set the table." and "The table has not been set yet."). In contrast, KRel leveraged eventuality knowledge instead of surface similarity to detect the relevance. Table 6.7b shows an example where KRel chose a situational statement that appears to be topically distant ("The doorbell has been rung." with respect to the utterance "Please let [someone] in.") using a generated knowledge path.

$$\text{The doorbell has been rung.} \quad \overrightarrow{\text{xWant}} \quad \text{open a door} \quad \overleftarrow{\text{xNeed}} \quad \text{let someone in}$$

On the other hand, such informative paths are not always available for KRel. Knowledge nodes can be disconnected due to the language variation of the generation output and contextually related knowledge may not appear in the top-K generation outputs, especially if the input eventuality is associated with many eventualities in different contexts (e.g., an event "to come back home" can be followed by many different events such as "to change clothes", "to take a shower", and "to eat dinner"). These problems were mitigated by combining TSim and KRel.

These results demonstrate that filtering methods can benefit from implicit knowledge. However, KRel has a major drawback in terms of its computational complexity, making it difficult to scale to a larger number of input texts. In our experiments, it took about 8.5 hours to generate a knowledge graph using KRel for top-5 generation up to 3 hops on 4,621 sentences. Additionally, the size of the knowledge graph also affects the computational cost of PPR. Therefore, future research should explore more efficient methods to incorporate implicit knowledge into situational context selection.

6.5 Discussion

Models' internal assumptions: As discussed in Section 2, the lack of sufficient situational information often makes the interpretation of utterances ambiguous. In such a setting, systems need to learn to make various assumptions about the world state to produce naturally-sounding language, which can be regarded as a form of hallucination. Responses generated in this way can

	TSim [0.99]	KRel ($K = 5$) [0.75]	TSim+KRel ($K = 5$) [0.85]
1	The table has not been set yet.	[user] has dishes and utensils to set the table with.	[user] has dishes and utensils to set the table with.
2	[user] has dishes and utensils to set the table with.	[user] is home in the kitchen.	The table has not been set yet.
3	The kitchen has a table.	The table has not been set yet.	[user] is home in the kitchen.
4	[user] has not had dinner yet.	[user] has finished cooking dinner.	[user] has finished cooking dinner.
5	[user] has finished cooking dinner.	The kitchen is dirty.	The kitchen has a table.

(a) **Input:** "Please set the table."'

	TSim [0.93]	KRel ($K = 5$) [1.00]	TSim+KRel ($K = 5$) [1.00]
1	[someone] is at the front door.	The doorbell has been rung.	[someone] is at the front door.
2	[user] is expecting [someone].	[someone] is at the front door.	The doorbell has been rung.
3	[user] has a bathtub in the house.	[user] is home.	[user] is home.
4	[user] is cooking right now.	[user] is cooking right now.	[user] is cooking right now.
5	[user] is home.	[user] is expecting [someone].	[user] is expecting [someone].

(b) **Input:** "Please let [someone] in."

Table 6.7: Examples of reranking results. Top-5 situational statements are presented. The values in the squared parentheses denote ROC-AUC scores.

be useless in real applications, where the world state is predetermined. Our empirical analysis also indicates that the systems' consistency can be improved with detailed situational information, which is also aligned with the initial motivation of background-based conversational tasks like persona chat.

Resource acquisition: Simple collections of textual conversations can be easily obtained at scale from the web, but acquiring their situational information is more difficult. For example, although conversations on Twitter may be grounded in the weather, sport events, and news on a particular day, automatically extracting such alignments may be challenging. The connection between utterances and related information is often obscure, and manual intervention is likely required to obtain high-quality annotations. As a potential remedy for this challenge, we attempted automatic generation of situational information in our case study. The quality of the generated result was fair, but we needed to manually revise the test instances. Recent studies have demonstrated promising results in inducing world knowledge from PLMs (Ghosal et al., 2022; West et al., 2022). The future advancement in this line of work may make it possible to annotate existing open-domain conversation datasets with situational information in a post-hoc manner.

Availability: Different platforms of conversational systems have access to different types of situational information. Smart speakers may be equipped with physical sensors to observe visual and audio information. On the other hand, virtual assistants and text-based chatbots may not have access to such information. However, it is likely that there are some available signals that human communicators and systems could refer to, such as approaching holidays and personal information obtained through previous conversations. Finch et al. (2019) demonstrated that mentioning recent events can improve user engagement in chit-chat. Furthermore, if conversation systems have access to the Internet, which is often the case, they can access diverse kinds of information through external APIs. Access to APIs can also facilitate conversational assistance with task-specific information in various domains (Liang et al., 2023).

Adequacy: When situations are taken into account, a different problem arises. Our findings indicate that it is not straightforward to identify relevant situational information and integrate it into a coherent response, even with just 10 situational statements. Additionally, there is a technical limitation on the length of input that a model can handle. situational information can typically be obtained from various sources, and often, an excessive amount of information is present. Humans can quickly focus on crucial information and discard the rest, otherwise, it would take forever to read, process, and reason over surrounding information. Researchers have identified the Frame Problem (McCarthy and Hayes, 1969) that describes the dilemma of a reasoning system in determining which aspects of a situation change and which remain constant after an action. To date, there has been no satisfactory solution to this questions, making the challenge of situated conversation an interesting open challenge.

Common ground: Knowledge about situations is closely related to common ground–the information shared by conversation participants. Without common ground, conversation participants would need to convey every parameter of their message, which is extremely inefficient. The importance of common ground is widely recognized, and decades of dialogue research have been devoted to developing systems that can effectively establish common ground with their interlocutors by inferring, presenting, requesting, accepting, and repairing individual beliefs about various information through conversations (Clark, 1996; Poesio and Rieses, 2010; Traum and Allen, 1994; *inter alia*). In this chapter, we did not delve into the problem of common ground, but the consideration of situations, which is our main proposal, is the first step towards computational modeling of grounding.

6.6 Related Work

Conversation history: There is a rich line of work on how to induce useful contextual information from conversation history, for example, by designing dedicated components for capturing contextual information (Sankar et al., 2019; Tian et al., 2017) and using external knowledge (Young et al., 2018; Wu et al., 2020; *inter alia*). While conversation history contains rich information, we need to also incorporate situational information, which is often unspoken, and to this end, we should think about how to design tasks and datasets.

Prompt design: Our analysis is closely related to work on in-context learning, or prompting, with PLMs. In particular, much attention has been paid to the effective provision of demonstrative examples (Liu et al., 2022a; Min et al., 2022; Zhao et al., 2021). This chapter discussed the problem from a different perspective, namely what clues should be included in prompts (situations) and how PLMs perform (misleading by distractors). Our observation regarding the latter is consistent with prior work that revealed the vulnerability to perturbations in input (Elazar et al., 2021; Pandia and Ettinger, 2021). Future work should explore ways to robustly identify relevant situational information to generate optimal responses.

6.7 Conclusion

Our main claim is that situational information, which may or may not be stated explicitly by humans, should be represented and incorporated as input in open-domain conversational tasks and datasets in order to advance the capabilities of conversation systems. We posited that the absence of situational information results in an underspecified problem space, causing a severe problem for both the development and evaluation of conversation systems. Our experiments highlight the benefits of providing context information explicitly to response generation systems, which motivates future research on context-specific conversation systems. On the other hand, we also showed the importance of selecting relevant situational statements, which is not a trivial problem for systems. To this end, we developed and evaluated filtering methods based on topical similarity and implicit knowledge and demonstrated that their combination achieves a high reranking accuracy. However, the tested knowledge-based method does not scale to a large-scale problem, and more efficient approaches should be explored in future research.

Limitations

Firstly, we did not address the fundamental challenge of determining *an adequate amount* of situational information. It is very difficult, if not impossible, to describe *all* the situations required to perform rationale reasoning, so we need to give up somewhere, relying on the reasoning capability of NLP systems.

Secondly, we did not use large-scale data or conduct an extensive search for optimal hyper-parameters and prompts (for GPT-3) in our experiments as the primary goal of this study was to raise attentions to potential issues and benefits associated with situational informaiton. The models may have performed better with different configurations. We did not examined the capabilities of larger PLMs in conducting situated conversations at scale. In our empirical analysis, we opted for GPT-3 due to its transparency about technical details compared with later versions of GPT.

Finally, while situational information can aid in the development of truthful and creative response generation systems, it does not address well-known issues associated with conversational technologies, such as safety and bias. In fact, poorly chosen situational information may even amplify undesired bias by linking two irelevant concepts together. To mitigate this problem, researchers and developers should exercise caution when collecting data and carefully monitor system output.

Chapter 7

Represention of Cooking Procedures

Summary

This chapter focuses on the implicit knowledge of physical events and addresses the issue of procedural text understanding. Procedural texts are commonly used to describe various human activities, and automatic procedural text understanding technologies have practical applications, such as how-to information retrieval. English cooking recipes have been widely studied in the NLP community due to data availability of data and manageable vocabulary size. Researchers have devised representation schemes, datasets, and systems that parse raw texts into structured semantic representations. Recent neural-based models have achieved promising parsing accuracy. However, the existing schemes are insufficient in representing common phenomena in cooking recipes, which can mislead language understanding systems and make evaluation results questionable. An empirical analysis presented in this chapter reveals that widely used representation schemes fail to capture essential phenomena in cooking instructions, and machine learning models trained on those schemes can learn undesired patterns. In light of these findings, this study proposes a representation scheme based on a graph of semantic frames and construct a dataset of 108 cooking recipes by crowdsourcing followed by manual verification. The proposed representation scheme achieves better coverage of cooking actions than the existing schemes. We also review existing studies and discuss directions for future research.[1]

7.1 Introduction

A procedural text, which instructs people to complete a particular activity, is a widely observed type of human communication found in a variety of areas such as cooking, engineering, and science. Our goal is to structure procedural knowledge, described in cooking recipes, into meaning representations that facilitate reasoning about the physical processes and states. This problem

[1]This chapter is based on Otani et al. (2023c).

has substantial implications for real-world applications such as information retrieval (Yang and Nyberg, 2015) and personalization (Shirai and Kim, 2022; Twomey et al., 2020). Furthermore, the task of procedural text understanding itself has received attention as an interesting sub-problem of natural language understanding (Dalvi et al., 2018; Tandon et al., 2020; Zellers et al., 2021).

This study focuses on English cooking recipes, which exhibit many unique linguistic characteristics (Culy, 1996) similar to other types of procedural texts.

Writers generally do not explicitly describe detailed information that they assume readers can infer. Even crucial actions are often assumed implicitly when they can be deduced from the surrounding context and common sense, as shown in the example below:

(11) Lay the dough over the tart pan and tuck the excess dough in around the edges. Brush with some oil and <u>bake</u> at 390F for approx. 20 min.

In this example, the action "bake" implies that the tart pan is transferred into an oven beforehand. Similarly, cutting actions such as "Finely dice the onions" often appear without mentioning a location, which is typically a cutting board.

To develop systems that can analyze the semantics of procedural texts as humans do, we must first explore how to represent underlying semantics. Researchers have designed various representation schemes and constructed annotated datasets. The existing schemes can be categorized into two approaches. The first approach focuses on a surface structure (Fang et al., 2022; Kulkarni et al., 2018; Maeta et al., 2015), where text spans of semantic elements, such as actions, ingredients, and tools, are first recognized and then connected by labeled edges to represent relationships between actions and objects. The second approach focuses on entity state tracking (Bosselut et al., 2018; Dalvi et al., 2018; Tandon et al., 2020), which identifies entities that undergo physical state changes at each step.

Although recent neural-based models have demonstrated impressive parsing accuracy on these meaning representations, their performance remains below human-level accuracy. We argue that the inadequacy of a representation scheme is a contributing factor to this gap (§7.3.2). The representations based on surface structures only model actions and entities that are explicitly stated in texts, although many are not. While the entity tracking approach captures implicit information to some extent, it represents the flow of cooking processes in a very simplified way, limiting grounding for reasoning and potentially leading machine learning systems to overfit spurious patterns.

To address these problems, we propose extending these schemes to model action sequences, their arguments, and underlying states based on a graph of semantic frames (§7.4). Our representation scheme is designed to achieve a wider coverage of implicit information, including unmentioned actions, ingredients, tools, and physical states, with higher accuracy than prior work. Specifically, we represent a cooking procedure by linking actions, denoted by frames, with their participants and modeling underlying state changes that ingredients undergo at each action. The resulting graph elucidates unmentioned actions and arguments, as well as how ingredients merge and separate during the cooking process. Following prior work on entity state tracking (Bosselut et al., 2018), we defined four internal physical states, including location, temperature, state of matter, and size. As a proof-of-concept, we annotated 108 English cooking recipes using the proposed representation scheme. The annotation result revealed the prevalence of implicit information that was not fully covered by prior work, suggesting the advantage of our scheme in representing, acquiring, and operationalizing physical knowledge. This chapter also offers insights into the computational models of recipe parsing for future research.

In summary, this study contributes to annotation research on cooking recipes by:

- Identifying prevalent linguistic phenomena in cooking recipes that was not addressed by previous work.
- Proposing an extension of existing schemes to address these issues.
- Providing 108 annotated English recipes.[2]

7.2 Related Work

This section provides a review of prior work on procedural text understanding.

7.2.1 Procedural Text

Procedural texts are written to explain to others how things work and how to perform actions to achieve a specific goal. Language technologies for procedural texts have various applications such as how-to information retrieval (Yang and Nyberg, 2015), recommendation (Twomey et al., 2020), question answering (Bisk et al., 2019; Dalvi et al., 2018), and instruction generation (Chandu et al., 2019; Paris et al., 2002).

English procedural texts have unique characteristics (Culy, 1996). First of all, instructions are usually written as imperative sentences, and secondly, writers often omit essential information that they assume readers can infer based on their knowledge and experience, and "too much detail is often considered too verbose and unnatural" (Gerhardt, 2013). These characteristics pose a unique challenge to standard NLP techniques.

7.2.2 Procedural Text Understanding

Different representation schemes favor different computational approaches. For SSR, many studies employ a sequential pipeline of linguistic analyzers to obtain structured output (Maeta et al., 2015; Shirai and Kim, 2022; *inter alia*). In contrast, the task of state tracking can be reduced

[2] A few examples are included as supplementary material for this submission. We plan to release our full annotation data upon acceptance.

to relatively simple classification problems, for which neural networks can be effectively applied (Bosselut et al., 2018; Dalvi et al., 2018; Huang et al., 2021). The comprehension of entity states often requires world knowledge. Motivated by this observation, several knowledge-based approaches have been proposed, such as the use of heuristic rules (Du et al., 2019; Tandon et al., 2018) and external knowledge resources (Clark et al., 2018; Kazeminejad et al., 2021; Zhang et al., 2021b).

Recent studies have demonstrated that PLMs can perform the task of procedural text understanding by prompting (Madaan et al., 2022; Spiliopoulou et al., 2022). This indicates that these models learn physical knowledge to some extent during pre-training, and we can induce the knowledge with the appropriate problem formulation and prompts.

7.3 What Makes Recipes Hard?

In this section, we discuss the challenges that arise in representing and analyzing cooking instructions. We first focus on basic linguistic analysis, then delve into knowledge representations.

7.3.1 Surface-level Challenges

English cooking instructions typically express actions using action verbs, with the participants of the actions often marked as direct objects (e.g., "mix eggs and salt") or prepositions (e.g. "season with pepper"). Semantic parsing of English recipes would be straightforward if a dependency parser worked almost perfectly as it does in newswire texts. However, English cooking recipes exhibit unique syntactic and semantic characteristics (Malmaud et al., 2014), such as the frequent use of imperative expressions and null core arguments. Standard NLP tools are developed mainly for texts in the indicative mood, making parsing imperative expressions challenging.

To measure the difficulty of POS tagging and dependency parsing, two annotators with a background in NLP manually analyzed 50 cooking recipes sampled from the NYC dataset (Bosselut et al., 2018). These texts were collected by Kiddon et al. (2016) from an online recipe repository for a recipe management software called "Now You're Cooking". Most of the recipes were originally sent to mailing lists of online cooking communities in the 1990s. Our 50 recipes contain 404 sentences with 4,462 tokens, with an average of 8.08 sentences per recipe.

The two annotators were first trained with the annotation guidelines of Universal Dependencies v1 (Nivre et al., 2016) and performed annotation on brat (Stenetorp et al., 2012), a web-based visual annotation tool. The annotation work primarily focused on the detection of verbs (universal POS of VERB), direct objects (dependency relation of `obj`), and obliques (dependency relation of `obl`), as they often correspond to predicates and themes in cooking instructions. We first ran StanfordNLP (Qi et al., 2018) to obtain initial annotations. The annotators then checked the texts and corrected POS tags and dependency relations when necessary. The inter-annotator agreement was measured using Cohen's κ, which resulted in a value of 0.891. Ambiguous cases

Prediction target		P	R	F1
POS	VERB	0.99	0.88	0.93
Dependency	obj	0.92	0.83	0.87
	obl	0.76	0.90	0.66

Table 7.1: The performance of StanfordNLP analyzer in the identification of verbs, direct objects, and obliques in the 50 cooking recipes.

were discussed by the annotators in the first 10 recipes, and one of the annotators adjusted the annotations on the rest accordingly.

Table 7.1 presents the results of manual annotation on the 50 recipes. We identified 747 verbs in 404 sentences, with 93.3% in the imperative mood. However, the StanfordNLP tagger misclassified some verbs as nouns, resulting in a recall of 0.88. This suggests that the tagger may miss 1-2 verbs per recipe. The StanfordNLP dependency parser achieved F1 scores of 0.87 and 0.66 in identifying direct objects and obliques, respectively, which are lower than typical scores for newswire texts. Several factors account for the parser's degraded performance. For example, 57.8% of verbs do not have an object, meaning that action targets must be inferred based on the context in many cases. Errors in detecting obliques can also be attributed to verb-particle phrases, such as "pour in" not being recognized as a compound verb. Similar errors were observed by Zhang et al. (2012). Moreover, oblique relations are often omitted, despite the fact that location changes are frequent and important information in instructions, which is consistent with the finding of Jiang et al. (2020) that only 17% of verbs had an argument that denotes a target location in their analysis. Note that the extent and frequency of omissions may vary across cooking recipes of different styles, with more casual recipes tending to omit more details Culy (1996). The NYC recipes in our dataset are relatively casual as they were primarily written for mailing lists by non-experts. We also observed frequent errors in sentence and word segmentation, such as "bake for approx. 3 min." being divided into two sentences.

In addition, semantic parsing also poses several unique challenges in interpreting predicates, arguments, and control structures (Malmaud et al., 2014; Momouchi, 1980). To examine this, the same two annotators annotated the 50 recipes with semantic roles, particularly patient/theme and locative. The results showed that 35.8% of predicates implicitly referred to the ingredients mentioned in a previous sentence as an argument, requiring cross-sentence semantic analyses. Furthermore, the referent of an implicit mention is often ambiguous, as in the example "Crack eggs into a large bowl. Whisk well." Implicit knowledge is required to determine that the target of "whisk" is the eggs, not the large bowl.

7.3.2 Representation Challenges

Researchers have proposed various representation schemes, which can be grouped into two categories: representations of surface structures and representations of entity states.

A surface-structure representation (SSR) identifies text spans that denote actions, events, ingredients, tools, and configurations, and connects them with labeled edges to denote predicate-argument structures and event sequences (Kulkarni et al., 2018; Maeta et al., 2015; Mori et al., 2014; Mysore et al., 2019; Shirai and Kim, 2022; Yamakata et al., 2020). While the resulting structures represent the flow of processes, this approach is primarily designed for structuring information that is explicitly stated in text. There are a few exceptions that focus on specific phenomena, such as anaphora resolution (Jiang et al., 2020), the existence of state changes (Fang et al., 2022), and tools (Nabizadeh et al., 2020).

