Unsupervised Learning and Modeling of Knowledge and Intent for Spoken Dialogue Systems

Yun-Nung (Vivian) Chen

CMU-LTI-15-018

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

Thesis Committee:

Dr. Alexander I. Rudnicky (chair), Carnegie Mellon University
Dr. Anatole Gershman (co-chair), Carnegie Mellon University
Dr. Alan W Black, Carnegie Mellon University
Dr. Dilek Hakkani-Tür, Microsoft Research

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

To my beloved parents and family,

Acknowledgements

> *If you have someone in your life that you are grateful for - someone to whom you want to write another heartfelt, slanted, misspelled thank you note - do it. Tell them they made you feel loved and supported. That they made you feel like you belonged somewhere and that you were not a freak. Tell them all of that. Tell them today.*
>
> Lisa Jakub, *Canadian-American writer and actress*

There is a long list of exceptional people to whom I wish to express my gratitude, who have supported and assisted me during my Ph.D. journey. I want to start with my excellent thesis advisor, Prof. Alexander I. Rudnicky, who provided me freedom to investigate and explore this field, guided me future directions, and not only appreciated but supported my ideas. Without Alex I might not be qualified as a good researcher. I also thank all committee members – Prof. Anatole Gershman, Prof. Alan W Black, and Dr. Dilek Hakkani-Tür. Thanks to their diverse areas of expertise and insightful advice, I was able to apply my experiments to real-world data, which makes my research work more valuable and more practical.

During my stay at CMU, I was very fortunate to work on different projects and collaborate with several faculty, including Prof. Jack Mostow, Prof. Florian Metze, Dr. Kai-Min Chang, Dr. Sunjing Lee, etc. As a student at CMU, I had chances to work and collaborate with people from the same group: Ming Sun, Matthew Marge, Aasish Pappu, Seshadri Sridharan, and Justin Chiu, and different amazing colleagues in LTI: William Y. Wang, Ting-Hao (Kenneth) Huang, Lingpeng Kong, Swabha Swayamdipta, Sujay Kumar Jauhar, Sunayana Sitaram, Sukhada Palkar, Alok Parlikar. I was very happy to know and interact with a lot of great people from CMU LTI: Shoou-I Yu, Yanchuan Sim, Miaomiao Wen, Yi-Chia Wang, Troy Hua, Duo Ding, Zi Yang, Chenyan Xiong, Diyi Yang, Di Wang, Hyeju Jang, Yanbo Xu, Wei Chen, Yajie Miao, Ran Zhou, Chu-Cheng Lin, Yu-Hsin Kuo, Sz-Rung Shiang, Po Yao Huang, Ting-Yao Hu, Han Lu, Yiu-Chang Lin, Wei-Cheng Chang, Joseph Chang, and Zhou Yu.

I am thankful to my roommates and great friends whom I met in Pittsburgh: Yu-Ying Huang, Jackie Yang, Yi-Tsen Pan, Pei-Hsuan Lee, Yin-Chen Chang, Wan-Ru Yu, Hsien-Tang Kao,

Ching-Heng Lu, Ting-Hao Chen, Chao-Lien Chen, Yu-Ting Lai, Yun-Hsuan Chu, Shannen Liu, Hung-Jing Huang, Kuang-Ching Cheng, Wei-Chun Lin, Po-Wei Chou, Chun-Liang Li, etc. Also, I am grateful that my great undergraduate fellow, Kerry Shih-Ping Chang, also studies at CMU, and it makes me be able to chat about our undergraduate life. Thanks to my best friend, Yu-Ying Lee, who stayed Pittsburgh with me for a awesome summer.

During the years of my Ph.D., I have benefited from conferences, workshops, and internships. I am thankful to have opportunities of discussing and interacting with a lot of smart people, such as Gokhan Tur, Asli Celikyilmaz, Andreas Stolcke, Geoffrey Zweig, Larry Heck, Jason Williams, Dan Bohus, Malcolm Slaney, Omer Levy, Yun-Cheng Ju, Li Deng, Xiaodong He, Wan-Tau Yih, Xiang Li, Qi Li, Pei-Hao Su, Tsung-Hsien Wen, David Vandyke, Dipanjan Das, Matthew Henderson, etc., who helped me find my research direction during my Ph.D. life.

I also want to thank my undergraduate fellows from NTU CSIE and colleagues from NTU Speech Lab. I started my research journey there and found what I am really interested in. Everyone helped me a lot, especially my master advisor – Prof. Lin-Shan Lee, who brought me into the area and motivated me to pursue my doctoral degree in this field.

In addition, I want to thank Luis Marujo for his beautiful LaTeXtemplate sharing, Jung-Yu Lin for her professional English revision, and my friends for their annotation work. I am so lucky to have all of your assistance, which speeds up my research and make it much better and more beautiful.

Last but not least to my dearest Mom and Dad, to my boyfriend and also my husband, Che-An Lu, to my big family, thank all of you for your unconditional love and support. Without all of you none of my success would be possible.

Yun-Nung (Vivian) Chen, Pittsburgh

Abstract

Various smart devices (smartphone, smart-TV, in-car navigating system, etc.) are incorporating spoken language interfaces, as known as spoken dialogue systems (SDS), to help users finish tasks more efficiently. The key role in a successful SDS is a spoken language understanding (SLU) component; in order to capture language variation from dialogue participants, the SLU component must create a mapping between natural language inputs and semantic representations that correspond to users' intentions.

The semantic representation must include "concepts" and a "structure": concepts are domain-specific topics, and the structure describes relations between concepts and conveys intention. Most of knowledge-based approaches originated from the field of artificial intelligence (AI). These methods leveraged deep semantics and relied heavily on rules and symbolic interpretations, which mapped sentences into logical forms: a context-independent representation of a sentence covering its predicates and arguments. However, most prior work focused on learning a mapping between utterances and semantic representations, where such organized concepts still remain predefined. The need of predefined structures and annotated semantic concepts results in extremely high cost and poor scalability in system development. Thus, current technology usually limits conversational interactions to a few narrow predefined domains/topics. Because domains used in various devices are increasing, to fill the gap, this dissertation focuses on improving generalization and scalability of building SDSs with little human effort.

In order to achieve the goal, two questions need to be addressed: 1) Given unlabeled conversations, how can a system automatically induce and organize the domain-specific concepts? 2) With the automatically acquired knowledge, how can a system understand user utterances and intents? To tackle above problems, we propose to acquire domain knowledge that captures human's salient semantics, intents, and behaviors. Then based on the acquired knowledge, we build an SLU component to understand users.

The dissertation focuses on several important aspects for above two problems: *Ontology Induction*, *Structure Learning*, *Surface Form Derivation*, *Semantic Decoding*, and *Intent Prediction*. To solve the first problem about automating knowledge learning, ontology induction extracts domain-specific concepts, and then structure learning infers a meaningful organization of these concepts for SDS design. With the structured ontology, surface form derivation

learns natural language variation to enrich its understanding cues. For the second problem about how to effectively understand users based on the acquired knowledge, we propose to decode users' semantics and to predict intents about follow-up behaviors through a matrix factorization model, which outperforms other SLU models.

Furthermore, the dissertation investigates the performance of SLU modeling for human-human conversations, where two tasks are discussed: actionable item detection and iterative ontology refinement. For actionable item detection, human-machine conversations are utilized to learn intent embeddings through convolutional deep structured semantic models for estimating the probability of appearing actionable items in human-human dialogues. For iterative ontology refinement, ontology induction is first performed on human-human conversations and achieves similar performance as human-machine conversations. The integration of actionable item estimation and ontology induction induces an improved ontology for manual transcripts. Also, the oracle estimation shows the feasibility of iterative ontology refinement and the room for further improvement.

In conclusion, the dissertation shows the feasibility of building a dialogue learning system that is able to understand how particular domains work based on unlabeled human-machine and human-human conversations. As a result, an initial SDS can be built automatically according to the learned knowledge, and its performance can be iteratively improved by interacting with users for practical usage, presenting a great potential for reducing human effort during SDS development.

Keywords

Convolutional Deep Structured Semantic Model (CDSSM)
Deep Structured Semantic Model (DSSM)
Distributional Semantics
Domain Ontology
Embeddings
Intent Modeling
Knowledge Graph
Matrix Factorization (MF)
Multimodality
Random Walk
Spoken Language Understanding (SLU)
Spoken Dialogue System (SDS)
Semantic Representation
Unsupervised Learning

List of Abbreviations

AF Average F-Measure is an evaluation metric that measures the performance of a ranking list by averaging the F-measure over all positions in the ranking list.

AMR Abstract Meaning Representation is a simple and readable semantic representation in AMR Bank.

AP Average Precision is an evaluation metric that measures the performance of a ranking list by averaging the precision over all positions in the ranking list.

ASR Automatic Speech Recognition, also known as computer speech recognition, is the process of converting the speech signal into written text.

AUC Area Under the Precision-Recall Curve is an evaluation metric that measures the performance of a ranking list by averaging the precision over a set of evenly spaced recall levels in the ranking list.

CBOW Continuous Bag-of-Words is an architecture for learning distributed word representations, which is similar to the feedforward neural net language model but uses continuous distributed representation of the context.

CDSSM Convolutional Deep Structured Semantic Model is a deep neural net model with a convolutional layer, where the objective is to maximize the similarity between semantic vectors of two associated elements.

CMU Carnegie Mellon University is a private research university in Pittsburgh.

DSSM Deep Structured Semantic Model is a deep neural net model, where the objective is to maximize the similarity between semantic vectors of two associated elements.

FE Frame Element is a descriptive vocabulary for the components of each frame.

IA Intelligent Assistant is a software agent that can perform tasks or services for an individual. These tasks or services are based on user input, location awareness, and the ability to access information from a variety of online sources.

ICSI International Computer Science Institute is an independent, non-profit research organization located in Berkeley, California, USA.

IR Information Retrieval is the activity of obtaining information resources relevant to an information need from a collection of information resources. Searches can be based on metadata or on full-text (or other content-based) indexing.

ISCA International Speech Communication Association is a non-profit organization that aims to promote, in an international world-wide context, activities and exchanges in all fields related to speech communication science and technology.

LTI Language Technologies Institute is a research department in the School of Computer Science at Carnegie Mellon University.

LU Lexical Unit is a word with a sense.

MAP Mean Average Precision is an evaluation metric that measures the performance of a ranking list by averaging the precision over all positions in the ranking list.

MF Matrix Factorization is a decomposition of a matrix into a product of matrices in the discpline of linear algebra.

MLR Multinomial Logistic Regression is a classification method that generalizes logistic regression to multiclass problems, i.e. with more than two possible discrete outcomes.

NLP Natural Language Processing is a field of artificial intelligence and linguistics that studies the problems intrinsic to the processing and manipulation of natural language.

POS Part of Speech tag, also known as word class, lexical class or lexical class are traditional categories of words intended to reflect their functions within a sentence.

RDF Resource Description Framework (RDF) is a family of World Wide Web Consortium (W3C) specifications originally designed as a metadata data model.

SDS Spoken Dialogue System is an intelligent agent that interacts with a user via natural spoken language in order to help the user obtain desired information or solve a problem more efficiently.

SGD Stochastic Gradient Descent is a gradient descent optimization method for miniming an objective function that is written as a sum of differentiable functions.

SLU Spoken Language Understanding is a component of a spoken dialogue system, which parses the natural languages into semantic forms that benefit the system's understanding.

SPARQL SPARQL Protocol and RDF Query Language is a semantic query language for databases, able to retrieve and manipulate data stored in RDF format.

SVM Support Vector Machine is a supervised learning method used for classification and regression based on the Structural Risk Minimization inductive principle.

WAP Weighted Average Precision is an evaluation metric that measures the performance of a ranking list by weighting the precision over all positions in the ranking list.

WOZ Wizard-of-Oz is a research experiment in which subjects interact with a computer system that subjects believe to be autonomous, but which is actually being operated or partially operated by an unseen human being.

Contents

List of Figures

List of Tables

1 Introduction

> *Computing is not about computers any more. It is about living.*
>
> Nicholas Negroponte, *Massachusetts Institute of Technology Media Lab founder and chairman emeritus*

A spoken dialogue system (SDS) is an intelligent agent that interacts with a user via natural spoken language in order to help the user obtain desired information or solve a problem more efficiently. Despite recent successful personal intelligent assistants (e.g. Google Now[1], Apple's Siri[2], Microsoft's Cortana[3], and Amazon's Echo[4]), spoken dialogue systems are still very brittle when confronted with out-of-domain information. The biggest challenge therefore results from limited domain knowledge. In this introductory chapter, we first introduce how an SDS is developed and articulate a research problem existing in the current development procedure. This chapter outlines the contributions this dissertation brings to address the challenges, and provides a roadmap for the rest of this document.

1.1 Introduction

Spoken language understanding (SLU) has also seen considerable advancements over the past two decades [150]. However, while language understanding remains unsolved, a variety of practical task-oriented dialogue systems have been built to operate on limited specific domains. For instance, the CMU Communicator system is a dialogue system for a air travel domain that provides information about flight, car, and hotel reservations [137]. Another example, the JUPITER system, is a dialogue system for a weather domain, which provides forecast information for the requested city [175]. More recently, a number of efforts in industry (e.g. Google Now, Apple's Siri, Microsoft's Cortana, Amazon's Echo, and Facebook's M) and

[1] http://www.google.com/landing/now/
[2] http://www.apple.com/ios/siri/
[3] http://www.microsoft.com/en-us/mobile/campaign-cortana/
[4] http://www.amazon.com/oc/echo

academia have focused on developing semantic understanding techniques for building better SDSs [1, 14, 56, 72, 103, 107, 120, 129, 131, 138, 160, 166].

These systems aim to automatically identify user intents as expressed in natural language, extract associated arguments or slots, and take actions accordingly to fulfill the user's requests. Typically, a SDS architecture is composed of the following components: an automatic speech recognizer (ASR), a spoken language understanding (SLU) module, a dialogue manager (DM), and an output manager. When developing a dialogue system in a new domain, we may be able to reuse some components that are designed independently of domain-specific information, for example, the speech recognizer. However, the components that are integrated with domain-specific information have to be reconstructed for each new domain, and the cost of development is expensive. With a rapidly increasing number of domains, the current bottleneck of the SDS is SLU.

1.2 The Problem

The classic development process of a dialogue system involves 1) specifying system requirements, 2) designing and implementing each module in a dialogue system to meet all requirements, and 3) evaluating the implemented system. In the first step, dialogue system developers need to specify the scope of a target dialogue system (i.e. the domain that the system can support and operate) to identify domain-specific concepts, a.k.a. slots, and arguments for slot filling; determine the structure of each task to specify the potential intents and associated slots for intent classification; indicate the desired interaction between the system and a user such as the dialogue flow for DM usage. Most of the prior studies focused on implementation of each component in the SDS pipeline under an assumption that the domain-specific schema is given. However, due to unlimited domains, identifying domain-specific information becomes a large issue during SDS development.

Conventionally, the domain-specific knowledge is manually defined by domain experts or developers . For common domains like a weather domain or a bus domain, system developers are usually able to identify such information. However, information of some niche domains (e.g. a military domain) is withheld by experts, making the knowledge engineering process more difficult [13]. Furthermore, the experts' decisions may be subjective and may not cover all possible real-world users' cases [168].

In the second step, implementing each component usually suffers from a common issue, *data scarcity*. Specifically, training intent detectors and slot taggers of SLU requires a set of utterances labeled with task-specific intents and arguments. One simple solution is to create hand-crafted grammars so that the component can be built without any annotated data.

Another solution is to simulate an environment to collect utterances, such as a Wizard-of-Oz (WOZ) method and crowd-sourcing, so that the collected data can be used to train the models [6, 76]. However, the collected data may be biased by developers' subjectivity, because users' perspectives of a task might not be foreseen by dialogue system developers [168].

Furthermore, poor generalization and scalability of current systems result in limited predefined information, and even biases the subsequent data collection and annotation. Another issue is about the efficiency: the manual definition and annotation process for domain-specific tasks can be very time-consuming, and have high financial costs. Finally, the maintenance cost is also non-trivial: when new conversational data comes in, developers, domain experts, and annotators have to manually analyze the audios or the transcripts for updating and expanding the ontologies. Identifying domain knowledge and collecting training data as well as annotations require domain experts and manual labors, resulting in high cost, long duration, and poor scalability of SDS development. The challenges, *generalization, scalability, efficiency*, are the main bottleneck in the current dialogue systems.

1.3 Towards Improved Scalability, Gernalization & Efficiency for SDS

Usually participants engage in a conversation in order to achieve a specific goal such as accomplishing a task or acquiring answers to questions, for example, to obtain a list of restaurants in a specific location. Therefore in the context of this dissertation, domain-specific information refers to the knowledge specific to an SDS-supported task rather than the knowledge about general dialogue mechanisms. To tackle the above problems, we aim to mine the domain-specific knowledge from unlabeled dialogues (e.g. conversations collected by a call center, recorded utterances that cannot be handled by existing systems, etc.) to construct a domain ontology, and then model the SLU component based on the acquired knowledge and unlabeled data in an unsupervised manner.

The dissertation mainly focuses on two parts:

- **Knowledge acquisition** is to learn the domain-specific knowledge that is used by an SLU component. The domain-specific knowledge is represented by a structured ontology, which allows SDS to support the target domain, and thus comprehend meanings. An example of the necessary domain knowledge about restaurant recommendation is shown in Figure 1.1, where the learned domain knowledge contains semantic slots and their relations[5]. The acquired domain ontology provides an overview of a domain or

[5]The slot is defined as a semantic unit usually used in dialogue systems.

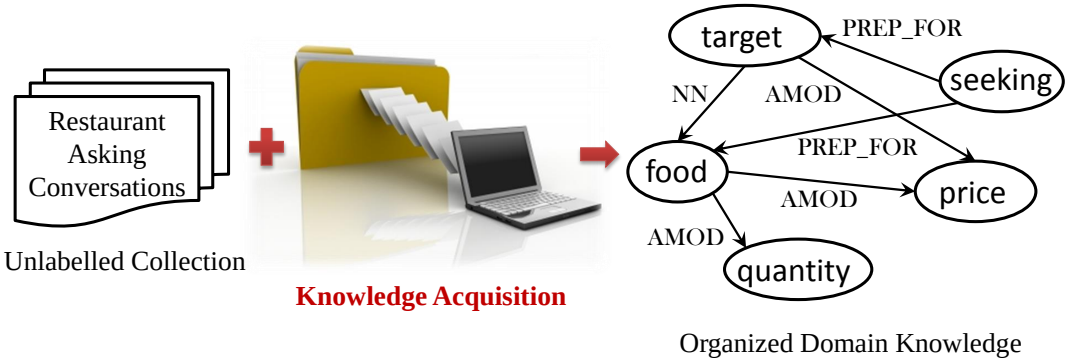


Figure 1.1: An example output of the proposed knowledge acquisition approach.

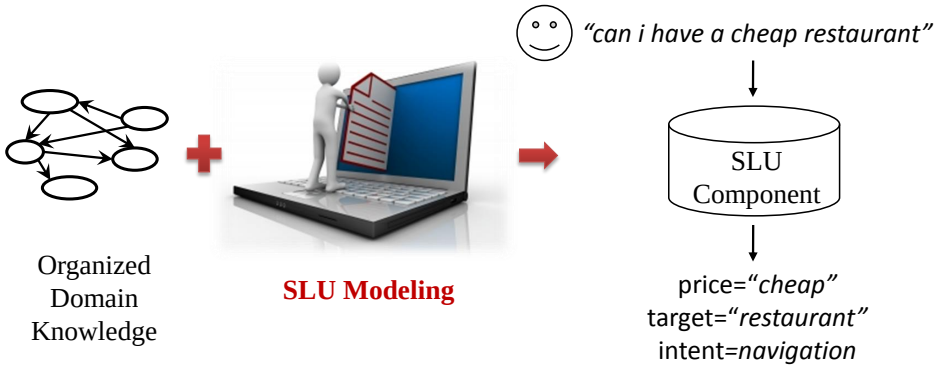


Figure 1.2: An example output of the proposed SLU modeling approach.

multiple domains, which can guide developers for designing the schema or be directly utilized by the SLU module.

- **SLU modeling** is to build an SLU module that is able to understand the actual meaning of domain-specific utterances based on the domain-specific knowledge and then further provide better responses. An example of the corresponding understanding procedure in a restaurant domain is shown in Figure 1.2, where the the SLU component analyzes an utterance "*can i have a cheap restaurant*" and output a semantic representation including low-level slots price="cheap" and target="restaurant" and a high-level intent navigation.

With more available conversational data, to acquire the domain knowledge, recent approaches are data-driven in terms of generalization and scalability. In the past decade, the computational linguistics community has focused on developing language processing algorithms that can leverage the vast quantities of available raw data. Chotimongkol et al. proposed a machine learning technique to acquire domain-specific knowledge, showing the potential for reducing human effort in the SDS development [44]. However, the work mainly focused on

the low-level semantic units like word-level concepts. With increasing high-level knowledge resources, such as FrameNet, Freebase and Wikipedia, this dissertation moves forward to investigate the possibility of developing a high-level semantic conversation analyzer for a certain domain using an unsupervised machine learning approach. The human's semantics, intent, and behavior can be captured from a collection of unlabelled raw conversational data, and then be modeled for building a good SDS.

In terms of practical usage, the acquired knowledge may be manually revised to improve system performance. Even though some revision might be required, the cost of revision is already significantly lower than the cost of analysis. Also, the automatically learned information may employ real-world users' cases and avoid biasing subsequent annotations. This thesis focuses on the highlighted parts, inducing acquiring domain knowledge from the dialogues using available resources, and modeling an SLU module using the automatically acquired information. The proposed approach combining both data-driven and knowledge-driven perspectives shows the potential for improving *generalization*, *maintenance*, *efficiency*, and *scalability* of dialogue system development.

1.4 Thesis Statement

The main purpose of this work is to automatically develop an SLU module for SDS by utilizing the automatically learned domain knowledge in an unsupervised fashion. This dissertation mainly focuses on acquiring the domain knowledge that is useful for better understanding and designing the system framework and further modeling the semantic meaning of the spoken language. For knowledge acquisition, there are two important stages – **ontology induction** and **structure learning**. After applying them, an organized domain knowledge is inferred from unlabeled conversations. For SLU modeling, there are two aspects – **semantic decoding** and **intent prediction**. Based on the acquired ontology, semantic decoding analyzes the semantic meaning in each individual utterance and intent prediction models user intents to predict possible follow-up behaviors. In conclusion, the thesis demonstrates the feasibility of building a dialogue learning system that is able to automatically learn salient knowledge and understand how the domains work based on unlabeled raw conversations. With the acquired domain knowledge, the initial dialogue system can be constructed and improved quickly by continuously interacting with users. The main contribution of the dissertation is presenting the potential for reducing human work and showing the feasibility of improving scalability and efficiency for dialogue system development by automating the knowledge learning process.

1.5 Thesis Structure

The dissertation is organized as below.

- **Chapter 2 - Background and Related Work**
  This chapter reviews background knowledge and summarizes related works. The chapter also discusses current challenges of the task, describes several structured knowledge resources and presents distributional semantics that may benefit understanding problems.

- **Chapter 3 - Ontology Induction for Knowledge Acquisition**
  This chapter focuses on inducing a domain ontology that are useful for developing SLU in SDS based on the available structured knowledge resources in an unsupervised way. Part of this research work has been presented in the following publications [31, 33]:

  – Yun-Nung Chen, William Yang Wang, and Alexander I. Rudnicky, "Unsupervised Induction and Filling of Semantic Slots for Spoken Dialogue Systems Using Frame-Semantic Parsing," in *Proceedings of 2013 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU'13)*, Olomouc, Czech Republic, 2013. **(Student Best Paper Award)**

  – Yun-Nung Chen, William Yang Wang, and Alexander I. Rudnicky, "Leveraging Frame Semantics and Distributional Semantics for Unsupervised Semantic Slot Induction for Spoken Dialogue Systems," in *Proceedings of 2014 IEEE Workshop on of Spoken Language Technology (SLT'14)*, South Lake Tahoe, Nevada, USA, 2014.

- **Chapter 4 - Structure Learning for Knowledge Acquisition**
  This chapter focuses on learning the structures, such as the inter-slot relations, for helping SLU development. Some of the contributions have been presented in the following publications [39, 40]:

  – Yun-Nung Chen, William Yang Wang, and Alexander I. Rudnicky, "Jointly Modeling Inter-Slot Relations by Random Walk on Knowledge Graphs for Unsupervised Spoken Language Understanding," in *Proceeding of The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT'15)*, Denver, Colorado, USA, 2015.

  – Yun-Nung Chen, William Yang Wang, and Alexander I. Rudnicky, "Learning Semantic Hierarchy for Unsupervised Slot Induction and Spoken Language Understanding," in *Proceedings of The 16th Annual Conference of the Interna-*

tional Speech Communication Association (INTERSPEECH'15), Dresden, Germany, 2015.

- **Chapter 5 - Surface Form Derivation for Knowledge Acquisition**
  This chapter focuses on deriving the surface forms conveying semantics for entities from the given ontology, where the derived information contributes to better understanding. Some of the work has been published [32]:

  – Yun-Nung Chen, Dilek Hakkani-Tür, and Gokhan Tur, "Deriving Local Relational Surface Forms from Dependency-Based Entity Embeddings for Unsupervised Spoken Language Understanding," in *Proceedings of 2014 IEEE Workshop of Spoken Language Technology (SLT'14)*, South Lake Tahoe, Nevada, USA, 2014.

- **Chapter 6 - Semantic Decoding in SLU Modeling**
  This chapter focuses on decoding users' spoken languages into corresponding semantic forms, which corresponds to the goal of SLU. Some of these contributions have been presented in the following publication [38]:

  – Yun-Nung Chen, William Yang Wang, Anatole Gershman, and Alexander I. Rudnicky, "Matrix Factorization with Knowledge Graph Propagation for Unsupervised Spoken Language Understanding," in *Proceeding of The 53rd Annual Meeting of the Association for Computational Linguistics and The 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP 2015)*, Beijing, China, 2015.

- **Chapter 7 - Intent Prediction in SLU Modeling**
  This chapter focuses on modeling user intents in SLU, so that the SDS is able to predict the users' follow-up actions and further provide better interactions. Some of the contributions have been presented by following publications [28, 36, 37, 42]:

  – Yun-Nung Chen and Alexander I. Rudnicky, "Dynamically Supporting Unexplored Domains in Conversational Interactions by Enriching Semantics with Neural Word Embeddings," in *Proceedings of 2014 IEEE Workshop of Spoken Language Technology (SLT'14)*, South Lake Tahoe, Nevada, USA, 2014.

  – Yun-Nung Chen, Ming Sun, Alexander I. Rudnicky, and Anatole Gershman, "Leveraging Behavioral Patterns of Mobile Applications for Personalized Spoken Language Understanding," in *Proceedings of The 17th ACM International Conference on Multimodel Interaction (ICMI'15)*, Seattle, Washington, USA, 2015.

  – Yun-Nung Chen, Ming Sun, and Alexander I. Rudnicky, "Matrix Factorization with Domain Knowledge and Behavioral Patterns for Intent Modeling," in *Extended Abstract of The 29th Annual Conference on Neural Information Processing*

Systems – Machine Learning for Spoken Language Understanding and Interactions Workshop (NIPS-SLU'15), Montreal, Canada, 2015.

- Yun-Nung Chen, Ming Sun, Alexander I. Rudnicky, and Anatole Gershman, "Unsupervised User Intent Modeling by Feature-Enriched Matrix Factorization," in *Proceedings of The 41st IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'16)*, Shanghai, China, 2016.

- **Chapter 8 - SLU in Human-Human Conversations**
  This chapter investigates the feasibility of applying the technologies developed for human-machine interactions to human-human interactions, expanding the application usage to more practical and broader genres. Part of the research work has been presented in the following publications [34, 35]:

  - Yun-Nung Chen, Dilek Hakkani-Tür, and Xiaodong He, "Detecting Actionable Items in Meetings by Convolutional Deep Structured Semantic Models," in *Proceedings of 2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU'15)*, Scottsdale, Arizona, 2015.
  - Yun-Nung Chen, Dilek Hakkani-Tür, and Xiaodong He, "Learning Bidirectional Intent Embeddings by Convolutional Deep Structred Semantic Models for Spoken Language Understanding," in *Extended Abstract of The 29th Annual Conference on Neural Information Processing Systems – Machine Learning for Spoken Language Understanding and Interactions Workshop (NIPS-SLU'15)*, Montreal, Canada, 2015.

- **Chapter 9 - Conclusions and Future Work**
  This chapter concludes the main contributions and discusses a number of interesting directions that can be explored in the future.

2 Background and Related Work

> *Everything that needs to be said has already been said. But since no one was listening, everything must be said again.*
>
> André Gide, *Nobel Prize in Literature winner*

With an emerging trend of using mobile devices, spoken dialogue systems (SDS) are being incorporating in several devices (e.g. smartphone, smart-TV, navigating system). In the architecture of SDSs, spoken language understanding (SLU) plays an important role and there are many unsolved challenges. The next section first introduces a typical pipeline of an SDS and elaborates the functionality of each individual component. Section 2.2 details how SLU works with different examples, reviews the related literature and discusses their pros and cons; following the literature review, we briefly sketch the idea of the proposed approaches and how it is related to prior studies. Then semantic resources that are used for benefiting language understanding are introduced, where the resources with explicit semantics, *Ontology and Knowledge Base*, are presented in Section 2.3, and implicit semantics based on the theory, *Distributional Semantics*, are presented in Section 2.4.

2.1 Spoken Dialogue System (SDS)

A typical SDS is composed of a recognizer, a spoken language understanding (SLU) module, a dialogue manager (DM), and an output manager. Figure 2.1 illustrates the system pipeline. The functionality of each component is summarized below.

- Automatic Speech Recognizer (ASR)
  The ASR component takes raw audio signals and then transcribes into word hypotheses with confidence scores. The top one hypothesis would then be transmitted into the next component.
- Spoken Language Understanding (SLU)
  The goal of SLU is to capture the core semantics given the input word hypothesis; And

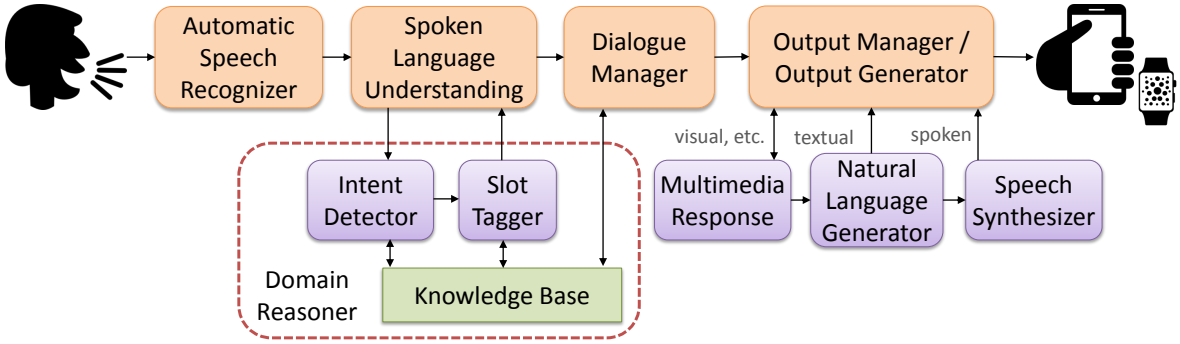


Figure 2.1: The typical pipeline in a dialogue system.

the extracted information can be populated into task-specific arguments in a given semantic frame [82]. Therefore the task of an SLU module is to identify user intents and fill associated slots based on the word hypotheses. This procedure is also called semantic parsing, semantic decoding, etc. The SLU component typically includes an intent detector and slot taggers. An example utterance "*I want to fly to Taiwan from Pittsburgh next week*" can be parsed into find_flight(origin="*Pittsburgh*", destination="*Taiwan*", departure_date="*next week*"), where find_flight is classified by the intent detector; and the associated slots are later filled by the slot taggers based on the detected intent. This component also estimates confidence scores of decoded semantic representations for next component usage.

- Dialogue Manager (DM) / Task Manager
  Subsequent to the SLU processing, the DM interacts with users to assist them in achieving their goals. Given the above example, DM should check whether required slots are properly assigned (departure_date may not properly specified) and then decide the system's action such as ask_date or return_flight(origin="*Pittsburgh*", destination="*Taiwan*"). This procedure should access knowledge bases as a retrieval database to acquire the desired information. Due to possible misrecognition and misunderstanding errors, this procedure involves dialogue state tracking and policy selection to make more robust decisions [85, 164].

- Output Manager / Output Generator
  Traditional dialogue systems are mostly used through phone calls, so the output manager mainly interacts with two modules, a natural language generation (NLG) module and a speech synthesizer. However, with increasing usage of various multimedia devices (e.g. smartphone, smartwatch, and smart-TV), the output manager does not need to focus on generating spoken responses. Instead, recent trend is moving toward displaying responses via different channels; for example, the utterance "*Play Lady Gaga's Bad Romance.*" should correspond to an output action that launches a music player and

then plays the specified song. Hence an additional component, multimedia response, is introduced in the infrastructure in order to handle diverse multimedia outputs.

- Multimedia Response
  Given the decided action, a multimedia response considers which channel is more suitable to present the returned information based on environmental contexts, user preference, and used devices. For example, return_flight(origin="*Pittsburgh*", destination="*Taiwan*") can be presented through visual responses by listing the flights that satisfy the requirement in desktops, laptops, etc., and through spoken responses by uttering "*There are seven flights from Pittsburgh to Taiwan. First is ...*" in the smartwatches.
- Natural Language Generation (NLG)
  Given the current dialogue strategy, the NLG component generates the corresponding natural language responses that humans can understand for the purpose of natural dialogues. For example, an action from DM, ask_date, can generate a response "*Which date will you plan to fly?*". Here the responses can be template-based or outputted by statistical models [29, 163].
- Speech Synthesizer / Text-to-Speech (TTS)
  In order to communicate with users via speech, a speech synthesizer simulates human speech based on the natural language responses generated by the NLG component.

All basic components in a dialogue system should interact with each other, so errors may propagate and then result in poor performance. In addition, several components (e.g. the SLU module) need to incorporate the domain knowledge in order to handle task-specific dialogues. Because domain knowledge is usually predefined by experts or developers, when there are more and more domains, making SLU scalable has been a main challenge of SDS development.

2.2 Spoken Language Understanding (SLU)

In order to allow machines to understand natural language, a semantic representation[1] is introduced. A semantic representation of an utterance carries its core content, so that the actual meaning behind the utterance can be inferred only through the representation. For example, an utterance "*show me action movies directed by james cameron*" can be represented

[1] In this document, we use the terms "semantic representation" and "semantic form" interchangeably.

as action="*show*", target="*movie*", genre="*action*", director="*james cameron*". Another utterance "*find a cheap taiwanese restaurant in oakland*" can be formed as action="*find*", target="*restaurant*", price="*cheap*", type="*taiwanese*", location="*oakland*". The semantic representations are able to convey the core meaning of the utterances, which can be more easily processed by machines. The semantic representation is not unique, and there are several forms for representing meanings. Below we describe two types of semantic forms:

- Slot-Based Semantic Representation
  The slot-based representation is a flat structure of semantic concepts, which are usually used in simpler tasks. Above examples belong to slot-based semantic representations, where semantic concepts are action, target, location, price, etc.

- Relation-Based Semantic Representation
  The relation-based representation includes structured concepts, which are usually used in tasks that have more complicate dependency relations. For instance, "*show me action movies directed by james cameron*" can be represented as movie.directed_by, movie.genre, director.name="*james cameron*", genre.name="*action*". This representation is the same as movie.directed_by(?, "*james cameron*") $\wedge$ movie.genre(?, "*action*"), which originated from the logic form in the artificial intelligence field. The semantic slots in the slot-based representation are formed as relations here.

The main purpose of an SLU component is to convert the natural language into semantic forms. In the natural language processing (NLP) field, natural language understanding (NLU) also refers to semantic decoding or semantic parsing. Therefore, this section reviews related literature and studies how they approach the problems for language understanding. After that, the following chapters focus on addressing the challenges that building an SDS suffers from, namely:

- How can we define semantic elements from unlabeled data to form a semantic schema?
- How can we organize semantic elements and then form a meaningful structure?
- How can we decode semantics for test data while considering noises in the mean time?
- How can we utilize the acquired information to predict user intents for improving system performance?

2.2.1 Leveraging External Resources

Building semantic parsing systems requires large training data with detailed annotations. With rich web-scaled resources, a lot of NLP research therefore leveraged external human

knowledge resources for semantic parsing. For example, Berant et al. proposed SEMPRE[2], which used the web-scaled knowledge bases to train the semantic parser [10]. Das *et al.* proposed SEMAFOR[3], which utilized a lexicon developed based on a linguistic theory – Frame Semantics to train the semantic parser [50]. However, such NLP tasks deal with individual and focused problems, ignoring how parsing results are used by applications.

Tur et al. were among the first to consider unsupervised approaches for SLU, where they exploited query logs for slot-filling [152, 154]. In a subsequent study, Heck and Hakkani-Tür studied the Semantic Web for an unsupervised intent detection problem in SLU, showing that results obtained from the unsupervised training process align well with the performance of traditional supervised learning [82]. Following their success of unsupervised SLU, recent studies have also obtained interesting results on the tasks of relation detection [32, 125, 75], entity extraction [159], and extending domain coverage [41, 28, 57]. Section 2.3 will introduce the exploited knowledge resources and the corresponding analyzers in detail. However, most of the prior studies considered semantic elements independently or only considered the relations appearing in the external resources, where the structure of concepts used by real users might be ignored.

2.2.2 Structure Learning and Inference

From a knowledge management perspective, empowering dialogue systems with large knowledge bases is of crucial significance to modern SDSs. While leveraging external knowledge is the trend, efficient inference algorithms, such as random walks, are still less-studied for direct inference on knowledge graphs of the spoken contents. In the NLP literature, Lao et al. used a random walk algorithm to construct inference rules on large entity-based knowledge bases, and leveraged syntactic information for reading the web [101, 102]. Even though this work has important contributions, the proposed algorithm cannot learn mutually-recursive relations, and does not to consider lexical items—in fact, more and more studies show that, in addition to semantic knowledge graphs, lexical knowledge graphs that model surface-level natural language realization, multiword expressions, and context, are also critical for short text understanding [91, 109, 110, 145, 158].

2.2.3 Neural Model and Representation

With the recently emerging trend of neural systems, a lot of work has shown the success of applying neural-based models in SLU. Tur et al. have shown that deep convex networks are effective for building better semantic utterance classification systems [153]. Following their

[2] `http://www-nlp.stanford.edu/software/sempre/`
[3] `http://www.ark.cs.cmu.edu/SEMAFOR/`

success, Deng et al. have further demonstrated the effectiveness of applying the kernel trick to build better deep convex networks for SLU [54]. Nevertheless, most of work used neural-based representations for supervised tasks, so there is a gap between approaches used for supervised and unsupervised tasks.

In addition, recently Mikolov proposed recurrent neural network based language models to capture long dependency and achieved the state-of-the-art performance in recognition [114, 116]. The proposed continuous representations as word embeddings have further boosted the state-of-the-art results in many applications, such as sentiment analysis, sentence completion, and relation detection [32, 117, 144]. The detail of distributional representations will be described in Section 2.4. Despite the advances of several NLP tasks, how unsupervised SLU can incorporate neural representations remains unknown.

2.2.4 Latent Variable Modeling

Most of the studies above did not explicitly learn latent factor representations from data, so they may neglect errors (e.g. misrecognition) and thus produce unreliable results of SLU [12]. Early studies on latent variable modeling in speech included the classic hidden Markov model for statistical speech recognition [94]. Recently, Celikyilmaz et al. were the first to study the intent detection problem using query logs and a discrete Bayesian latent variable model [23]. In the field of dialogue modeling, the partially observable Markov decision process (POMDP) model is a popular technique for dialogue management [164, 172], reducing the cost of hand-crafted dialogue managers while producing robustness against speech recognition errors. More recently, Tur et al. used a semi-supervised LDA model to show improvement on the slot filling task [155]. Also, Zhai and Williams proposed an unsupervised model for connecting words with latent states in HMMs using topic models, obtaining interesting qualitative and quantitative results [174]. However, for unsupervised SLU, it is unclear how to take latent semantics into account.

2.2.5 The Proposed Method

Towards unsupervised SLU, this dissertation proposes an SLU model to integrate the advantages of prior studies and overcome the disadvantages mentioned above. The model leverages the external knowledge while combining frame semantics and distributional semantics, and learns latent feature representations while taking various local and global lexical, syntactic and semantic relations into account in an unsupervised manner. The details will be presented in the following chapters.

Table 2.1: The frame example defined in FrameNet.

Frame: Food	
Semantics	physical_object noun: *almond, apple, banana, basil, beef, beer, berry, ...*
Frame Element	constituent_parts, descriptor, type

2.3 Domain Ontology and Knowledge Base

There are two main types of knowledge resources available, generic concept and entity-based, both of which may benefit SLU modules for SDSs. Below we first introduce the detailed definition of each resource and discuss the corresponding work and tools that are useful for leveraging such resources.

2.3.1 Definition

For generic concept and entity-based knowledge bases, the former covers the concepts that are more common, such as a food domain and a weather domain. The latter usually contains a lot of named entities that are specific for certain domains, for example, a movie domain and a music domain. The following describes examples of these knowledge resources, which contain the rich semantics and may be beneficial for understanding tasks.

2.3.1.1 Generic Concept Knowledge

There are two semantic knowledge resources for generic concepts, FrameNet and Abstract Meaning Representation (AMR).

- **FrameNet**[4] is a linguistically semantic resource that offers annotations of predicate-argument semantics, and associated lexical units for English [4]. FrameNet is developed based on semantic theory, Frame Semantics [64]. For example, the phrase "*low fat milk*" should be analyzed with "*milk*" evoking the food frame, where "*low fat*" fills the descriptor FE of that frame and the word "*milk*" is the actual LU. A defined frame example is shown in Table 2.1.
- **Abstract Meaning Representation (AMR)** is a semantic representation language including the meanings of thousands of English sentences. Each AMR is a single rooted, directed graph. AMRs include PropBank semantic roles, within-sentence coreference, named entities and types, modality, negation, questions, quantities, etc [5]. The AMR

[4] `http://framenet.icsi.berkeley.edu`

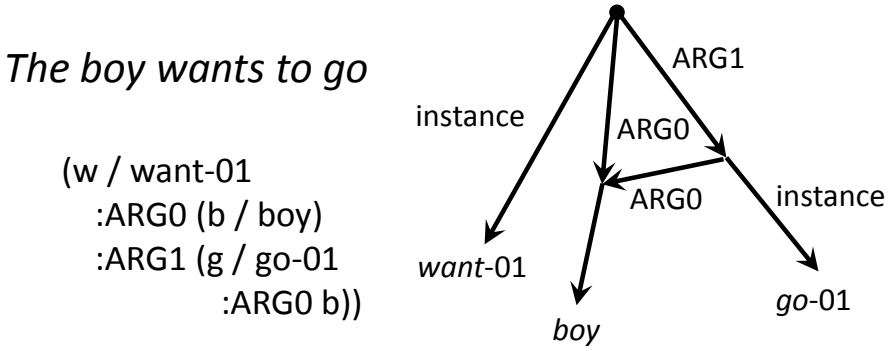


Figure 2.2: A sentence example in AMR Bank.

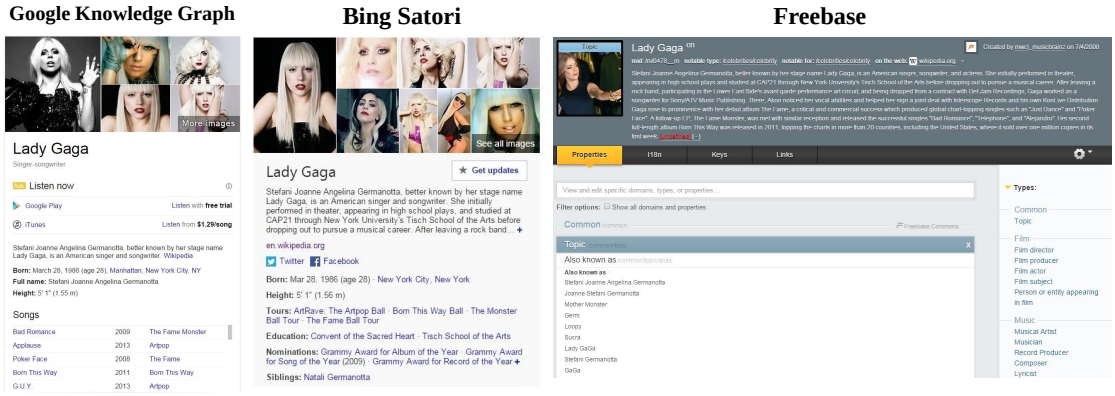


Figure 2.3: Three famous semantic knowledge graph examples (Google's Knowledge Graph, Bing Satori, and Freebase) corresponding to the entity "Lady Gaga".

feature structure graph of an example sentence is illustrated in Figure 2.2, where the "*boy*" appears twice, once as the ARG0 of "*want-01*", and once as the ARG0 of "*go-01*".

2.3.1.2 Entity-Based Knowledge

- **Semantic Knowledge Graph** is a knowledge base that provides structured and detailed information about the topic with a lists of related links. Three different knowledge graph examples, Google's knowledge graph[5], Microsoft's Bing Satori, and Freebase, are shown in Figure 2.3. The semantic knowledge graph is defined by a schema and composed of nodes and edges connecting the nodes, where each node represents an entity-type and the edge between each node pair describes their relation, as called as property. An example from Freebase is shown in Figure 2.4, where nodes represent core entity-types for the movie domain. The domains in the knowledge graphs span the web, from "American Football" to "Zoos and Aquariums".

[5] `http://www.google.com/insidesearch/features/search/knowledge.html`

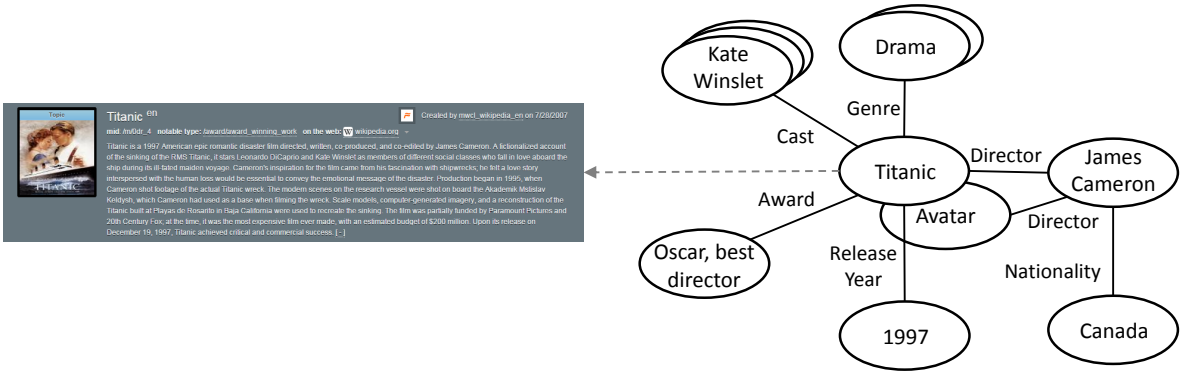


Figure 2.4: A portion of the Freebase knowledge graph related to the movie domain.

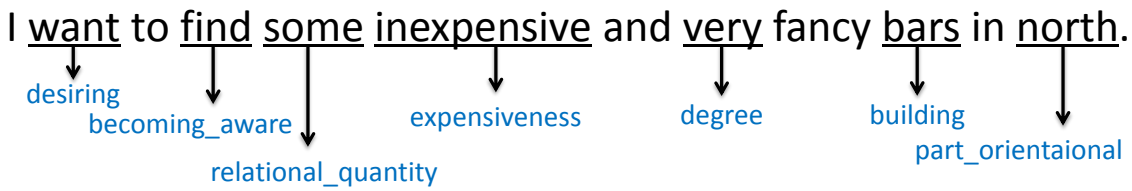


Figure 2.5: An example of FrameNet categories for an ASR output labelled by probabilistic frame-semantic parsing.

- **Wikipedia**[6] is a free-access, free content Internet encyclopedia, which contains a large number of pages/articles related to a specific entity [124]. It provides basic background knowledge for help understanding tasks in the natural language processing (NLP) field.

2.3.2 Knowledge-Based Semantic Analyzer

With the available knowledge resources mentioned above, there are many work that utilizes such knowledge for different tasks. The prior approaches or tools can serve as analyzers and facilitate the target task of this dissertation, unsupervised SLU for dialogue systems.

2.3.2.1 Generic Concept Knowledge

- FrameNet
  SEMAFOR[7] is a state-of-the-art semantic parser for frame-semantic parsing [48, 49]. Trained on manually annotated sentences in FrameNet, SEMAFOR is relatively accurate in predicting semantic frames, FE, and LU from raw text. SEMAFOR is augmented by the dual decomposition techniques in decoding, and thus produces semantically-

[6] http://en.wikipedia.org/wiki/Wikipedia
[7] http://www.ark.cs.cmu.edu/SEMAFOR/

show me what richard lester directed

```
(s / show-01
      :ARG1 (d / direct-01
            :ARG0 (p / person
                  :name (n / name
                        :op1 "lester”
                        :op2 "richard"))))
```

Figure 2.6: An example of AMR parsed by JAMR on an ASR output.

Michael Jordan is my favorite player.

Michael Jordan is a machine learning expert.

Figure 2.7: An example of Wikification.

labeled outputs in a timely manner. Note that SEMAFOR does not consider the relations between frames but treat each frame independently. Figure 2.5 shows the output of probabilistic frame-semantic parsing.

- Abstract Meaning Representation (AMR)
  JAMR[8] is the first semantic parser that parses the sentences into AMRs [65]. Trained on manually defined AMR Bank, JAMR applied an algorithm for finding the maximum, spanning, connected subgraph and showed how to incorporate extra constraints with Lagrangian relaxation. Figure 2.6 shows the output of JAMR on an example sentence.