The other line of work, an entity-state representation (ESR), employs intermediate representations that encode semantic information, including the type of actions, composition of ingredients, and physical properties of ingredients. This approach, called entity state tracking, has representative datasets such as the Now You're Cooking (NYC) dataset in a cooking domain (Bosselut et al., 2018), the ProPara dataset in a science domain (Dalvi et al., 2018) and the OpenPI dataset in a general domain (Tandon et al., 2020).

SSR structures only the actions, ingredients, tools, and parameters that are explicitly stated in the instruction texts. However, many actions and entities may be implicitly assumed in the texts (Culy, 1996; Gerhardt, 2013; Gil, 2015; Malmaud et al., 2014). For instance, modifiers and combinations of verbs can express a cooking action (Donatelli et al., 2021). Consider the following examples taken from different cooking recipes.

(12) Add oil and <u>crashed</u> garlic to the pan.

(13) Boil pennes in the saucepan with <u>salted</u> water.

(14) <u>Put</u> the tray <u>in the oven.</u> <u>Take out</u> after 8 minutes.

In Examples 12 and 13, the modifiers of nouns imply cooking actions. In Example 14, the combination of two actions "put in the oven" and "take out" suggest that the food on the tray is baked between the two actions. Moreover, the change of location associated with common cooking actions is often implicitly assumed. In the example below, the onions are transferred to a cutting board before being cut.

(15) Finely chop the onions.

These examples suggest the limitations of surface-level representations.

ESR falls short of modeling cooking flows. The ESR approach captures the entities involved in each instruction and their physical states, which may not be explicitly stated in the text. How-

		garlic clove	shallot	salt	spice	paprika	oil	chicken
1.	**Mince and mash** the garlic and shallots **to a paste** with the salt.	$\checkmark$	$\checkmark$	$\checkmark$				
2.	In a bowl **stir the paste together with** the spice powder, paprika, and the oil.	$\underline{\checkmark}$	$\underline{\checkmark}$	$\underline{\checkmark}$	$\checkmark$	$\checkmark$	$\checkmark$	
3.	Add the chicken.	$\underline{\checkmark}$	$\underline{\checkmark}$	$\underline{\checkmark}$	$\underline{\checkmark}$	$\underline{\checkmark}$	$\underline{\checkmark}$	$\checkmark$

Table 7.2: A recipe containing merging actions (1, 2). In the entity-state representation scheme, ingredients are labeled separately even after being merged (denoted by $\underline{\checkmark}$).

ever, due to the simplification in the representations, ESR does not accurately represent certain prevalent actions, such as merging and separating.

1. Once entities are merged, they will undergo the same state changes afterward, but in the ESR scheme, merged entities are analyzed separately, as denoted by $\underline{\checkmark}$ in Table 7.2. The labels assigned to merged entities account for 31.8% of predictive targets in the development data of the NYC dataset, making the task of entity tracking virtually easy. Just by detecting the first occurrences of entities with heuristics and predicting them in the next steps, we can achieve over 70% accuracy (Gupta and Durrett, 2019). Thus, accuracy scores with ESR are likely to be inflated due to the representation of a merging action. In addition, it is possible that data-driven models overfit the spurious pattern and do not learn generalizable patterns.
2. In cooking recipes, it is often the case to separate one ingredient and use it in different steps, as in the examples below:

   (16) Season salmon with salt. Heat on the pan. In a large bowl, season tomatoes with salt.

   In this example, the ingredient "salt" is used twice separately, but this separation is not accurately described in ESR, as shown in Table 7.3. This representation leads to incorrect procedural knowledge extracted from cooking recipes (e.g., salt must be extracted from salted salmon.).

Therefore, ESR falls short in representing the full range of cooking actions found in recipes.

7.4 Semantic Frame Graph

To address the challenges discussed in the previous section, we propose a representation scheme for cooking recipes based on a graph of semantic frames. We define a cooking recipe as a series of

		salmon	tomato	salt
1.	Season salmon with salt.	✓		$\underline{✓}$
2.	Heat in the pan.	✓		$\underline{✓}$
3.	In a bowl, season tomatoes with salt.		✓	$\underline{✓}$

Table 7.3: A recipe containing separation of ingredients. Salt is used for salmon and tomatoes separately but is represented as a single entity in the entity-state scheme (denoted by $\underline{✓}$). Consequently, the pre-location of salt in step 3 is considered to be the pan.

T sentences $S = \{s_1, s_2, \cdots, s_T\}$ that instruct a reader to complete the steps required to make a target dish. The task is to identify the cooking actions instructed in each sentence, as well as the entities, such as ingredients, tools, and containers, that are affected by each action. We denote the entities as $E = \{e_1, e_2, \cdots, e_N\}$.

7.4.1 Entity States

Important details are often left implicit due to the assumption that readers can guess them based on common sense and contextual information. Such implicit comprehension includes knowledge about a cooking action's pre-conditions and effects. For example, the "cut" action is typically applied to solid ingredients, and the "melt" action indicates that the input ingredient will become liquid. Knowledge about entity states can provide crucial information to resolve semantic gaps. we consider the following physical states to cooking processes based on prior work (Bosselut et al., 2018; Malmaud et al., 2014; Tasse and Smith, 2008) and our preliminary annotation study.

Location: The location of an entity, which is another entity available in the cooking process such as an ingredient or a container. One entity can have only one location value. If entities are nested, such as a pancake placed in a baking pan inside an oven, the location points to the inner location (i.e., the baking pan for the pancake). The location information often constitutes the pre-conditions of actions such as a cutting board for "cut" and an oven for "bake".

Temperature: The temperature of an entity, either "high", "low", or "unknown". The change in temperature is prevalent in cooking procedures but is often assumed implicitly.

State of matter: The state of matter of an entity, either "solid", "liquid", or "unknown". As mentioned before, some English action verbs denote the state of matter of target ingredients.

Size: The size of ingredients, either "small", "large", or "unknown". Cutting actions are often required before other actions such as "sauté". Thus, the size of an entity provides some clue in identifying missing arguments for those actions.

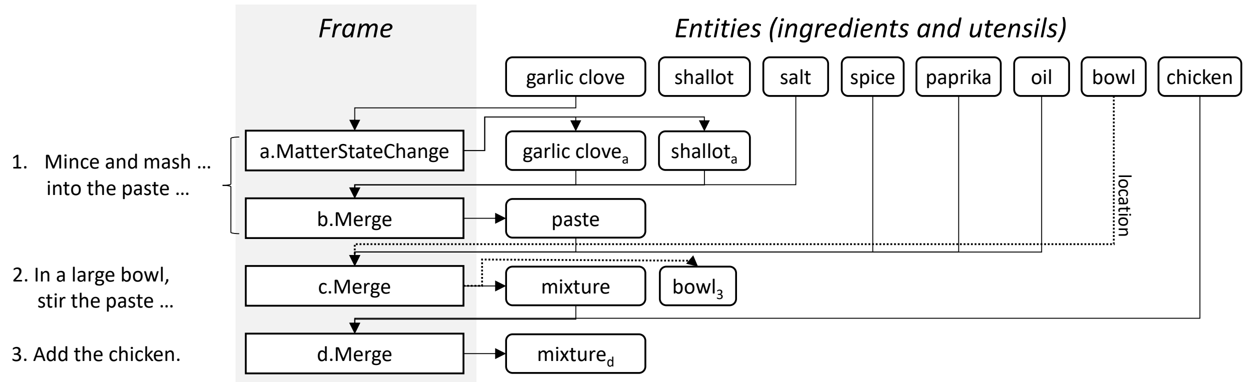


Figure 7.1: The graph representation of merging actions. This example corresponds to the example in Table 7.2. LocationChange frames for the first and second cooking instructions are not shown in this figure for brevity.

In addition, we denote mixtures of ingredients by a binary flag. Once entities have mixed, they will not be divided afterward in general. This information will be useful to resolve the referent of mentions such as “the mixture”.

We intend to use these entity states as supplementary clues for identifying unmentioned actions and entities. Therefore, we opted to simplify the values of entity states.

7.4.2 Semantic Frames

We represent a cooking action using a structured representation called a frame (Fillmore et al., 2012; Minsky, 1974) that maintains terminals (semantic elements) of an action with their roles. We focus on four semantic roles: input, output, location, and tool. Input entities undergo changes in their underlying physical fluency, such as temperature, size, and state of matter, and output entities correspond to the entities after their state changes. A location entity represents where the output entity will be afterward, and tool entities denote the tool used for performing the action.

While one sentence can evoke multiple frames, each frame f_i is evoked by exactly one sentence, denoted by $s^{(i)}$. In most existing schemes, actions are limited to ones explicitly triggered by verbs. However, in the SFG scheme, frames can be evoked by a construction (e.g., “fold _ into” and “bake _ in the oven” for a change of location), non-verbs (e.g., “salted water” for a merge action), or without explicit trigger terms (e.g., a transfer action implied by “cut the onions”).

We defined six semantic frames: Merge, LocationChange, SizeChange, TemperatureChange, MatterStateChange, and OtherChange. The first five frames are concerned with entity states that are frequent and often essential to make a target dish. OtherChange covers miscellaneous actions that are not vital or frequent in cooking recipes, such as tilting, reshaping, soaking, and washing. Table 7.4 shows their definitions and examples.

We assume that each frame has different default assumptions about pre-conditions and effects.

Frame	Definition	Effect	Example
Merge	This frame represents the combination of input entities, resulting in a single mixture entity.	Input entities are linked with one output entity which is marked as "mixed".	"Mix milk, salt, and pepper."
LocationChange	This frame changes the location of entities by transferring them.	A specified location is assigned to the location of output entities.	"Pour in milk," "Season the steak with salt and pepper."
TemperatureChange	This frame changes the temperature of entities by heating or cooling.	The temperature of output entities is set to "high" (higher than before) or "low" (lower than before).	"Heat butter," "Preheat an oven."
MatterStateChange	This frame changes the state of matter of entities through actions such as melting and freezing.	The state of matter of output entities are set to "solid" or "liquid".	"Melt butter," "Freeze juice in a fridge."
SizeChange	This frame changes the size of ingredients, usually making them smaller through cutting, chopping, etc.	The size of output entities is set to "small" (smaller than before) or "large" (larger than before).	"Cut carrots into small pieces."
OtherChange	This frame covers various actions such as reshaping, soaking, drying, washing, rotating, and peeling.	None	"Roll up the tortillas," "Wash potatoes well," "Flip the omelet," "Peel carrots."

Table 7.4: Semantic Frames

For instance, SizeChange requires input entities to be solid ingredients, and LocationChange requires input entities to be in places other than the destination. We can also consider more specific subframes, such as LocationChange triggered by an action verb "pour" requiring input to be liquid. We leave these extensions as future work.

We construct a directed acyclic graph of semantic frames where nodes denote frames and entities, and edges denote the relations between frames and entities. We use three edge types to denote frame-ingredient and frame-location, and frame-tool connections. A merge frame combines the ingredient entities in input and yields one intermediate ingredient entity, but does not merge tool entities. Frames of the other types take ingredients/tool nodes as input and yield the same number of output nodes. The separation and repeated use of ingredients are represented as nodes with two or more outgoing edges. This representation approach resolves the problem

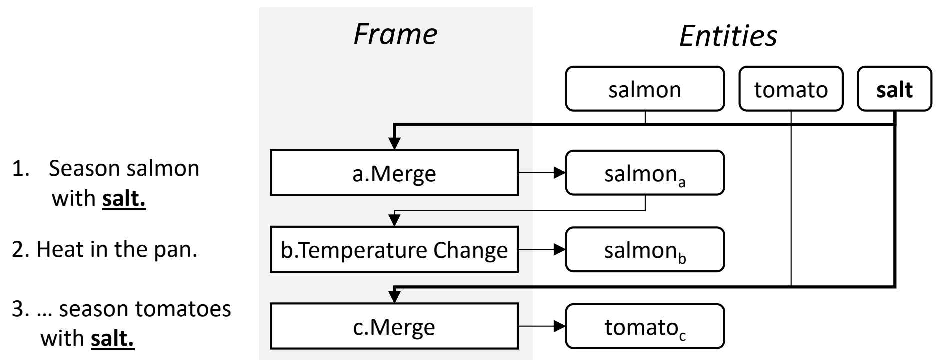


Figure 7.2: The graph representation of separated ingredients. This example corresponds to the example in Table 7.3. LocationChange frames, location entities (pan and bowl), and their edges are not shown for brevity.

Major category	Minor category	Recipes
Appetizers	Salad, Soup	10 for each
	Stew	5
Entree	Burger, Wrap, Pizza, Casserole, Curry, Meatball	5 for each
Entree	Sandwich, Pasta	4 for each
Side	Omelette, Quiche	5 for each
Drink	Smoothie	10
Dessert	Cheesecake, Chocolate cake, Muffin, Custard/pudding, Ice pop	5 for each

Table 7.5: 108 recipes collected from the KitchenStories website.

with the entity-state representations, as illustrated in Figures 7.1 and 7.2.

7.5 Annotation of Cooking Recipes

As a proof-of-concept, we annotated cooking recipes using the SFG schema. This section outlines our annotation process and results.

7.5.1 Data Collection

We collected English cooking recipes authored by professional writers from Kitchen Stories[3], a website that provides a variety of recipes. We selected common and diverse dishes based on dish genres and user ratings on the website, resulting in a total of 108 recipes covering various categories (Table 7.5). We extracted lists of ingredients and tools and instruction sentences from

[3] https://www.kitchenstories.com/en

each recipe. This data is formatted using a standardized structure on the website[4]. We defined E using the ingredient and tool lists provided in each recipe. When necessary, we supplemented E with entities often implicitly assumed, such as water, a container, a refrigerator, and a cooling rack. The dataset comprises 1,263 sentences and 15,198 tokens in total.[5]

7.5.2 Manual Annotation

Initially, we employed crowdsourcing to identify frame types and their arguments (see Appendix E.1 for details). Although the crowdsourced annotations were largely accurate, we found that all annotated recipes contained a few incorrect and/or missing frames. Consequently, one of the authors manually re-annotated the data and refined annotation guidelines through regular discussions with the other authors.[6]

7.5.2.1 General Instructions

The goal of this annotation task is to collect semantic annotations of cooking instructions from which an automatic analysis system can use to learn explicit and implicit changes of ingredients and utensils. To this end, we aim to:

1. Identify the nature and extent of state changes in each instruction.
2. Identify the entities (ingredients and tools) that undergo each state chages.
3. Identify the specific changes in each state, including changes in temperature and location.

All changes in location, temperature, state of matter, and size should be marked by a corresponding frame. For instance, if the temperature of an ingredient changes, a TemperatureChange frame must be assigned, and TemperatureChange frames can only change temperature. For example, "Heat butter in a pan." includes changes in location, temperature, and state of matter. Therefore, the following three frames are assigned to this sentence.

- `LocationChange`(in=butter, out=butter$_1$, effect=pan)
- `TemperatureChange`(in=butter$_1$, out=butter$_2$, effect=pan)
- `MatterStateChange`(in=butter$_2$, out=butter$_3$, effect=pan)

7.5.2.2 Entities

Entities are ingredients, tools, and containers involved in cooking actions. In our annotation, we use lists of ingredients and tools provided in each recipe from Kitchen Stories to define an entity

[4] https://schema.org/Recipe

[5] We tokenized texts by spaCy (en_core_web_sm) (Honnibal et al., 2020).

[6] The primary annotator is a Ph.D. student studying NLP in a university in the USA, and the other (3) authors hold Ph.D. degrees with NLP backgrounds. All authors are fluent in English.

list E. Some entities may not be specified in the list. In such cases, we create a new entity, as demonstrated in the examples below.

(17) Transfer the mixture into baking tins and bake in a preheated oven at 110C/230F for approx. 40–45 min. until fully set. Then, remove from oven and let cool. (A new entity "cooling rack" was created as the destination of the underlined LocationChange action.)

(18) In a saucepan, cook an egg in boiling water for approx. 2-3 min. (A new entity "water" was created for the underlined LocationChange (→ a saucepan) and TemperatureChange actions.)

Implicit ingredients are typically limited to common entities such as water and seasonings, which can be easily determined based on context. In contrast, implicit location entities cannot always be determined uniquely. To maintain consistency, we pre-define and use frequently observed entities or general entities. Specifically, our vocabulary includes: container, oven, refrigerator, cooling rack, and countertop.

We do not mark entity mentions in our annotation, as many entities, like inferred entities and intermediate ingredients, do not have corresponding language expressions.

7.5.3 Semantic Frames

We represent intended cooking actions using frames. Each of these frames takes one or more input entities, such as ingredients and tools, and produces corresponding output entities, which could be either a single entity (for Merge frames only) or the same number of entities as there are input entities.

Triggers: When cooking actions are explicitly stated, we mark a triggering expression for each semantic frame. A triggering expression can be an action verb (e.g., heat), a phrasal verb (e.g., heat up), or non-verbs (e.g., *melted* butter, and stir _ into). In either case, we consider minimal expressions as triggers. For example, "cool" should be marked for an expression "allow to cool". We intend to utilize the information of triggering expressions to facilitate knowledge acquisition associated with language expressions. For instance, LocationChange frames triggered by the verb "pour" are likely to take liquid ingredients as input. Future work should explore methods to automatically induce this kind of knowledge and leverage it for more accurate recipe understanding.

7.5.4 Analysis

Table 7.6 displays the average number of frames per recipe.

Implicit Actions in Recipes: In all recipes, at least one action does not have an explicit mention in the text. 81% of such frames involve a change of location, followed by actions such as a

Frame	Per recipe
Merge	3.32
LocationChange	10.36
TemperatureChange	3.53
MatterStateChange	1.51
SizeChange	2.65
OtherChange	1.85

Table 7.6: Average number of frames per recipe

change of size (e.g., cutting and slicing; 30%), change of temperature (e.g., baking and heating; 18%), and merging (e.g., combining and whisking; 8%). Notably, a considerable number of actions are indicated by non-verbs, such as prepositional phrases (e.g., "In a large bowl, mix..." and "Bake in a preheated oven."). These results highlight the need to consider actions without linguistic expressions or with non-verb triggers and demonstrate the benefits of our representations for addressing this phenomenon. Table 7.7 presents some examples.

Merging Ingredients: Mixing ingredients is a common action in cooking recipes, with an average of 3.3 Merge frames per recipe. It is important to note that even if ingredients are brought together in one location, they are not necessarily combined into a single mixture in our representation scheme. For instance, beef stew is not a single mixture because some of its ingredients (beef, roux, etc.) can still be identified individually, and they do not share the same states (i.e., beef is solid, and roux is liquid). Instead, the Merge frame represents an action that produces a non-separable ingredient with largely shared physical states (e.g., roux, smoothie, dough). We believe that accurately representing the merge action is essential for conducting semantic analysis based on knowledge of physical states.

Separation of Ingredients: Our annotation work revealed that the separate use of a single ingredient is quite common in cooking recipes. In our dataset, 69% of the recipes have at least one ingredient used separately in different steps. Most of these ingredients are seasonings such as salt and pepper, but other types of ingredients can also be used separately, as shown in Table 7.8. We found that many cases could be handled based on prior knowledge (i.e., seasonings are often used multiple times) and linguistic cues such as "half of" and "remaining." Yet, our scheme provides an opportunity to explore this important and interesting phenomenon concerning anaphora resolution, which was not addressed in existing entity-state representations.