2.3.2.2 Entity-Based Knowledge

- Semantic Knowledge Graph
  Freebase API[9] is an API for accessing the data, and the data can also be dumped directly.

- Wikipedia

[8] `http://github.com/jflanigan/jamr`
[9] `https://developers.google.com/freebase/`

Wikifier[10] is an entity linking (also known as Wikification, Disambiguation to Wikipedia (D2W)) tool. The task is to identify concepts and entities in texts and disambiguate them into the corresponding Wikipedia pages. An example is shown in Figure 2.7, where the entities "Micheal Jordan" in two sentences refer to different people, pointing to different Wikipedia pages.

2.4 Distributional Semantics

The distributional view of semantics hypothesizes that words occurring in the same contexts may have similar meanings [79]. As the foundation for modern statistical semantic, an early success that implements this distributional theory is latent semantic analysis (a.k.a. LSA) [53, 66]. Brown et al. proposed an early hierarchical clustering algorithm that extracts word clusters from large corpora [21], which has been used successfully in many NLP applications [111]. Recently, with the advance of deep learning techniques, continuous word embeddings (a.k.a. word representations, or neural embeddings) have further boosted the state-of-the-art results in many NLP tasks, due to its rich continuous representations (e.g. vectors, or sometimes matrices, and tensors) that capture the context of the target semantic unit [7, 156].

The continuous word vectors are derived from a recurrent neural network architecture [115]. The recurrent neural network language models use the context history to include long-distance information, outperforming standard bag-of-words n-gram language models. ,Interestingly, the vector-space word representations learned from the language models were shown to capture syntactic and semantic regularities [118, 119]. The word relationships are characterized by vector offsets, where in the embedded space, all pairs of words sharing a particular relation are related by the same constant offset.

Based on the considered contexts of embedding training, there are two types, linear and dependency-based embeddings, which are described in Section 2.4.1 and Section 2.4.2 respectively. Also, a considered unit can be a word or a word sequence (sentence, paragraph), and Section 2.4.3 and Section 2.4.4 detail training procedures.

2.4.1 Linear Word Embedding

Typical neural embeddings use linear word contexts, where a window size is defined to produce contexts of the target words, a.k.a. `word2vec` [117, 118, 119]. There are two model architectures for learning distributed word representations: continuous bag-of-words (CBOW) model

[10] `http://cogcomp.cs.illinois.edu/page/software_view/Wikifier`

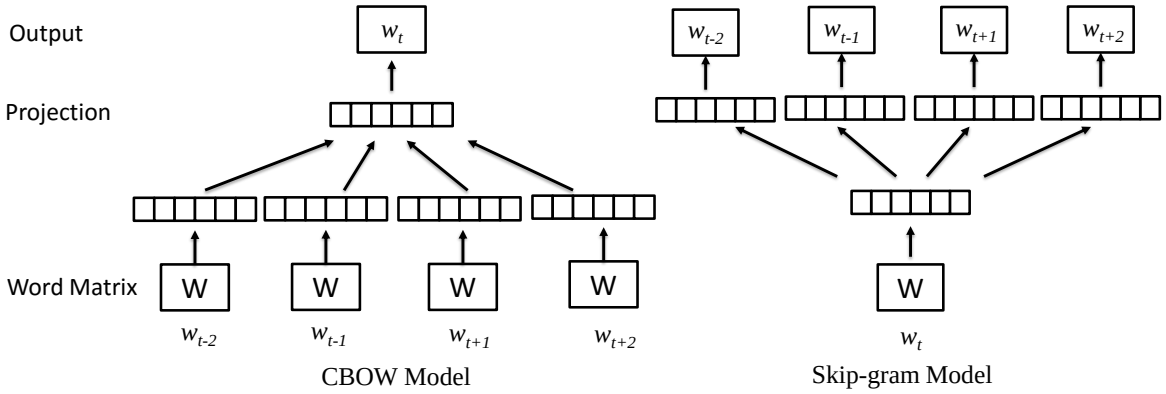


Figure 2.8: The CBOW and Skip-gram architectures. The CBOW model predicts the current word based on the context, and the Skip-gram model predicts surrounding words given the target word [117].

and continuous skip-gram model, where the former predicts the current word based on the context and the latter predicts surrounding words given the current word.

- Continuous Bag-of-Words (CBOW) Model
  The word representations are learned by a recurrent neural network language model [115], as illustrated in Figure 2.8. The architecture contains an input layer, a hidden layer with recurrent connections, and the corresponding weight matrices. Given a word sequence $w_1, ..., w_T$, the objective function of the model is to maximize the probability of observing the target word w_t given its contexts $w_{t-c}, w_{t-c+1}, ..., w_{t+c-1}, w_{t+c}$, where c is the window size:

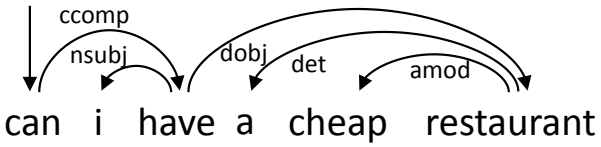
$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq i\leq c, i\neq 0}\log p(w_t \mid w_{t+i}). \tag{2.1}$$

  The objective can be trained using stochastic gradient updates via backpropagation over the observed corpus.

- Continuous Skip-Gram Model
  The training objective of the skip-gram model is to find word representations that are useful for predicting the surrounding words, which is similar to the CBOW architecture. Given a word sequence as the training data $w_1, ..., w_T$, the objective function of the model is to maximize the average log probability:

$$\frac{1}{T}\sum_{i=t}^{N}\sum_{-c\leq i\leq c, i\neq 0}\log p(w_{t+i} \mid w_t) \tag{2.2}$$

  The objective can also be obtained in the similar way.



Word	Contexts
can	have/ccomp
i	have/nsubj^{-1}
have	can/ccomp^{-1}, i/nsubj, restaurant/dobj
a	restaurant/det^{-1}
cheap	restaurant/amod^{-1}
restaurant	have/dobj^{-1}, cheap/amod

Figure 2.9: The target words and associated dependency-based contexts extracted from the parsed sentence for training depedency-based word embeddings.

The rich semantics contained by word embeddings helps learn relations of slot-fillers, so following chapters utilize the properties for the SLU task.

2.4.2 Dependency-Based Word Embedding

Most neural embeddings use linear bag-of-words contexts, where a window size is defined to produce contexts of the target words [117, 118, 119]. However, some important contexts may be missing due to smaller windows, while larger windows capture broad topical content. To solve the above problems, a dependency-based embedding approach was proposed to derive contexts based on the syntactic relations the word participates in for training embeddings, where the embeddings are less topical but offer more functional similarity compared to original embeddings [108].

Figure 2.9 shows the extracted dependency-based contexts for each target word from the dependency-parsed sentence, where headwords and their dependents can form the contexts by following the arc on a word in the dependency tree, and -1 denotes the directionality of the dependency. After replacing original bag-of-words contexts with dependency-based contexts, we can train dependency-based embeddings for all target words [16, 17, 169].

For training dependency-based word embeddings, each target x is associated with a vector $\mathbf{v_x} \in \mathbb{R}^d$ and each context c is represented as a context vector $\mathbf{v_c} \in \mathbb{R}^d$, where d is the embedding dimensionality. We learn vector representations for both targets and contexts such that the dot product $\mathbf{v_x} \cdot \mathbf{v_c}$ associated with "good" target-context pairs belonging to the training data $\mathcal{D}$ is maximized, leading to the objective function:

$$\arg\max_{\mathbf{v_x},\mathbf{v_c}} \sum_{(w,c)\in\mathcal{D}} \log \frac{1}{1+\exp(-\mathbf{v_c}\cdot\mathbf{v_x})}, \tag{2.3}$$

which can be trained using stochastic-gradient updates [108]. We thus expect the syntactic contexts to yield more focused embeddings, capturing more functional and less topical

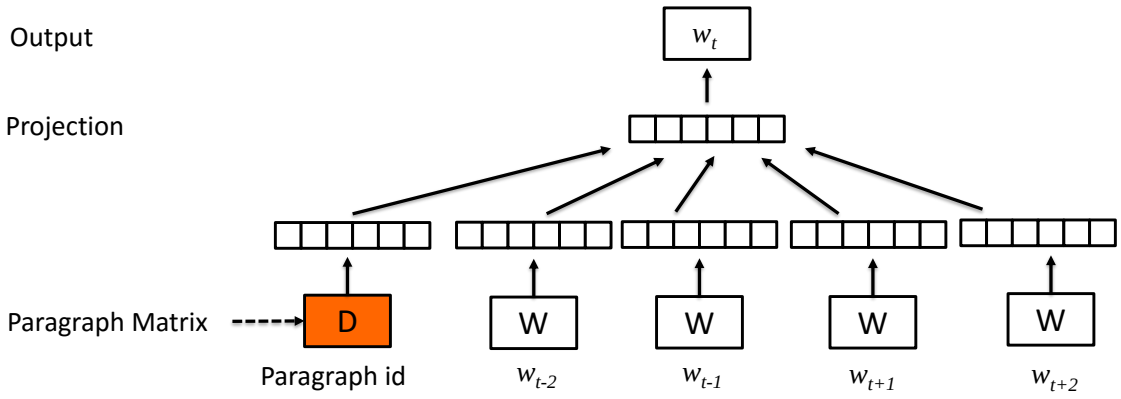


Figure 2.10: The framework for learning paragraph vectors.

similarity. The dependency-based word embeddings capture more structure information, and their properties are used for the proposed approaches presented in following chapters.

2.4.3 Paragraph Embedding

The paragraph vector is a distributed memory model, a.k.a. `doc2vec`, which holds the similar assumption as CBOW word vectors: the paragraph vectors are asked to contribute to the prediction task of the next word given many contexts sampled from the paragraph. Figure 2.10 illustrates the framework for learning paragraph vectors. Comparing to the architecture of the CBOW model, the additional paragraph token that is mapped to a vector via a matrix D. In this model, the concatenation or average of this vector with a context of three words is used to predict the fourth word. The paragraph vector can be viewed as another word, acting as a memory that represents the missing information from the current context or the topic of the paragraph [104]. Then the paragraph vector is also the semantic representations of each paragraph, so called paragraph embeddings.

2.4.4 Sentence Embedding

The sentence embeddings originated from information retrieval (IR), where the user click logs were utilized as implicit information to mine the relations between queries and documents [90, 139, 140]. In the deep neural network architecture, a vector in the semantic layer can represent the semantics of a word sequence (a query or a document). Therefore, the vector representations can be treated as sentence embeddings, a.k.a. `sent2vec`. Different from above embeddings, training sentence embeddings requires the paired data, which can be viewed as supervised training. Although the goal of this thesis is unsupervised SLU, training sentence embeddings from available data may provide additional knowledge and then help

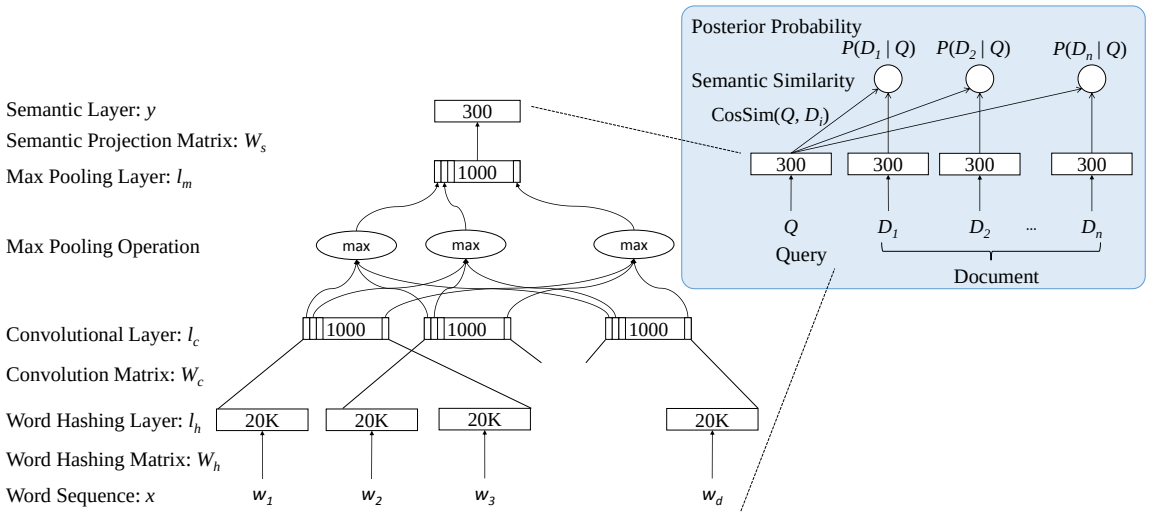


Figure 2.11: Illustration of the CDSSM architecture for the IR task.

the target task. Chapter 8 will apply the technique for improved SLU.

2.4.4.1 Architecture

The model is a deep neural network with a convolutional layer, called convolutional deep structured semantic model (CDSSM), whose architecture is illustrated in Figure 2.11 [69, 90, 139, 140]. The model contains: 1) a word hashing layer obtained by converting one-hot word representations into tri-letter vectors, 2) a convolutional layer that extracts contextual features for each word with its neighboring words defined by a window, 3) a max-pooling layer that discovers and combines salient features to form a fixed-length sentence-level feature vector, and 4) a semantic layer that further transform the max-pooling layer to a low-dimensional semantic vector for the input sentence.

Word Hashing Layer l_h. Each word from a word sequence (i.e. an utterance) is converted into a tri-letter vector [90]. For example, the tri-letter vector of the word "#email#" (# is a word boundary symbol) has non-zero elements for "#em", "ema", "mai", "ail", and "il#" via a word hashing matrix W_h. Then we build a high-dimensional vector l_h by concatenating all word tri-letter vectors. The advantages of tri-letter vectors include: 1) OOV words can be represented by tri-letter vectors, where the semantics can be captured based on the subwords such as prefix and suffix; 2) the tri-letter space is smaller. For example, each word in a 40K word vocabulary can be represented by a 10,306-dimensional vector using letter trigrams, showing a four-fold dimensionality reduction with few collisions. When the vocabulary size is larger, the reduction of dimensionality is even more significant. Therefore, incorporating tri-letter vectors improves the representation power of word vectors and also reduces the OOV

problem while maintaining a small vocabulary size.

Convolutional Layer l_c. A convolutional layer extracts contextual features c_i for each target word w_i, where c_i is the vector concatenating the word vector of w_i and its surrounding words within a window (the window size is set to 3). For each word, a local feature vector l_c is generated using a tanh activation function and a global linear projection matrix W_c:

$$l_{ci} = \tanh(W_c^T c_i), \text{ where } i = 1, ..., d, \tag{2.4}$$

where d is the total number of windows.

Max-Pooling Layer l_m. The max-pooling layer forces the network to only retain the most useful local features by applying the max operation over each dimension of l_{ci} across i in (2.4),

$$l_{mj} = \max_{i=1,...,d} l_{ci}(j). \tag{2.5}$$

The convolutional and max-pooling layers are able to capture prominent words of the word sequences [69, 139]. As illustrated in Fig. 8.3, if we view the local feature vector $l_{c,i}$ as a topic distribution of the local context window, e.g., each element in the vector corresponds to a hidden topic and the value corresponds to the activation of that topic, then taking the max operation at each element keeps the max activation of that hidden topic across the whole sentence.

Semantic Layer y. The global feature vector l_m in (2.5) is fed to feed-forward neural network layers to output the final non-linear semantic features y as the output layer.

$$y = \tanh(W_s^T l_m), \tag{2.6}$$

where W_s is a learned linear projection matrix. The output semantic vector can be either document embeddings y_D or query embeddings y_Q.

2.4.4.2 Training Procedure

With the pairs of queries and clicked documents, the idea of this model is to learn the embeddings for queries and documents such that the documents with the same queries can be close to each other in the continuous space. Below we define the semantic score between a query Q and a document D using the cosine similarity between their embeddings:

$$\text{CosSim}(Q, D) = \frac{y_Q \cdot y_D}{|y_Q||y_D|}. \tag{2.7}$$

The posterior probability of a document given a query is computed based on the semantic

score through a softmax function,

$$P(D \mid Q) = \frac{\exp(\text{CosSim}(Q, D))}{\sum_{D'} \exp(\text{CosSim}(Q, D'))}, \tag{2.8}$$

where D' is a document from the archive.

For training the model, we maximize the likelihood of the associated documents given the queries through the click logs. The parameters of the model $\theta = \{W_c, W_s\}$ is optimized by an objective:

$$\Lambda(\theta) = \log \prod_{(Q,D^+)} P(D^+ \mid Q). \tag{2.9}$$

The model is optimized using mini-batch stochastic gradient descent (SGD) [90]. Then each document can be represented as a vector, and we can transform the input queries into the vector to estimate the relevance scores for the retrieval task.

2.5 Evaluation Metrics

In the rest of the document, we evaluate the proposed approaches using different metrics. This section introduces how the metrics are computed and when they are used for measurement. In most of binary classification tasks, precision, recall, f-measure, accuracy are standard metrics for evaluation.

- ◇ *Precision (P)* (also called positive predictive value) is the fraction of retrieved instances that are true positives.
- ◇ *Recall (R)* is the fraction of relevant instances that are retrieved.
- ◇ *F-Measure (F_1)* is the harmonic mean that combines precision and recall.

$$F_1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{2.10}$$

- ◇ *Accuracy (ACC)* is the fraction of all instances that have correct labels, which is mostly used for prediction problems.

Precision and recall are single-value metrics based on the whole list of documents returned by the system. For systems that return a ranked sequence of instances, it is desirable to also consider the order in which the returned instances are presented.

- ◇ *Precision at K (P@K)* corresponds to the number of relevant instances in the top K returned results, but fails to take into account the positions of the relevant documents

among the top K. The metric is usually used for some tasks where measuring recall is not meaningful. Especially for modern retrieval tasks, as many queries have thousands of relevant instances, it is more suitable to evaluate the top returned instances.

- ◇ *Average Precision (AP)* is a measure that considers the ranked positions of instances. By computing the precision and recall at every position in the ranking list, a precision-recall curve can be plotted, where $P(r)$ as a function of recall r. AP computes the average value of $P(r)$ over the interval from $r = 0$ to $r = 1$. For a ranked list of returned instances $l = s^1, ..., s^k, ..., s^n$, where the s^k is the instance ranked at k-th position, the AP is

$$AP(l) = \frac{\sum_{k=1}^{n} P@k \times \mathbb{1}[s^k \text{ is relevant}]}{\text{number of reference instances}}, \tag{2.11}$$

  where $P@k$ is the precision at cut-off k in the list and $\mathbb{1}$ is an indicator function equaling 1 if k-th ranked instance s^k is relevant, 0 otherwise.

- ◇ *Area Under the Curve (AUC)* computes average precision over a set of evenly spaced recall levels based on predicted scores, not from positions in the ranking list. The difference between AUC and AP is that AUC focuses on the predicted scores more while AP is more sensitive to the positions. For ranking problems, AP and AUC are used interchangeablely because both evaluate the ranking performance [18]. In addition, AUC can be measured as the area under the precision-recall curve or the receiver operating characteristic (ROC) curve.

When there are multiple ranking lists (e.g. multi-class labels, multiple queries), there are three common ways to average the values, *micro*, *macro*, and *weighted*.

micro calculates metrics globally by considering each element of the label indicator as a label.

macro calculates metrics for each label, and find their unweighted mean. This does not take label imbalance into account.

weighted calculates metrics for each label, and find their average, weighted by the importance of each label.

Following metrics are reported for measuring the whole performance.

- ◇ *Mean Average Precision (MAP)* is the unweighted mean (macro average) of AP scores for all lists $l_1, ..., l_q, ..., l_Q$.

$$MAP = \frac{1}{Q} \sum_{q=1}^{Q} AP(l_q), \tag{2.12}$$

where $AP(l_q)$ is AP performance of the q-th ranking list.

◇ *Weighted Average Precision (WAP)* is the weighted mean (weighted average) of AP scores for all lists $l_1, ..., l_q, ..., l_Q$.

$$WAP = \frac{\sum_{q=1}^{Q} w(l_q) \cdot AP(l_q)}{\sum_{q=1}^{Q} w(l_q)}, \tag{2.13}$$

where $AP(l_q)$ is AP performance of the q-th ranking list and $w(l_q)$ is the defined importance of the list q.

◇ *Micro F-Measure* is the global F-measure by considering all labels together. This is usually used for imbalanced data when the metric prefers to be biased towards the most populated ones.

In the following chapters, we basically evaluate the proposed approaches using above standard metrics. Multiple evaluation metrics are sometimes integrated to fit for some complicated cases.

3 Ontology Induction for Knowledge Acquisition

> *The deductive method is the mode of using knowledge, and the inductive method the mode of acquiring it.*
>
> Henry Mayhew, *English social researcher and journalist*

When building a dialogue system, a domain-specific knowledge base is required. To acquire the such knowledge, we allow a system to automatically induce an ontology for a certain domain from unlabeled dialogues by leveraging external resources (e.g. frame semantics and news articles). The chapter focuses on automatically extracting domain-specific concepts that can be used for building SDSs from unlabeled conversations.

3.1 Introduction

The distributional view of semantics hypothesizes that words occurring in the same contexts may have similar meanings [79]. Recently, with the advance of deep learning techniques, the continuous representations as word embeddings have further boosted the state-of-the-art results in many applications. Frame semantics, on the other hand, is a linguistic theory that has not been explored in the speech community, although there has been some successful applications in NLP [47, 81, 84]. The theory defines meaning as a coherent structure of related concepts, and provides an existing foundation for ontology induction [63].

Recently, a lot of prior work leveraged available semantic resources for unsupervised SLU [152, 154]. The major difference between our work and previous unsupervised studies is that, instead of leveraging the discrete representations of web search queries or knowledge graphs, we build our model on top of the recent success of deep learning—we utilize the continuous-valued word embeddings trained on Google News to induce semantic ontologies for task-oriented SDSs [117, 118, 119]. Our approach is clearly relevant to recent studies on deep learning for SLU. To the best of our knowledge, our work is the pioneering study that combines a distributional view of meaning from the deep learning community, and a linguistic frame semantic view for improved SLU [31].

The chapter focuses on using probabilistic frame-semantic parsing to automatically induce and adapt a semantic ontology for designing SDS in an unsupervised fashion [31, 50], alleviating some of the challenging problems for developing and maintaining SLU-based interactive systems [**?**]. Comparing to the traditional approach where domain experts and developers manually define the semantic ontology for SDS, the proposed approach has the advantages to reduce annotation effort, avoid human-induced biases, and lower maintenance cost [31].

Given unlabeled dialogues, we investigate an unsupervised approach for automatic induction of semantic slots, basic semantic units used in SDSs. To do this, we use a state-of-the-art probabilistic frame-semantic parsing approach [48], and perform an unsupervised approach to adapt, rerank, and map generic FrameNet-style semantic parses to the target semantic space that is suitable for domain-specific conversation settings [4]. We utilize continuous word embeddings trained on very large external corpora (e.g. Google News) to improve the adaptation process. To evaluate the performance of our approach, we compare the automatically induced semantic slots with the reference slots created by domain experts. Empirical experiments show that the slot creation results generated by our approach align well with those of domain experts. Our main contributions of this chapter are three-fold:

- We exploit continuous-valued word embeddings for unsupervised SLU;
- We propose the first approach of combining distributional and frame semantics for inducing a semantic ontology from unlabeled speech data;
- We show that this synergized method yields the state-of-the-art performance.

3.2 Proposed Framework

The main motivation of the work is to use a FrameNet-trained statistical probabilistic semantic parser to generate initial frame-semantic parses from ASR decodings of the raw audio conversation files. Then we adapt the FrameNet-style frame-semantic parses to the semantic slots in the target semantic space, so that they can be used practically in the SDSs. The semantic mapping and adaptation problem are formulated as a ranking problem, where the domain-specific slots should be ranked higher than the generic ones. The framework is illustrated in Figure 3.1. This thesis proposes the use of unsupervised clustering methods to differentiate generic semantic concepts from target semantic concepts for task-oriented dialogue systems [31]. Also, considering that using the small in-domain conversations as the training data may not be robust enough, this thesis proposes a radical extension: we aim at improving the semantic adaptation process by leveraging distributed word representations trained on very large external datasets [118, 119].

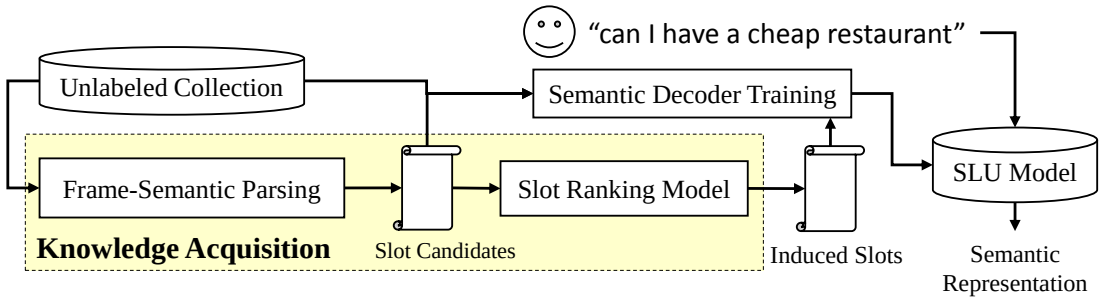


Figure 3.1: The proposed framework for ontology induction

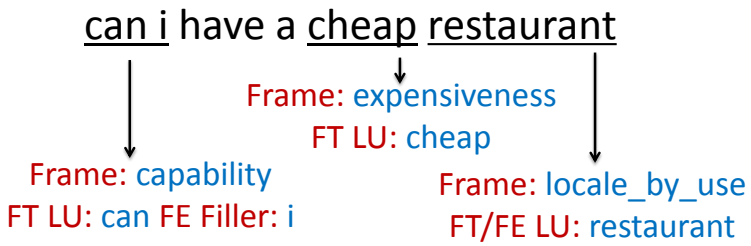


Figure 3.2: An example of probabilistic frame-semantic parsing on ASR output. FT: frame target. FE: frame element. LU: lexical unit.

3.2.1 Probabilistic Semantic Parsing

In our approach, we parse all ASR-decoded utterances in our corpus using SEMAFOR introduced in Section 2.3.2, and extract all frames from semantic parsing results as slot candidates and corresponding lexical units (LU) as slot-fillers. For example, Figure 3.2 shows an example of an ASR-decoded text output parsed by SEMAFOR. SEMAFOR generates three frames (capability, expensiveness, and locale_by_use) for the utterance, which we consider as slot candidates. Note that for each slot candidate, SEMAFOR also includes the corresponding lexical unit ("*can i*", "*cheap*", and "*restaurant*"), which we consider as possible slot-fillers.

Since SEMAFOR was trained on FrameNet annotation, which has a more generic frame-semantic context, not all the frames from the parsing results can be used as the actual slots in the domain-specific dialogue systems. For instance, in Figure 3.2, we see that the frames expensiveness and locale_by_use are essentially the key slots for the purpose of understanding in the restaurant query domain, whereas the capability frame does not convey particular valuable information for SLU in this domain. In order to fix this issue, we first compute the prominence of these slot candidates, then use a slot ranking model to rank the most important slots, and eventually generate a list of induced slots for use in domain-specific dialogue systems.

3.2.2 Independent Semantic Decoder

With outputted semantic parses, we extract the frames with the top 50 highest frequency as our slot candidates for training SLU in an unsupervised manner. The features for training are generated by word confusion network, where a confusion network features are shown to be useful in developing more robust systems for SLU [73]. We build a vector representation of an utterance as $\mathbf{u} = [x_1, ..., x_j, ...]$.

$$x_j = \mathbb{E}[C_u(n\text{-gram}_j)]^{1/|n\text{-gram}_j|}, \tag{3.1}$$

where $C_u(n\text{-gram}_j)$ counts how many times $n\text{-gram}_j$ occurs in the utterance u, $\mathbb{E}(C_u(n\text{-gram}_j))$ is the expected frequency of $n\text{-gram}_j$ in u, and $|n\text{-gram}_j|$ is the number of words in $n\text{-gram}_j$.

For each slot candidate s_i, we generate a set of pseudo training data $\mathcal{D}^i$ to train a binary classifier $\mathcal{M}^i$ for predicting the existence of s_i in an utterance, $\mathcal{D}^i = \{(\mathbf{u_k}, l_k^i) \mid \mathbf{u_k} \in \mathbb{R}^+, l_k^i \in \{-1, +1\}\}_{k=1}^K$, where $l_k^i = +1$ when the utterance u_k contains the slot candidate s_i in its semantic parse, $l_k^i = -1$ otherwise, and K is the number of utterances.

3.2.3 Adaptation Process and SLU Model

The generic concepts should be distinguished from the domain-specific concepts in the adaptation process. With the trained independent semantic decoders for all slot candidates, adaptation process computes the prominence of slot candidates for ranking and then selects a list of induced slots associated with their corresponding semantic decoders for use in domain-specific dialogue systems. Then with each induced slot s_i and its corresponding trained semantic decoder $\mathcal{M}^i$, an SLU model can be built to predict whether the semantic slot occurs in the given utterance in a fully unsupervised way. In other words, the SLU model is able to transform testing utterances into semantic representations without human involvement. The detail of the adaptation is described in the following section.

3.3 Slot Ranking Model

The purpose of the ranking model is to distinguish between generic semantic concepts and domain-specific concepts that are relevant to an SDS. To induce meaningful slots for the purpose of SDS, we compute the prominence of the slot candidates using a slot ranking model described below.

With the semantic parses from SEMAFOR, the model ranks the slot candidates by integrating

two scores [31, 33]: (1) the normalized frequency of each slot candidate in the corpus, with the assumption that slots with higher frequency may be more important; (2) the semantic coherence of slot-fillers corresponding to the slot. Assuming that domain-specific concepts focus on fewer topics, the coherence of the corresponding slot-fillers can help measure the prominence of the slots because they are similar to each other.

$$w(s) = (1 - \alpha) \cdot \log f(s) + \alpha \cdot \log h(s), \tag{3.2}$$

where $w(s)$ is the ranking weight for the slot candidate s, $f(s)$ is its normalized frequency from semantic parsing, $h(s)$ is its coherence measure, and α is the weighting parameter within the interval $[0, 1]$, which balances the frequency and coherence.

For each slot s, we have a set of corresponding slot-fillers, $V(s)$, constructed from the utterances including the slot s in the parsing results. The coherence measure of the slot s, $h(s)$, is computed as the average pair-wise similarity of slot-fillers to evaluate if slot s corresponds to centralized or scattered topics.

$$h(s) = \frac{\sum_{x_a, x_b \in V(s)} \mathrm{Sim}(x_a, x_b)}{|V(s)|^2}, \tag{3.3}$$

where $V(s)$ is the set of slot-fillers corresponding slot s, $|V(s)|$ is the size of the set, and $\mathrm{Sim}(x_a, x_b)$ is the similarity between the slot-filler pair x_a and x_b. The slot s with higher $h(s)$ usually focuses on fewer topics, which is more specific and more likely to be a slot for the dialogue system. The detail about similarity measure is introduced in the following section.

3.4 Word Representations for Similarity Measure

To capture the semantics of each word, we transform each token x into a corresponding vector $\mathbf{x}$ by following methods. Given that word representations can capture semantic meanings, the topical similarity between each slot-filler pair x_a and x_b can be computed as

$$\mathrm{Sim}(x_a, x_b) = \frac{\mathbf{x_a} \cdot \mathbf{x_b}}{\|\mathbf{x_a}\| \|\mathbf{x_b}\|}. \tag{3.4}$$

We assume that words occurring in similar domains have similar semantic representations, and thus $\mathrm{Sim}(x_a, x_b)$ will be larger when x_a and x_b are semantically related and their vectors should be close to each other in the topic space. To build the vector representations for words, we consider three techniques: in-domain clustering vector, in-domain embedding vector, and external embedding vector.

3.4.1 In-Domain Clustering Vector

Section 2.4 introduces distributional semantics, and a lot of studies have utilized the semantically-rich continuous word representations to benefit many NLP tasks. The in-domain data is used to cluster words through different clustering algorithms, such as K-means clustering [80], spectral clustering [122]. For each word x, we construct a vector $\mathbf{x} = [c_1, c_2, ..., c_K]$, where $c_i = 1$ when the word x is clustered into the i-th cluster, and $c_i = 0$ otherwise, and K is the number of clusters. The assumption is that topically similar words may be clustered together because they occur with the same contexts more frequently. Therefore, the cluster-based vectors that carry the such information can help measure similarity between words.

3.4.2 In-Domain Embedding Vector

Considering that continuous space word representations may capture more robust topical information, we leverage word embeddings trained on the in-domain data to involve distributional semantics of slot-fillers [119]. More specifically, to better adapt the FrameNet-style parses to the target task-oriented SDS domain, we make use of continuous word embeddings derived from a recurrent neural network architecture introduced in Section 2.4.1 [115]. Therefore, for all slot fillers, we build word embeddings as their word vectors for computing the coherence in (3.3).

3.4.3 External Embedding Vector

Since the distributional semantic theory may benefit our SLU task, we leverage word representations trained from large external data to better differentiate semantic concepts. The rationale behind applying the distributional semantic theory to our task is straight-forward: because spoken language is a very distinct genre comparing to the written language on which FrameNet is constructed, it is necessary to borrow external word representations to help bridge these two data sources for the unsupervised adaptation process. The word embeddings trained on the external data are able to capture both syntactic and semantic relations, which provide more robust relatedness information between words and may help distinguish the domain-specific information from the generic concepts compared to ones trained on the internal data [118, 119].

Table 3.1: The statistics of training and testing corpora.

	Train	Test	Total
Dialogue	1522	644	2166
Utterance	10571	4882	15453
Male : Female	28 : 31	15 : 15	43 : 46
Native : Non-Native	33 : 26	21 : 9	54 : 47
Avg. #Slot	0.959	0.952	0.957

3.5 Experiments

To evaluate the effectiveness of our induced slots, we performed two evaluations. First, we examine the slot induction accuracy by comparing the ranked list of frame-semantic parsing induced slots with the reference slots created by developers of the corresponding system [171]. Secondly, based on the ranked list of induced slots, we can train a semantic decoder for each slot to build an SLU component, and then evaluate the performance of our SLU model by comparing against the human annotated semantic forms. For the experiments, we evaluate both on ASR results of the raw speech, and on the manual transcripts.

3.5.1 Experimental Setup

We used the Cambridge In-Car SLU corpus[1], which has been used on several other SLU tasks such as supervised slot filling and dialogue act classification [30, 86]. The dialogue corpus was collected via a restaurant information system for Cambridge. Users can specify restaurant suggestions by area, price range and food type and can then query the system for additional restaurant specific information such as phone number, post code, signature dish and address [71, 148].

Subjects were asked to interact with multiple SDSs in an in-car setting [70, 148]. There were multiple recording settings for different noise conditions: 1) a stopped car with the air condition control on and off; 2) a driving condition; and 3) in a car simulator. The distribution of each condition in this corpus is uniform. The corpus contains a total number of 2,166 dialogues, and 15,453 utterances, which is separated into training and testing parts as shown in Table 3.1. The training part is for self-training the SLU model.

The data is gender-balanced, with slightly more native than non-native speakers. The vocabulary size is 1,868. An ASR system was used to transcribe the speech; the word error rate (WER) was reported as 37%. There are 10 slots created by domain experts: addr, area, food, name, phone, postcode, price range, signature, task and type.

[1] http://www.repository.cam.ac.uk/handle/1810/248271

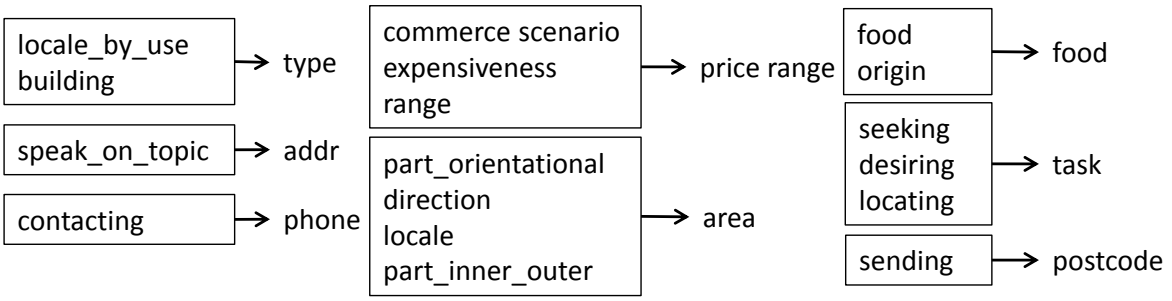


Figure 3.3: The mappings from induced slots (within blocks) to reference slots (right sides of arrows).

3.5.2 Implementation Detail

For clustering, we perform K-means clustering. The parameter K, the number of clusters, can be empirically set, where we use $K = 50$ for all experiments. For the parameter α in (2.7), we tune it by using a development set, which contains first 302 dialogues and total 2,515 utterances. For training independent semantic decoders, we apply support vector machine (SVM) with linear kernel to classify whether each utterance contain a semantic concept.

To include distributional semantics information for the external data, we use the distributed vectors trained on 10^9 words from Google News[2]. Training was performed using the CBOW architecture, which predicts the current word based on the context, with sub-sampling using threshold $1 \times e^{-5}$, and with negative sampling using 3 negative examples per each positive one. The resulting vectors have dimensionality 300, vocabulary size is 3×10^6; the entities contain both words and automatically derived phrases. The dataset provides a larger vocabulary and better coverage.

3.5.3 Evaluation Metrics

Our metrics take the entire list into account and evaluate the performance by the metrics that are independent on the selected threshold, in order to eliminate the influence of different thresholds when producing inducing induced slots.

3.5.3.1 Slot Induction

To evaluate the accuracy of the induced slots, we measure their quality as the proximity between induced slots and reference slots. Figure 3.3 shows the mappings that indicate semantically related induced slots and reference slots [31]. For example, expensiveness →

[2] `https://code.google.com/p/word2vec/`

Table 3.2: The performance with different α tuned on a development set (%).

Approach		ASR					Transcripts				
		α	Slot Induction		SLU Model		α	Slot Induction		SLU Model	
			AP	AUC	WAP	AF		AP	AUC	WAP	AF
(a)	Baseline	.0	58.17	56.16	35.39	36.76	.0	55.03	53.52	36.99	35.96
(b)	In. Cluster.	.4	64.76	63.55	40.54	37.28	.5	63.59	62.89	42.25	36.95
(c)	In. Embed.	.6	66.98	65.82	42.74	37.50	.4	57.96	56.51	39.78	36.61
(d)	Ex. Embed.	.8	**74.51**	**73.51**	**46.04**	**37.88**	.8	**64.99**	**64.17**	**43.28**	**38.57**
Max RI (%)		-	+39.9	+44.1	+32.9	+3.0	-	+18.1	+19.9	+18.0	+7.3

price, food $\rightarrow$ food, and direction $\rightarrow$ area show that these induced slots can be mapped into the reference slots defined by experts and carry important semantics in the target domain for developing the task-oriented SDS. Note that two slots, name and signature, do not have proper mappings, because they are too specific on restaurant-related domain, where name records the name of restaurant and signature refers to signature dishes. This means that the 80% recall is achieved by our approach because we consider all outputted frames as slot candidates. Since we define the adaptation task as a ranking problem, with a ranked list of induced slots and their associated scores, we can use the standard AP defined in (2.11) and AUC as our metrics, where the induced slot is counted as correct when it has a mapping to a reference slot.

3.5.3.2 SLU Model

Semantic slot induction is essential for providing semantic categories and imposing semantic constraints for SLU modeling. Therefore, we are also interested in understanding the performance of our unsupervised SLU models. For each induced slot with the mapping to a reference slot, we can compute an F-measure of the corresponding semantic decoder, and weight AP with corresponding F-measure as WAP defined in (2.13) to evaluate the performance of slot induction and SLU tasks together. The metric scores the ranking result higher if the induced slots corresponding to better semantic decoders are ranked higher. Another metric is the average F-measure (AF), which is the average micro F-measure of SLU models at all cut-off positions in the ranked list. Compared to WAP, AF additionally considers the slot popularity in the dataset.

3.5.4 Evaluation Results

Table 3.2 shows all results. The row (a) is the baseline, which only considers the frequency of slot candidates for ranking. It is found that the performance of slot induction for ASR is better than for manual results. The better AP and AUC scores of ASR results from biased

recognition results, which are optimized to recognize domain-specific words better. Thus the ASR results contain more accurate slot-fillers and the performance of slot induction is biased and better than the results on manual transcripts.

Rows (b)-(d) show performance after leveraging distributional word representations, in-domain clustering vector, in-domain embedding vector and external embedding vector. In terms of both slot induction and SLU modeling, we find that most results are improved by including distributed word information. With in-domain data (row (b) and row (c)), the performance of slot induction can be significantly improved, from 58% to 67% on AP and from 56% to 65% on AUC for ASR results, and from 55% to 64% on AP and from 54% to 63% on AUC for manual transcripts. Also, for SLU modeling, in-domain clustering and in-domain embedding approaches outperform the baseline. Using external data to train word embeddings (row (d)), the performance for both slot induction and SLU modeling is significantly improved for ASR and manual transcripts, which shows the effectiveness of involving external data for the similarity measurement. The reason may be that external word embeddings have more accurate vector representations to measure similarity because they are trained on the large data, while in-domain approaches rely on a small in-domain training set, which may be biased by the data and may be sensitive to recognition errors.

We see that leveraging distributional semantics with frame-semantic parsing produces promising slot ranking performance; this demonstrates the effectiveness of our proposed approaches for slot induction. The 72% of AP indicates that our proposed approach can generate good coverage for domain-specific slots in a real-world SDS, reducing labor cost of system development.

3.6 Discussion

3.6.1 Balance between Frequency and Coherence

To analyze the influence of coherence, we analyze the performance with different α values in (3.2) for slot induction and SLU modeling in Figure 3.4 and 3.5 respectively.

Figure 3.4 shows that the best performance of the in-domain clustering vector (blue line with circles) is when the frequency and coherence have similar weights for both ASR and manual transcripts. However, the in-domain embedding (red line with squares) and the external embedding (green line with triangles) perform best when $\alpha = 0.8, 0.9$, indicating that the coherence plays an important role for inducing slots and that the coherence is more reliable when our word representations are better.

Among three proposed approaches, the external embedding vector often performs better and

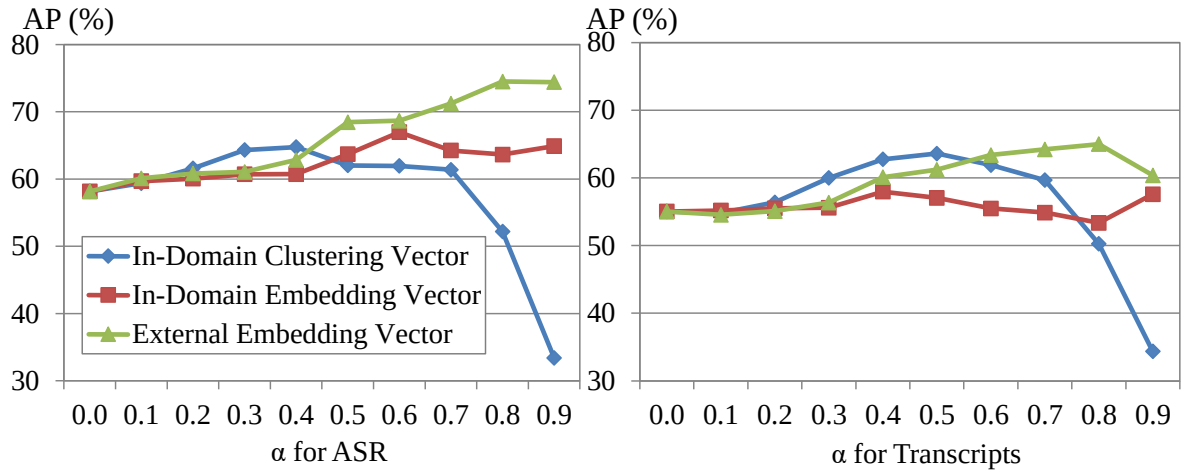


Figure 3.4: The performance of slot induction learned with different α values.

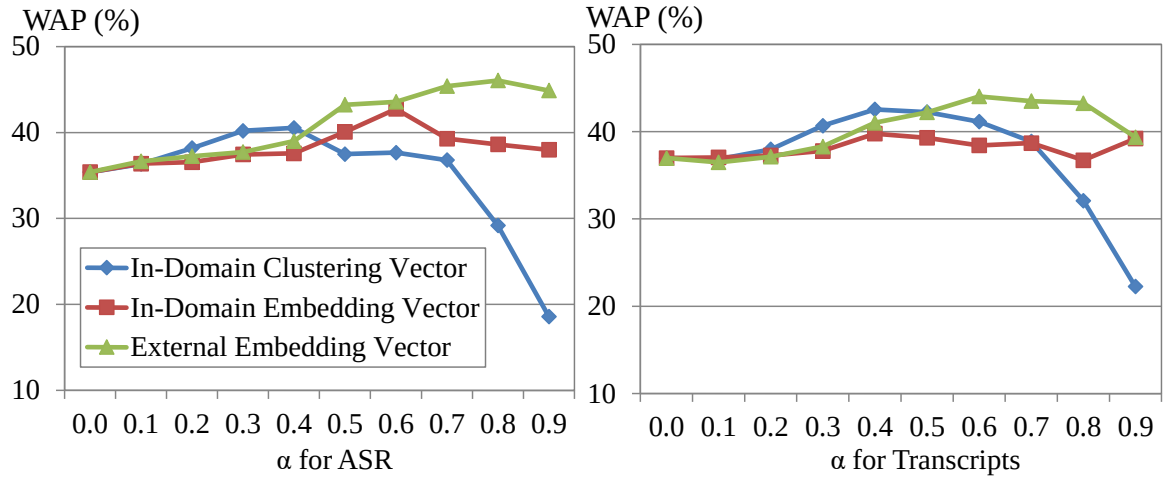


Figure 3.5: The performance of SLU modeling with different α values.

more stable, and the reason may be that word representations trained on the larger data carry more accurate semantic information for better coherence estimation. For SLU modeling, Figure 3.5 shows the similar trend as slot induction. In sum, when word representations are reliable, the coherence helps both slot induction and SLU modeling.

3.6.2 Sensitivity to Amount of Training Data

To further analyze the effectiveness of the proposed approach in a real world, we examine the performance of the induced slots learned from different amount of training data, where α is optimized and fixed for each approach. We compare the performance of all approaches using 20%, 40%, 60%, 80% and 100% amount of the training set in Figure 3.6. It can be found that

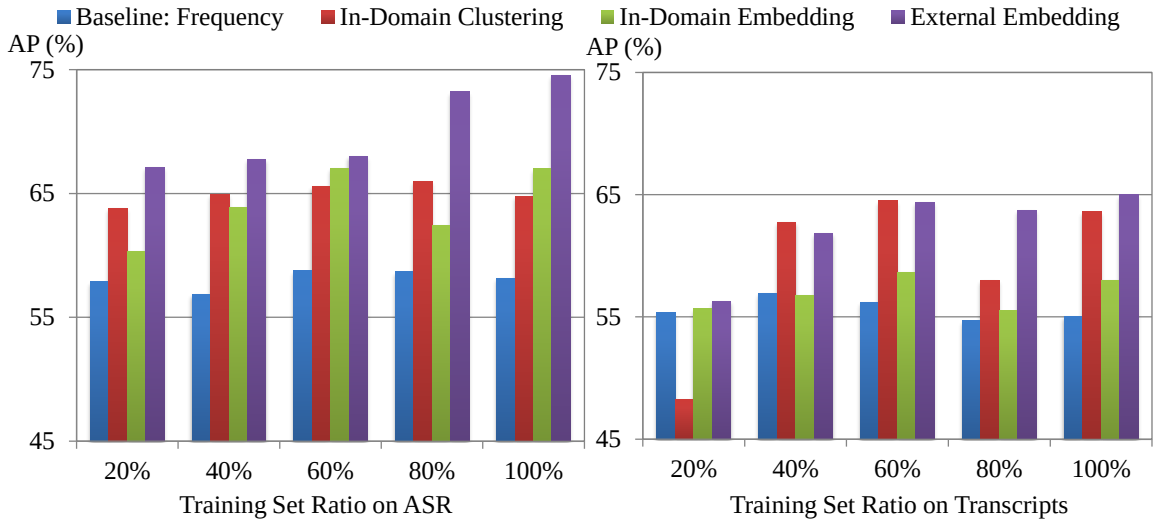


Figure 3.6: The performance of slot induction learned from different amount of training data.

the baseline approach is insensitive to the data size, probably because the slot distribution is similar across dialogues. However, the best approach that uses external embeddings shows higher sensitivity to the training amount, where at least 40% training data can achieve higher than 55% of AP. In conclusion, scarce training data may not cover a complete set of important slot-fillers, so that the approach produces less reliable coherence measurements and is more sensitive to the amount of training data.

3.7 Summary

This chapter proposes the first unsupervised approach unifying distributional and frame semantics for domain ontology induction. Our work makes use of a state-of-the-art semantic parser, and adapts the generic linguistic FrameNet representation to a semantic space characteristic of a domain-specific SDS. With the incorporation of distributional word representations, we show that our automatically induced semantic slots align well with reference slots, yielding the state-of-the-art performance. Also, we demonstrate that it is feasible for the automatically induced ontology to benefit SLU tasks. The automating process of ontology induction reduces the cost of human annotations, speeding up SDS development.

4 Structure Learning for Knowledge Acquisition

> *A scene has to have a rhythm of its own, a structure of its own.*
>
> Michelangelo Antonioni, *Academy Award winner*

Ontology induction extracts the domain-specific concepts, but the induced information is flat and unstructured. Assuming that a well-organized ontology may help understanding, inter-slot relations, which represent dependencies between semantic concepts, should be considered for organizing the domain knowledge. In order to acquire the relations, a structure learning approach is introduced in this chapter.