Dish	Cooking instructions and evoked frames for the underlined text
Pudding	[...] Add milk, heavy cream, a third of the sugar, the seeds of one vanilla bean , and a small pinch of salt to a saucepan. Bring mixture to a slight boil. Whisk together eggs and another third of the sugar in a bowl. Now, add the hot milk mixture to the beaten egg mixture. [...] 1. `LocationChange`(in=[eggs, sugar], out=[eggs_1, sugar_1] effect=bowl, **trigger=None**) 2. `Merge`(in=[eggs_1, sugar_1], out=[egg mixture], effect="mixed", trigger="whisk")
Pizza	[...] Discard top layer of parchment paper, transfer to a baking sheet, and bake in preheated oven at 180c/350f for approx. 15–20 min. Spread ricotta evenly over crust. Then, top pizza off with salmon, shrimp, pine nuts, and freshly grated horseradish. [...] 1. `LocationChange`(in=[dough], out=[dough_1] effect=preheated oven, **trigger=None**) 2. `TemperatureChange`(in=[dough_1], out=[dough_2], effect="high", trigger="bake")

Table 7.7: Examples of implicit actions in the KitchenStories dataset. We assigned Location-Change frames corresponding to the implicit actions presented in this table.

7.6 Computational Instantiation

So far, we have identified the limitations of prior representation schemes and discussed how they were addressed by the proposed representation scheme, SFG. This section discusses potential technical instantiations of the SFG scheme.

7.6.1 Semantic Parsing

Automatic parsing of cooking recipes requires computational modeling of actions, entities, and their states, for which the proposed representation scheme introduces some important and interesting challenges. First, we must identify implicit actions and entities that may not be present in the text. Therefore, we cannot rely solely on textual cues to compute their representations. Second, the comprehension of merging actions often requires complex reasoning. For example, the mixture of milk and sugar is likely to be liquid, but the mixture of milk and flour is typically not. This problem will be challenging for previous approaches that were built on simplified representations. Finally, our problem subsumes the resolution of complex anaphora phenomena. This subproblem is an open challenge for language technologies (Fang et al., 2022; Jiang et al., 2020).

Recent studies have demonstrated that pre-trained language models (PLMs) learn world knowl-

Dish	Cooking instructions
Meatballs	[...] In a bowl, combine onion, garlic, ground meat, panko breadcrumbs, milk, and half of the harissa paste. Season generously with salt and pepper. [...] Combine bell peppers, onions, and chickpeas in a bowl, then add remaining harissa paste, olive oil, salt, and pepper. Stir to combine. [...]
Popsicles	Place some of the orange juice and sugar in saucepan and cook for approx. 5 min. over medium-low heat until the sugar has completely dissolved. Add lemon juice and remove from heat . Add the remaining orange juice to measuring cup along with the sugar-juice mixture and stir. [...]

Table 7.8: Examples of ingredients that are used separately in different steps.

edge through pre-training, which can be induced for performing various language tasks with a few or no training examples (Brown et al., 2020; Raffel et al., 2020). For procedural language understanding, Spiliopoulou et al. (2022) demonstrated that PLMs can perform the task of state change prediction with proper task encodings, although they still fall short of complex physical reasoning. Furthermore, Madaan et al. (2022) revealed that PLMs trained on human-generated computer programs can describe sequential entity state changes expressed by procedural texts in the form of Python code. Combining PLMs with physical knowledge, such as symbolic knowledge representation introduced in early studies (Forbus, 1984; Hayes, 1985), will be an interesting direction for future research.

7.6.2 Knowledge Acquisition

While implicit knowledge can be acquired to some degree from annotated recipes and external resources, it is difficult to fully cover the required knowledge on the first try. Knowledge bases are often incomplete (Ilievski et al., 2021), and the range of semantic knowledge is so broad that manual knowledge engineering takes a significant amount of time (Fillmore et al., 2012; Lenat, 1995).

A potential approach is to progressively acquire knowledge. We could discover new knowledge in parsed cooking recipes. For example, suppose we know that a cutting action requires a target object to be solid and we read an instruction like “Cut *[unknown ingredient]*”; then, we can learn that the ingredient is solid. Conversely, if we encounter many sentences saying “*[unknown action]* carrots/apples/beef/...”, we can deduce that the pre-condition of the action is an input ingredient being edible and solid. Moreover, multiple texts for the same procedure with different information granularity levels (e.g., recipes of omelette for beginners and experts) can provide clues for assumed knowledge more reliably.

7.7 Conclusion

This chapter addressed the problem of meaning representations for English cooking recipes, which has significant implications for practical applications. We discussed challenges in understanding implicit cooking knowledge and identified shortcomings in current representation approaches. To address these limitations, we expanded existing schemes and proposed representing cooking procedures using a graph of semantic frames. Our approach captures crucial phenomena in cooking recipes with greater accuracy and coverage compared to previous efforts. As a proof-of-concept, we annotated 108 cooking recipes, and the results suggest that the proposed scheme offers advantages in operationalizing physical knowledge for reasoning.

Chapter 8

Conclusion

This chapter summarizes the key findings and contributions of the thesis, discusses the limitations of the study, and suggests possible avenues for future research.

8.1 Findings and Contributions

This thesis has highlighted the benefits of incorporating implicit eventuality-centric knowledge for enhancing our understanding of language in a more controllable and interpretable manner. In this section, I summarize the key findings and contributions of this study.

8.1.1 Representation, Acquisition, and Incorporation of Implicit Knowledge Facilitates Deeper Understanding of Language

The accuracy and coverage of language analysis greatly depends on how we define a task and represent associated knowledge, as emphasized in Chapter 7. Ill-defined problem settings hinder obtaining optimal results through NLP systems and fail to capture important language phenomena. Conversely, NLP systems benefit from accurate and adequate task designs in acting on language at a higher level.

This thesis has presented several improved task designs and knowledge representations. Chapter 3 demonstrates that the inclusion of basic human motives provides a domain-agnostic view of sentiment and helps us understand why and how human sentiment is realized in various eventualities. Chapters 5 and 6 show that situational context in input enables systems to grasp a high-level semantic and pragmatic understanding of a user request. Chapter 7 proposes a new representation method of cooking recipes that addresses the accuracy and coverage of prior work. These studies also introduce annotated resources to facilitate future research.

8.1.2 The Consideration of Implicit Knowledge Leads to More Controllable and Interpretable Language Technologies

Effective communication between individuals would be challenging even for humans if there were not sufficient situational context available. Consider a scenario where you receive a phone request from an unknown person without prior knowledge of their identity or the conversation's purpose. In such a situation, meaningful interactions would be difficult, if not impossible, without seeking clarifications to establish context. However, the development and evaluation of language technologies often rely on this setting, which could be a cause of the common errors observed in modern generative NLP systems, such as the controllability of system behavior. Chapter 6 provides empirical evidence that response generation systems generate coherent output with provided situational context, but when it is not sufficiently presented, systems can learn to improvise contextual information, generating mutually-contradictory responses within N-best candidates. This result highlights the importance of appropriately representing and incorporating eventuality knowledge that is omitted from human communication.

In addition to improving controllability, implicit knowledge also enables systems to produce more interpretable output. The human motive detection proposed in Chapter 3 provides a modular approach toward sentiment analysis, where human sentiment is decomposed into underlying human motives, their associated eventualities and conditions (i.e., satisfied or dissatisfied). The system output has rich implications for service and product development. Chapter 4 demonstrates that very short, under-specified descriptions of To-Do tasks can be linked with eventuality knowledge that humans are also likely to possess. Training on such supervision enables text encoders to evaluate the similarity of To-Do tasks from the eventuality-centric viewpoint (e.g., judging if To-Do tasks share similar prerequisite or next actions).

8.2 Limitations and Future Directions

Finally, this section discusses the limitations of the thesis and potential opportunities for future research.

8.2.1 Extensions to Large-scale PLMs

As discussed in Chapter 2, cutting-edge large-scale PLMs can store and utilize world knowledge to perform various language tasks, even with no or only a few task-specific training signals. Researchers are actively exploring the application of such advanced models by prompting them to perform step-by-step reasoning (Wei et al., 2023), framing a reasoning task as the generation of programming code (Madaan et al., 2022), and feeding supplementary background knowledge (Zhou et al., 2022b). These emerging techniques were not extensively discussed in this thesis. However, in a nutshell, they are showing that we need to represent, acquire, and feed essential knowledge for making the most of pre-trained models, which aligns with the main claim of this thesis.

Nevertheless, the opaque nature of these complex models makes it unclear whether they ground language in world knowledge or simply reuse patterns found in training data. Furthermore, it remains unclear how and when these models use implicit knowledge to process language. To advance the state-of-the-art in language technologies, future research should establish systematic methods for evaluating the behavior of pre-trained language models.

8.2.2 Types of Knowledge

This thesis focuses on eventuality-centric knowledge due to its close connection to the target applications. However, other types of knowledge also play significant roles in language processing, such as knowledge specific to a certain domain or centered around entities. When certain facts and inferences are shared and evident to participants in communication, they can constitute implicit knowledge, regardless of the topic. Future research could explore extending the methods presented in this thesis to other types of implicit knowledge.

Chapters 5 and 6 demonstrate that proactive conversations require knowledge about the user's behavior and possessions. This underlines the value of personalized knowledge, which is a type of situational knowledge that focuses on individuals. While personalized knowledge has already been recognized as valuable in some applications, including dialogue systems, search engines, and recommendation systems, other NLP applications can also benefit from personalized knowledge for disambiguating situated language use. In Chapter 4 we observed that users of To-Do management apps describe To-Do tasks as a reminder for themselves, resulting in short and under-specified texts that are often difficult to interpret without knowing the author and context. While obtaining real data on personalized knowledge may be challenging due to privacy concerns, studying language comprehension based on personalized knowledge is an important area for future research.

8.2.3 Modality of Knowledge

Text is commonly used as the primary input modality in various NLP applications due to its availability and flexibility in representing a wide range of information and its ease of use in conjunction with PLMs. For this reason, this thesis has focused on textual representations of knowledge. However, knowledge can also be efficiently represented and acquired through other modalities such as visual images and audio. For instance, in cooking recipes, essential yet obvious information may be explicitly observable in attached pictures, even if it is not included in the text (e.g., "*using a knife* to cut vegetables"). In fact, information that can be observed from non-verbal sources is often omitted from texts. Therefore, future research should investigate how non-verbal contextual information can be utilized to handle semantic gaps in communication more robustly.

8.2.4 Common ground

In language communication, a speaker may assume that the listeners know or can easily infer certain facts, even if the information is highly personal, specialized, or context-dependent. This is one of the important aspects associated with the prevalence of knowledge. To handle this, NLP systems must model how common ground is established during communication through co-observations, clarification, and acknowledgment.

Modeling common ground computationally is a challenging task. It remains unclear whether data-driven models can acquire the concept of common ground solely from data. Empirical analyses presented in the response generation study indicated that GPT-3 may skip necessary clarifications and produce responses that sound abrupt. This result underscores the need for dedicated processing of common ground, as implemented in modularized dialogue systems. It is also possible that other types of non-conversational NLP applications can benefit from considering the notion of common ground.

Bibliography

Armen Aghajanyan, Anchit Gupta, Akshat Shrivastava, Xilun Chen, Luke Zettlemoyer, and Sonal Gupta. 2021. Muppet: Massive Multi-task Representations with Pre-Finetuning. *arXiv*. 4.3.4, 4.5

Reina Akama, Sho Yokoi, Jun Suzuki, and Kentaro Inui. 2020. Filtering Noisy Dialogue Corpora by Connectivity and Content Relatedness. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 941–958, Online. Association for Computational Linguistics. 5.2.2, 5.3.2, 5.5

James F Allen and C. Raymond Perrault. 1980. Analyzing Intention in Utterances. *Artificial Intelligence*, 15(3):143–178. 1, 2.1.4.1, 5.1, 5.6

Tawfiq Ammari, Jofish Kaye, Janice Y. Tsai, and Frank Bentley. 2019. Music, Search , and IoT: How People (Really) Use Voice Assistants. *ACM Transactions on Computer-Human Interaction*, 26(3):17:1–17:28. 5.1

Akari Asai, Sara Evensen, Behzad Golshan, Alon Halevy, Vivian Li, Andrei Lopatenko, Daniela Stepanov, Yoshihiko Suhara, Wang-Chiew Tan, and Yinzhan Xu. 2018. HappyDB: A Corpus of 100,000 Crowdsourced Happy Moments. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation*, Miyazaki, Japan. European Language Resources Association. 2.1.4.1

Emmon Bach. 1986. The Algebra of Events. *Linguistics and Philosophy*, 9(1):5–16. 2.1.4

Lisa Bauer and Mohit Bansal. 2021. Identify, Align, and Integrate: Matching Knowledge Graphs to Commonsense Reasoning Tasks. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2259–2272, Online. Association for Computational Linguistics. 2.2.2

Nicola Bellini and Laetitia Convert. 2016. The Concierge. Tradition, Obsolescence and Innovation in Tourism. *Symphonya. Emerging Issues in Management*, 0(2):17. 5.1

Emily M Bender and Alexander Koller. 2020. Climbing towards NLU: On Meaning, Form, and Understanding in the Age of Data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5185–5198, Online. Association for Computational Linguistics. 2.2.1

Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Scott Wen-tau Yih, and Yejin Choi. 2020. Abductive Commonsense Reasoning. *The Eighth International Conference on Learning Representations*. 5.2.2, 5.3.2, C.2.1

Yonatan Bisk, Ari Holtzman, Jesse Thomason, Jacob Andreas, Yoshua Bengio, Joyce Chai, Mirella Lapata, Angeliki Lazaridou, Jonathan May, Aleksandr Nisnevich, Nicolas Pinto, and Joseph Turian. 2020. Experience Grounds Language. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 8718–8735, Online. Association for Computational Linguistics. 2.2.1

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. PIQA: Reasoning about Physical Commonsense in Natural Language. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, New York City, USA. AAAI Press. 7.2.1

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146. B.4

Antoine Bosselut, Ronan Le Bras, and Yejin Choi. 2021. Dynamic Neuro-Symbolic Knowledge Graph Construction for Zero-shot Commonsense Question Answering. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*, pages 4923–4931. 2.2.2

Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. 2018. Simulating Action Dynamics with Neural Process Networks. *Sixth International Conference on Learning Representations*. 2.1.4.2, 7.1, 7.2.2, 7.3.1, 7.3.2, 7.4.1

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: Commonsense Transformers for Automatic Knowledge Graph Construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, Florence, Italy. 2.2.2, 4.1, 8, 5.1, 6

Gordon H Bower, John B Black, and Terrence J Turner. 1979. Scripts in Memory for Text. *Cognitive Psychology*, 11(2):177–220. 2.1.4

Andrei Broder. 2002. A Taxonomy of Web Search. *SIGIR Forum*, 36(2):3–10. 4.2.2

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. *arXiv*. 1, 2.2.1, 3.4.1.4, 6.1, 7.6.1

Paweł Budzianowski and Ivan Vulić. 2019. Hello, It's GPT-2 - How Can I Help You? Towards the

Use of Pretrained Language Models for Task-Oriented Dialogue Systems. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 15–22, Hong Kong. Association for Computational Linguistics. 5.4.1

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics. 2.1.4.1, 5.1, 5.2.1, 6.2.1

Bill Byrne, Karthik Krishnamoorthi, Chinnadhurai Sankar, Arvind Neelakantan, Ben Goodrich, Daniel Duckworth, Semih Yavuz, Amit Dubey, Kyu-Young Kim, and Andy Cedilnik. 2019. Taskmaster-1: Toward a Realistic and Diverse Dialog Dataset. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4516–4525, Hong Kong. Association for Computational Linguistics. 2.1.4.1, 5.2.1

Jose Camacho-Collados and Mohammad Taher Pilehvar. 2018. From Word to Sense Embeddings: A Survey on Vector Representations of Meaning. *Journal of Artificial Intelligence Research*, 63(1):743–788. 4.1

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, Brussels, Belgium. Association for Computational Linguistics. 4.1

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised Learning of Narrative Event Chains. In *Proceedings of ACL-08: HLT*, pages 789–797, Columbus, Ohio, USA. Association for Computational Linguistics. 2.1.4, 3.6

Khyathi Chandu, Eric Nyberg, and Alan W Black. 2019. Storyboarding of Recipes: Grounded Contextual Generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6040–6046, Florence, Italy. Association for Computational Linguistics. 2.1.4.2, 7.2.1

Snigdha Chaturvedi, Dan Goldwasser, and Hal Daumé III. 2016. Ask, and Shall You Receive?: Understanding Desire Fulfillment in Natural Language Text. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2697–2703, Phoenix, Arizona, USA. 2.1.4.1

Sneha Chaudhari, Varun Mithal, Gungor Polatkan, and Rohan Ramanath. 2021. An Attentive Survey of Attention Models. *ACM Transactions on Intelligent Systems and Technology*, 12(5). 4.3.2

Ethan A. Chi, John Hewitt, and Christopher D. Manning. 2020. Finding Universal Grammatical

Relations in Multilingual BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5564–5577, Online. Association for Computational Linguistics. 1, 2.2.2

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734. Association for Computational Linguistics. 4.3.4

Wonae Cho, Robert T Sumichrast, and Michael D Olsen. 1996. Expert-System Technology for Hotels: Concierge Application. *Cornell Hotel and Restaurant Administration Quarterly*, 37(1):54–60. 5.1

Cuong Xuan Chu, Niket Tandon, and Gerhard Weikum. 2017. Distilling Task Knowledge from How-To Communities. In *Proceedings of the 26th International Conference on World Wide Web*, pages 805–814, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee. 2.1.4.2

Herbert H. Clark. 1996. *Using language*. Cambridge University Press. 6.5

Herbert H. Clark and Edward F. Schaefer. 1989. Collaborating on Contributions to Conversations. In *Language Processing in Social Context*, volume 54 of *North-Holland Linguistic Series: Linguistic Variations*, pages 123–152. Elsevier. 5.2.1

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019. What Does BERT Look at? An Analysis of BERT's Attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics. 2.1.1

Peter Clark, Bhavana Dalvi, and Niket Tandon. 2018. What Happened? Leveraging VerbNet to Predict the Effects of Actions in Procedural Text. *arXiv*. 2.1.4.2, 7.2.2

Paul A. Crook, Shivani Poddar, Ankita De, Semir Shafi, David Whitney, Alborz Geramifard, and Rajen Subba. 2019. SIMMC: Situated Interactive Multi-Modal Conversational Data Collection And Evaluation Platform. *IEEE Workshop on Automatic Speech Recognition and Understanding*. 5.1, 5.2.1

Leyang Cui, Yu Wu, Shujie Liu, Yue Zhang, and Ming Zhou. 2020. MuTual: A Dataset for Multi-Turn Dialogue Reasoning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1406–1416, Online. Association for Computational Linguistics. 6.1, 6.2.1, 2

Christopher Culy. 1996. Null objects in English recipes. *Language Variation and Change*, 8(1):91–124. 7.1, 7.2.1, 7.3.1, 7.3.2

Jeff Da, Ronan Le Bras, Ximing Lu, Yejin Choi, and Antoine Bosselut. 2021. Analyzing Commonsense Emergence in Few-shot Knowledge Models. In *Proceedings of the 3rd Conference on Automated Knowledge Base Construction*. Automated Knowledge Base Construction. 2.2.2

Bhavana Dalvi, Lifu Huang, Niket Tandon, Wen-tau Yih, and Peter Clark. 2018. Tracking State Changes in Procedural Text: a Challenge Dataset and Models for Process Paragraph Comprehension. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1595–1604, New Orleans, Louisiana, USA. Association for Computational Linguistics. 2.1.4.2, 7.1, 7.1, 7.2.1, 7.2.2, 7.3.2

Marie-Catherine de Marneffe, Scott Grimm, and Christopher Potts. 2009. Not a Simple Yes or No: Uncertainty in Indirect Answers. In *Proceedings of the SIGDIAL 2009 Conference*, pages 136–143, London, UK. Association for Computational Linguistics. 5.2.1

Jan Deriu, Alvaro Rodrigo, Arantxa Otegi, Guillermo Echegoyen, Sophie Rosset, Eneko Agirre, and Mark Cieliebak. 2020. Survey on Evaluation Methods for Dialogue Systems. *Artificial Intelligence Review*, pages 1–56. 5.2.2

Jan Deriu, Don Tuggener, Pius Von Däniken, and Mark Cieliebak. 2022. Probing the robustness of trained metrics for conversational dialogue systems. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 750–761, Dublin, Ireland. Association for Computational Linguistics. 9, 11

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics. 2.2.1, 3.4.1.3, 4.1, 4.3.2, 4.4.3, 2, B.6, B.6

Emily Dinan, Varvara Logacheva, Valentin Malykh, Alexander Miller, Kurt Shuster, Jack Urbanek, Douwe Kiela, Arthur Szlam, Iulian Serban, Ryan Lowe, Shrimai Prabhumoye, Alan W. Black, Alexander Rudnicky, Jason Williams, Joelle Pineau, Mikhail Burtsev, and Jason Weston. 2020. The Second Conversational Intelligence Challenge (ConvAI2). *The NeurIPS '18 Competition*, pages 187–208. 5.2.1, 6.2.1, 3

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019. Wizard of Wikipedia: Knowledge-Powered Conversational Agents. *The Seventh International Conference on Learning Representations*. 2.1.2, 5.2.1, 6.2.1

Haibo Ding and Ellen Riloff. 2018. Human Needs Categorization of Affective Events Using Labeled and Unlabeled Data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1919–1929, New Orleans, Louisiana, USA. Association for Computational Linguistics. 2.1.4.1, 3.1

Lucia Donatelli, Theresa Schmidt, Debanjali Biswas, Arne Köhn, Fangzhou Zhai, and Alexander Koller. 2021. Aligning Actions Across Recipe Graphs. In *Proceedings of the 2021 Conference on*

Empirical Methods in Natural Language Processing, pages 6930–6942, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. 7.3.2

Rotem Dror, Gili Baumer, Marina Bogomolov, and Roi Reichart. 2017. Replicability Analysis for Natural Language Processing: Testing Significance with Multiple Datasets. *Transactions of the Association for Computational Linguistics*, 5:471–486. 4.4.4

Xinya Du, Bhavana Dalvi, Niket Tandon, Antoine Bosselut, Wen-tau Yih, Peter Clark, and Claire Cardie. 2019. Be Consistent! Improving Procedural Text Comprehension using Label Consistency. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2347–2356, Minneapolis, Minnesota. Association for Computational Linguistics. 7.2.2

Yanai Elazar, Victoria Basmov, Shauli Ravfogel, Yoav Goldberg, and Reut Tsarfaty. 2020. The Extraordinary Failure of Complement Coercion Crowdsourcing. In *Proceedings of the First Workshop on Insights from Negative Results in NLP*, pages 106–116, Online. Association for Computational Linguistics. 5.5

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. Measuring and Improving Consistency in Pretrained Language Models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031. 6.6

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, Adarsh Kumar, Anuj Goyal, Peter Ku, and Dilek Hakkani-Tur. 2020. MultiWOZ 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 422–428, Marseille, France. European Language Resources Association. 5.2.1

Biaoyan Fang, Timothy Baldwin, and Karin Verspoor. 2022. What does it take to bake a cake? The RecipeRef corpus and anaphora resolution in procedural text. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3481–3495, Dublin, Ireland. Association for Computational Linguistics. 7.1, 7.3.2, 7.6.1

Richard E Fikes and Nils J Nilsson. 1971. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2(3):189–208. 2.1.4.2

Charles Fillmore, Russell Lee-Goldman, and Russell Rhodes. 2012. The FrameNet Constructicon. In Hans C. Boas and Ivan A. Sag, editors, *Sign-Based Construction Grammar*. Center for the Study of Language and Information. 7.4.2, 7.6.2

Sarah E. Finch, James D. Finch, Ali Ahmadvand, Choi Ingyu (Jason), Xiangjue Dong, Ruixiang Qi, Harshita Sahijwani, Sergey Volokhin, Zihan Wang, Zihao Wang, and Jinho D. Choi. 2019. Emora: An Inquisitive Social Chatbot Who Cares For You. In *3rd Proceedings of Alexa Prize*. 6.5

Joseph L. Fleiss. 1971. Measuring Nominal Scale Agreement among Many Raters. *Psychological Bulletin*, 76(5):378–382. E.1.1

Kenneth D Forbus. 1984. Qualitative Process Theory. *Artificial Intelligence*, 24(1):85–168. 2.1.4.2, 7.6.1

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software*, pages 1–6, Melbourne, Australia. Association for Computational Linguistics. A.1

Cornelia Gerhardt. 2013. Food and language – language and food. In *Culinary Linguistics: The Chef's Special*, pages 3–49. John Benjamins Publishing Company. 7.2.1, 7.3.2

Deepanway Ghosal, Siqi Shen, Navonil Majumder, Rada Mihalcea, and Soujanya Poria. 2022. CICERO: A Dataset for Contextualized Commonsense Inference in Dialogues. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 5010–5028, Dublin, Ireland. Association for Computational Linguistics. 6.2.1, 2, 6.5

Raymond W Gibbs and Gregory A Bryant. 2008. Striving for Optimal Relevance When Answering Questions. *Cognition*, 106(1):345–369. 5.2.1

Yolanda Gil. 2015. Human Tutorial Instruction in the Raw. *ACM Transactions on Interactive Intelligent Systems*, 5(1):2:1–2:29. 7.3.2

Yolanda Gil, Varun Ratnakar, Timothy Chklovski, Paul Groth, and Denny Vrandecic. 2012. Capturing Common Knowledge about Tasks: Intelligent Assistance for To-Do Lists. *ACM Transactions on Information Systems*, 2(3). 4.5

J.J. Godfrey, E.C. Holliman, and J. McDaniel. 1992. SWITCHBOARD: telephone speech corpus for research and development. In *[Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 517–520 vol.1. 6.2.1

Andrew B Goldberg, Nathanael Fillmore, David Andrzejewski, Zhiting Xu, Bryan Gibson, and Xiaojin Zhu. 2009. May All Your Wishes Come True: A Study of Wishes and How to Recognize Them. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 263–271, Boulder, Colorado. Association for Computational Linguistics. 2.1.4.1

Andrew S. Gordon and Jerry R. Hobbs. 2004. Formalizations of Commonsense Psychology. *AI Magazine*, 25(4):49–49. 2.1.4.1, 5.2.1

Jonathan Gordon and Benjamin Van Durme. 2013. Reporting Bias and Knowledge Acquisition. In *Proceedings of the 3rd Workshop on Automated Knowledge Base Construction*. ACM Press. 2.2.2

Kyle Gorman and Steven Bedrick. 2019. We Need to Talk about Standard Splits. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2786–2791, Florence, Italy. Association for Computational Linguistics. 4.4.2, 4.4.4

David Graus, Paul N Bennett, Ryen W White, and Eric Horvitz. 2016. Analyzing and Predicting Task Reminders. In *Proceedings of the 2016 Conference on User Modeling Adaptation and*

Personalization, pages 7–15, Halifax, NS, Canada. ACM. 4.4.1, 4.5

H. P. Grice. 1975. Logic and Conversation. In *Speech Acts*, number 3 in Syntax and Semantics, pages 41 – 58. Academic Press, New York, NY. 2.1.3, 5.1

Aditya Gupta and Greg Durrett. 2019. Effective Use of Transformer Networks for Entity Tracking. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 759–769, Hong Kong, China. Association for Computational Linguistics. 1

Prakhar Gupta, Yulia Tsvetkov, and Jeffrey Bigham. 2021. Synthesizing Adversarial Negative Responses for Robust Response Ranking and Evaluation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3867–3883, Online. Association for Computational Linguistics. 5.3.2

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't Stop Pretraining: Adapt Language Models to Domains and Tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics. 4.4.3

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation Artifacts in Natural Language Inference Data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 107–112, New Orleans, Louisiana. Association for Computational Linguistics. 5.5

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. REALM: Retrieval Augmented Language Model Pre-Training. In *Proceedings of the 37th International Conference on Machine Learning*, pages 3929–3938. PMLR. 2.2.2

Janosch Haber, Tim Baumgärtner, Ece Takmaz, Lieke Gelderloos, Elia Bruni, and Raquel Fernández. 2019. The PhotoBook dataset: Building common ground through visually-grounded dialogue. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1895–1910, Florence, Italy. Association for Computational Linguistics. 6.2.1

Janghoon Han, Taesuk Hong, Byoungjae Kim, Youngjoong Ko, and Jungyun Seo. 2021. Fine-grained Post-training for Improving Retrieval-based Dialogue Systems. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1549–1558, Online. Association for Computational Linguistics. 5.4.1, 1, 2, 5.4.1

Patrick J. Hayes. 1985. The Second Naive Physics Manifesto. In *Formal Theories of the Commonsense World*, Ablex Series in Artificial Intelligence, pages 1–36. Ablex Publishing Corp., Norwood, New jersey, USA. 2.1.4.2, 7.6.1

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-Answer Driven Semantic Role

Labeling: Using Natural Language to Annotate Natural Language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 643–653, Lisbon, Portugal. Association for Computational Linguistics. E.1.2

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021a. DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing. *arXiv*. 2.2.1, 3.4.1.3, 4

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021b. DeBERTa: Decoding-Enhanced BERT with Disentangled Attention. *The Ninth International Conference on Learning Representations*. 4

Behnam Hedayatnia, Di Jin, Yang Liu, and Dilek Hakkani-Tur. 2022. A Systematic Evaluation of Response Selection for Open Domain Dialogue. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 298–311, Edinburgh, UK. Association for Computational Linguistics. 5.2.2, 5.3.2, 2, 4, 6.2.2

Matthew Henderson, Iñigo Budzianowski Pawełand Casanueva, Sam Coope, Daniela Gerz, Girish Kumar, Nikola Mrkšić, Georgios Spithourakis, Pei-Hao Su, Ivan Vulić, and Tsung-Hsien Wen. 2019. A Repository of Conversational Datasets. In *Proceedings of the First Workshop on NLP for Conversational AI*, pages 1–10, Florence, Italy. Association for Computational Linguistics. 5.1, 5.2.2, 5.3.2, 5.3.2, 6.2.1

John Hewitt and Christopher D Manning. 2019. A Structural Probe for Finding Syntax in Word Representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics. 2.1.1

Beth Ann Hockey, Deborah Rossen-Knill, Beverly Spejewski, Matthew Stone, and Stephen Isard. 1997. Can you predict answers to yes/no questions? Yes, no and stuff. In *Proceedings of the Fifth European Conference on Speech Community and Technology*, pages 2267–2270, Rhodes, Greece. 5.2.1

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python. 4.2.1, 5

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning Whom to Trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130. Association for Computational Linguistics. 3.3, E.1.1

Hao Huang, Xiubo Geng, Jian Pei, Guodong Long, and Daxin Jiang. 2021. Reasoning over Entity-Action-Location Graph for Procedural Text Understanding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 5100–5109, Online. Association for Computational

Linguistics. 7.2.2

Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2020. Poly-encoders: Transformer Architectures and Pre-training Strategies for Fast and Accurate Multi-sentence Scoring. *The Eighth International Conference on Learning Representations*. 6.2.1

Jena D. Hwang, Chandra Bhagavatula, Ronan Le Bras, Jeff Da, Keisuke Sakaguchi, Antoine Bosselut, and Yejin Choi. 2021. COMET-ATOMIC 2020: On Symbolic and Neural Commonsense Knowledge Graphs. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*, pages 6384–6392, Online. 2.2.2, 4.1, 4.3.3.2, 8, 5.1, 5.3.2

Filip Ilievski, Alessandro Oltramari, Kaixin Ma, Bin Zhang, Deborah L. McGuinness, and Pedro Szekely. 2021. Dimensions of Commonsense Knowledge. *arXiv:2101.04640 [cs]*. 2.1.3, 7.6.2

Peter Jansen, Niranjan Balasubramanian, Mihai Surdeanu, and Peter Clark. 2016. What's in an Explanation? Characterizing Knowledge and Inference Requirements for Elementary Science Exams. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2956–2965, Osaka, Japan. The COLING 2016 Organizing Committee. 2.1.1

Sujay Kumar Jauhar, Nirupama Chandrasekaran, Michael Gamon, and Ryen White. 2022. MS-LaTTE: A Dataset of Where and When To-do Tasks are Completed. In *Proceedings of the Language Resources and Evaluation Conference*, pages 5393–5403, Marseille, France. European Language Resources Association. 4.1, 4.4.1

Yiwei Jiang, Klim Zaporojets, Johannes Deleu, Thomas Demeester, and Chris Develder. 2020. Recipe Instruction Semantics Corpus (RISeC): Resolving Semantic Structure and Zero Anaphora in Recipes. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 821–826, Suzhou, China. Association for Computational Linguistics. 7.3.1, 7.3.2, 7.6.1

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2022. Large Language Models Struggle to Learn Long-Tail Knowledge. *arXiv*. 2.2.2

Katharina Kann, Abteen Ebrahimi, Joewie Koh, Shiran Dudy, and Alessandro Roncone. 2022. Open-domain Dialogue Generation: What We Can Do, Cannot Do, And Should Do Next. In *Proceedings of the 4th Workshop on NLP for Conversational AI*, pages 148–165, Dublin, Ireland. Association for Computational Linguistics. 5

Ghazaleh Kazeminejad, Martha Palmer, Tao Li, and Vivek Srikumar. 2021. Automatic Entity State Annotation using the VerbNet Semantic Parser. In *Proceedings of the Joint 15th Linguistic Annotation Workshop and 3rd Designing Meaning Representations Workshop*, pages 123–132, Punta Cana, Dominican Republic. Association for Computational Linguistics. 7.2.2

Atsushi Keyaki and Jun Miyazaki. 2017. Part-of-speech Tagging for Web Search Queries Using a

Large-scale Web Corpus. In *Proceedings of the Symposium on Applied Computing*, pages 931–937, New York, NY, USA. ACM. 4.2.1

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally Coherent Text Generation with Neural Checklist Models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 329–339, Austin, Texas. Association for Computational Linguistics. 2.1.4.2, 7.3.1

Hyunwoo Kim, Youngjae Yu, Liwei Jiang, Ximing Lu, Daniel Khashabi, Gunhee Kim, Yejin Choi, and Maarten Sap. 2022. ProsocialDialog: A Prosocial Backbone for Conversational Agents. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4005–4029, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics. 6.2.1

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics. 3.4.1.2

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. *The Third International Conference for Learning Representations*. 5.4.1, A.1

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2020. Attention is Not Only a Weight: Analyzing Transformers with Vector Norms. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 7057–7075, Online. Association for Computational Linguistics. 4.4.5

Mojtaba Komeili, Kurt Shuster, and Jason Weston. 2022. Internet-Augmented Dialogue Generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 8460–8478, Dublin, Ireland. Association for Computational Linguistics. 2.2.2, 6.1

Satwik Kottur, José M. F. Moura, Devi Parikh, Dhruv Batra, and Marcus Rohrbach. 2019. CLEVR-Dialog: A Diagnostic Dataset for Multi-Round Reasoning in Visual Dialog. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 582–595, Minneapolis, Minnesota. Association for Computational Linguistics. 5.1

Klaus Krippendorff. 2006. Reliability in Content Analysis: Some Common Misconceptions and Recommendations. *Human Communication Research*, 30(3):411–433. 5.3.2, E.1.1

Chaitanya Kulkarni, Wei Xu, Alan Ritter, and Raghu Machiraju. 2018. An Annotated Corpus for Machine Reading of Instructions in Wet Lab Protocols. In *Proceedings of the 2018 Conference of the North {A}merican Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 97–106, New Orleans, Louisiana. Association for Computational Linguistics. 2.1.4.2, 7.1, 7.3.2

Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. A Survey on Complex Knowledge Base Question Answering: Methods, Challenges and Solutions. In

Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, pages 4483–4491. International Joint Conferences on Artificial Intelligence Organization. 1

Paul Landes. 2018. Supervised approach imperative to-do list categorization. `https://github.com/plandes/todo-task`. Accessed: 2022-03-04. 4.1

Paul Landes and Barbara Di Eugenio. 2018. A Supervised Approach To The Interpretation Of Imperative To-Do Lists. *arXiv*. 4.1, 4.2.1, 4.4.1, 4.5

Douglas B. Lenat. 1995. CYC: A Large-scale Investment in Knowledge Infrastructure. *Communications of the ACM*, 38(11):33–38. 1, 2.1.3, 7.6.2

Stephen C Levinson. 1983. *Pragmatics*. Cambridge University Press. 2.1.3

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics. 2.2.1, 4.3.3.1, 4.3.3.2, 5.3.2

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020b. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc. 2.2.2

Jia Li, Chongyang Tao, Wei Wu, Yansong Feng, Dongyan Zhao, and Rui Yan. 2019. Sampling Matters! An Empirical Study of Negative Sampling Strategies for Learning of Matching Models in Retrieval-based Dialogue Systems. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 1291–1296, Hong Kong, China. Association for Computational Linguistics. 5.3.2

Jiwei Li and Eduard Hovy. 2017. Reflections on Sentiment/Opinion Analysis. In *A Practical Guide to Sentiment Analysis*, pages 41–59. Springer, Cham. 2.1.4.1, 3.1, 3.5

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. DailyDialog: A Manually Labelled Multi-turn Dialogue Dataset. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing*, pages 986–995, Taipei, Taiwan. Asian Federation of Natural Language Processing. 5.1, 6.1, 3, 6.2.1, 2

Yaobo Liang, Chenfei Wu, Ting Song, Wenshan Wu, Yan Xia, Yu Liu, Yang Ou, Shuai Lu, Lei Ji, Shaoguang Mao, Yun Wang, Linjun Shou, Ming Gong, and Nan Duan. 2023. TaskMatrix.AI: Completing Tasks by Connecting Foundation Models with Millions of APIs. *arXiv*. 6.5

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring How Models Mimic Human Falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computa-*

tional Linguistics, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics. 1, 2.2.2

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535. 2.1.1

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132, Austin, Texas. Association for Computational Linguistics. 6.2.2, 9, 11

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022a. What Makes Good In-Context Examples for GPT-3? In *Proceedings of Deep Learning Inside Out: The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online. Association for Computational Linguistics. 6.6

Jiacheng Liu, Alisa Liu, Ximing Lu, Sean Welleck, Peter West, Ronan Le Bras, Yejin Choi, and Hannaneh Hajishirzi. 2022b. Generated Knowledge Prompting for Commonsense Reasoning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 3154–3169, Dublin, Ireland. Association for Computational Linguistics. 2.2.2

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Computing Surveys*, 55(9):195:1–195:35. 2.2.1

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Multi-Task Deep Neural Networks for Natural Language Understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, Florence, Italy. Association for Computational Linguistics. 4.5

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv*. 2.2.1, 3.4.1.3, 4.3.2, 4.4.3, 3

Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. *The Seventh International Conference on Learning Representations*. 4.4.2.1

Annie Louis, Dan Roth, and Filip Radlinski. 2020. "I'd rather just go to bed": Understanding Indirect Answers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 7411–7425, Online. Association for Computational Linguistics. 5.2.1

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 285–294, Prague, Czech Republic. Association for Computational Linguistics. 5.1, 5.2.2, 5.4.1, 1