4.1 Introduction

From a engineering perspective, quick and easy development turnaround time for domain-specific dialogue applications is critical [28]. From a knowledge management perspective, empowering dialogue systems with large knowledge bases is of crucial significance to modern SDSs. Our work that builds on top of frame-semantic parsing clearly aligns with recent studies on leveraging semantic knowledge resources for SLU, which combines two above perspectives [32, 57, 75, 76, 83]. However, an important requirement for building a successful SDS is to define a coherent slot set and the associated slot-fillers for an SLU component. Unfortunately, since the semantic slots are often mutually-related, it is not easy for domain experts and professional annotators to design a such slot set for better semantic representation of SLU.

It is challenging to manually design such a coherent and complete slot set, while considering various lexical, syntactic, and semantic dependencies in the mean time [40, 86]. Take the restaurant domain for example, "*restaurant*" is the target slot, and important adjective modifiers such as "*Asian*" (a restaurant type) and "*cheap*" (the price of food in the restaurant) should be included in the slot set, so that the semantic representation of SLU can be more coherent and complete.

However, most work treats each slot independently and have not considered the inter-slot relations when inducing the semantic slots. Assuming that a well-organized ontology may benefit a better SLU component construction and the system development, the semantic structure of the ontology should be included when inducing the domain knowledge. Due to the importance of lexical knowledge and structure information, we first use syntactically-informed random walk algorithms to combine the semantic and lexical knowledge graphs, and globally inducing the semantic slots for building better unsupervised SLU components.

Specifically, instead of considering slots independently, this chapter takes a data-driven approach to model word-to-word relations via syntactic dependencies and further infer slot-to-slot relations. To do this, we incorporate the typed dependency grammar theory in a state-of-the-art, frame-semantic driven, and unsupervised slot induction framework [31, 51]. In particular, we build two knowledge graphs: a slot-based semantic knowledge graph and a word-based lexical knowledge graph with typed dependency triples. We then study stochastic relations between slots and words, using a mutually-reinforced random walk inference procedure to combine these two knowledge graphs. To produce a structured ontology, we use the jointly learned inter-slot relations to induce a coherent slot set in an unsupervised fashion. Our contributions in this chapter are three-fold:

- We are among the first to combine semantic and lexical knowledge graphs for unsupervised SLU;
- We propose a novel typed syntactic dependency grammar driven random walk model for relation discovery;
- Our experimental results suggest that jointly considering inter-slot relations helps obtain a more coherent and complete semantic slot set, showing that the ontology structure is essential to build a better SLU component.

4.2 The Proposed Framework

The approach is built on top of the success of an unsupervised frame-semantic parsing approach introduced in Chapter 3 [31]. The main motivation is to use a FrameNet-trained statistical probabilistic semantic parser to generate initial frame-semantic parses from ASR decodings of the raw conversations, and then adapt the FrameNet-style frames to the semantic slots in the target semantic space, so that they can be used practically in the SDSs. In stead of inducing an unstructured ontology, this chapter improves the adaptation process by leveraging distributed word embeddings associated with typed syntactic dependencies between words to infer inter-slot relations in order to learn a well-organized ontology [108, 118, 119].

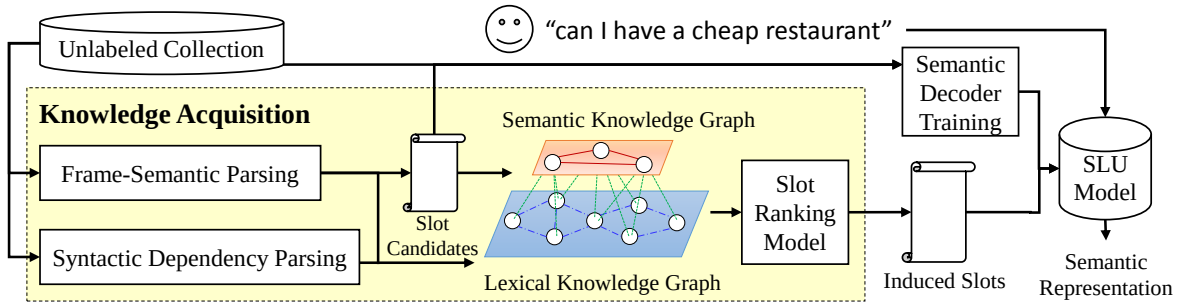


Figure 4.1: The proposed framework of structure learning.

The proposed framework is shown in Figure 4.1. Frame-semantic parsing, independent semantic decoder, and adaptation process are similar to ones from Chapter 3, except that the slot ranking model does consider the relation information. Finally, we build an SLU model based on the learned semantic decoders and induced slots to evaluate whether the ontology structure helps SLU modeling.

4.3 Slot Ranking Model

The goal of the ranking model is to extract domain-specific concepts from all fine-grained frames outputted by frame-semantic parsing. To induce meaningful slots for the purpose of SDS, we compute the prominence of slot candidates by additionally considering their structure information.

With the semantic parses from SEMAFOR, where each frame is viewed independently and inter-slot relations are not included, our model ranks slot candidates by integrating two information: (1) the frequency of each slot candidate in the corpus, and (2) the relations between slot candidates. Assuming that domain-specific concepts are usually related to each other, globally considering inter-slot relations induces a more coherent slot set. As the baseline in Chapter 3, we consider only the frequency of each slot candidate as its prominence without the structure information.

Since syntactic dependency relations between fillers may help measure the prominence of corresponding slots. First we construct two knowledge graphs, one is a slot-based semantic knowledge graph and another is a word-based lexical knowledge graph, both of which encode the typed dependency relations in their edge weights. We also connect two graphs to model the relations between slot-filler pairs. The integrated graph that incorporates dependency relations of semantic and lexical elements can compute the prominence of slot candidates by a random walk algorithm. The details are described as follows.

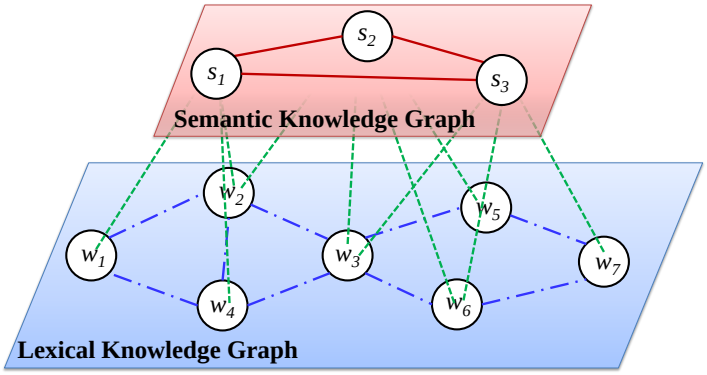


Figure 4.2: A simplified example of the integration of two knowledge graphs, where a slot candidate s_i is represented as a node in a semantic knowledge graph and a word w_j is represented as a node in a lexical knowledge graph.

4.3.1 Knowledge Graphs

We construct two undirected graphs, semantic and lexical knowledge graphs. Each node in the semantic knowledge graph is a slot candidate s_i outputted by the frame-semantic parser, and each node in the lexical knowledge graph is a word w_j.

- **Slot-based semantic knowledge graph** is built as $G_s = \langle V_s, E_{ss} \rangle$, where $V_s = \{s_i\}$ and $E_{ss} = \{e_{ij} \mid s_i, s_j \in V_s\}$.
- **Word-based lexical knowledge graph** is built as $G_w = \langle V_w, E_{ww} \rangle$, where $V_w = \{w_i\}$ and $E_{ww} = \{e_{ij} \mid w_i, w_j \in V_w\}$.

With two knowledge graphs, we build edges between slots and slot-fillers to integrate them as shown in Figure 4.2. Thus the integrated graph can be formulated as $G = \langle V_s, V_w, E_{ss}, E_{ww}, E_{ws} \rangle$, where $E_{ws} = \{e_{ij} \mid w_i \in V_w, s_j \in V_s\}$. E_{ss}, E_{ww}, and E_{ws} correspond to slot-to-slot relations, word-to-word relations, and word-to-slot relations respectively [26, 27].

4.3.2 Edge Weight Estimation

To incorporate different strengths of dependency relations in the knowledge graphs, we assign weights for edges. The edge weights for E_{ww} and E_{ss} are measured based on the dependency parsing results. The example utterance "*can i have a cheap restaurant*" and its dependency parsing result are illustrated in Figure 4.3. The arrows denote the dependency relations from headwords to their dependents, and words on arcs denote types

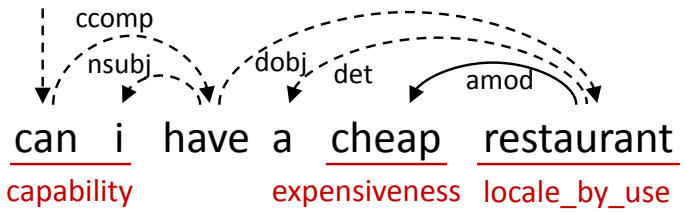


Figure 4.3: The dependency parsing result on an utterance.

of dependencies. All typed dependencies between two words are encoded in triples and form a word-based dependency set $\mathcal{T}_w = \{\langle w_i, t, w_j \rangle\}$, where t is the typed dependency between the headword w_i and the dependent w_j. For example, Figure 4.3 generates $\langle$*restaurant*, AMOD, *cheap*$\rangle$, $\langle$*have*, DOBJ, *restaurant*$\rangle$, etc. for $\mathcal{T}_w$. Similarly, we build a slot-based dependency set $\mathcal{T}_s = \{\langle s_i, t, s_j \rangle\}$ by transforming dependencies between slot-fillers into ones between slots. For example, $\langle$*restaurant*, AMOD, *cheap*$\rangle$ from $\mathcal{T}_w$ is transformed into $\langle$locale_by_use, AMOD, expensiveness$\rangle$ for building $\mathcal{T}_s$, because both sides of the non-dotted line are parsed as slot-fillers by SEMAFOR.

For all edges within a single knowledge graph, we assign the weight of the edge connecting nodes x_i and x_j as $\hat{r}(x_i, x_j)$, where x is either s (slot) or w (word). Since weights are measured based on relations between nodes regardless of directions, we combine the scores for two directional dependencies:

$$\hat{r}(x_i, x_j) = r(x_i \rightarrow x_j) + r(x_j \rightarrow x_i), \tag{4.1}$$

where $r(x_i \rightarrow x_j)$ is the score that estimates the dependency including x_i as a head and x_j as a dependent. In Section 4.3.2.1 and 4.3.2.2, we propose two scoring functions for $r(\cdot)$, frequency-based as $r_1(\cdot)$ and embedding-based as $r_2(\cdot)$ respectively.

For edges of E_{ws}, we estimate edge weights based on the frequency that slot candidates and words are parsed as slot-filler pairs. In other words, the edge weight between a slot-filler w_i and a slot candidate s_j, $\hat{r}(w_i, s_j)$, is equal to how many times the filler w_i corresponds to the slot candidate s_j in the parsing results.

4.3.2.1 Frequency-Based Measurement

Based on the parsed dependency set $\mathcal{T}_x$, we use $t^*_{x_i \rightarrow x_j}$ to denote the most frequent typed dependency with x_i as a head and x_j as a dependent.

$$t^*_{x_i \rightarrow x_j} = \arg\max_t C(x_i \underset{t}{\rightarrow} x_j), \tag{4.2}$$

Table 4.1: The contexts extracted for training dependency-based word/slot embeddings from the utterance of Figure 3.2.

	Typed Dependency Relation	Target Word	Contexts
Word	$\langle$*restaurant*, AMOD, *cheap*$\rangle$	*restaurant* *cheap*	*cheap*/AMOD *restaurant*/AMOD^{-1}
Slot	$\langle$locale_by_use, AMOD, expensiveness$\rangle$	locale_by_use expansiveness	expensiveness/AMOD locale_by_use/AMOD^{-1}

where $C(x_i \underset{t}{\rightarrow} x_j)$ counts how many times the dependency $\langle x_i, t, x_j \rangle$ occurs in the dependency set $\mathcal{T}_x$. Then the scoring function that estimates the dependency $x_i \rightarrow x_j$ is measured as

$$r_1(x_i \rightarrow x_j) = C(x_i \xrightarrow[t^*_{x_i \rightarrow x_j}]{} x_j), \tag{4.3}$$

which equals to the highest observed frequency of the dependency $x_i \rightarrow x_j$ among all types from $\mathcal{T}_x$.

4.3.2.2 Embedding-Based Measurement

It is shown that a dependency-based embedding approach introduced in Section 2.4.2 is able to capture more functional similarity because it uses dependency-based syntactic contexts for training word embeddings [108]. Table 4.1 shows some extracted dependency-based contexts for each target word from the example in Figure 4.3, where headwords and their dependents can form the contexts by following the arc on a word in the dependency tree, and -1 denotes the directionality of the dependency. We learn vector representations for both words and contexts such that the dot product $\mathbf{v_w} \cdot \mathbf{v_c}$ is maximized when they are associated with "good" word-context pairs belonging to the training data.

Then we can obtain the dependency-based slot and word embeddings using $\mathcal{T}_s$ and $\mathcal{T}_w$ respectively.

With trained dependency-based embeddings, we estimate the probability that x_i is a headword and x_j is its dependent via a typed dependency t as

$$P(x_i \underset{t}{\rightarrow} x_j) = \frac{Sim(x_i, x_j/t) + Sim(x_j, x_i/t^{-1})}{2}, \tag{4.4}$$

where $Sim(x_i, x_j/t)$ is the cosine similarity between word/slot embeddings $\mathbf{v_{x_i}}$ and context embeddings $\mathbf{v_{x_j/t}}$ after normalizing to $[0, 1]$. Then we can measure the scoring function $r_2(\cdot)$ as

$$r_2(x_i \rightarrow x_j) = C(x_i \xrightarrow[t^*_{x_i \rightarrow x_j}]{} x_j) \cdot P(x_i \xrightarrow[t^*_{x_i \rightarrow x_j}]{} x_j), \tag{4.5}$$

which is similar to (4.3) but additionally weighted with the estimated probability. The estimated probability smooths the observed frequency to avoid overfitting due to the smaller dataset.

4.3.3 Random Walk Algorithm

We first compute $L_{ww} = [\hat{r}(w_i, w_j)]_{|V_w|\times|V_w|}$ and $L_{ss} = [\hat{r}(s_i, s_j)]_{|V_s|\times|V_s|}$, where $\hat{r}(w_i, w_j)$ and $\hat{r}(s_i, s_j)$ are either from frequency-based ($r_1(\cdot)$) or embedding-based measurements ($r_2(\cdot)$). Similarly, $L_{ws} = [\hat{r}(w_i, s_j)]_{|V_w|\times|V_s|}$ and $L_{sw} = [\hat{r}(w_i, s_j)]^T_{|V_w|\times|V_s|}$ are computed, where $\hat{r}(w_i, s_j)$ is the frequency that s_j and w_i are a slot-filler pair computed in Section 4.3.2. Then we only keep the top N highest weights for each row in L_{ww} and L_{ss} ($N = 10$), which means that we filter out edges with smaller weights within a single knowledge graph. Column-normalization are performed for L_{ww}, L_{ss}, L_{ws}, L_{sw} [141]. They can be viewed as word-to-word, slot-to-slot, and word-to-slot relation matrices.

4.3.3.1 Single-Graph Random Walk

Here we perform a random walk algorithm only on the semantic knowledge graph to propagate scores based on inter-slot relations through the edges E_{ss}.

$$R_s^{(t+1)} = (1-\alpha)R_s^{(0)} + \alpha L_{ss} R_s^{(t)}, \tag{4.6}$$

where $R_s^{(t)}$ denotes importance scores of slot candidates V_s in t-th iteration. In the algorithm, the score is the interpolation of two scores, the normalized baseline importance of slot candidates ($R_s^{(0)}$), and scores propagated from the neighboring nodes in the semantic knowledge graph based on slot-to-slot relations L_{ss}. The algorithm will converge when $R_s^* = R_s^{(t+1)} \approx R_s^{(t)}$ and R_s^* satisfies the equation,

$$R_s^* = (1-\alpha)R_s^{(0)} + \alpha L_{ss} R_s^*. \tag{4.7}$$

We can solve R_s^* as

$$R_s^* = \left((1-\alpha)R_s^{(0)} e^T + \alpha L_{ss}\right) R_s^* = M_1 R_s^*, \tag{4.8}$$

where the $e = [1, 1, ..., 1]^T$. It has been shown that the closed-form solution R_s^* of (4.8) is the dominant eigenvector of M_1, or the eigenvector corresponding to the largest absolute eigenvalue of M_1 [100]. The solution of R_s^* denotes the updated importance scores for all utterances. Similar to the PageRank algorithm, the solution can also be obtained by iteratively updating $R_s^{(t)}$ [19].

4.3.3.2 Double-Graph Random Walk

We borrow the idea from two-layer mutually reinforced random walk to propagate scores based on not only internal importance propagation within the same graphs but also external mutual reinforcement between different knowledge graphs [26, 27, 93].

$$\begin{cases} R_s^{(t+1)} = (1-\alpha)R_s^{(0)} + \alpha L_{ss} L_{sw} R_w^{(t)} \\ R_w^{(t+1)} = (1-\alpha)R_w^{(0)} + \alpha L_{ww} L_{ws} R_s^{(t)} \end{cases} \tag{4.9}$$

In the algorithm, they are the interpolations of two scores, the normalized baseline importance ($R_s^{(0)}$ and $R_w^{(0)}$) and the scores propagated from another graph. For the semantic knowledge graph, $L_{sw}R_w^{(t)}$ is the score from the word set weighted by slot-to-word relations, and then the scores are propagated based on slot-to-slot relations L_{ss}. Similarly, nodes in the lexical knowledge graph also include scores propagated from the semantic knowledge graph. Then $R_s^{(t+1)}$ and $R_w^{(t+1)}$ can be mutually updated by the latter parts in (4.9) iteratively. When the algorithm converges, R_s^* and R_w^* can be derived similarly.

$$\begin{cases} R_s^* = (1-\alpha)R_s^{(0)} + \alpha L_{ss} L_{sw} R_w^* \\ R_w^* = (1-\alpha)R_w^{(0)} + \alpha L_{ww} L_{ws} R_s^* \end{cases} \tag{4.10}$$

$$\begin{aligned} R_s^* &= (1-\alpha)R_s^{(0)} + \alpha L_{ss} L_{sw} \left((1-\alpha)R_w^{(0)} + \alpha L_{ww} L_{ws} R_s^* \right) \\ &= (1-\alpha)R_s^{(0)} + \alpha(1-\alpha) L_{ss} L_{sw} R_w^{(0)} + \alpha^2 L_{ss} L_{sw} L_{ww} L_{ws} R_s^* \\ &= \left((1-\alpha)R_s^{(0)} e^T + \alpha(1-\alpha) L_{ss} L_{sw} R_w^{(0)} e^T + \alpha^2 L_{ss} L_{sw} L_{ww} L_{ws} \right) R_s^* \\ &= M_2 R_s^*. \end{aligned} \tag{4.11}$$

The closed-form solution R_s^* of (4.11) can be obtained from the dominant eigenvector of M_2.

4.4 Experiments

The goal of the experiments is to validate the effectiveness of including the structure information for SLU modeling. We evaluate our approach in two ways. First, we examine slot induction performance by comparing the ranking list of induced slots with the reference slots created by system developers [171]. Second, with the ranked list of induced slots and their associated semantic decoders, we evaluate the SLU performance. In the following experiments, we evaluate performance on both ASR transcripts and manual transcripts.

Table 4.2: The performance of induced slots and corresponding SLU models (%)

Approach			ASR				Transcripts			
			Slot Induction		SLU Model		Slot Induction		SLU Model	
			AP	AUC	WAP	AF	AP	AUC	WAP	AF
(a)	Baseline ($\alpha = 0$)		56.69	54.67	35.82	43.28	53.01	50.80	36.78	44.20
(b)	Single	Freq.	63.88	62.05	41.67	47.38	63.02	61.10	43.76	48.53
(c)		Embed.	69.04	68.25	**46.29**	**48.89**	75.15	74.50	**54.50**	**50.86**
(d)	Double	Freq.	56.83	55.31	32.64	44.91	52.12	50.54	34.01	45.05
(e)		Embed.	**71.48**	**70.84**	44.06	47.91	**76.42**	**75.94**	52.89	50.40

4.4.1 Experimental Setup

The data is the Cambridge University SLU corpus described in the previous chapter. For the parameter setting, the damping factor for random walk α is empirically set as 0.9 for all experiments[1]. For training semantic decoders, we use SVM with linear kernel to predict the probability of each semantic slot. We use the Stanford Parser to obtain the collapsed typed syntactic dependencies and set the dimensionality of embeddings $d = 300$ in all experiments [143].

For evaluation, we measure their quality as the proximity between induced slots and reference slots. Figure 3.3 shows the mappings between induced slots and reference slots [31]. As the metrics in Chapter 3, we use AP and AUC for evaluating slot induction, and WAP and AF for evaluating slot induction and SLU tasks together.

4.4.2 Evaluation Results

Table 4.2 shows results on both ASR and transcripts. The row (a) is the baseline considering only the frequency of each slot candidate for ranking. Rows (b) and (c) show performance after leveraging a semantic knowledge graph through random walk. Rows (d) and (e) are results after combining two knowledge graphs. We find that almost all results are improved by additionally considering inter-slot relations in terms of single- and double-graph random walk for both ASR and manual transcripts.

4.4.2.1 Slot Induction

For both ASR and manual transcripts, almost all results outperform the baseline, which shows that inter-slot relations significantly influence the performance of slot induction. The best performance is from results using double-graph random walk with the embedding-based

[1]The performance is different from results in Chapter 3 since we do not need a dev set in the experiments.

measurement, which integrate a semantic knowledge graph and a lexical knowledge graph together and jointly consider slot-to-slot, word-to-word, and word-to-slot relations when scoring the prominence of slot candidates to generate a coherent slot set.

4.4.2.2 SLU Model

For both ASR and manual transcripts, almost all results outperform the baseline, which shows the practical usage for training dialogue systems. The best performance is from the results of single-graph random walk with embedding-based measurement, which only use the semantic knowledge graph to involve inter-slot relations. The semantic knowledge graph is not as precise as the lexical one and may be influenced more by the performance of the semantic parser. Although the row (e) does not show better performance than the row (c), double-graph random walk may be more robust because it additionally includes word relations to avoid from relying only on relations tied with slot candidates.

4.4.3 Discussion and Analysis

4.4.3.1 Comparing Frequency- and Embedding-Based Measurements

Table 4.2 shows that all results with the embedding-based measurement perform better than ones with frequency-based measurement. The frequency-based measurement also brings large improvement for single-graph approaches, but not for double-graph ones. The reason is probably that using observed frequencies in the lexical knowledge graph may result in overfitting issues due to the smaller dataset. Additionally incorporating embedding information can smooth edge weights and deal with data sparsity to improve the performance, especially for the lexical knowledge graph.

4.4.3.2 Comparing Single- and Double-Graph Approaches

Considering that the embedding-based measurement performs better, we only compare results of single- and double-graph random walk using the measurement (rows (c) and (e)). It can be seen that the difference between them is not consistent in terms of slot induction and SLU modeling.

For evaluating slot induction (AP and AUC), double-graph random walk (row (e)) performs better on both ASR and manual results, which implies that additionally integrating the lexical knowledge graph helps decide a more coherent and complete slot set because we can model the score propagation more precisely (not only slot-level but word-level information).

Table 4.3: The top inter-slot relations learned from the training set of ASR outputs.

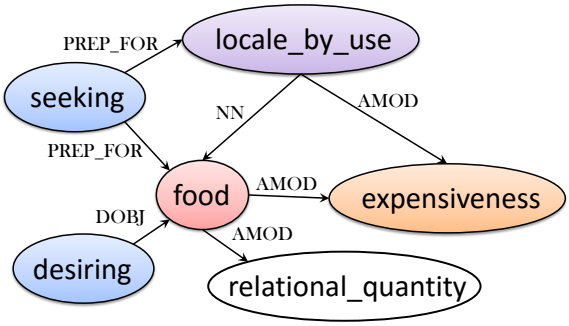
Rank	Relation
1	⟨locale_by_use, NN, food⟩
2	⟨food, AMOD, expensiveness⟩
3	⟨locale_by_use, AMOD, expensiveness⟩
4	⟨seeking, PREP_FOR, food⟩
5	⟨food, AMOD, relational_quantity⟩
6	⟨desiring, DOBJ, food⟩
7	⟨seeking, PREP_FOR, locale_by_use⟩
8	⟨food, DET, quantity⟩

However, for SLU evaluation (WAP and AF), single-graph random walk (row (c)) performs better, which may imply that the slots carrying coherent relations from the row (e) may not have good semantic decoder performance so that the performance is decreased a little. For example, double-graph random walk scores the slots local_by_use and expensiveness higher than the slot contacting, while the single-graph method ranks the latter higher. The slots, local_by_use and expensiveness, are more important on this domain but contacting has very good performance of its semantic decoder, so the double-graph approach does not show the improvement when evaluating SLU. This allows us to try an improved method of jointly optimizing the slot coherence and SLU performance in the future.

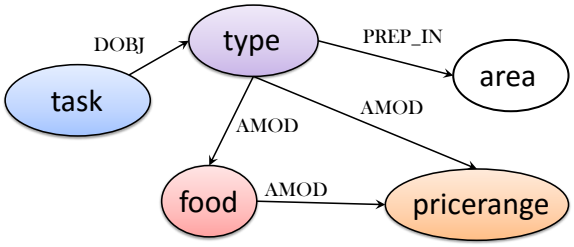
4.4.3.3 Relation Discovery Analysis

To interpret inter-slot relations, we output the relations that connect slots with highest scores from the best results (row (e)) in Table 4.3. It can be shown that the outputted inter-slot relations are reasonable and usually connect two important semantic slots. The automatically learned structure is able to construct a corresponding slot-based semantic knowledge graph as Figure 4.4a.

To evaluate the performance of the automatically learned knowledge graph, we also construct a semantic knowledge graph based on human-annotated data, where the shown relations are the most frequent typed dependencies between two slots in the domain-expert annotated data. The reference knowledge graph is shown in Figure 4.4b. Here we can clearly see similar structures between the generated one and the reference one, where the nodes with same colors represent semantically similar concepts. This proves that inter-slot relations help decide a coherent and complete slot set and enhance the interpretability of semantic slots. Thus, from a practical perspective, developers are able to design the framework of dialogue systems more easily.



(a) A simplified example of the automatically derived knowledge graph.



(b) The reference knowledge graph.

Figure 4.4: The automatically and manually created knowledge graphs for a restaurant domain.

4.5 Summary

The chapter proposes an approach of considering inter-slot relations for slot induction to output a more coherent slot set, where two knowledge graphs, a slot-based semantic knowledge graph and a word-based lexical knowledge graph, are built and jointly modeled by a random walk algorithm. The automatically induced slots carry coherent and interpretable relations and can be used for better understanding, showing that relation information helps SLU modeling.

5 Surface Form Derivation for Knowledge Acquisition

> *From principles is derived probability, but truth or certainty is obtained only from facts.*
>
> Tom Stoppard, *Academy Award and Tony Award winner*

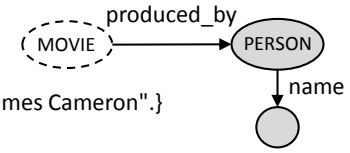
With the available structured ontology, the domain knowledge can be learned for building a domain-specific dialogue system. However, entities in the ontology have various surface forms, for example, in a movie domain, "*movie*" and "*film*" can be used to refer to the same entity of an ontology. This chapter focuses on deriving surface forms and shows that the derived surface forms can benefit SLU modeling performance.

5.1 Introduction

An SLU component aims to detect semantic frames that include domain-related information. Traditional SDSs are trained with annotated examples and support limited domains. Recent studies utilized structured semantic knowledge such as Freebase, FrameNet, etc. to obtain domain-related knowledge and help SLU for tackling open domain problems in SDSs [3, 15, 31, 33, 76, 82, 83].

Knowledge graphs, such as Freebase, usually carry rich information for named entities, which is encoded in triples of entity pairs and their relations. Such information is usually used for interpretation of natural language in SDSs [76, 89, 152]. However, the entity lists/gazatteers may bring noises and ambiguity to SLU; for example, the commonly used words "*Show me*" and "*Up*" can be movie names, and "*Brad Pitt*" can be an actor name or a producer name, which makes interpretation more difficult. Some work focused on assigning weights for entities or entity types to involve prior background knowledge of entities, where the probabilistic confidences offer better cues for SLU [89, 76]. Also, a lot of work focused on mining natural language forms based on the ontology by web search or query click logs, which benefit discovering new relation types from large text corpora [75, 74]. The mined data can also be used to help SLU by adaptation from the text domain to the spoken domain [83].

User Utterance:
find movies produced by james cameron
SPARQL Query (simplified):
SELECT ?movie {?movie. ?movie.produced_by?producer. ?producer.name"James Cameron".}
Logical Form:
λx. ∃y. movie.produced_by(x, y) ∧ person.name(y, z) ∧ z="James Cameron"
Relation:
movie.produced_by producer.name



User Utterance:
who produced avatar
SPARQL Query (simplified):
SELECT ?producer {?movie.name"Avatar". ?movie.produced_by?producer.}
Logical Form:
λy. ∃x. movie.produced_by(x, y) ∧ movie.name(x, z) ∧ z="Avatar"
Relation:
movie.name movie.produced_by

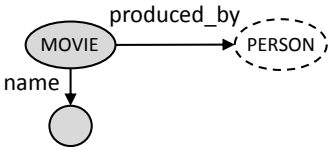


Figure 5.1: The relation detection examples.

On the other hand, the distributional view of semantics hypothesizes that words occurring in the same contexts may have similar meanings, and words can be represented as high dimensional vectors [119, 118, 117, 169, 16, 17]. Furthermore, dependency-based word embeddings were proposed to capture more functional similarity ,based on the dependency-based contexts instead of the linear contexts using a similar training procedure [108].

Following the successes brought by word embeddings, we leverage dependency-based entity embeddings to learn relational information including entity surface forms and entity contexts from the text data. We further integrate derived relational information as local cues and gazetteers as background knowledge to improve the performance for relation detection in a fully unsupervised fashion.

5.2 Knowledge Graph Relation

Given an utterance, we can form a set of relations that encode user intents for informational queries based on the semantic graph ontology. Figure 5.1 presents two user utterances and their invoked relations, which can be used to create requests in query languages (i.e., SPARQL Query Language for RDF[1]). Two examples in this figure include two nodes and the same relation `movie.produced_by`, and we differentiate these examples by including the originating node types in the relation (`movie.name`, `producer.name`) instead of just plain names (the nodes with gray color denote specified entities). Given all these, this task is richer than the regular relation detection task. The motivation to do that is, since we are trying to write queries to the knowledge source, we need to make sure that the queries are well-

[1] `http://www.w3.org/TR/ref-sparql-query/`

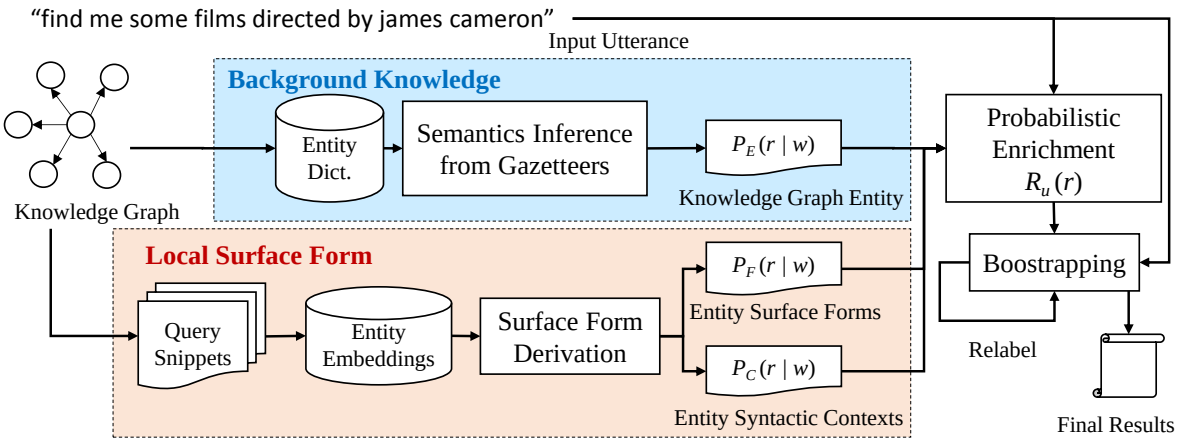


Figure 5.2: The proposed framework of surface form derivation.

formed; and relation arcs originate from the correct nodes in them. Therefore, this chapter focuses on detecting not only the relation `movie.produced_by` but also the specified entities `producer.name` and `movie.name`, so that we can obtain a better understanding of user utterances.

5.3 Proposed Framework

The whole system framework is shown in Figure 5.2. There are two major components: 1) we first utilize background knowledge as a prior to infer relations, and 2) we capture natural language surface forms for detecting local observations, which are described in Sections 5.4 and Section 5.5 respectively. Then probabilistic enrichment is used to integrate probabilistic information from background knowledge and local relational observations given the input utterance. Finally, an unsupervised learning approach is proposed to boost the performance. The detail is presented in Section 5.6.

5.4 Relation Inference from Gazetteers

Due to ambiguity about entity mentions, we utilize prior knowledge from gazetteers to estimate the probability distribution of associated relations for each entity [76]. For example, "*James Cameron*" can be a director or a producer, which infers `movie.directed_by` or `movie.produced_by` relations respectively. Given a word w_j, the estimated probability of an inferred relation r_i is defined as

$$P_E(r_i \mid w_j) = P_E(t_i \mid w_j) = \frac{C(w_j, t_i)}{\sum_{t_k \in T(w_j)} C(w_j, t_k)}, \tag{5.1}$$

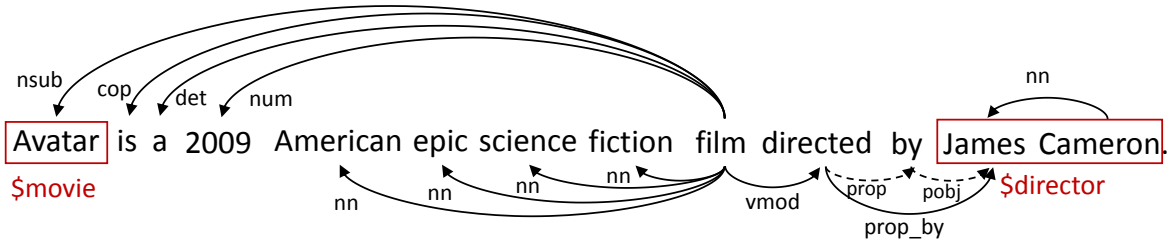


Figure 5.3: An example of dependency-based contexts.

where t_i is the type corresponding to the relation r_i (e.g. the entity type `director.name` infers the relation `movie.directed_by`), $T(w_j)$ denotes a set of all possible entity types for a word w_j, and $C(w_j, t_i)$ is the number of times the specific entity w_j is observed with a specific type t_i in the knowledge graph. For example, $C(w_j, t_i)$ is the number of movies James Cameron has directed.

5.5 Relational Surface Form Derivation

5.5.1 Web Resource Mining

Based on the ontology, we extract all possible entity pairs that are connected with specific relations. Following the previous work, we get search snippets for entity pairs tied with specific relations by web search[2] [75, 82]. Then we mine the patterns used in natural language realization of the relations. With the mined query snippets, we use dependency relations to learn natural language surface forms of each specific relation by dependency-based entity embeddings introduced below.

5.5.2 Dependency-Based Entity Embedding

As Section 2.4.2 introduces, dependency-based embeddings contain more relational information because they are trained on dependency-based contexts [108]. An example sentence "*Avatar is a 2009 American epic science fiction film directed by James Cameron.*" and its dependency parsing result are illustrated in Figure 5.3. Here the sentence comes from snippets returned by searching the entity pair, "*Avatar*" (movie) and "*James Cameron*" (director). The arrows denote dependency relations from headwords to their dependents, and words on arcs denote types of dependency relations. Relations that include a preposition are "collapsed" prior to context extraction (dashed arcs in Figure 5.3), by directly connecting a head and the object of a preposition, and subsuming the preposition itself into the dependency

[2] `http://www.bing.com`

Table 5.1: The contexts extracted for training dependency entity embeddings in the example of the Figure 5.3.

Word	Contexts
`$movie`	film/nsub^{-1}
is	film/cop^{-1}
a	film/det^{-1}
2009	film/num^{-1}
american	film/nn^{-1}
epic	film/nn^{-1}
science	film/nn^{-1}
fiction	film/nn^{-1}
film	avatar/nsub, is/cop, a/det, 2009/num, american/nn epic/nn, science/nn, fiction/nn, directed/vmod
directed	`$director`/prep_by
`$diretor`	directed/prep_by^{-1}

label. Before training embeddings, we replace entities with their entity tags such as `$movie` for "*Avatar*" and `$director` for "*James Cameron*".

The dependency-based contexts extracted from the example are given in Table 5.1, where headwords and their dependents can form the contexts by following the arc on a word in the dependency tree, and -1 denotes the directionality of the dependency. With the target words and associated dependency-based contexts, we can train dependency-based entity embeddings for all target words [169, 16, 17].

5.5.3 Surface Form Derivation

In addition to named entities detected by gazetteers, there are two different relational surface forms used in natural language, entity surface forms and entity syntactic contexts, which are derived from trained embeddings through following approaches.

5.5.3.1 Entity Surface Forms

With only background knowledge gazetteers provided in Section 5.4, the unspecified entities cannot be captured because a knowledge graph does not contain such information like words "*film*" and "*director*". This procedure is to discover words that play the same role and carry similar functional dependency as the specified entities. For example, the entity `$character` may derive the word "*role*", and `$movie` may derive "*film*", "*movie*" as their entity surface forms. The unspecified entities provide important cues for inferring corresponding relations.

We first define a set of entity tags $E = \{e_i\}$ and a set of words $W = \{w_j\}$. Based on the

trained dependency-based entity embeddings, for each entity tag e_i, we compute the score of a word w_j as

$$S_i^F(w_j) = \frac{\text{FormSim}(w_j, e_i)}{\sum_{e_k \in E} \text{FormSim}(w_j, e_k)}, \tag{5.2}$$

where $\text{FormSim}(w, e)$ is the cosine similarity between the embeddings of the word w and the entity tag e. $S_i^F(w_j)$ can be viewed as the normalized weights of words and indicate the importance for discriminating different entities. Based on $S_i^F(w_j)$, we propose to extract top N similar words for each entity tag e_i, to form a set of entity surface forms F_i, where F_i includes surface form candidates of entity e_i. The derived words may have similar embeddings as the target entity, for example, "*director*" and `$director` may encode the same context information such as *directed/prep_by*$^{-1}$ in their embeddings. Therefore, the word "*director*" can be extracted by the entity tag `$director` to serve as its surface form. With derived words F_i for entity tag e_i, we can normalize relation probabilities a word $w_j \in F_i$ infers.

$$P_F(r_i \mid w_j) = P_F(e_i \mid w_j) = \frac{S_i^F(w_j)}{\sum_{k, w_j \in F_k} S_k^F(w_j)}, \tag{5.3}$$

where r_i is a relation inferred from the entity tag e_i, $S_k^F(w_j)$ is the score of a word w_j that belongs to the set F_k extracted by the entity tag e_k, and $P_F(r_i \mid w_j)$ is similar to $P_E(r_i \mid w_j)$ in (5.1) but based on derived words instead of specified entities.

5.5.3.2 Entity Syntactic Contexts

Another type of relational cues comes from contexts of entities; for example, a user utterance "*find movies produced by james cameron*" includes an unspecified movie entity "*movies*" and a specified entity "*james cameron*", which may be captured by entity surface forms via P_F and gazetteers via P_E respectively. However, it does not consider local observations "*produced by*". In this example, the most likely relation of an entity "*james cameron*" from the background knowledge is `director.name`, which infers `movie.directed_by`, and local observations are not used to derive the correct relation `movie.produced_by` for this utterance.

This procedure is to discover the relational entity contexts based on syntactic dependencies. With dependency-based entity embeddings and their context embeddings, for each entity tag e_i, we extract top N syntactic contexts to form a set of entity contexts C_i, which includes the words that are the most activated by a given entity tag e_i. The extraction procedure is similar to one in Section 5.5.3.1; for each entity tag e_i, we compute the score of the word w_j as

$$S_i^C(w_j) = \frac{\text{CxtSim}(w_j, e_i)}{\sum_{e_k \in E} \text{CxtSim}(w_j, e_k)}, \tag{5.4}$$

where $\text{CxtSim}(w_j, e_i)$ is the cosine similarity between the context embeddings of a word w_j

and the embeddings of a entity tag e_i.

The derived contexts may serve as indicators of possible relations. For instance, for the entity tag `$producer`, the most activated contexts include "*produced/prep_by*$^{-1}$", so the word "*produced*" can be extracted by this procedure for detecting local observations other than entities. Then we can normalize the relation probabilities the contexts imply to compute $P_C(r_i \mid w_j)$ similar to (5.3):

$$P_C(r_i \mid w_j) = P_C(e_i \mid w_j) = \frac{S_i^C(w_j)}{\sum_{k, w_j \in C_k} S_k^C(w_j)}. \tag{5.5}$$

5.6 Probabilistic Enrichment and Bootstrapping

Hakkani-Tür et al. proposed to use probabilistic weights for unsupervised relation detection [76]. We extend the approach to integrate induced relations from prior knowledge $P_E(r_i \mid w_j)$ and from local relational surface forms $P_F(r_i \mid w_j)$ and $P_C(r_i \mid w_j)$ to enrich the relation weights for effectively detecting relations given utterances. The experiments integrate multiple distributions in three ways:

- Unweighted

$$R_w(r_i) = \begin{cases} 1 & \text{, if } P_E(r_i \mid w) > 0 \text{ or } P_F(r_i \mid w) > 0 \text{ or } P_C(r_i \mid w) > 0. \\ 0 & \text{, otherwise.} \end{cases} \tag{5.6}$$

  This method combines possible relations from all sources, which tends to capture as many as possible relations (higher recall).

- Weighted

$$R_w(r_i) = \max(P_E(r_i \mid w), P_F(r_i \mid w), P_C(r_i \mid w)) \tag{5.7}$$

  This method assumes that the relation r_i invoked in word w comes from the source that carries the highest probability, so it simply selects the highest one among the three sources.

- Highest Weighted

$$R_w(r_i) = \max(P'_E(r_i \mid w), P'_F(r_i \mid w), P'_C(r_i \mid w)),$$

$$P'(r_i \mid w) = \mathbb{1}[i = \arg\max_i P(r_i \mid w)] \cdot P(r_i \mid w). \tag{5.8}$$

  This method only combines the most likely relation for each word, because $P'(r_i \mid w) = 0$ when a relation r_i is not the most likely relation of a word w.

Table 5.2: An example of three different methods in the probabilistic enrichment ($w =$ "*pitt*").

Relation Probabilistic Weight		actor	produced_by	location
Original	$P_E(r \mid w)$	0.7	0.3	0
	$P_F(r \mid w)$	0.4	0	0.6
	$P_C(r \mid w)$	0	0	0
Enrichment	Unweighted $R_w(r)$	1	1	1
	Weighted $R_w(r)$	0.7	0.3	0.6
	Highest Weighted $R_w(r)$	0.7	0	0.6

Algorithm 1: Bootstrapping

Data: a set of user utterances $U = \{u_j\}$; relation weights for utterances, $R_{u_j}(r_i)$, $u_j \in U$;
Result: a multi-class multi-label classifier E that estimates relations given an utterance
Initializing relation labels $L^0(u_j) = \{r_i \mid R_{u_j}(r_i) \geq \delta\}$;
repeat
  Training ensemble of M weak classifiers E^k on U and $L^k(u_j)$;
  Classifying the utterance u_j by E^k and output the probability distribution of relations as $R_{u_i}^{(k+1)}(r_i)$;
  Creating relation labels $L^{(k+1)}(u_j) = \{r_i \mid R_{u_j}^{(k+1)}(r_i) \geq \delta\}$;
until $L^{(k+1)}(u_j) \sim L^k(u_j)$;
return E^k;

An example of relation weights about the word "*pitt*" with three different methods is shown in Table 5.2. The final relation weight of the relation r_i given an utterance u, $R_u(r_i)$, can be compute as

$$R_u(r_i) = \max_{w \in u} R_w(r_i). \tag{5.9}$$

With enriched relation weights, $R_u(r_i)$, we train a multi-class, multi-label classifier in an unsupervised way, where we learn ensemble of weak classifiers by creating pseudo training labels in each iteration for boosting the performance [52, 75, 106, 149]. The detail of the algorithm is shown in Algorithm 1. Then the returned classifier E^k can be used to detect relations given unseen utterances.

5.7 Experiments

5.7.1 Experimental Setup

The experiments use a list of entities/gazetteers from the publicly available Freebase knowledge graph. The list includes 670K entities of 78 entity types, including movie names, actors, release dates, etc. after filtering out the movie entities with lower confidences [89].

The relation detection datasets include crowd-sourced utterances addressed to a conversa-

Table 5.3: Relation detection datasets used in the experiments.

Query Statistics	Train	Test
% entity only	8.9%	10.7%
% relations only with specified movie names	27.1%	27.5%
% relations only with specified other names	39.8%	39.6%
% more complicated relations	15.4%	14.7%
% not covered	8.8%	7.6%
#utterance with SPARQL annotations	3338	1084

tional agent and are described in Table 5.3. Both train and test sets are manually annotated with SPARQL queries, which are used to extract relation annotations. Most of data includes the relations with either specified movie names or specified other names. In addition, the relations only with specified movie names are difficult to capture by gazetteers, which emphasizes the contribution of this task. We use 1/10 training data as a development set to tune the parameters δ, M, and the optimal number of iterations in Algorithm 1. The training set is only used to train the classifier of Algorithm 1 for bootstrapping in an unsupervised way; note that manual annotations are not used here.

For retrieving snippets, we use 14 entity pairs from a knowledge graph related to movie entities, which include director, character, release date, etc. We extract snippets related to each pair from web search results, and we end up with 80K snippets, where the pairs of entities are marked in the returned snippets[3]. For all query snippets, we parse all with the Berkeley Parser, and then convert output parse trees to dependency parses using the LTH Constituency-to-Dependency Conversion toolkit[4] for training dependency-based entity embeddings [95, 128]. The trained entity embeddings have dimension 200 and vocabulary size is 1.8×10^5.

In the experiments, we train multi-class, multi-label classifiers using `icsiboost` [61], a boosting-based classifier, where we extract word unigrams, bigrams, and trigrams as classification features. The evaluation metric we use is micro F-measure for relation detection [76]. The performance with all of the proposed approaches before and after bootstrapping with $N = 15$ (top 15 similar words of each tag) is shown in Table 5.4 and Table 5.5 respectively.

5.7.2 Results

The first baseline here (row (a)) uses gazetteers to detect entities and then infers relations by background knowledge described in Section 5.4. Row (b) is another baseline, which uses the retrieved snippets and their inferred relations as labels to train a multi-class multi-label

[3]In this work, we use top 10 results from Bing for each entity pair.

[4]`http://nlp.cs.lth.se/software/treebank_converter`

Table 5.4: The micro F-measure of the first-pass SLU performance before bootstrapping ($N = 15$) (%).

Approach			Unweighted	Weighted	Highest
(a)	Baseline	Gazetteer	35.21	37.93	36.08
(b)		Gazetteer + Weakly Supervised	25.07	39.04	39.40
(c)	BOW	Gazetteer + Surface Form	34.23	36.57	34.69
(d)	Dep.-Based	Gazetteer + Surface Form	37.44	**41.01**	39.19
(e)		Gazetteer + Context	35.31	38.04	37.25
(f)		Gazetteer + Surface Form + Context	**37.66**	40.29	**40.07**

Table 5.5: The micro F-measure of SLU performance with bootstrapping ($N = 15$) (%).

Approach			Unweighted	Weighted	Highest
(a)	Baseline	Gazetteer	36.91	40.10	38.89
(b)		Gazetteer + Weakly Supervised	37.39	39.07	39.98
(c)	BOW	Gazetteer + Surface Form	34.91	38.13	37.16
(d)	Dep.-Based	Gazetteer + Surface Form	38.37	41.10	42.74
(e)		Gazetteer + Context	37.23	38.88	38.04
(f)		Gazetteer + Surface Form + Context	**38.64**	**41.98**	**43.34**

classifier, then outputs relation probabilities for each utterance as $R_u(w)$, and integrates with the first baseline [75]. Here the data for training only uses patterns between entity pairs in the paths of dependency trees. Row (c) is results of adding entity surface forms derived from original embeddings, which is shown for demonstrating the effectiveness of dependency-based entity embeddings (row (d)). Row (e) is results of adding entity contexts, and row (f) combines both of entity surface forms and entity contexts. Below we analyze the effectiveness of proposed approaches.

5.8 Discussion

We analyze the effectiveness of learned entity surface forms and entity contexts, and compare different probabilistic enrichment methods, and validate the effectiveness of bootstrapping below.

5.8.1 Effectiveness of Entity Surface Forms

Row (c) and row (d) show the performance of using entity surface forms derived from CBOW and dependency-based embeddings respectively. It can be found that the words derived from original embeddings do not successfully capture surface forms of entity tags, and the results cannot be improved. On the other hand, results from dependency-based embeddings outper-

Table 5.6: The examples of derived entity surface forms based on dependency-based entity embeddings.

Entity Tag	Derived Word
`$character`	*character, role, who, girl, she, he, officer*
`$director`	*director, dir, filmmaker*
`$genre`	*comedy, drama, fantasy, cartoon, horror, sci*
`$language`	*language, spanish, english, german*
`$producer`	*producer, filmmaker, screenwriter*

form baselines for all enrichment methods, which demonstrate the effectiveness of including entity surface forms based on dependency relations for relation detection. To analyze results of entity surface forms, we show some examples about derived words in Table 5.6. It can be shown that the functional similarity carried by dependency-based entity embeddings effectively benefits relation detection task.