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics. 4.3.4

Kaixin Ma, Filip Ilievski, Jonathan Francis, Yonatan Bisk, Eric Nyberg, and Alessandro Oltramari. 2021. Knowledge-driven Data Construction for Zero-shot Evaluation in Commonsense Question Answering. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*, volume 35, pages 13507–13515. 2.2.2

Aman Madaan, Shuyan Zhou, Uri Alon, Yiming Yang, and Graham Neubig. 2022. Language Models of Code are Few-Shot Commonsense Learners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1384–1403, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics. 7.2.2, 7.6.1, 8.2.1

Hirokuni Maeta, Tetsuro Sasada, and Shinsuke Mori. 2015. A Framework for Procedural Text Understanding. In *Proceedings of the 14th International Conference on Parsing Technologies*, pages 50–60, Bilbao, Spain. Association for Computational Linguistics. 7.1, 7.2.2, 7.3.2

Bodhisattwa Prasad Majumder, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Julian McAuley. 2020. Like hiking? You probably enjoy nature: Persona-grounded Dialog with Commonsense Expansions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 9194–9206, Online. 6.2.1

Jonathan Malmaud, Earl Wagner, Nancy Chang, and Kevin Murphy. 2014. Cooking with Semantics. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, pages 33–38, Baltimore, Maryland. Association for Computational Linguistics. 7.3.1, 7.3.1, 7.3.2, 7.4.1

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland, USA. Association for Computational Linguistics. 3

Abraham Harold Maslow. 1943. A theory of Human Motivation. *Psychological Review*, 50(4):370–396. 2.1.4.1, 3.1, 3.2

John McCarthy and Patrick J. Hayes. 1969. Some Philosophical Problems from the Standpoint of Artificial Intelligence. *Machine Intelligence 4*, pages 463—-502. 15, 6.5

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119, Stateline, Nevada, USA. MIT Press. 2.2.1, 5, B.4

A. H. Miller, W. Feng, A. Fisch, J. Lu, D. Batra, A. Bordes, D. Parikh, and J. Weston. 2017. ParlAI: A Dialog Research Software Platform. *arXiv*. 1

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the Role of Demonstrations: What Makes In-Context Learning Work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics. 6.6

Marvin Minsky. 1974. A Framework for Representing Knowledge. *MIT-AI Laboratory Memo*, 306. 2.1.4, 7.4.2

Ishan Misra, C. Lawrence Zitnick, Margaret Mitchell, and Ross Girshick. 2016. Seeing through the Human Reporting Bias: Visual Classifiers from Noisy Human-Centric Labels. In *2016 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2930–2939. IEEE. 2.2.2

Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Justin Betteridge, Andrew Carlson, Bhavana Dalvi Mishra, Matthew Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, Thahir Mohamed, Ndapa Nakashole, Emmanouil Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, Richard Wang, Derry Wijaya, Abhinav Gupta, Xinlei Chen, Abulhair Saparov, Malcolm Greaves, and Joel Welling. 2015. Never-Ending Learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2302–2310, Austin, Texas, USA. AAAI Press. 2.1.2

Yasuhide Miura, Yuhao Zhang, Emily Tsai, Curtis Langlotz, and Dan Jurafsky. 2021. Improving Factual Completeness and Consistency of Image-to-Text Radiology Report Generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5288–5304, Online. Association for Computational Linguistics. 5.5

Nikita Moghe, Siddhartha Arora, Suman Banerjee, and Mitesh M Khapra. 2018. Towards Exploiting Background Knowledge for Building Conversation Systems. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2322–2332, Brussels, Belgium. Association for Computational Linguistics. 2.1.2

Yoshio Momouchi. 1980. Control Structures For Actions in Procedural Texts and PT-Chart. In *Proceedings of the 8th International Conference on Computational Linguistics*. 7.3.1

Seungwhan Moon, Satwik Kottur, Paul Crook, Ankita De, Shivani Poddar, Theodore Levin, David Whitney, Daniel Difranco, Ahmad Beirami, Eunjoon Cho, Rajen Subba, and Alborz Geramifard. 2020. Situated and interactive multimodal conversations. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1103–1121, Barcelona, Spain (Online). International Committee on Computational Linguistics. 5.2.1, 6.2.1

Robert C. Moore. 1984. A Formal Theory of Knowledge and Action. In *Formal Theories of the Commonsense World*, Ablex Series in Artificial Intelligence. Ablex Publishing Corp., Norwood, New jersey, USA. 2.1.4

Shinsuke Mori, Hirokuni Maeta, Yoko Yamakata, and Tetsuro Sasada. 2014. Flow Graph Corpus from Recipe Texts. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, Reykjavik, Iceland. European Language Resources Association. 2.1.4.2, 7.3.2

Katharina Morik, Peter Brockhausen, and Thorsten Joachims. 1999. Combining Statistical Learning with a Knowledge-Based Approach - A Case Study in Intensive Care Monitoring. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 268–277, Bled, Slovenia. Morgan Kaufmann Publishers. 3.4.2

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural Language Inference by Tree-Based Convolution and Heuristic Matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 130–136, Berlin, Germany. Association for Computational Linguistics. 4.4.2

Sudipto Mukherjee, Subhabrata Mukherjee, Marcello Hasegawa, Ahmed Hassan Awadallah, and Ryen White. 2020. Smart To-Do: Automatic Generation of To-Do Items from Emails. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8680–8689, Online. Association for Computational Linguistics. 4.1, 4.5

Sheshera Mysore, Zach Jensen, Edward Kim, Kevin Huang, Haw-Shiuan Chang, Emma Strubell, Jeffrey Flanigan, Andrew McCallum, and Elsa Olivetti. 2019. The Materials Science Procedural Text Corpus: Annotating Materials Synthesis Procedures with Shallow Semantic Structures. In *Proceedings of the 13th Linguistic Annotation Workshop*, pages 56–64, Florence, Italy. Association for Computational Linguistics. 7.3.2

Nima Nabizadeh, Dorothea Kolossa, and Martin Heckmann. 2020. MyFixit: An Annotated Dataset, Annotation Tool, and Baseline Methods for Information Extraction from Repair Manuals. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 2120–2128, Marseille, France. European Language Resources Association. 7.3.2

Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020. BERTweet: A pre-trained language model for English Tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14, Online. Association for Computational Linguistics. 4.5

Yixin Nie, Mary Williamson, Mohit Bansal, Douwe Kiela, and Jason Weston. 2021. I like fish, especially dolphins: Addressing Contradictions in Dialogue Modeling. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 1699–1713, Online. Association for Computational Linguistics. 6.2.2

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A Multilingual Treebank Collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation*, pages

1659–1666, Portorož, Slovenia. European Language Resources Association. 7.3.1

Elnaz Nouri, Robert Sim, Adam Fourney, and Ryen W White. 2020. Step-wise Recommendation for Complex Task Support. In *Proceedings of the 2020 Conference on Human Information Interaction and Retrieval*, pages 203–212, New York, NY, USA. ACM. 4.1, 4.5

Naoki Otani, Jun Araki, HyeongSik Kim, and Eduard Hovy. 2023a. A Textual Dataset for Situated Proactive Response Selection. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (To Appear)*, Toronto, Canada. Association for Computational Linguistics. 1

Naoki Otani, Jun Araki, HyeongSik Kim, and Eduard Hovy. 2023b. On the Underspecification of Situations in Open-domain Conversational Datasets. *In Submission*. 1

Naoki Otani, Jun Araki, HyeongSik Kim, and Eduard Hovy. 2023c. Semantic Frame Graphs for Cooking Recipe Understanding. *In Submission*. 1

Naoki Otani, Michael Gamon, Sujay Kumar Jauhar, Mei Yang, Sri Raghu Malireddi, and Oriana Riva. 2022. LITE: Intent-based Task Representation Learning Using Weak Supervision. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2410–2424, Seattle, United States. Association for Computational Linguistics. 1

Naoki Otani and Eduard Hovy. 2019. Toward Comprehensive Understanding of a Sentiment Based on Human Motives. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4672–4677. Association for Computational Linguistics. 1

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35*. 2.2.1

Alexander Panchenko, Eugen Ruppert, Stefano Faralli, Simone P Ponzetto, and Chris Biemann. 2018. Building a Web-Scale Dependency-Parsed Corpus from CommonCrawl. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation*, Miyazaki, Japan. European Language Resources Association. 4.2.1

Lalchand Pandia and Allyson Ettinger. 2021. Sorting through the Noise: Testing Robustness of Information Processing in Pre-trained Language Models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1583–1596, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. 1, 6.6

Nikolaos Pappas and James Henderson. 2019. GILE: A Generalized Input-Label Embedding for Text Classification. *Transactions of the Association for Computational Linguistics*, 7:139–155. 4.3.4

Cécile Paris, Keith Vander Linden, and Shijian Lu. 2002. Automated Knowledge Acquisition for Instructional Text Generation. In *Proceedings of the 20th Annual International Conference on Computer Documentation*, pages 142–151, New York, NY, USA. Association for Computing Machinery. 2.1.4.2, 7.2.1

Hogun Park and Hamid Reza Motahari Nezhad. 2018. Learning Procedures from Text: Codifying How-to Procedures in Deep Neural Networks. In *Companion Proceedings of the The Web Conference 2018*, pages 351–358, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee. 2.1.4.2

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc. 4.4.2.1, D.1

Ajay Patel, Alexander Sands, Chris Callison-Burch, and Marianna Apidianaki. 2018. Magnitude: A Fast, Efficient Universal Vector Embedding Utility Package. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 120–126, Brussels, Belgium. Association for Computational Linguistics. B.4

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830. 4.4.2

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics. 3.4.3, A.1

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. 2017. Regularizing Neural Networks by Penalizing Confident Output Distributions. *The Fifth International Conference on Learning Representations*. 4.3.4

C. Raymond Perrault. 1980. A Plan-Based Analysis of Indirect Speech Act. *American Journal of Computational Linguistics*, 6(3-4):167–182. 5.1, 5.2.1

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2227–2237. Association for Computational Linguistics. 2.2.1

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language Models as Knowledge Bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics. 1, 2.2.2

Massimo Poesio and Hannes Rieses. 2010. Completions, Coordination, and Alignment in Dialogue. *Dialogue & Discourse*, 1(1). 6.5

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad AL-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphee De Clercq, Veronique Hoste, Marianna Apidianaki, Xavier Tannier, Natalia Loukachevitch, Evgeniy Kotelnikov, Núria Bel, Salud María Jiménez-Zafra, and Gülşen Eryiğit. 2016. SemEval-2016 Task 5: Aspect Based Sentiment Analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, pages 19–30, San Diego, California. Association for Computational Linguistics. 3.1, 3.1, 3.3

Ellen F. Prince. 1978. On the function of existential presupposition in discourse. *Papers from the Fourteenth Regional Meeting of the Chicago Linguistic Society*, pages 362–376. 2.1.3

Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D. Manning. 2018. Universal Dependency Parsing from Scratch. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 160–170, Brussels, Belgium. Association for Computational Linguistics. 7.3.1

XiPeng Qiu, TianXiang Sun, YiGe Xu, YunFan Shao, Ning Dai, and XuanJing Huang. 2020. Pretrained Models for Natural Language Processing: A Survey. *Science China Technological Sciences*, 63(10):1872–1897. 2.2.1, 4.1

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2018. Language Models are Unsupervised Multitask Learners. Technical report, OpenAI. 2.2.1, 4.1, 4.3.2, 6.3.2, B.4, D.1.1

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140):1–67. 2.2.1, 2.2.1, 4.3.2, 7.6.1

Elahe Rahimtoroghi, Jiaqi Wu, Ruimin Wang, Pranav Anand, and Marilyn Walker. 2017. Modelling Protagonist Goals and Desires in First-Person Narrative. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 360–369, Saarbrücken, Germany. Association for Computational Linguistics. 5.2.1

Hannah Rashkin, Antoine Bosselut, Maarten Sap, Kevin Knight, and Yejin Choi. 2018. Modeling Naive Psychology of Characters in Simple Commonsense Stories. In *Proceedings of the 56th*

Annual Meeting of the Association for Computational Linguistics, pages 2289–2299, Melbourne, Australia. Association for Computational Linguistics. 2.1.4.1, 3.1

Hannah Rashkin, Sameer Singh, and Yejin Choi. 2016. Connotation Frames: A Data-Driven Investigation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 311–321, Berlin, Germany. Association for Computational Linguistics. 2.1.4.1

Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y-Lan Boureau. 2019. Towards Empathetic Open-domain Conversation Models: A New Benchmark and Dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5370–5381, Florence, Italy. Association for Computational Linguistics. 2.1.4.1, 5.2.1, 6.2.1

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. DeepSpeed: System Optimizations Enable Training Deep Learning Models with Over 100 Billion Parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, page 3505–3506, New York, NY, USA. Association for Computing Machinery. D.1.1

Nils Reimers. 2021. Pretrained Models — Sentence-Transformers documentation. `https://www.sbert.net/docs/pretrained_models.html`. Accessed: 2022-03-04. B.4

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics. 3.4.1.4, 4.1, 4.4.3, 4.4.4, 5.3.2, A.1, B.4, B.5, C.2.1

Steven Reiss. 2004. Multifaceted Nature of Intrinsic Motivation: The Theory of 16 Basic Desires. *Review of General Psychology*, 8(3):179–193. 2.1.4.1, 3.1, 3.2

Danilo Ribeiro, Thomas Hinrichs, Maxwell Crouse, Kenneth Forbus, Maria Chang, and Michael Witbrock. 2019. Predicting State Changes in Procedural Text using Analogical Question Answering. *Advances in Cognitive Systems*, pages 1–6. 2.1.4.2

Chuck Rieger. 1976. An Organization of Knowledge for Problem Solving and Language Comprehension. *Artificial Intelligence*, 7(2):89–127. 1

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised Modeling of Twitter Conversations. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 172–180, Los Angeles, California. Association for Computational Linguistics. 6.2.1

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-Driven Response Generation in Social Media. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 583–593, Edinburgh, Scotland, UK. Association for Computational Linguistics. 6.1

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How Much Knowledge Can You Pack Into the Parameters of a Language Model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 5418–5426, Online. Association for Computational Linguistics. 1, 2.2.2

Arpita Roy and Shimei Pan. 2021. Incorporating Medical Knowledge in BERT for Clinical Relation Extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5357–5366, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. 2.1.1

J Ruppenhofer, M Ellsworth, M R L Petruck, C R Johnson, and J Scheffczyk. 2016. FrameNet II: Extended Theory and Practice. 4.1, 4.3.3.3

Ananya B. Sai, Akash Kumar Mohankumar, Siddhartha Arora, and Mitesh M. Khapra. 2020. Improving Dialog Evaluation with a Multi-reference Adversarial Dataset and Large Scale Pretraining. *Transactions of the Association for Computational Linguistics*, 8:810–827. 6.2.2

Keisuke Sakaguchi, Chandra Bhagavatula, Ronan Le Bras, Niket Tandon, Peter Clark, and Yejin Choi. 2021. proScript: Partially Ordered Scripts Generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2138–2149, Punta Cana, Dominican Republic. Association for Computational Linguistics. 2.1.4

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. WinoGrande: An adversarial Winograd Schema Challenge at scale. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, New York City, USA. AAAI Press. 5.2.2, 5.3.2

Chinnadhurai Sankar, Sandeep Subramanian, Chris Pal, Sarath Chandar, and Yoshua Bengio. 2019. Do Neural Dialog Systems Use the Conversation History Effectively? An Empirical Study. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 32–37, Florence, Italy. Association for Computational Linguistics. 6.6

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A. Smith, and Yejin Choi. 2019. ATOMIC: An Atlas of Machine Commonsense for If-Then Reasoning. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, volume 33, pages 3027–3035. Association for the Advancement of Artificial Intelligence. 2.1.3, 2.1.4.1, 5.1, 5.3.2

Shoetsu Sato, Naoki Yoshinaga, Masashi Toyoda, and Masaru Kitsuregawa. 2017. Modeling Situations in Neural Chat Bots. In *Proceedings of {ACL} 2017, Student Research Workshop*, pages 120–127, Vancouver, Canada. Association for Computational Linguistics. 6.2.1

Roger C. Schank and Robert P. Abelson. 1977. *Scripts, Plans, Goals, and Understanding: An Inquiry Into Human Knowledge Structures*. Psychology Press, New York, NY, USA. 1, 2.1.4, 5.2.1

Lenhart Schubert. 2002. Can We Derive General World Knowledge from Texts? In *Proceedings of the Second International Conference on Human Language Technology Research*, pages 94–97. Morgan Kaufmann Publishers Inc. 2.1.2

Lenhart Schubert. 2015. What Kinds of Knowledge are Needed for Genuine Understanding? In *Proceedings of the First Workshop on Cognitive Knowledge Acquisition and Applications*, Buenos Aires, Argentina. 2.1.1

Karin Kipper Schuler. 2005. *VerbNet: A Broad-coverage, Comprehensive Verb Lexicon*. Ph.D. dissertation, University of Pennsylvania, Ann Arbor. 2.1.4.2

Iulian Vlad Serban, Ryan Lowe, Peter Henderson, Laurent Charlin, and Joelle Pineau. 2017. A Survey of Available Corpora for Building Data-Driven Dialogue Systems. *arXiv*. 5

Iulian Vlad Serban, Ryan Lowe, Peter Henderson, Laurent Charlin, and Joelle Pineau. 2018. A Survey of Available Corpora for Building Data-Driven Dialogue Systems. *Dialogue and Discourse*, 9(1):1–49. 5.2.1

Chirag Shah and Ryen W White. 2021. Bridging Task Expressions and Search Queries. In *Proceedings of the 2021 Conference on Human Information Interaction and Retrieval*, pages 319–323, New York, NY, USA. Association for Computing Machinery. 4.5

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural Responding Machine for Short-Text Conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1577–1586, Beijing, China. Association for Computational Linguistics. 6.2.1

Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. 2018. Baseline Needs More Love: On Simple Word-Embedding-Based Models and Associated Pooling Mechanisms. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 440–450, Melbourne, Australia. Association for Computational Linguistics. 3.4.1.2

Xiaoyu Shen, Ernie Chang, Hui Su, Cheng Niu, and Dietrich Klakow. 2020. Neural Data-to-Text Generation via Jointly Learning the Segmentation and Correspondence. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7155–7165, Online. Association for Computational Linguistics. 5.5

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed Chi, Nathanael Schärli, and Denny Zhou. 2023. Large Language Models Can Be Easily Distracted by Irrelevant Context. 2.2.1

Sola S. Shirai and HyeongSik Kim. 2022. EaT-PIM: Substituting Entities in Procedural Instructions Using Flow Graphs and Embeddings. *The Semantic Web – ISWC 2022*, pages 161–178. 7.1, 7.2.2, 7.3.2

Changjian Shui, Mahdieh Abbasi, Louis-Émile Robitaille, Boyu Wang, and Christian Gagné. 2019. A Principled Approach for Learning Task Similarity in Multitask Learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 3446–3452. International Joint Conferences on Artificial Intelligence Organization. 4.5

Kurt Shuster, Da Ju, Stephen Roller, Emily Dinan, Y-Lan Boureau, and Jason Weston. 2020. The Dialogue Dodecathlon: Open-Domain Knowledge and Image Grounded Conversational Agents. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2453–2470, Online. Association for Computational Linguistics. 4.5, 6.2.1

Kurt Shuster, Jack Urbanek, Arthur Szlam, and Jason Weston. 2022. Am I Me or You? State-of-the-Art Dialogue Models Cannot Maintain an Identity. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2367–2387, Seattle, United States. Association for Computational Linguistics. 6.2.2