5.8.2 Effectiveness of Entity Contexts

Row (e) shows results of adding entity contexts learned from dependency-based contexts. It does not show significantly improvement compared to baselines. Nevertheless, combining with dependency-based entity surface forms, F-measure achieves 43% by highest weighted probabilistic enrichment, which implies that including local observations based on syntactic contexts may help relation detection, but the influence is not significant. The derived entity contexts of an entity tag `$actor` include "*plays*" and "*starring*", which help capture the implied relations of utterances.

5.8.3 Comparison of Probabilistic Enrichment Methods

From Table 5.4 and Table 5.5, among the three probabilistic enrichment methods, unweighted method performs worst, because it does not differentiate the relations with higher and lower confidence, and some relations with lower probabilities will be mistakenly outputted. Comparing between weighted and highest weighted methods, the first baseline using the weighted method performs better, while other approaches using the highest weighted method perform better. The reason probably is that the weighted method can provide more possible relations for the baseline only using gazetteers to increase the recall, so the weighted method benefits the first baseline. On the other hand, proposed approaches have higher recall and the highest weighted method provides more precise relations, resulting in better performance when applying the highest weighted method.

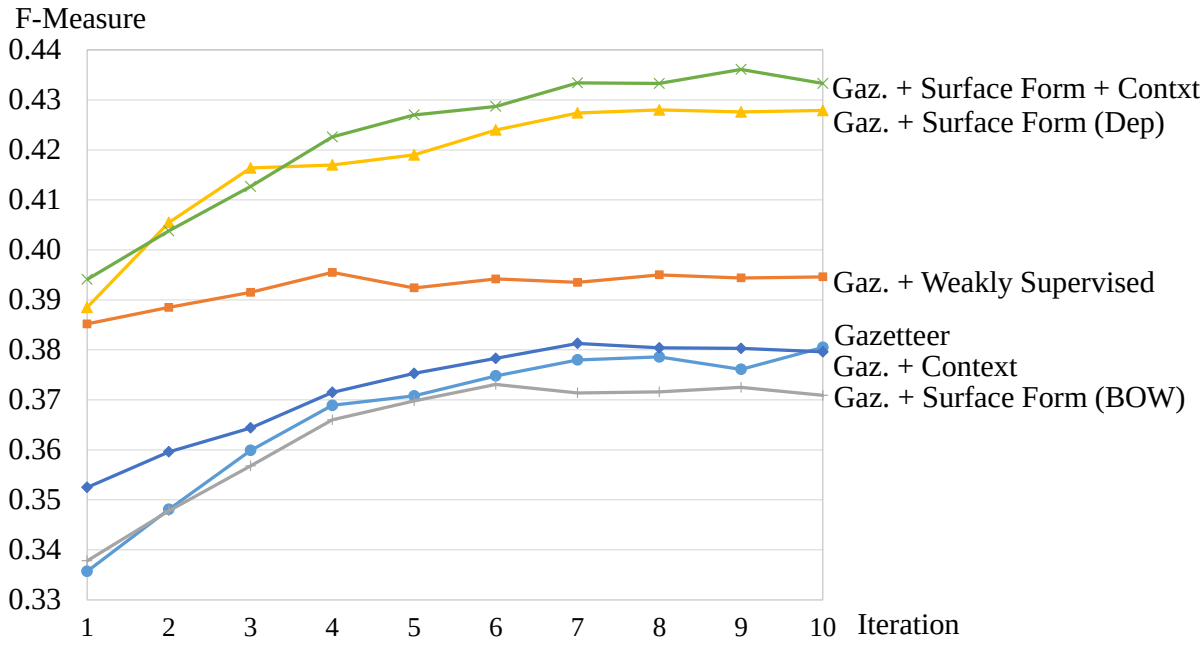


Figure 5.4: Learning curves over incremental iterations of bootstrapping.

5.8.4 Effectiveness of Bootstrapping

The F-measure learning curves of all results using highest weighted probabilistic enrichment on the test set are presented in Figure 5.4. The light blue line marked with circles is the first baseline, which applies only gazetteers with probability distribution of entity types to relation detection. After bootstrapping, the performance is significantly improved and achieves about 39% of F-measure. Another baseline using a weakly supervised classifier (orange line marked with squares) performs well before bootstrapping, while the performance cannot be significantly improved with increased iterations. All other results show significant improvements after bootstrapping. The best result is the combination of all approaches (green line marked with crosses), and the curve shows the effectiveness and efficiency of bootstrapping. The probable reason is that the probabilities came from different sources can complement each other, and then benefit classifiers. Also, only adding dependency-based entity surface forms (yellow line marked with triangles) performs similar to the combination result, showing that the major improvement comes from relational entity surface forms. The figure demonstrates the effectiveness of bootstrapping for improving relation detection.

5.8.5 Overall Results

The proposed approaches successfully capture local information other than background knowledge, where relational surface forms can be learned by dependency-based entity embeddings

trained on query snippets. After combining with prior relations induced by gazetteers, relational information from the text domain can benefit relation detection for the spoken domain. Also, the fully unsupervised approach shows the effectiveness of applying structured knowledge to SLU for tackling open domain problems.

5.9 Summary

This chapter proposes to automatically capture relational surface forms including entity surface forms and entity contexts based on dependency-based entity embeddings. The detected semantics viewed as local observations can be integrated with background knowledge by probabilistic enrichment methods. Experiments show that open-domain SLU can be significantly improved by involving derived entity surface forms as local cues together with prior knowledge. Therefore, it is shown that the surface forms corresponding to an ontology carry important knowledge for building a good SLU component.

6 Semantic Decoding in SLU Modeling

> *Semantics is about the relation of words to thoughts, but it is also about the relation of words to other human concerns. Semantics is about the relation of words to reality - the way that speakers commit themselves to a shared understanding of the truth, and the way their thoughts are anchored to things and situations in the world.*
>
> Steven Pinker, *Johnstone Family Professor at Harvard University*

With an organized ontology automatically learned by knowledge acquisition, this chapter further introduces a novel matrix factorization (MF) approach to learn latent feature vectors for utterances and semantic concepts. More specifically, our model learns semantic slots for a domain-specific SDS in an unsupervised fashion, and then performs semantic decoding using latent MF techniques. To further consider the global semantic structure, such as inter-word and inter-slot relations, we augment the latent MF-based model with the ontology structure. The final goal of the model is to predict semantic slots and word patterns of each given utterance, considering their inference relations and domain-specificity in a joint fashion.

6.1 Introduction

A key component of an SDS is the SLU module—it parses the users' utterances into semantic representations; for example, the utterance "*find a cheap restaurant*" can be parsed into action="*find*", price="*cheap*", target="*restaurant*" [129]. To design the SLU module of an SDS, most previous studies relied on predefined slots for training the decoder [14, 56, 72, 138]. However, these predefined semantic slots may bias the subsequent data collection process, and the cost of manually labeling utterances for updating the ontology is high [**?**].

In recent years, these problems led to the development of unsupervised SLU techniques [82, 83, 31, 33]. However, a challenge of SLU is inference of hidden semantics. Taking the utterance "*can i have a cheap restaurant*" as an example, from its surface patterns, we can see

that it includes explicit semantic information about "price (cheap)" and "target (restaurant)"; however, it also includes hidden semantic information, such as "food" and "seeking", since the SDS needs to infer that the user wants to "find" some cheap "food", even though they are not directly observed in the surface patterns. Nonetheless, these implicit semantics are important semantic concepts for domain-specific SDSs. Traditional SLU models use discriminative classifiers to predict whether the predefined slots occur in the utterances or not, ignoring the unobserved concepts and the hidden semantic information [86].

As mentioned in Chapter 2, most of prior studies did not explicitly learn latent factor representations from the data—while we hypothesize that the better robustness in noisy data can be achieved by explicitly modeling the measurement errors (usually produced by ASR) using latent variable models and taking additional local and global semantic constraints into account. To the best of our knowledge, this work is the first to learn latent feature representations in unsupervised SLU, taking various local and global lexical, syntactic, and semantic information into account.

In this chapter, we take a rather radical approach: we propose a novel matrix factorization (MF) model for learning latent features for SLU, taking account of additional structure information such as word relations, induced slots, and slot relations simultaneously. To further consider global coherence of induced slots, we combine the MF model with a knowledge graph propagation based model, fusing both a word-based lexical knowledge graph and a slot-based semantic knowledge graph. In fact, as it is shown in the Netflix challenge, MF is credited as the most useful technique for recommendation systems [98]. Also, the MF model considers unobserved patterns and estimates their probabilities instead of viewing them as negative examples. However, to the best of our knowledge, the MF technique is not yet well explored in the SLU and SDS communities, and it is not very straight-forward to use MF methods to learn latent feature representations for semantic parsing in SLU. To evaluate the performance of our model, we compare it to standard discriminative SLU baselines, and show that our MF-based model is able to produce strong results in semantic decoding, and the knowledge graph propagation model further improves the performance. Our contributions are three-fold:

- We are among the first to study MF techniques for unsupervised SLU, taking account of additional information;
- We augment the MF model with a knowledge graph propagation model, increasing the global coherence of semantic decoding using induced slots;
- Our experimental results show that the unsupervised MF-SLU outperforms strong discriminative baselines, obtaining promising results.

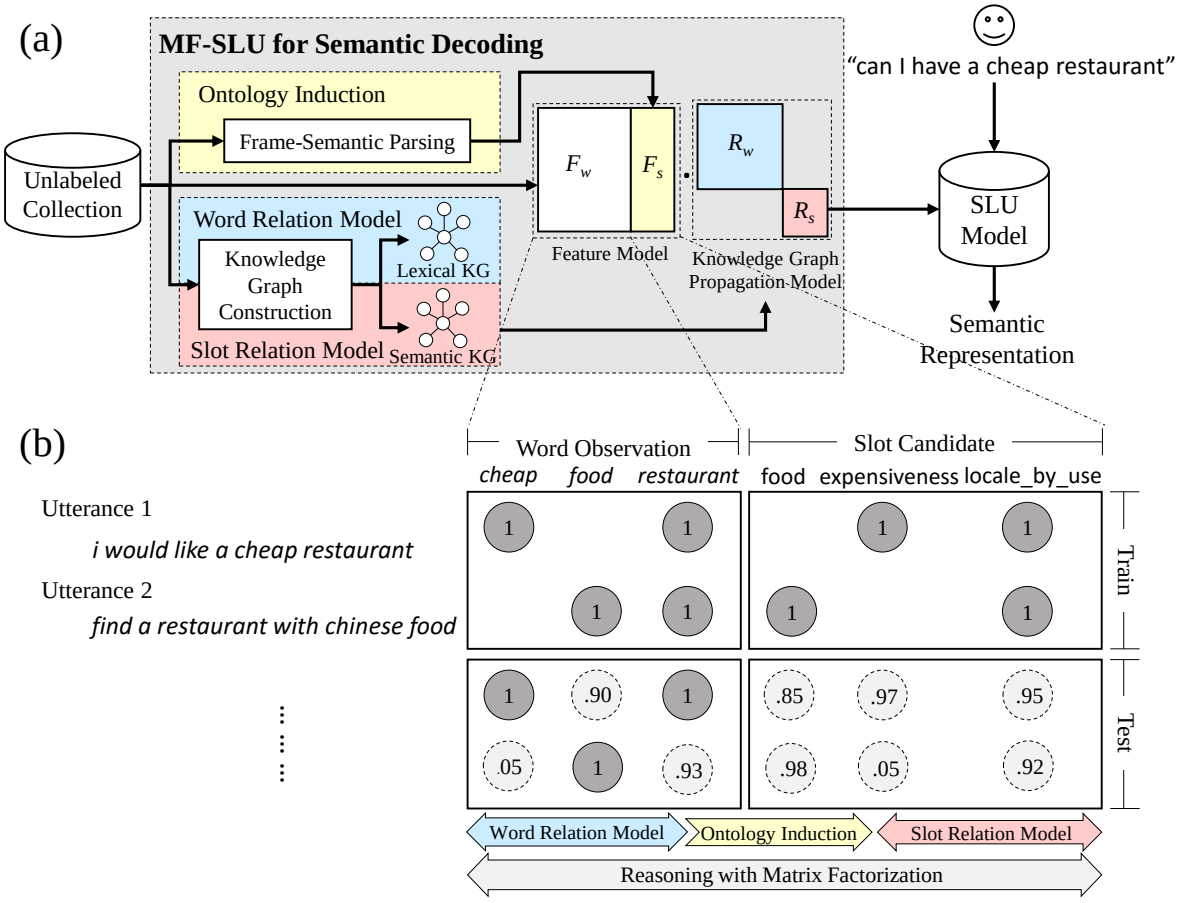


Figure 6.1: (a): The proposed framework of semantic decoding. (b): Our MF method completes a partially-missing matrix for implicit semantic parsing. Dark circles are observed facts, and shaded circles are inferred facts. Ontology induction maps observed surface patterns to semantic slot candidates. Word relation model constructs correlations between surface patterns. Slot relation model learns slot-level correlations based on propagating the automatically derived semantic knowledge graphs. Reasoning with matrix factorization incorporates these models jointly, and produces a coherent and domain-specific SLU model.

6.2 Proposed Framework

This chapter introduces an MF technique for unsupervised SLU. The proposed framework is shown in Figure 6.1a. Given utterances, the task of the SLU model is to decode their surface patterns into semantic forms and simultaneously differentiate the target semantic concepts from the generic semantic space for task-oriented SDSs. Note that our model does not require any human-defined slots and domain-specific semantic representations for utterances.

In the proposed model, we first build a feature matrix to represent training utterances, where each row represents an utterance, and each column refers to an observed surface pattern or an induced slot candidate. Figure 6.1b illustrates an example of the matrix. Given a testing

utterance, we convert it into a vector based on the observed surface patterns, and then fill in the missing values of all slot candidates. In the first utterance in the figure, although the semantic slot food is not observed, the utterance implies the meaning facet food. The MF approach is able to learn latent feature vectors for utterances and semantic elements, inferring implicit semantic concepts to improve the decoding process—namely, by filling the matrix with probabilities (lower part of the matrix).

The feature model is built on the observed word patterns and slot candidates, where slot candidates are obtained from the ontology induction component through frame-semantic parsing (the yellow block in Figure 6.1a) [31]. Section 6.3.1 explains the detail of the feature model. In order to additionally consider inter-word and inter-slot relations, we propose a knowledge graph propagation model based on two knowledge graphs, which includes a word relation model (blue block) and a slot relation model (pink block), described in Section 6.3.2. The method of automatic knowledge graph construction is introduced in Section 4.3.1, where we leverage distributed word embeddings associated with typed syntactic dependencies to model the relations [39, 108, 118, 119].

Finally, we train the SLU model by learning latent feature vectors for utterances and slot candidates through MF techniques. Combining with a knowledge graph propagation model based on word/slot relations, the trained SLU model estimates a probability that each semantic slot occurs in a testing utterance, and how likely each slot is domain-specific simultaneously. In other words, the SLU model is able to transform testing utterances into domain-specific semantic representations without human involvement.

6.3 Matrix Factorization for Spoken Language Understanding (MF-SLU)

Considering the benefits brought by MF techniques, including 1) modeling noisy data, 2) modeling hidden semantics, and 3) modeling long-range dependencies between observations, in this work we apply an MF approach to SLU modeling for SDS. In our model, we use U to denote a set of input utterances, W as a set of word patterns, and S as a set of semantic slots that we would like to predict. The pair of an utterance $u \in U$ and a word pattern/semantic slot $x \in \{W \cup S\}$, $\langle u, x \rangle$, is a *fact*. The input to our model is a set of observed facts $\mathcal{O}$, and the observed facts for a given utterance is denoted by $\{\langle u, x \rangle \in \mathcal{O}\}$. The goal of our model is to estimate, for a given utterance u and a given word pattern/semantic slot x, the probability, $p(M_{u,x} = 1)$, where $M_{u,x}$ is a binary random variable that is true if and only if x is the word pattern/domain-specific semantic slot in the utterance u. We introduce a series of exponential family models that estimate the probability using a natural parameter $\theta_{u,x}$ and

the logistic sigmoid function:

$$p(M_{u,x} = 1 \mid \theta_{u,x}) = \sigma(\theta_{u,x}) = \frac{1}{1 + \exp(-\theta_{u,x})} \tag{6.1}$$

We construct a matrix $M_{|U|\times(|W|+|S|)}$ as observed facts for MF by integrating a feature model and a knowledge graph propagation model below.

6.3.1 Feature Model

First, we build a word pattern matrix F_w with binary values based on observations, where each row represents an utterance and each column refers to an observed unigram. In other words, F_w carries basic word vectors for utterances, which is illustrated as the left part of the matrix in Figure 6.1b.

To induce the semantic elements, we parse all ASR-decoded utterances in our corpus using SEMAFOR [48, 49], and extract all frames from semantic parsing results as slot candidates [31, 55]. Figure 3.2 shows an ASR-decoded utterance example "*can i have a cheap restaurant*" parsed by SEMAFOR. Three FrameNet-defined frames capability, expensiveness, and locale_by_use are generated for the utterance, which we consider as slot candidates for a domain-specific dialogue system [4]. Then we build a slot matrix F_s with binary values based on the induced slots, and the matrix also denotes slot features for all utterances (right part of the matrix in Figure 6.1b).

To build a feature model M_F, we concatenate two matrices:

$$M_F = [\ F_w \quad F_s\ ], \tag{6.2}$$

which is the upper part of the matrix in Figure 6.1b for training utterances. Note that we do not use any annotations, so all slot candidates are included.

6.3.2 Knowledge Graph Propagation Model

As mentioned in Chapter 3, because SEMAFOR was trained on FrameNet annotation, which has a more generic frame-semantic context, not all the frames from the parsing results can be used as the actual slots in the domain-specific dialogue systems. For instance, we see that the frames expensiveness and locale_by_use are essentially key slots for the purpose of understanding in the restaurant query domain, whereas the capability frame does not convey particularly valuable information for SLU.

Assuming that domain-specific concepts are usually related to each other, considering global

relations between semantic slots induces a more coherent slot set. It is shown that the relations on knowledge graphs help make decisions on domain-specific slots [39]. Similar to Chapter 4, we consider two directed graphs, semantic and lexical knowledge graphs, where each node in the semantic knowledge graph is a slot candidate s_i generated by the frame-semantic parser, and each node in the lexical knowledge graph is a word w_j.

- **Slot-based semantic knowledge graph** is built as $G_s = \langle V_s, E_{ss} \rangle$, where $V_s = \{s_i \in S\}$ and $E_{ss} = \{e_{ij} \mid s_i, s_j \in V_s\}$.
- **Word-based lexical knowledge graph** is built as $G_w = \langle V_w, E_{ww} \rangle$, where $V_w = \{w_i \in W\}$ and $E_{ww} = \{e_{ij} \mid w_i, w_j \in V_w\}$.

The edges connect two nodes in the graphs if there is a typed dependency between them. The structured graph helps define a coherent slot set. To model the relations between words/slots based on the knowledge graphs, we define two relation models below.

- Semantic Relation
  To model word semantic relations, we compute a matrix $R_w^S = [\text{Sim}(w_i, w_j)]_{|W|\times|W|}$, where $\text{Sim}(w_i, w_j)$ is the cosine similarity between the dependency embeddings of word patterns w_i and w_j after normalization. For slot semantic relations, we compute $R_s^S = [\text{Sim}(s_i, s_j)]_{|S|\times|S|}$ similarly[1]. The matrices R_w^S and R_s^S model not only semantic similarity but functional similarity since we use dependency-based embeddings [108].
- Dependency Relation
  Assuming that important semantic slots are usually mutually related to each other, that is, connected by syntactic dependencies, our automatically derived knowledge graphs are able to help model the dependency relations. For word dependency relations, we compute a matrix $R_w^D = [\hat{r}(w_i, w_j)]_{|W|\times|W|}$, where $\hat{r}(w_i, w_j)$ measures a dependency relation between two word patterns w_i and w_j based on the word-based lexical knowledge graph, and the detail is described in Section 4.3.1. For slot dependency relations, we similarly compute $R_s^D = [\hat{r}(s_i, s_j)]_{|S|\times|S|}$ based on the slot-based semantic knowledge graph.

With the built word relation models (R_w^S and R_w^D) and slot relation models (R_s^S and R_s^D), we combine them as a knowledge graph propagation matrix M_R[2].

$$M_R = \begin{bmatrix} R_w^{SD} & 0 \\ 0 & R_s^{SD} \end{bmatrix}, \tag{6.3}$$

[1]For each column in R_w^S and R_s^S, we only keep top 10 highest values, which correspond the top 10 semantically similar nodes.

[2]The values in the diagonal of M_R are 0 to model the propagation from other entries.

where $R_w^{SD} = R_w^S + R_w^D$ and $R_s^{SD} = R_s^S + R_s^D$ to integrate semantic and dependency relations. The goal of this matrix is to propagate scores between nodes according to different types of relations in the knowledge graphs [26].

6.3.3 Integrated Model

With a feature model M_F and a knowledge graph propagation model M_R, we integrate them into a single matrix.

$$\begin{aligned} M &= M_F \cdot (\alpha I + \beta M_R) \\ &= \begin{bmatrix} F_w & F_s \end{bmatrix} \cdot \begin{bmatrix} \alpha I + \beta R_w & 0 \\ 0 & \alpha I + \beta R_s \end{bmatrix} \\ &= \begin{bmatrix} \alpha F_w + \beta F_w R_w & 0 \\ 0 & \alpha F_s + \beta F_s R_s \end{bmatrix}, \end{aligned} \tag{6.4}$$

where M is the final matrix and I is an identity matrix. α and β are weights for balancing original values and propagated values, where $\alpha + \beta = 1$. The matrix M is similar to M_F, but some weights are enhanced through the knowledge graph propagation model, M_R. The word relations are built by $F_w R_w$, which is the matrix with internal weight propagation on the lexical knowledge graph (the blue arrow in Figure 6.1b). Similarly, $F_s R_s$ models the slot correlations, and can be treated as the matrix with internal weight propagation on the semantic knowledge graph (the pink arrow in Figure 6.1b). The propagation models can be treated as running a random walk algorithm on the graphs.

F_s contains all slot candidates generated by SEMAFOR, which may include some generic slots (such as capability), so the original feature model cannot differentiate domain-specific and generic concepts. By integrating with R_s, the semantic and dependency relations can be propagated via edges in the knowledge graph, and domain-specific concepts may have higher weights based on the assumption that slots for dialogue systems are often mutually related [39]. Hence, the structure information can be automatically involved in the matrix. Also, the word relation model brings the same function, but on the word level. In conclusion, for each utterance, the integrated model not only predicts probabilities that semantic slots occur but also considers whether the slots are domain-specific. The following sections describe the learning process.

6.3.4 Parameter Estimation

The proposed model is parameterized through weights and latent component vectors, where the parameters are estimated by maximizing the log likelihood of observed data [46].

$$\begin{aligned}\theta^* &= \arg\max_{\theta} \prod_{u \in U} p(\theta \mid M_u) \\ &= \arg\max_{\theta} \prod_{u \in U} p(M_u \mid \theta) p(\theta) \\ &= \arg\max_{\theta} \sum_{u \in U} \ln p(M_u \mid \theta) - \lambda_\theta,\end{aligned} \tag{6.5}$$

where M_u is the vector corresponding to the utterance u, because we assume that each utterance is independent of others.

To avoid treating unobserved facts as designed negative facts and to complete missing entries of the matrix, our model can be factorized by a matrix completion technique with a low-rank latent semantics assumption [97, 134]. Bayesian personalized ranking (BPR) is an optimization criterion that learns from implicit feedback for MF by a matrix completion technique, which uses a variant of the ranking: giving observed true facts higher scores than unobserved (true or false) facts to factorize the given matrix [134]. BPR was shown to be useful in learning implicit relations for improving semantic parsing [38, 135].

6.3.4.1 Objective Function

To estimate the parameters in (6.5), we create a dataset of *ranked pairs* from M in (6.4): for each utterance u and each observed fact $f^+ = \langle u, x^+ \rangle$, where $M_{u,x} \geq \delta$, we choose each word pattern/slot x^- such that $f^- = \langle u, x^- \rangle$, where $M_{u,x} < \delta$, which refers to the word pattern/slot we have not observed to be in utterance u. That is, we construct the observed data $\mathcal{O}$ from M. Then for each pair of facts f^+ and f^-, we want to model $p(f^+) > p(f^-)$ and hence $\theta_{f^+} > \theta_{f^-}$ according to (6.1). BPR maximizes the summation of each ranked pair, where the objective is

$$\sum_{u \in U} \ln p(M_u \mid \theta) = \sum_{f^+ \in \mathcal{O}} \sum_{f^- \notin \mathcal{O}} \ln \sigma(\theta_{f^+} - \theta_{f^-}). \tag{6.6}$$

The BPR objective is an approximation to the per utterance AUC (area under the ROC curve), which directly correlates to what we want to achieve – well-ranked semantic slots per utterance.