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. MPNet: Masked and Permuted Pre-training for Language Understanding. In *Advances in Neural Information Processing Systems*, volume 33, pages 16857–16867. 3.4.1.4, 5.3.2, A.1, B.4, C.2.1

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A Neural Network Approach to Context-Sensitive Generation of Conversational Responses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 196–205, Denver, Colorado. Association for Computational Linguistics. 6.2.1

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 4444–4451, San Francisco, California, USA. AAAI Press. 1, 2.1.2, 2.1.3, 2.1.4.2, 8, 5.1, 5.3.2

Evangelia Spiliopoulou, Artidoro Pagnoni, Yonatan Bisk, and Eduard Hovy. 2022. EvEntS ReaLM: Event Reasoning of Entity States via Language Models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1982–1997, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics. 7.2.2, 7.6.1

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. brat: a Web-based Tool for NLP-Assisted Text Annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107, Avignon, France. Association for Computational Linguistics. 7.3.1

Jon Stevens, Anton Benz, Sebastian Reuße, and Ralf Klabunde. 2015. A Strategic Reasoning Model for Generating Alternative Answers. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 534–542, Beijing, China. Association for Computational Linguistics. 5.1, 5.2.1

Kai Sun, Dian Yu, Jianshu Chen, Dong Yu, Yejin Choi, and Claire Cardie. 2019. DREAM: A Challenge Data Set and Models for Dialogue-Based Reading Comprehension. *Transactions of the*

Association for Computational Linguistics, 7:217–231. 6.1, 6.2.1, 2

Swabha Swayamdipta, Sam Thomson, Chris Dyer, and Noah A. Smith. 2017. Frame-Semantic Parsing with Softmax-Margin Segmental RNNs and a Syntactic Scaffold. *arXiv*. 4.3.3.3

Mona Taghavi, Ahmed Patel, Nikita Schmidt, Christopher Wills, and Yiqi Tew. 2012. An Analysis of Web Proxy Logs with Query Distribution Pattern Approach for Search Engines. *Computer Standards & Interfaces*, 34(1):162–170. 4.2.2, 4.5

Jennifer R. Talevich, Stephen J. Read, David A. Walsh, Ravi Iyer, and Gurveen Chopra. 2017. Toward a Comprehensive Taxonomy of Human Motives. *PLOS ONE*, 12(2):e0172279. 2.1.4.1, 3.1, 3.2

Niket Tandon, Bhavana Dalvi, Joel Grus, Wen-tau Yih, Antoine Bosselut, and Peter Clark. 2018. Reasoning about Actions and State Changes by Injecting Commonsense Knowledge. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 57–66, Brussels, Belgium. Association for Computational Linguistics. 2.1.4.2, 7.2.2

Niket Tandon, Gerard de Melo, Fabian Suchanek, and Gerhard Weikum. 2014. WebChild: Harvesting and Organizing Commonsense Knowledge from the Web. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, pages 523–532, New York, New York, USA. ACM Press. 1, 2.1.2, 2.1.4.2

Niket Tandon, Keisuke Sakaguchi, Bhavana Dalvi, Dheeraj Rajagopal, Peter Clark, Michal Guerquin, Kyle Richardson, and Eduard Hovy. 2020. A Dataset for Tracking Entities in Open Domain Procedural Text. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 6408–6417, Online. Association for Computational Linguistics. 2.1.4.2, 7.1, 7.1, 7.3.2

Chongyang Tao, Jiazhan Feng, Rui Yan, Wei Wu, and Daxin Jiang. 2021. A Survey on Response Selection for Retrieval-based Dialogues. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 4619–4626. International Joint Conferences on Artificial Intelligence Organization. 5.4.1

Dan Tasse and Noah A Smith. 2008. SOUR CREAM : Toward Semantic Processing of Recipes MILK : Minimal Instruction Language for the Kitchen. *CMU-LTI-08-005*. 7.4.1

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, YaGuang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Vincent Zhao, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Pranesh Srinivasan, Laichee Man, Kathleen Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Soraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Diaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravi Rajakumar, Alena Butryna, Matthew Lamm, Viktoriya Kuzmina, Joe Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Aguera-Arcas, Claire Cui, Marian Croak, Ed Chi, and Quoc Le. 2022. LaMDA: Language Models for Dialog Applications. *arXiv*. 6.3.3, 6.3.3

Zhiliang Tian, Rui Yan, Lili Mou, Yiping Song, Yansong Feng, and Dongyan Zhao. 2017. How to Make Context More Useful? An Empirical Study on Context-Aware Neural Conversational Models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 231–236, Vancouver, Canada. Association for Computational Linguistics. 6.6

David R. Traum and James F. Allen. 1994. Discourse Obligations in Dialogue Processing. In *32nd Annual Meeting of the Association for Computational Linguistics*, pages 1–8, Las Cruces, New Mexico, USA. Association for Computational Linguistics. 6.5

Niall Twomey, Mikhail Fain, Andrey Ponikar, and Nadine Sarraf. 2020. Towards Multi-Language Recipe Personalisation and Recommendation. In *Proceedings of the 14th ACM Conference on Recommender Systems*, RecSys '20, pages 708–713, New York, NY, USA. Association for Computing Machinery. 7.1, 7.2.1

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008, Long Beach, California, USA. 2.2.1, 3.4.1.3, 4.3.2, 4.3.2

Ivan Vulić, Edoardo Maria Ponti, Robert Litschko, Goran Glavaš, and Anna Korhonen. 2020. Probing Pretrained Language Models for Lexical Semantics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 7222–7240, Online. Association for Computational Linguistics. 2.1.1

Sarah Theres Völkel, Daniel Buschek, Malin Eiband, Benjamin R. Cowan, and Heinrich Hussmann. 2021. Eliciting and Analysing Users' Envisioned Dialogues with Perfect Voice Assistants. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, pages 1–15, New York, NY, USA. Association for Computing Machinery. 5.2.1

Traci Walker, Paul Drew, and John Local. 2011. Responding Indirectly. *Journal of Pragmatics*, 43(9):2434–2451. 5.1

Hao Wang, Zhengdong Lu, Hang Li, and Enhong Chen. 2013. A Dataset for Research on Short-Text Conversations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 935–945, Seattle, Washington, USA. Association for Computational Linguistics. 6.2.1

Lili Wang, Chongyang Gao, Jason Wei, Weicheng Ma, Ruibo Liu, and Soroush Vosoughi. 2020a. An Empirical Survey of Unsupervised Text Representation Methods on Twitter Data. In *Proceedings of the Sixth Workshop on Noisy User-generated Text*, pages 209–214, Online. Association for Computational Linguistics. 4.5

Peifeng Wang, Nanyun Peng, Filip Ilievski, Pedro Szekely, and Xiang Ren. 2020b. Connecting the Dots: A Knowledgeable Path Generator for Commonsense Question Answering. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4129–4140, Online. Association for Computational Linguistics. 2.2.2

Shufan Wang, Laure Thompson, and Mohit Iyyer. 2021a. Phrase-BERT: Improved Phrase Embeddings from BERT with an Application to Corpus Exploration. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10837–10851, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. D.3

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021b. KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation. *Transactions of the Association for Computational Linguistics*, 9:176–194. 2.2.2

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. BLiMP: A Benchmark of Linguistic Minimal Pairs for English. In *Proceedings of the Society for Computation in Linguistics 2020*, pages 409–410, New York, New York. Association for Computational Linguistics. 2.1.1

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems 35*, New Orleans, Louisiana, USA. 8.2.1

Gerhard Weikum. 2017. What Computers Should Know, Shouldn't Know, and Shouldn't Believe. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion, pages 1559–1560, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee. 2.1.1

Peter West, Chandra Bhagavatula, Jack Hessel, Jena Hwang, Liwei Jiang, Ronan Le Bras, Ximing Lu, Sean Welleck, and Yejin Choi. 2022. Symbolic Knowledge Distillation: from General Language Models to Commonsense Models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4602–4625, Seattle, United States. Association for Computational Linguistics. 2.2.2, 6.3.2, 6.4.1, 6.5, D.1.1, D.3, D.3, D.3

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The Dialog State Tracking Challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413, Metz, France. Association for Computational Linguistics. 5.1, 5.1

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics. 4.4.2.1, B.5, D.1

Sixing Wu, Ying Li, Dawei Zhang, Yang Zhou, and Zhonghai Wu. 2020. Diverse and Informative Dialogue Generation with Context-Specific Commonsense Knowledge Awareness. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5811–5820, Online. Association for Computational Linguistics. 6.6

Jing Xu, Arthur Szlam, and Jason Weston. 2022. Beyond Goldfish Memory: Long-Term Open-Domain Conversation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 5180–5197, Dublin, Ireland. Association for Computational Linguistics. 6.1

Yoko Yamakata, Shinsuke Mori, and John Carroll. 2020. English Recipe Flow Graph Corpus. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5187–5194, Marseille, France. European Language Resources Association. 7.3.2

Zi Yang and Eric Nyberg. 2015. Leveraging Procedural Knowledge for Task-Oriented Search. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 513–522, Santiago, Chile. Association for Computing Machinery. 2.1.4.2, 7.1, 7.2.1

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546, Online. Association for Computational Linguistics. 2.2.2

Tom Young, Erik Cambria, Iti Chaturvedi, Hao Zhou, Subham Biswas, and Minlie Huang. 2018. Augmenting End-to-End Dialogue Systems With Commonsense Knowledge. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 4970–4977, New Orleans, LA, USA. AAAI Press. 6.6

Wenhao Yu, Chenguang Zhu, Zaitang Li, Zhiting Hu, Qingyun Wang, Heng Ji, and Meng Jiang. 2022. A Survey of Knowledge-Enhanced Text Generation. *ACM Computing Surveys*. 1

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 93–104, Brussels, Belgium. Association for Computational Linguistics. 5.2.2, 5.3.2

Rowan Zellers, Ari Holtzman, Matthew Peters, Roozbeh Mottaghi, Aniruddha Kembhavi, Ali Farhadi, and Yejin Choi. 2021. PIGLeT: Language Grounding Through Neuro-Symbolic Interaction in a 3D World. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 2040–2050, Online. Association for Computational Linguistics. 7.1

Hongfei Zhang, Xia Song, Chenyan Xiong, Corby Rosset, Paul N. Bennett, Nick Craswell, and Saurabh Tiwary. 2019. Generic Intent Representation in Web Search. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 65–74, New York, NY, USA. Association for Computing Machinery. 4.5

Houyu Zhang, Zhenghao Liu, Chenyan Xiong, and Zhiyuan Liu. 2020. Grounded Conversation Generation as Guided Traverses in Commonsense Knowledge Graphs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2031–2043, Online. 5.6

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 2204–2213, Melbourne, Australia. Association for Computational Linguistics. 5.2.1, 6.2.1, 3

Yi Zhang, Sujay Kumar Jauhar, Julia Kiseleva, Ryen White, and Dan Roth. 2021a. Learning to Decompose and Organize Complex Tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2726–2735, Online. Association for Computational Linguistics. 4.1, 4.5

Zhihan Zhang, Xiubo Geng, Tao Qin, Yunfang Wu, and Daxin Jiang. 2021b. Knowledge-Aware Procedural Text Understanding with Multi-Stage Training. In *Proceedings of the Web Conference 2021*, pages 3512–3523, New York, NY, USA. Association for Computing Machinery. 7.2.2

Ziqi Zhang, Philip Webster, Victoria Uren, Andrea Varga, and Fabio Ciravegna. 2012. Automatically Extracting Procedural Knowledge from Instructional Texts using Natural Language Processing. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation*, pages 520–527, Istanbul, Turkey. European Language Resources Association. 2.1.4.2, 7.3.1

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate Before Use: Improving Few-shot Performance of Language Models. In *Proceedings of the 38th International Conference on Machine Learning*, pages 12697–12706. PMLR. 6.6

Dengyong Zhou, Qiang Liu, John Platt, and Christopher Meek. 2014. Aggregating Ordinal Labels from Crowds by Minimax Conditional Entropy. In *Proceedings of the 31st International Conference on Machine Learning*, pages 262–270, Beijing, China. ACM Press. 5.3.2

Kangyan Zhou, Shrimai Prabhumoye, and Alan W Black. 2018. A Dataset for Document Grounded Conversations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 708–713, Brussels, Belgium. Association for Computational Linguistics. 6.2.1

Pei Zhou, Hyundong Cho, Pegah Jandaghi, Dong-Ho Lee, Bill Yuchen Lin, Jay Pujara, and Xiang Ren. 2022a. Reflect, Not Reflex: Inference-Based Common Ground Improves Dialogue Response Quality. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10450–10468, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics. 6.2.1, 6.3.3, 10, 6.3.3, 12, D.2

Pei Zhou, Karthik Gopalakrishnan, Behnam Hedayatnia, Seokhwan Kim, Jay Pujara, Xiang Ren, Yang Liu, and Dilek Hakkani-Tur. 2022b. Think Before You Speak: Explicitly Generating Implicit Commonsense Knowledge for Response Generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 1237–1252, Dublin, Ireland. Association for Computational Linguistics. 2.2.2, 5.6, 8.2.1

Appendix A

Supplementary Materials for Chapter 3

A.1 Implementation Details

We implemented baseline methods in Python and conducted experiments on a machine with NVIDIA GeForce GTX TITAN X GPUs.

SVM: For SVM, we used the LinearSVC implementation in scikit-learn (v0.20.2). We tested the following hyper-parameter grid on validation splits:

- Regularization: {L1, L2}
- Loss function: {Hinge loss, Squared hinge loss}
- Penalty parameter: $\{2^{-3}, 2^{-2}, 2^{-1}, 0\}$

MLP: We used the AllenNLP library (v0.8.2) (Gardner et al., 2018) with PyTorch (v1.0.1) for MLP and encoders. For all encoders, we initialized an embedding layer with 100-D pre-trained GloVe embeddings (Pennington et al., 2014) fixed the values throughout training. We also experimented with other embedding algorithms, but they yielded similar results. The details of the hyper-parameter settings of the encoders are as follows:

- **SWEM:** No tunable hyperparameters
- **CNN:**
  - Window size: 3, 4, 5
  - Number of filters: picked from {50, 100}
- **BiLSTM:**
  - Number of layers: 2
  - Number of hidden units: picked from {50, 100}

The hyper-parameters of MLP are set as follows:

- Number of hidden layers: 1
- Number of hidden units: picked from {50, 100, 200}
- Activation: picked from {ReLU, tanh}
- Dropout: 0.5[1]

We used the Adam optimizer (Kingma and Ba, 2015) with learning rate 0.001 to optimize parameters. The weight decay parameter (L2 regularization) were chosen from {0, 0.1} Finally, we tuned out-domain discounting factor λ from $\{2^{-3}, 2^{-2}, 2^{-1}\}$.

Fine-tuned Transformers: We used the Transformers library to use pre-trained Transformer models.

- Number of epochs: 10
- Batch size: picked from {32, 64}
- Learning rate: picked from {5e-5, 1e-5}
- Number of hidden units of the classifier: the same as the hidden units of each Transformer model (e.g., 768 for $\text{BERT}_{\text{base}}$)

For other hyperparameters such as a dropout rate, we folloed the setting of each PLM. We optimzied model parameters by Adam and performed early stopping based on a validation score.

GPT-3: We provided GPT-3 with a prompt through OpenAI's API and generated results. For each input sentence, we retrieved similar example sentences in terms of cosine similarity sentence embeddings so that each of the six motive labels has at least one example sentence. We computed sentence embeddings through Sentence-Transformer (Reimers and Gurevych, 2019) with MPNet (Song et al., 2020), whose dimension is 768. Table A.1 and Table A.2 show the prompt for performing motive detection (§3.4) and motive-based sentiment analysis (§3.5), respectively.

[1] The same dropout rate was applied to LSTM encoder.

Read a sentence from a restaurant review website and identify relevant basic human motives.

We have six human motive categories.
1. Self-fulfill: A motive for finding meaning in life, feeling satisfied with one's life, being entertained, and exploring a new thing.
2. Appreciating Beauty: A motive for enjoying fine design/natural beauty and being creative.
3. Social Relation: A motive for being treated well by others and belonging to a social group.
4. Health: A motive for being physically healthy.
5. Ambition & Ability: A motive for being competent/knowledgeable, keeping things in order, and being efficient.
6. Finance: A motive for saving money and getting things worth the financial cost.

Review sentence: "this place had ALL the trimmings and i mean all."
Consider the following human motives: Self-fulfill, Appreciating Beauty, Social Relation, Health, Ambition & Ability, Finance. Which motives are relevant?: Self-fulfill

Review sentence: "Over all it was a very nice romantic place ."
Consider the following human motives: Self-fulfill, Appreciating Beauty, Social Relation, Health, Ambition & Ability, Finance. Which motives are relevant?: Self-fulfill, Appreciating Beauty
...

Review sentence: "I thought this place was totally overrated."
Consider the following human motives: Self-fulfill, Appreciating Beauty, Social Relation, Health, Ambition & Ability, Finance. Which motives are relevant?: `[GPT-3 generates text starting here.]`

Table A.1: Prompt for motive detection using GPT-3

Please extract the underlying human motive(s), relevant goal(s), and whether the goals are met from the following restaurant review text.

Text: "Ess-A-Bagel is by far the best bagel in NY."
Motive: (1) A motive for finding meaning in life and feeling satisfied with one's life.
Goal: (1) Having tasty bagels.
Result: (1) Satisfied.

Text: "A beautifully designed dreamy restaurant."
Motive: (1) A motive for enjoying fine design/natural beauty and being creative.
Goal: (1) Having a meal at a restaurant with a nice interior design.
Result: (1) Satisfied.

Text: "The service was a bit slow, but everyone was cheerfully cooperative."
Motive: (1) A motive for being competent/knowledgeable, keeping things in order, and being efficient. (2) A motive for being treated well by others and belonging to a social group.
Goal: (1) Having prompt service. (1) Being treated friendly by a waiter/waitress.
Result: (1) Dissatisfied. (2) Satisfied.
. . .

Text: "The food here is rather good, but only if you like to wait for it."
Motive: (1) A motive for finding meaning in life and feeling satisfied with one's life. (2) A motive for being competent/knowledgeable, keeping things in order, and being efficient.
Goal:(1)`[GPT-3 generates text starting here.]`

Table A.2: Prompt for motive-based sentiment analsyis using GPT-3. The motive labels were generated by GPT-3 in a different prompt (Table A.1) and then converted into language expressions through templates before being used in this process.

Appendix B

Supplementary Materials for Chapter 4

B.1 Ethical Considerations

The proprietary Wunderlist data was anonymized and personally identifiable information was scrubbed. Names were replaced by random names. In addition, k-anonymization was performed on the data so that tasks that were created by fewer than five users or fewer than 100 times in total were automatically discarded. The result is an *aggregate* view of the logs, devoid of any identifiers, private information or infrequent tasks that can be correlated back to a user. The data cleaning process was approved by an internal legal review board before the data was cleared for internal use. None of the internal data has been exposed to the public throughout this study, and the example texts presented in Chapter 4 were created by the authors.

Although the proposed method is not specifically designed for English, it will require significant cost to deliver the outcome to other languages due to the dependence on English resources (knowledge bases used for training COMET and English FrameNet).

As LITE is essentially built on pre-trained language models, biases existing in the original language models can still remain in the final model (e.g., biased associations between gender and actions). We did not observe any undesired associations caused by the models in our experiments, but it may be required to monitor biases and apply debiasing techniques before deploying the model to production systems.

B.2 Templates for Autocompletion Data

We used 312 hand-crafted templates for collecting the autocompletion data. We first created templates for common nouns used in list names such as today, monday, mom, home. We then used a publicly available dataset[1] to mine list names that represent company names like "costco" and "target". We show representative examples in Table B.1.

[1]kaggle.com/peopledatalabssf/free-7-million-company-dataset/version/1

list name	Expansion
<date>	(description) **on <date>**
<shop>	(description w/ verb) **at <shop>**
	buy (description w/o verb) **at <shop>**
netflix	**watch** (description w/ verb) **on netflix**
mom	(description) **for mom**

Table B.1: Examples of the templates used to generate autocompletion data. Tokens added by the templates are denoted in **bold** (<date>=monday, tuesday, weekend, $\cdots$, <shop>=costco, wholefoods, $\cdots$, etc.)

	UIT_{F1}	AT_{Acc}	CoL_{F1}	CoT_{F1}	ID_{Acc}
None	**.860**	.900	.860	.443	.676
Input	.851	.923	.856	.448	.670
Intent ext.	.856	**.929**	**.863**	**.453**	**.714**

Table B.2: Best validation scores with different type embedding settings: no token-type embeddings, injection into the input layer, and injection into the intent extractor. The models were trained on a 500k subset of the Wunderlist dataset.

B.3 Architecture Search

We present the validation scores with different architectural choices in Table B.2 (how to inject type embeddings) and Table B.3 (number of attention heads in the intent extractor). We used $\text{BERT}_{\text{base}}$as the base text encoder and trained BERT-LITE on 500k samples of our dataset.

B.4 Additional Baseline Results

In this section, we present experimental results with the following additional baselines:

GPT-2 (Radford et al., 2018): We take the average of the last hidden states to represent an input sequence as we do for RoBERTa. Unlike BERT and RoBERTa, GPT-2 is a unidirectional encoder.

Sentence-MPNet: MPNet is a Transformer-based pre-trained language model that is reported to outperform BERT and RoBERTa (Song et al., 2020). Sentence-Transformer (Reimers and Gurevych, 2019) based on MPNet (Sentence-MPNet) is trained on 1.2B sentences from various tasks and is considered to be the best-quality general-purpose encoder (Reimers, 2021).

	UIT_{F1}	AT_{Acc}	CoL_{F1}	CoT_{F1}	ID_{Acc}
1	.857	.893	.856	.446	.650
4	**.862**	.925	.862	.450	.700
8	.860	.923	.862	.447	**.740**
12	.856	**.929**	**.863**	**.453**	.714

Table B.3: Best validation scores with different numbers of attention heads (T). The models were trained on a 500k subset of the Wunderlist dataset.

	UIT			**AT**	**CoL**			**CoT**			**ID**
	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Prec.	Rec.	F1	Acc.
GPT-2	.845	.803	.823	.908	.805	.907	.853	.501	.353	.414	.544
Sentence-MPNet	.865	.834	.849	.919	.798	.924	.856	.499	.386	.435	.654
word2vec	.856	.804	.829	.789	.780	<u>.925</u>	.846	.493	.284	.360	.628
word2vec-DA	.857	.816	.835	.805	.798	.896	.844	.506	.279	.360	.604
fastText	.856	.816	.835	.797	.780	.923	.845	.492	.282	.358	.678
BERT-LITE	<u>.871</u>	<u>.855</u>	<u>.863</u>	.932	.826	.901	.862	.511	.409	.454	.670
RoBERTa-LITE	<u>.871</u>	.847	.859	.919	.826	.905	.864	.509	.402	.449	.674
$\text{BERT}_{\text{large}}$-LITE	.863	.849	.855	<u>.936</u>	<u>.830</u>	.907	<u>.867</u>	<u>.516</u>	<u>.441</u>	<u>.475</u>	<u>.718</u>

Table B.4: Performance of additional baseline models and LITE (from Table 4.5) on downstream applications. The overall best scores are denoted in <u>underlines.</u>

word2vec and fastText: Unlike the other baseline encoders, word2vec (Mikolov et al., 2013) and fastText (Bojanowski et al., 2017) do not contextualize embeddings. We use a 300D word2vec model trained on Google News 100B and extend it by Magnitude (Patel et al., 2018) for OOV words. For fastText, we use a 300D model trained on CommonCrawl 2M. We also train a word2vec model from scratch on the same texts without special tokens as the domain-adapted version (DA).

Results (Table B.4): GPT-2 performed worse than BERT and RoBERTa. Sentence-MPNet is trained with a huge amount of additional training data but still under-performs LITE. word2vec and fastText performed similarly and outperform vanilla BERT and RoBERTa on UIT and ID. The two datasets do not provide list names as input and have fewer data points than the other datasets. Thus, we conjecture that (1) there is not enough word context that vanilla BERT and RoBERTa can leverage and (2) the dimension of embeddings is too high for a classifier to find generalizable patterns from a small amount of data.

	UIT			AT	CoLoc			CoTim			ID
	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Prec.	Rec.	F1	Acc.
BERT	.828	.840	.833	.938	.898	.942	.919	.542	.608	.563	.320
RoBERTa	.849	.859	.853	.940	.864	.952	.905	.411	.432	.384	.288

Table B.5: Result of in-dataset fine-tuning.

B.5 Implementation Details of Baselines

We implemented the baseline encoders with the following libraries.

Transformers: We used Huggingface's `transformers` library (Wolf et al., 2020) to run pre-trained Transformer models.

Sentence Transformers: We use the Sentence-BERT library (Reimers and Gurevych, 2019)[2] to run pre-trained sentence encoders. We used the following pre-trained models:
BERT: `roberta-base-nli-stsb-mean-tokens`[3]
RoBERTa: `roberta-base-nli-stsb-mean-tokens`[4]
MPNet: `all-mpnet-base-v2`[5]

B.6 Fine-tuning BERT and RoBERTa

We present the performance of BERT and RoBERTa fine-tuned on downstream datasets. Note that our main goal is to train a general-purpose encoder that does not need to be re-trained for each downstream task as we describe in §B.5. We aim to answer the following two hypothetical questions.

Q1 (In-dataset fine-tuning): How well could BERT and RoBERTa perform if they were fine-tuned on the target dataset? This approach is commonly practiced for *task-specific* representations (Devlin et al., 2019).

Q2 (Cross-dataset fine-tuning): How well could BERT and RoBERTa perform on the target dataset if they were fine-tuned on another dataset? (Were the fine-tuned encoders generalizable to multiple to-do datasets?)

[2] www.sbert.net/
[3] huggingface.co/sentence-transformers/bert-base-nli-stsb-mean-tokens
[4] huggingface.co/sentence-transformers/roberta-base-nli-stsb-mean-tokens
[5] huggingface.co/sentence-transformers/all-mpnet-base-v2

	→Test				
↓Train	UIT	AT	CoLoc	CoTim	ID
UIT	**.833**	.638	.793	.394	.110
AT	.604	**.938**	.801	.405	.134
CoLoc	.497	.560	**.919**	.394	.116
CoTim	.325	.512	.814	**.563**	.148
ID	.362	.541	.782	.394	**.320**
LITE	.863	.932	.862	.454	.670

(a) $\text{BERT}_{\text{base}}$

	→Test				
↓Train	UIT	AT	CoLoc	CoTim	ID
UIT	**.853**	.645	.793	.373	.112
AT	.536	**.940**	.798	.372	.110
CoLoc	.412	.570	**.905**	.328	.104
CoTim	.276	.513	.823	**.384**	.106
ID	.359	.509	.796	.360	**.288**
LITE	.859	.919	.864	.449	.674

(b) $\text{RoBERTa}_{\text{base}}$

Table B.6: Test performance of fine-tuned BERT and RoBERTa. **The diagonal cells** show the performance of the models trained with the in-dataset fine-tuning setting.

Setup: We fine-tune and evaluate $\text{BERT}_{\text{base}}$ and $\text{RoBERTa}_{\text{base}}$ models on the 20 random splits used in the main experiments. We follow Devlin et al. (2019) and add a linear classification layer that takes in the final hidden state of the first token (`[CLS]` token). For fine-tuning, the encoder and classifier are trained to optimize a binary cross entropy loss (UIT, CoLoc, and CoTim) or a cross entropy loss (ID and AT). We use the same optimization configurations described in §B.5. We continue training for 5 epochs and take the checkpoint that achieves the best validation score. For the cross-dataset experiment, we initialize the encoder with the fine-tuned parameters and freeze it during training. We use the same optimization settings except that we set a learning rate to 0.001.

A1 (Table B.5): As expected, the fine-tuned models perform better than LITE on several datasets (AT, CoLoc, and CoTim with BERT, and AT with RoBERTa). When the main goal is to build task-specific representations, and there is a sufficiently large training dataset, task-specific fine-tuning will be a better solution than LITE. However, the result shows the fine-tuned models do not *al-*

ways outperform LITE. We conjecture that for datasets without a sufficient number of training instances like UIT and AT, a fine-tuning strategy is not very effective.

A2 (Table B.6): Performance consistently drops when the encoders are trained on another dataset, and all the scores are far below those of BERT/RoBERTa-LITE. This result indicates that LITE is more effective for training generalizable encoders than fine-tuning on a single dataset.

Appendix C

Supplementary Materials for Chapter 5

C.1 Manual Annotation

We recruited non-expert crowd workers in Mturk in annotation steps (2-5). In all steps, crowd workers were required to meet the following qualification requirements: (i) Their number of tasks approved $\geq$ 5k, (ii) the task approval rate $\geq$ 99%, (iii) their location is the US, and (iv) they answer an exercise question correctly.

Two of the authors were involved in the annotation steps (1), (4), (5), and (8). They are ESL with a degree in computer science from a school in the US (one holds a master's degree, and the other holds a Ph.D.). They all have backgrounds in NLP/CL research.

C.2 Distractor Selection

This section presents the technical details of the distractor selection method (Step 7). Below, tunable parameters like thresholds on scores and the number of iterations were empirically selected based on several pilot runs.

C.2.1 Response Selection

Our method selects distractor responses from all the responses in the dataset in two steps: We first create an initial dataset by a light-weight method (Algorithm 1) and then perform adversarial filtering (Algorithm 2).

First Step (Algorithm 1)

The objective of the first step is to avoid including false-negative responses (Lines 3-6). We discard responses that are too similar to r_1 in terms of the overlap coefficient of content words (noun,

Algorithm 1 Create an initial dataset by light-weight filtering

Input: m, Dataset $\mathcal{D} = \{(u^{(i)}, g^{(i)}, r_1^{(i)}, S_1^{(i)})\}_{i=1,\cdots,N}$, ▷ $N :=$ num. of examples in the dataset.
Output: $\mathcal{D}' = \{(u^{(i)}, g^{(i)}, R^{(i)}, S_1^{(i)})\}_{i=1,\cdots,N}$ ▷ $R^{(i)} := \{r_1^{(i)}, \cdots, r_m^{(i)}\}$ ▷ Initial dataset

1: **function** InitDataset($m, \mathcal{D}$)
2: $\mathcal{D}' \leftarrow \varnothing$
3: **for** $i : 1..N$ **do**
4: $\mathcal{P} \leftarrow \{r_1^{(j)}\}_{j=i,\cdots,i-1,i+1,\cdots,N}$ ▷ All the responses in $\mathcal{D}$ but $r_1^{(i)}$
5: `# (1) Remove too similar responses`
6: **for** $j : 1..N$ **do**
7: **if** i=j **then**
8: **continue**
9: **if** Overlap($u^{(i)}, u^{(j)}$) ≥ 0.75 **or** Overlap($g^{(i)}, g^{(j)}$) ≥ 0.75 **or** EmbSim($u_1^{(i)}, r_1^{(j)}$) ≥ 0.5) **then**
10: Remove $r_1^{(j)}$ from $\mathcal{P}$
11: `# (2) Pick` $m-1$ `similar responses`
12: $R^{(i)} \leftarrow \{r_1^{(i)}\}$
13: **for** $j : 1..m-1$ **do**
14: Sample $r \in \mathcal{P}$
15: Add r to $R^{(i)}$
16: `# (3) Remove similar responses from the pool`
17: **for all** $r' \in \mathcal{P}$ **do**
18: **if** Overlap(r, r') ≥ 0.75 **then**
19: Remove r' from $\mathcal{P}$
20: Add $(u^{(i)}, g^{(i)}, R^{(i)}, S_1^{(i)})$ to $\mathcal{D}'$
21: return $\mathcal{D}'$

verb, adjective, and adverb).

$$\text{Overlap}(x, y) = \frac{|\text{CW}(x) \cap \text{CW}(y)|}{\min(|\text{CW}(x)|, |\text{CW}(y)|)}, \tag{C.1}$$

where $\text{CW}(x)$ is a set of content words in x. We set the threshold of overlap coefficient to 0.75. We use the same constraint on their goal texts. We also measure their closeness by the cosine similarity of their sentence embeddings (denoted as EmbSim) and discard candidates whose similarity is 0.5 or higher. We then sample $m-1$ responses from this filtered response pool one by one (Lines 11-15). To diversify response options, we remove similar responses to the picked one from the pool based on overlap coefficient (Line 16-19).

Algorithm 2 Adversarial filtering (AF) for R

Input: m, Dataset $\mathcal{D} = \{(u^{(i)}, g^{(i)}, r_1^{(i)}, S_1^{(i)})\}_{i=1,\cdots,N}$, ▷ $N :=$ number of examples in the dataset.

Output: $\mathcal{D}' = \{(u^{(i)}, g^{(i)}, R^{(i)}, S_1^{(i)})\}_{i=1,\cdots,N}$ ▷ $R^{(i)} := \{r_1^{(i)}, \cdots, r_m^{(i)}\}$

1: $\mathcal{P} \leftarrow \{(r_0)_i\}$ ▷ All responses in $\mathcal{D}$

2: `(1) Create an initial dataset` D_0

3: $\mathcal{D}_0 \leftarrow$ InitDataset$(m, \mathcal{D})$ ▷ See Algorithm 1

4: `(2) Run AF for` J `rounds`

5: **for** $j : 1..J$ **do** ▷ We set $J = 3$

6:     Split $\mathcal{D}_{j-1}$ into K-folds $\{(\mathcal{T}^k, \mathcal{V}^k)\}_{k=1,\cdots,K}$ ▷ We set $K = 10$

7:     **for** $k : 1..K$ **do**

8:         Train a binary logistic regression classifier $\mathcal{M}$ on $\mathcal{T}^k$

9:         **for all** $(u, g, R, S_1) \in \mathcal{V}^k$ **do**

10:             **for all** $r \in R \setminus \{r_1\}$ **do**

11:                 (f: $\mathcal{M}$'s score function)

12:                 **if** $f(r) + \gamma \leq f(r_1)$ **then** ▷ γ is a margin, which we set to 0.05.

13:                     Remove r from R

14:                     Pick r' s.t. $f(r') - \gamma > f(r_1)$

15:                     Add r' to R

16:                     Update $\mathcal{V}^k$ with the new R

17:     $\mathcal{D}_j \leftarrow \bigcup_{k=1}^{K} \mathcal{V}_k$

18: $\mathcal{D}' \leftarrow \mathcal{D}_K$ ▷ End

Second Step (Algorithm 2)

We then perform $J = 3$ rounds of adversarial filtering. Our method is a slightly modified version of the algorithm used by Bhagavatula et al. (2020). In each round, we split the dataset into $K = 10$ folds (Line 6), and for each split, we train a binary logistic regression classifier that takes sentence embeddings of u, S_1, and a response candidate $r \in R$ (Line 8). We pre-compute their sentence embeddings with the pre-trained SentenceTransformers (Reimers and Gurevych, 2019) with MPNet (Song et al., 2020). Once the classifier is trained, we score response candidates in each example and identify distractors whose scores are lower than that of the reference response r_1 plus a margin $\gamma = 0.05$. We replace these *easy* distractors with more confusing ones (Line 14-16). In this way, we repeatedly update the dataset (Line 17) and output the final result (Line 18).

C.2.2 Situation Selection

Next, we update S_1, which only contains relevant information to u and r_1, to include l statements in total such that some of them are associated with distractors or not directly related to the conversation. Otherwise, reference responses can be easily identified by superficial clues. Having irrelevant situation statements is also for simulating real use cases, where a conversational system has access to a wide range of sensory information or external APIs, but most of them are unimportant for addressing a user's request.

It is required that (a) additional situation statements do not disqualify the reference response, and (b) they do not contradict others. To this end, we again use sentence embeddings with keyword-based heuristics. We first combine the statements associated with distractor responses and create a pool of candidates. Here, we drop statements that are similar to the response candidates in terms of the overlap coefficient of content words with a threshold of 0.75. We also used manually defined keywords to discard situation statements that tend to contradict others (e.g., the time is midnight, the user is injured, etc.). We then iterate over six categories and pick situation statements from the pool one by one. We score statement s of category c using the function below:

$$f(s; R, S') = \max_{r \in R} \mathrm{EmbSim}(s, r) - \max_{s' \in S'_c} \mathrm{EmbSim}(s, s') - \frac{1}{2} \max_{s' \in S'_{\mathcal{C} \setminus \{c\}}} \mathrm{EmbSim}(s, s'), \tag{C.2}$$

where S' is the current situation statements, $S'_c \subset S'$ represents the statements in S of category c, and $\mathcal{C}$ denotes a set of situation categories. We pick distractor statements until we exhaust all the candidates in the pool or the maximum score does not reach 0. We then draw statements from the entire dataset in the same way until $|S|$ reaches $l = 12$. For time, date, behavior, and location categories, we pick zero or one statement as those categories are not likely to have more than one value.

Appendix D

Supplementary Materials for Chapter 6

D.1 Implementation Details

Throughout the experiments, we used the models implemented in Python 3.8 with PyTorch v1.13.1 (Paszke et al., 2019) and the Transformers library (Wolf et al., 2020). We preprocessed texts by spaCy[1] (*en-core-web-sm* model) and NLTK[2]. Our tools and resources do not involve license restrictions on the use for research purposes. We will release our code and pre-trained model parameters.

D.1.1 Situation Generation

We employed COMET$^{\text{DIS}}_{\text{TIL}}$ (West et al., 2022), which is based on GPT2-XL (Radford et al., 2018) (1.5B parameters), for situation generation. COMET$^{\text{DIS}}_{\text{TIL}}$is trained on a large-scale collection of event-centric common-sense triples, ATOMIC$^{20}_{20}$, which may serve as a useful inductive bias for situation generation. The goal of situation generation is to generate statements of observable situational information for a given conversation. Reference responses were added to the input along with an previous utterance for the training and validation data. However, to prevent introducing clues about the reference result, responses were not included in generating situational statements for the test instances in CICERO and ConvAI2.

We fine-tuned a model on the SUGAR dataset using two different input settings. The first setting concatenats a previous utterance, a response, and a reference situational information into one sequence. The second setting concatenated a previous utterance and a reference situational information into one sequence for generating situational statements on test instances, for the aforementioned reason. In both cases, each text was headed by special symbols indicating the text type: <utterance> for an utterance, <response> for a response, and <situation category> for

[1] https://spacy.io/
[2] https://www.nltk.org/

Max iterations	5,000
Batch size	16
Gradient accumulation	16
Optimizer	Adam
Weight decay	0.01
Gradient clipping	max norm of 1.0
Learning rate (LR)	0.000005
LR warmup (linear)	300 steps
Dropout	0.1

Table D.1: Hyperparameters for the COMET$^{\text{DIS}}_{\text{TIL}}$-based context generator

a situational statement. The <situation category> symbol is one of date, time, location, behavior, environment, and possession. The model was optimized to minimize a cross-entropy loss with a label smoothing factor of 0.1 for the tokens in the situational information. Table D.1 shows the hyperparameters for the training step. We evaluated the average token-level perplexity on the validation split every 100 steps and terminated training if the value did not improve for 5 consecutive validations. The training process took approximately four hours on an NVIDIA TITAN RTX GPU with the DeepSpeed (Rasley et al., 2020) library.

To generate situations on the CICERO and ConvAI2 datasets, we concatenated a conversation history and a response (for the training and validation splits) followed by one of the situation categories as input. We generated three candidates for each category using nucleus sampling ($p = 0.9$). As the model was trained on SUGAR, which only contains single-turn conversations, we observed that feeding many previous utterances impaired the generation quality. Therefore, we limited the number of previous utterances in the input to 3. Finally, for quality control, one of the authors manually checked the test samples from CICERO and ConvAI2 (25 for each) and corrected situational statements when required (e.g., conflicting facts). The reference responses were hidden during the manual verification to avoid bias. This manual verification process ensures the quality of the test dataset in order to minimize the confusion of annotators in the following manual evaluation of responses.

D.1.2 Response Generation

BlenderBot2: We used the pre-trained BlenderBot2 model with 400M parameters[3] with web search turned off. We concatenated persona statements (for ConvAI2), context descriptions, and a conversation history with newline symbols
n. We denoted text types by dedicated prefixes as practiced in pre-training of BlenderBot2, namely, a persona statement is headed by text `your persona:`, a context description is

[3] https://parl.ai/projects/blenderbot2/

Max epochs	10
Batch size	16
Optimizer	Adam
Weight decay	None
Gradient clipping	max norm of 1.0
Learning rate (LR)	0.00001
LR warmup (linear)	100 steps
LR decay (based on validation)	coef. of 0.5
Dropout	0.1

Table D.2: Hyperparameters for BlenderBot2

headed by `context:`, and each utterance in a conversation history is headed by either `<speaker1>` or `<speaker2` which corresponds to the speaker of the utterance. a We followed the original configuration of hyperparameters (Table D.2). We evaluated a model on the validation set every 1/4 epoch and terminated training if the average token-level perplexity score on the validation set did not improve five times in a row. In our experiments, training finished at around two epochs, taking about 4 hours on one NVIDIA TITAN RTX. For generation, we used nucleus sampling with $p = 0.9$.

GPT-3: We generated responses with GPT-3 with a few-shot learning mannar. We picked four high-quality examples from the training and validation splits for each dataset and provided them with a short instruction in a prompt. Table D.3 shows an example of our prompt. We generated responses with top-p=0.9 and temperature=0.7.

ChatGPT: We used the same prompt as that of GPT-3 for generating responses with ChatGPT through OpenAI's interactive demo page [4]. As ChatGPT's technical details are not fully transparent, we only used ChatGPT for performing a few case studies like the example in Table 6.1 in this study.

D.2 Crowdsourced Evaluation

In the first experiment we rectuited crowd workers on Amazon Mechanical Turk. We set the following qualification requirements for filtering workers: (1) at least 1,000 HITs are approved so far, (2) $\geq 99\%$ approval rate, (iii) living in US. Each HIT involves judgment of three response candidates. Workes were paid $\$0.30$ for each HIT. We used the guidelines and interface developed by (Zhou et al., 2022a). Figure D.1 shows the annotation guidelines. To monitor the performance

[4] https://chat.openai.com/

Instructions

1. You will see **one incomplete conversation** between two people.
2. **You are given three responses** that automatic conversation systems generated for each conversation. Systems are supposed to be friendly and cooperative to a conversation partner.
3. Your task is **to evaluate the responses in four dimensions:** (1) Is the response grammatically correct? (2) Does the response make sense? (3) Is the response context-specific? (4) Is the response interesting?

Before starting the task, please read carefully the criteria below.

(a) General instructions

Treat each of the three dimensions below as separate, independent measure. A response can be context-specific/interesting but factually wrong.

1. Does the response make sense?

- Use your common-sense here. Is the response completely reasonable in the given context?
- If anything seems **confusing, out of context, or factually wrong,** then rate it as **"No (Does not make sense.)"**
- If the response seems wrong but you're not sure, choose "Does not make sense."

2. Is the response specific?

- Is the response context-specific? **Check if the response can be used in many different contexts of different topics.**
  1. If SpeakerA says "I love tennis" and SpeakerB replies "That's nice", then B's response is **"Not specific."**This response can apprear in many different contexts of topics other than tennis.
  2. If SpeakerB replies "Mee too, I can't get enough of Roger Federer!", then mark this response as **"Specific."** This response is closely related to the context and is unlikely to appear in other contexts, for example, when people are talking about baseball.
- You're not sure, choose "Does not make sense."

3. Is the response interesting?

- Choose "Interesting" if the response is likely to **catch someone's attention or arouse curiosity,** or the response is **insightful, unexpected, or witty.**
- If the response is monotonous and predictable, or if you're unsure, then pick "Not interesting."

(b) Evaluation criteria

Figure D.1: Evaluation guidelines. Note that the third criterion was removed from the final results due to the low inter-annotator agreement.

of workers, we embedded one dummy response in each HIT. We created the dummy responses to be a clearly bad response.

Initially, we followed Zhou et al. (2022a) and also evaluated if the responses are interesting or not, but we found the inter-annotator agreement of this criterion is high enough to draw a reliable conclusion (Fleiss' kappa of 0.2). Therefore, we removed this criterion from our final results.

D.2.1 Second Experiment

In the second experiment, we recruited workers who met the following qualifications: (1) The Mechanical Turk *Masters Qualification* has been granted by the platform, (2) Number of HITs approved $\geq$ 1,000, (3) HIT approval rate $\geq$ 95%, (4) Location is US. We increased a reward based on the numbder of distractors. ($0.35 for 10 distractors and $0.40 for 15 distractors.)

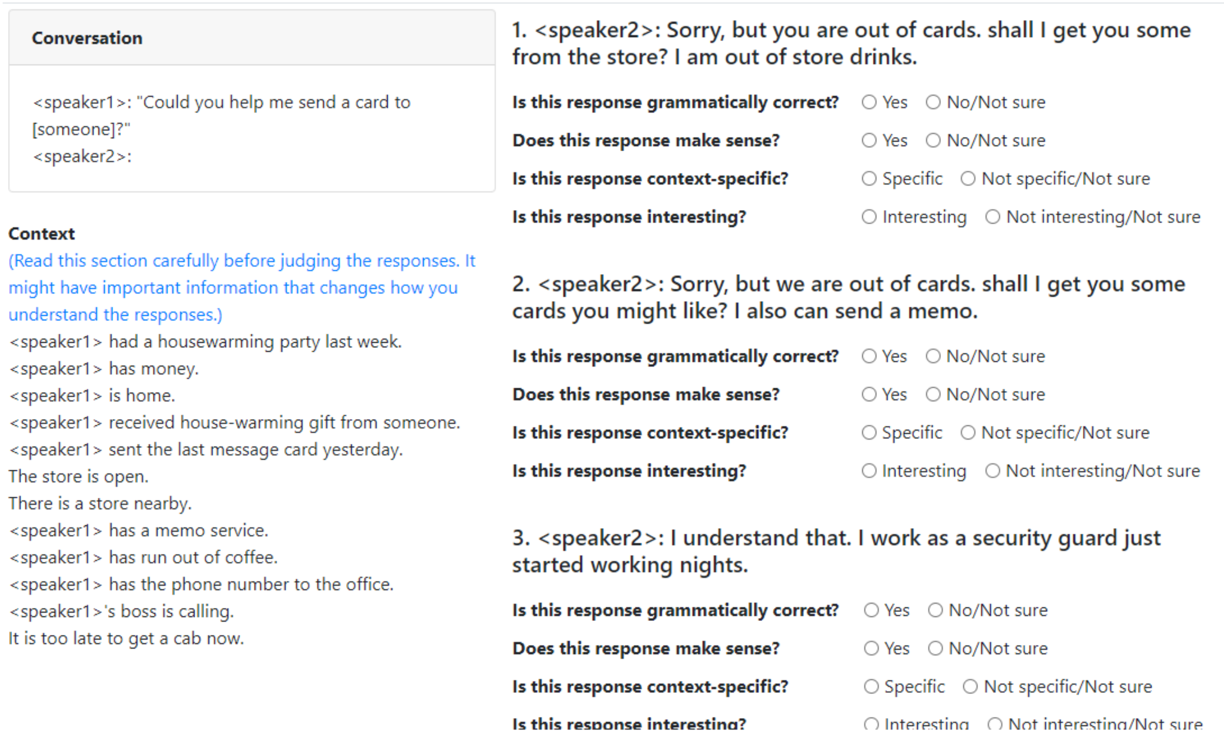


Figure D.2: Evaluation interface.

D.3 Knowledge Generation for KRel

We used the pre-trained COMET$^{\text{DIS}}_{\text{TIL}}$-high model released by West et al. (2022).[5]

Input format: As defined by West et al., the model was provided a prompt in the following format for a target input text and relation label (either xNeed, xIntent, or xWant): `[CLS] <head> (input text) </head> <relation> (relation label) </relation> [GEN] to`, where `[CLS]`, `[GEN]`, `<head></head>`, and `<relation></relation>` are special tokens to inform the model of the input format.

Generation: We generated edges up to five hops through the transformer library's generation API with the arguments presented in Table D.4. We used a pre-trained Transformer-based classifier [6] to filter out ungrammatical outputs.

[5] https://github.com/peterwestai2/symbolic-knowledge-distillation

[6] The model is available at https://huggingface.co/madhurjindal/autonlp-Gibberish-Detector-492513457. We set the minimum score for the "clean" class to 0.3.

Normalization: To form a densely connected graph, we grouped synonymous expressions by heuristics. We performed the following post-processing at each round of the generation step.

- Lemmatization: We used spaCy to lemmatize generated expressions into verb or noun phrases without articles and modifiers (e.g., PersonX reads interesting books → read book).
- Clustering of sentence embeddings: We encoded the lemmatized phrases into 768-D vectors by PhraseBERT (Wang et al., 2021a) and clustered them using the Fast Clustering algorithm implemented in the SentenceTransformers library.
- Representative forms: Finally, for each cluster, we picked the most frequent expression as the cluster's representative form.

Edge scoring: The resulting graph contains some edges that represent implausible or incorrect knowledge. We used the pre-trained classification model relaesed by West et al. (2022) to estimate the confidence score of each edge, which was fed to the PageRank algorithm to discount the influence of noisy edges.

Two people are having a conversation in the following examples. Both people are helpful and friendly.

Example 1
Context:
1. Today is Monday.
2. It is afternoon now.
3. <speaker1> and <speaker2> are at school.
4. <speaker2> is studying English.
5. <speaker1> has a phone.
6. <speaker1> has alrady finished lunch.
7. <speaker2> has an English book with her.
8. The nearby restaurant is open.
9. Final exams are coming soon.
10. <speaker2> has not had lunch yet.
Conversation:
<speaker1>: Hi, Lily. Where were you at lunchtime? I was looking for you in the dining hall.
<speaker2>: Oh, sorry, I missed you . My English class ran late again.
<speaker1>: That's been happening quite often recently . Maybe it's because the final exams are coming up.

. . .

Example 5
Context:
1. Today is Sunday.
2. It is daytime now.
3. <speaker9> and <speaker10> are in the hotel.
4. <speaker10> is working at the hotel.
5. <speaker9> has a car.
6. <speaker9> is carrying a suitcase.
7. <speaker10> has a computer.
8. The door is closed.
9. <speaker9>'s keys are on the desk.
10. It is raining outside.
Conversation:
<speaker9>: Hello. I'm leaving. Here is my key.
<speaker10>:

Table D.3: Example of the prompt for GPT-3 and ChatGPT. The examples are taken from CICERO.

do_sample	False
max_new_tokens	10
num_return_sequences	3 or 5 (=K)
num_beams	num_generate $\times 3$
top_p	0.9
top_k	40
temperature	1.0
repetition_penalty	1.0
no_repeat_ngram_size	3
bad_words_ids	See Table D.5

Table D.4: Hyperparameters for knowledge generation

"I", "me", "you", "your", "we", "us", "they", "them",
"this", "that", "these", "those", "it",
"some", "any", "all", "few", "many", "most",
"PersonX", "PersonY"

Table D.5: Disallowed words for knowledge generation

Appendix E

Supplementary Materials for Chapter 7

E.1 Crowdsourced Annotation

This section provides details of crowdsourced annotation tasks to identify frame types and their arguments.

E.1.1 Step 1: Frame Identification

The first step is to identify cooking actions and their types. In this step, we considered 11 semantic frames, including the six frames that were later grouped into OtherChange. To ensure accurate annotations, we first conducted a trial round with 100 sentences, where we solicited 10 workers for each sentence and updated annotation instructions and interface based on the results. Figure E.1 displays the annotation interface.

We recruited five workers per sentence for the remaining sentences. To estimate worker accuracy, we added one test question to every nine sentences, and workers who failed to answer the test question correctly were filtered out. We aggregated the collected responses using MACE (Hovy et al., 2013) to obtain the final frame type labels. In our manual evaluation of 100 sentences, MACE outperformed majority voting with an F1 score of 0.1.

Quality: We achieved satisfactory inter-annotator agreement, with macro average Krippendorff's α (Krippendorff, 2006) and Fleiss' κ (Fleiss, 1971) of 0.648 and 0.657, respectively (Table E.1).

E.1.2 Step 2: Argument Identification

In this step, we identified arguments of a frame that were detected in Step 1. To simplify our annotation work, we used natural language questions, such as "A cook moves ___ somewhere.

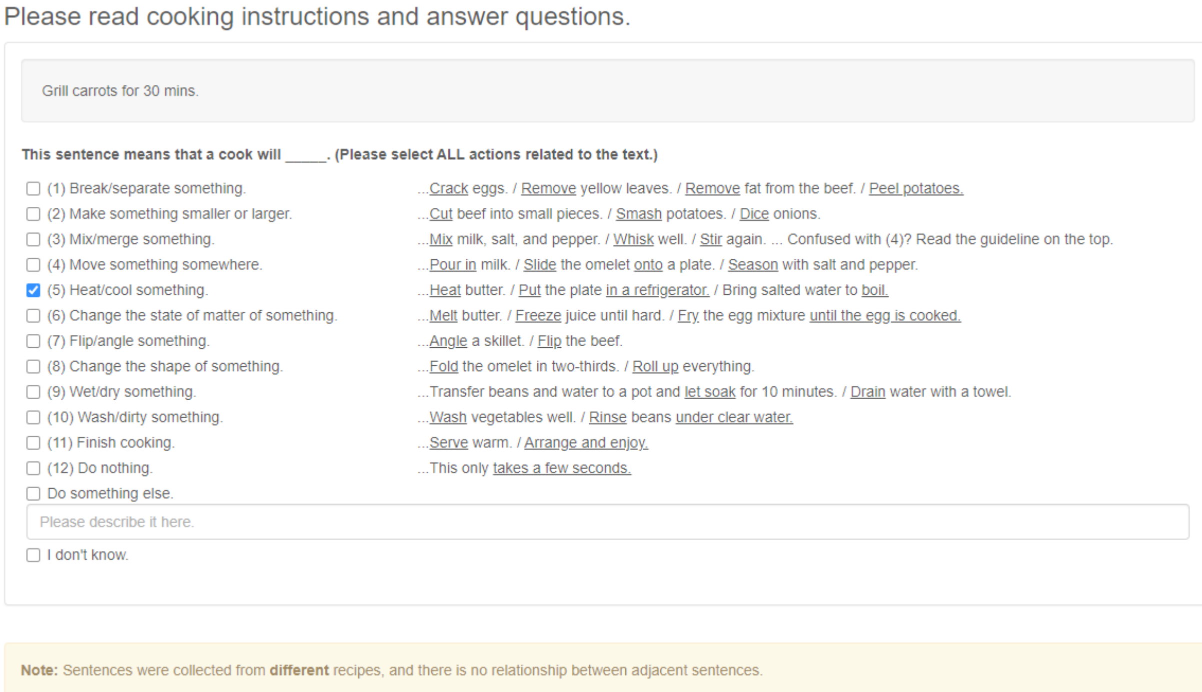


Figure E.1: Annotation interface used in Step 1. Crowd workers pick one or more actions that a cook performs in a given cooking instruction.

What's ___?" inspired by work on question-answering-style semantic role annotations (He et al., 2015). Figure E.2 provides an example of the annotation questions.

The main challenge in this step was missing or inferred arguments. We instructed annotators to identify an action where the output of the previous action is involved ("Result of Step 1" in Figure E.2). If arguments were not present in the text, annotators were asked to write the most likely participants in a text field (e.g., seasonings for "Season salmon."). We conducted several rounds of internal annotation practice to develop annotation guidelines and collect stereotypical and challenging cases. We provided an exercise question and several examples in the annotation instructions to train the crowd workers.

Quality To evaluate the annotations made by crowd workers, one of the authors annotated a subset of the annotation targets (287 frames) to create reference annotations. The F1 score between the crowdsourced annotations and the reference annotations was 0.880 and 0.833 for input and location slots, respectively, indicating substantial agreement. One exception was the input slots of TemperatureChange, where the precision was 0.449. Major disagreements were found when a location entity was heated or cooled, as in "Bake the crust in *an oven*." Some crowd workers tagged an oven as both an input and location entity, while the reference annotation considered the oven to be only a location.

Frame type	Responses	α	κ
Divide	290	.535	.548
SizeChange	1233	.894	.894
Merge	1479	.649	.655
LocationChange	3001	.640	.639
TemperatureChange	1805	.820	.822
MatterStateChange	442	.399	.401
AngleChange	20	.323	.349
ShapeChange	244	.579	.601
WetnessChange	161	.718	.718
CleanlinessChange	84	.829	.852
Serve	466	.737	.741
Macro avg.		.648	.657
Micro avg.		.751	.753

Table E.1: Inter-annotator agreements (Krippendorff's α and Fleiss' κ) of responses against 1,263 sentences in step 1. Note that, for Fleiss' κ, we randomly sampled 5 workers per sentence in the first 90 sentences, for which we recruited 10 workers. The values report the average over 100 trials.

Simplification of Semantic Frames: We considered 11 semantic frames but found it difficult to identify some frames (e.g., AngleChange) with a satisfactory level of inter-annotator agreement in Step 1. We also found some frames were not frequent or indispensable (e.g., ShapeChange is often concerned with a stylistic feature of food) in cooking recipes. Thus, we decided to focus primarily on the four action types that are crucial to completing cooking and combined 6 frame types into a category called OtherChange as we described in Section 7.4.

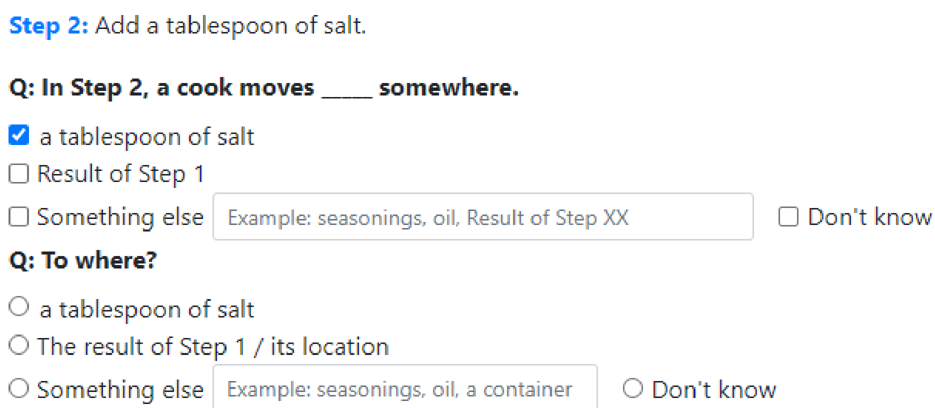


Figure E.2: Annotation interface used in Step 2. For each frame identified in Step 1, another set of crowd workers annotated participants and locations (if exist).