6.3.4.2 Optimization

To maximize the objective in (6.6), we employ a stochastic gradient descent (SGD) algorithm [134]. For each randomly sampled observed fact $\langle u, x^+ \rangle$, we sample an unobserved fact $\langle u, x^- \rangle$, which results in $|\mathcal{O}|$ fact pairs $\langle f^-, f^+ \rangle$. For each pair, we perform an SGD update using the gradient of the corresponding objective function for MF [68].

6.4 Experiments

6.4.1 Experimental Setup

In this experiment, we used the Cambridge University SLU corpus as previous chapters [86, 30]. The domain about restaurant recommendation in Cambridge; subjects were asked to interact with multiple SDSs in an in-car setting. The corpus contains a total number of 2,166 dialogues, including 15,453 utterances (10,571 for self-training and 4,882 for testing). The vocabulary size is 1,868. There are 10 slots created by domain experts: addr, area, food, name, phone, postcode, price range, signature, task, and type.

For the parameter setting, weights for balancing feature models and propagation models, α and β in (6.4), are set as 0.5 to give the same influence, and the threshold for defining the unobserved facts δ is set as 0.5 for all experiments. We use the Stanford Parser[3] to obtain the collapsed typed syntactic dependencies and set the dimensionality of embeddings $d = 300$ in all experiments [143].

To evaluate the performance of automatically decoded slots, we measure their quality as the proximity between predicted slots and reference slots via mappings shown in Figure 3.3. To eliminate the influence of threshold selection, we take the whole ranking list into account and use the metric that are independent of the selected threshold for evaluation. For each utterance, with the predicted probabilities of all slot candidates, we can compute AP to evaluate the performance of SLU by treating the slots with mappings as positive. MAP is computed for evaluating all utterances. For all experiments, we perform paired t-test on AP scores of results to test the significance.

6.4.2 Evaluation Results

Table 6.1 shows the MAP performance of predicted slots for all experiments on ASR and manual transcripts. For the first baseline using explicit semantics, we use the observed data

[3] http://nlp.stanford.edu/software/lex-parser.shtml

Table 6.1: The MAP of predicted slots (%); † indicates that the result is significantly better than MLR (row (b)) with $p < 0.05$ in t-test.

Approach				ASR		Transcripts	
				w/o	w/ Explicit	w/o	w/ Explicit
(a)	Explicit	SVM		32.48		36.62	
(b)		MLR		33.96		38.78	
(c)	+ Implicit	Baseline	Random	3.43	22.45	2.63	25.09
(d)			Majority	15.37	32.88	16.43	38.41
(e)		MF	Feature	24.24	$37.61^{\dagger}$	22.55	$45.34^{\dagger}$
(f)			Feature + KGP	$\mathbf{40.46}^{\dagger}$	$\mathbf{43.51}^{\dagger}$	$\mathbf{52.14}^{\dagger}$	$\mathbf{53.40}^{\dagger}$

to self-train models for predicting the probability of each semantic slot by SVM with linear kernel and multinomial logistic regression (MLR) (row (a)-(b)) [127, 86]. It is shown that SVM and MLR perform similarly, and MLR is slightly better than SVM because it has better capability of estimating probabilities. For modeling implicit semantics, two baselines are performed as references, random (row (c)) and majority (row (d)) approaches, where the former assigns random probabilities for all slots, and the later assigns probabilities for slots based on their frequency distribution. To improve probability estimation, we further integrate the results from implicit semantics with one from explicit approaches, MLR (row (b)), by averaging their probability distributions.

Two baselines, random and majority, cannot model the implicit semantics, producing poor results. The random results integrated with MLR significantly degrades the performance of MLR for both ASR and manual transcripts. Also, results of the majority approach integrated with MLR does not produce any difference compared to the MLR baseline. Among the proposed MF approaches, only using feature model for building the matrix (row (e)) achieves 24.2% and 22.6% of MAP for ASR and manual results respectively, which are worse than two baselines using explicit semantics. However, with the combination of explicit semantics, using only the feature model significantly outperforms the baselines, where the performance comes from about 34.0% to 37.6% and from 38.8% to 45.3% for ASR and manual results respectively. Additionally integrating a knowledge graph propagation (KGP) model (row (e)) outperforms the baselines for both ASR and manual transcripts, and the performance is further improved by combining with explicit semantics (achieving MAP of 43.5% and 53.4%). The experiments show that the proposed MF models successfully learn implicit semantics and consider structural relations and domain-specificity simultaneously.

6.4.3 Discussion and Analysis

With promising results obtained by the proposed models, we analyze detailed difference between different relation models in Table 6.2.

Table 6.2: The MAP of predicted slots using different types of relation models in M_R (%); † indicates that the result is significantly better than the feature model (column (a)) with $p < 0.05$ in t-test.

Model	Feature	Feature + Knowledge Graph Propagation Model				
Rel.	(a) None	(b) Semantic	(c) Dependent	(d) Word	(e) Slot	(f) All
M_R	-	$\begin{bmatrix} R_w^S & 0 \\ 0 & R_s^S \end{bmatrix}$	$\begin{bmatrix} R_w^D & 0 \\ 0 & R_s^D \end{bmatrix}$	$\begin{bmatrix} R_w^{SD} & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & R_s^{SD} \end{bmatrix}$	$\begin{bmatrix} R_w^{SD} & 0 \\ 0 & R_s^{SD} \end{bmatrix}$
ASR	37.61	$41.39^\dagger$	$41.63^\dagger$	$39.19^\dagger$	$42.10^\dagger$	$\mathbf{43.51}^\dagger$
Manual	45.34	$51.55^\dagger$	$49.04^\dagger$	45.18	$49.91^\dagger$	$\mathbf{53.40}^\dagger$

6.4.3.1 Effectiveness of Semantic and Dependency Relation Models

To evaluate the effectiveness of semantic and dependency relations, we consider each of them individually in M_R of (6.3) (columns (b) and (c) in Table 6.2). Comparing to the original model (column (a)), both modeling semantic relations and modeling dependency relations significantly improve the performance for ASR and manual results. It is shown that semantic relations help the SLU model infer implicit meanings, and then prediction becomes more accurate. Also, dependency relations successfully differentiate the generic concepts from the domain-specific concepts, so that the SLU model is able to predict more coherent set of semantic slots [39]. Integrating two types of relations (column (f)) further improves the performance.

6.4.3.2 Comparing Word/ Slot Relation Models

To analyze performance results from inter-word and inter-slot relations, the columns (d) and (e) show results considering only word relations and only slot relations respectively. It can be seen that the inter-slot relation model significantly improves the performance for both ASR and manual results. However, the inter-word relation model only performs slightly better results for ASR output (from 37.6% to 39.2%), and there is no difference after applying the inter-word relation model on manual transcripts. The reason may be that inter-slot relations carry high-level semantics that align well with the structure of SDS, but inter-word relations do not. Nevertheless, combining two relations (column (f)) outperforms both results for ASR and manual transcripts, showing that different types of relations can compensate each other and then benefit the SLU performance.

6.5 Summary

This chapter presents an MF approach to self-train the SLU model for semantic decoding with consideration of a well-organized ontology in an unsupervised way. The purpose of

the proposed model is not only to predict the probability of each semantic slot but also to distinguish between generic semantic concepts and domain-specific concepts that are related to an SDS. The experiments show that the MF-SLU model obtains promising results of semantic decoding, outperforming strong discriminative baselines.

7 Intent Prediction in SLU Modeling

> *It's really interesting to me how all of us can experience the exact same event, and yet come away with wildly disparate interpretations of what happened. We each have totally different ideas of what was said, what was intended, and what really took place.*
>
> Marya Hornbacher, *Pulitzer Prize nominee*

SLU modeling has different aspects: shallow understanding and deep understanding. In addition to low-level semantic concepts from semantic decoding, deeply understanding users involves more high-level intentions. Since users usually take observable actions motivated by their intents, predicting intents along with follow-up actions can be viewed as another aspect about SLU. A good SLU module is able to accurately understand low-level semantic meanings of utterances and high-level user intentions, which allows SDSs to further predict users' follow-up actions in order to offer better interactions. This chapter focuses on predicting user intents, which correspond to observable actions, using a feature-enriched MF technique, in order to deeply understand users in the unsupervised and semi-supervised manners.

7.1 Introduction

In a dialogue system, SLU and DM modules play important roles because they first map utterances into semantics and then into intent-triggered actions. The semantic representations of user utterances usually refer to the specified information that directly occurs in the utterances. To involve deeper understanding, high-level intentions should be considered. For example, in a restaurant domain, "*find me a taiwanese restaurant*" has a semantic form as action="*find*", type="*taiwanese*", target="*restaurant*", and a follow-up intended action might be asking for its location or navigation, which can be viewed as the high-level intention. Assuming an SDS is able to predict user intents (e.g. navigation), the system not only returns the user a restaurant list but also asks whether the user needs the corresponding location or navigating instructions, and then automatically launches the corresponding application (e.g.

MAPS), providing better conversational interactions and more friendly user experience [147] To design the SLU module of an SDS, most previous studies relied on the predefined ontology and schema to bridge intents and semantic slots [14, 41, 56, 72, 138].

Recently, dialogue systems are appearing on smart-phones and allowing users to launch applications[1] via spontaneous speech. Typically, an SDS needs predefined task domains to understand corresponding functions, such as setting_alert_clock and query_words_via_browser, Each app supports a *single-turn request* task. However, traditional SDSs are unable to dynamically support functions provided by newly installed or not yet installed apps, so that open domain requests cannot be handled due to lack of predefined ontologies. We address the following question: with an open domain single-turn request, how can a system dynamically and effectively provide the corresponding functions to fulfill users' requests? This chapter first focuses on understanding a user's intent and identifying apps that can support such open domain requests.

In addition to the difficulty caused by language ambiguity, behavioral patterns also influence user intents. Typical intelligent assistants (IA) treat each domain (e.g. restaurant search, messaging, etc.) independent of each other, where only current user utterances are considered to decide the desired apps in SLU [28]. Some IAs model user intents by using contexts from previous utterances, but they do not take into account behavioral patterns of individual users [11]. This work improves intent prediction based on our observation that the intended apps usually depend on 1) individual preference (some people prefer MESSAGE to EMAIL) and 2) behavioral patterns at the app level (MESSAGE is more likely to follow CAMERA, and EMAIL is more likely to follow EXCEL). Since behavioral contexts from previous turns affect intent prediction, we refer it as a *multi-turn interaction* task.

To improve understanding, some studies utilized non-verbal contexts like eye gaze and head nod as cues to resolve the referring expression ambiguity and to improve driving performance [77, 99]. Because human users often interact with their phones to carry out complicated tasks that span multiple domains and applications, user behavioral patterns as additional non-verbal signals may provide deeper insights into user intents [142, 24]. For example, if a user always texts his friend via MESSAGE instead of EMAIL right after finding a good restaurant via YELP, this behavioral pattern helps disambiguate apps corresponding to the communicating utterance "*send to alex*".

Another challenge of SLU is inference of hidden semantics, which is mentioned in the previous chapter. Considering a user utterance "*i would like to contact alex*", we can see that its surface patterns includes explicit semantic information about "*contact*"; however, it also includes hidden semantic information such as "*message*" and "*email*", since the user is likely

[1]In the rest of the document, we use the word "app" in stead of the word "application" for simplification.

to launch some apps such as MESSENGER (message) or OUTLOOK (email) even though they are not directly observed in the surface patterns. Traditional SLU models use discriminative classifiers to predict whether predefined slots occur in the utterances or not and ignore hidden semantic information. However, in order to provide better interactions with users, modeling hidden intents helps predict user-desired apps. Therefore, this chapter proposes a feature-enriched MF model to learn low-ranked latent features for SLU [98]. Specifically, an MF-SLU model is able to learn relations between observed features and unobserved features, and estimate probabilities of all unobserved patterns instead of viewing them as negative instances as described in Chapter 6. Therefore, the feature-enriched MF-SLU incorporates rich features, including semantic and behavioral cues, to infer high-level intents.

For the single-turn request task, the model takes account of app descriptions and spoken utterances along with enriched semantic knowledge in a joint fashion. More specifically, we use entity linking methods based on structured knowledge resources, which are to locate slot fillers in a given utterance, and then types of identified fillers are extracted as semantic seeds to enrich the features of the utterance. In additional to slot types, low-level semantics of the utterance is further enriched with related knowledge that is automatically extracted through neural word embeddings. Then applying feature-enriched MF-SLU enables an SDS to dynamically support non-predefined domains based on the semantics-enriched models. We evaluate the performance by examining whether predicted apps are capable of fulfilling users' requests.

For the multi-turn interaction task, the model additionally incorporates contextual behavior history to improve intent prediction. Here we take personal app usage history into account, where the behavioral patterns are used to enrich utterance features, in order to infer user intents better. Finally the system is able to provide behavioral and context-aware personalized prediction by feature-enriched MF techniques. To evaluate the personalized performance, we examine whether predicted apps are what the users actually launch.

We evaluate the performance by examining whether predicted applications can satisfy users' requests. The experiments show that our MF-based approach can model user intents and allow an SDS to provide better responses for both unsupervised single-turn requests and supervised multi-turn interactions. Our contributions include:

- This is among the first attempts to apply feature-enriched MF techniques for intent modeling, incorporating different sources of rich information (app description, semantic knowledge, behavioral patterns);
- The feature-enriched MF-SLU approach jointly models spoken observations, available text information, and structured knowledge to infer user intents for single-turn requests, taking hidden semantics into account;

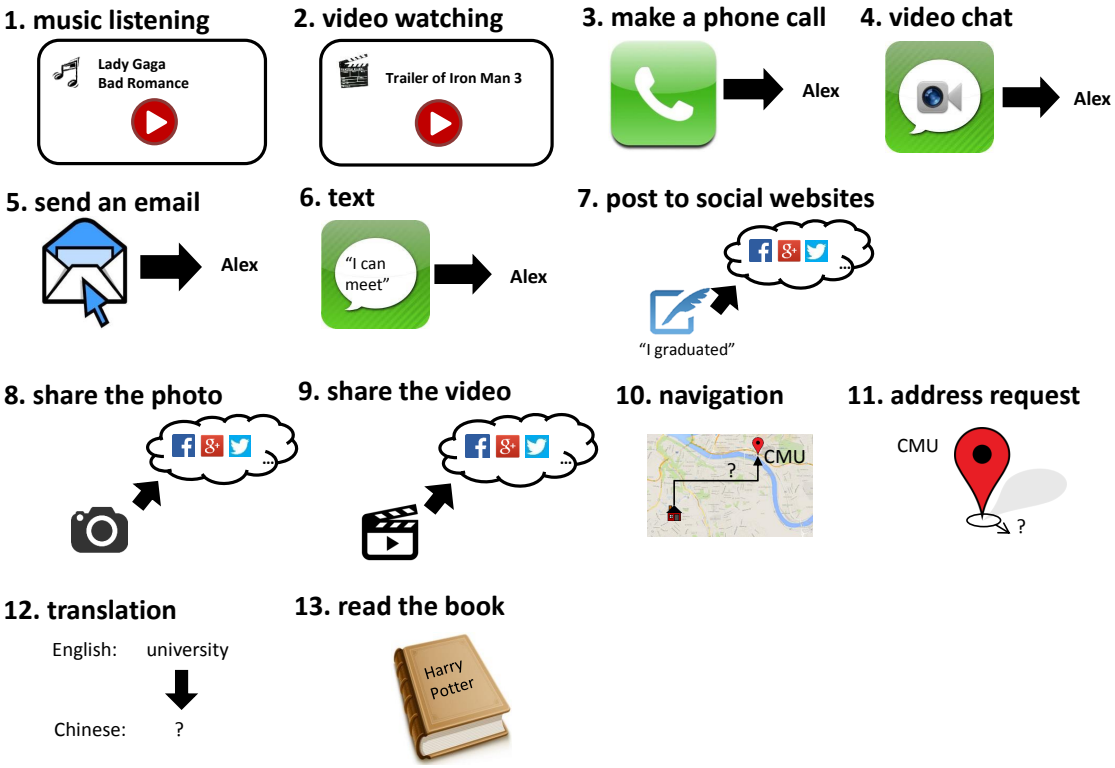


Figure 7.1: Total 13 tasks in the corpus (only pictures are shown to subjects for making requests).

- The behavioral patterns can be incorporated into the feature-enriched MF-SLU approach to model user preference for personalized understanding in multi-turn interactions;
- Our experimental results indicate that feature-enriched MF-SLU approaches outperform most strong baselines and achieve better intent prediction performance.

7.2 Data Description

The data for intent prediction focuses on mobile app interactions, where there are two datasets, *single-turn request* and *multi-turn interaction*. The detail is described below.

7.2.1 Single-Turn Request for Mobile Apps

Considering to expand useful domains of SDS, we extract the most popular apps from mobile app stores for defining important domains users tend to access frequently, and the defined

Table 7.1: The recording examples collected from some subjects for single-turn requests.

ID	Task Description	Utterance Transcript
3	Phone Call	*please dial a phone call to alex* *can i have a telcon with alex*
10	Navigation	*how can i go from my home to cmu* *i'd like navigation instruction from my home to cmu*

domains are used to design the experiments for this task. Figure 7.1 shows total 13 domains we define for experiments. The speech data is collected from 5 non-native subjects (1 female and 4 males). They are only provided with pictures referring to domain-specific tasks in a random order. For each picture/task, a subject is asked to use 3 different ways to make requests for fulfilling the task implied by the displayed picture, and then subjects are asked to manually annotate apps from Google Play[2] that can support the corresponding tasks. These annotated apps are treated as our ground truth for evaluation.

Thus 39 utterances (total 13 tasks and 3 ways for each) are collected from each subject. Figured 7.1 shows provided pictures and implied tasks, and some recording examples are shown in Table 7.1. The corpus contains 195 utterances. An ASR system was used to transcribe speech into text, and WER is reported as 19.8%. Here we use Google Speech API to perform better recognition results because it covers more named entities, which may be out-of-vocabulary words for most recognizers. The average number of words in an utterance is 6.8 for ASR outputs and 7.2 for manual transcripts, which implies the challenge of retrieving relevant apps with limited information in a short spoken request.

The data to populate the database was collected from Google Play in November 2012. Each Android app in Google Play has its own description page, and the extracted metadata includes its name, number of downloads, and content description[3]. Total 140,854 apps were available[4]; only 1,881 apps that have more than one million downloads were considered to return to users.

7.2.2 Multi-Turn Interaction for Mobile Apps

To understand how user spoken language produced in the course of multi-app tasks might be modeled, we conducted a longitudinal study with 14 participants to investigate how people structure and perform such tasks via speech [37]. We investigated users' behaviors across multiple domains/apps. An Android logging app was installed on users' phones to record app

[2] `https://play.google.com/store`

[3] Google does not provide the absolute number of downloads. Instead, it discretizes this number into several ranges.

[4] Google defines two major categories for the programs, "game" and "application". This paper only uses apps with category "application".

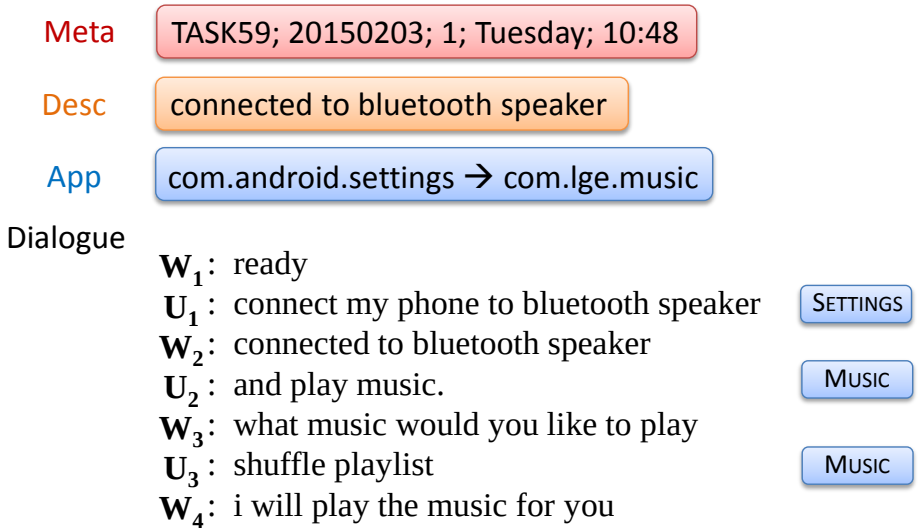


Figure 7.2: The dialogue example for multi-turn interaction with multiple apps

invocations, time and location (users had the ability to drop sensitive personal activities). Participants came in regularly to annotate their activities including [146]:

1. **Task Structure** — link apps that served a common goal.

2. **Task Description** — type in a brief description of the goal or intention of the task.

For example, in the upper part of Figure 7.2, SETTINGS and MUSIC are linked together since they were used for the goal of "*play music via bluetooth speaker*".

Then users were later shown tasks that they had annotated earlier along with the meta-data (date, location, and time), the task description they furnished earlier, and the apps that had been grouped (Meta, Desc, App lines in Figure 7.2). They were asked to use a WoZ system to repeat the same task via speech. The wizard arrangement was not concealed and the human wizard was in the same space (albeit not directly visible). The wizard was instructed to respond directly to participants' goal-directed requests and to not accept out-of-domain inputs. Participants were informed that they do not need to follow the order of the apps used on their smart-phones. Other than for being on-task, we did not constrain what users could say.

Conversations between users (U) and the wizard (W) were recorded, segmented into utterances and transcribed by both a human and a cloud speech recognizer. An example dialogue is shown in the lower part of Figure 7.2. Each user utterance was further associated with the apps that are able to handle it. As in Figure 7.2, SETTINGS would deal with the utterance

"*connect my phone to bluetooth speakers*" (U_1) and MUSIC would take care of music-related utterances such as "*and play music*" (U_2) and "*shuffle playlist*" (U_3).

The smart-phone app usage data was collected from 14 participants with Android OS version 4. This included 533 multi-app spoken dialogs with 1607 utterances (about 3 user utterances per dialogue). Among these dialogues, there are 455 multi-turn dialogues (82.3%), which provides behavioral information for intent prediction. With Google Speech API, the WER is 22.7%[5].

7.3 Feature-Enriched MF-SLU

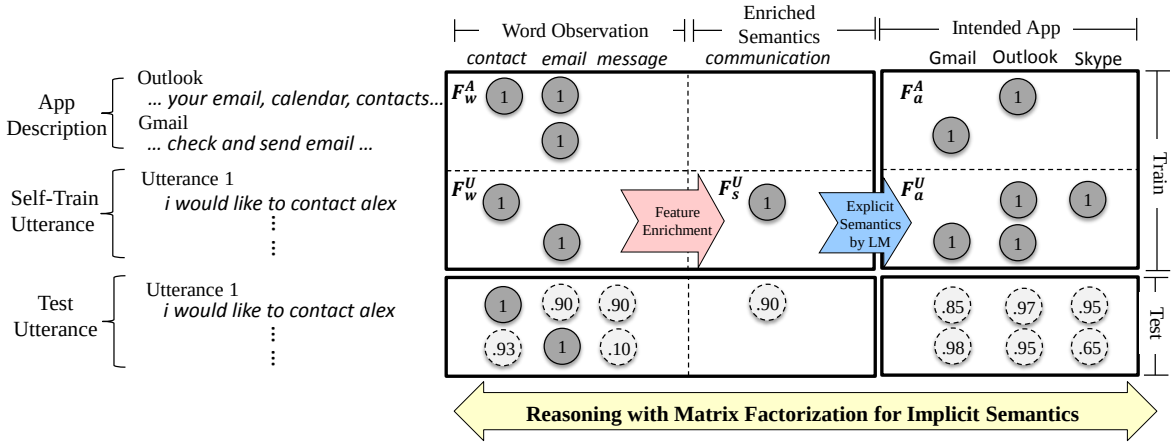
Under the app-oriented SDS, the main idea is to predict user intents along with corresponding apps. For single-turn requests, given a user utterance, how can an SDS dynamically support functions corresponding to requests beyond predefined domains in an unsupervised manner [28]? For multi-turn interactions, the goal is to predict the apps that are more likely to be used to handle the user requests given input utterances and behavioral contexts, considering not only the desired functionality but also user preference. We build an SLU component to model user intents: we frame the task as a multi-class classification problem, where we estimate the probability of each intent/app a given an utterance u, $P(a \mid u)$, using a proposed feature-enriched MF approach.

As Chapter 6 described, an MF model considers the unobserved patterns and estimates their probabilities instead of viewing them as negative, allowing it to model the implicit information [38]. Due to several advantages of MF techniques, such as modeling noisy data, hidden semantics, and long-range dependencies, an MF approach is applied to intent modeling. First we define $\langle x, y \rangle$ as a *fact*, which refers to an entry in a matrix. The input of our model is a set of observed facts $\mathcal{O}$, and the observed facts for a given utterance is denoted by $\{\langle x, y \rangle \in \mathcal{O}\}$. The goal of our model is to estimate, for a given utterance x and an app-related intent y, the probability, $P(M_{x,y} = 1)$, where $M_{x,y}$ is a binary random variable that is true if and only if y is the app for supporting the utterance x. Similarly, a series of exponential family models are introduced to estimate the probability using a natural parameter $\theta_{x,y}$ and the logistic sigmoid function:

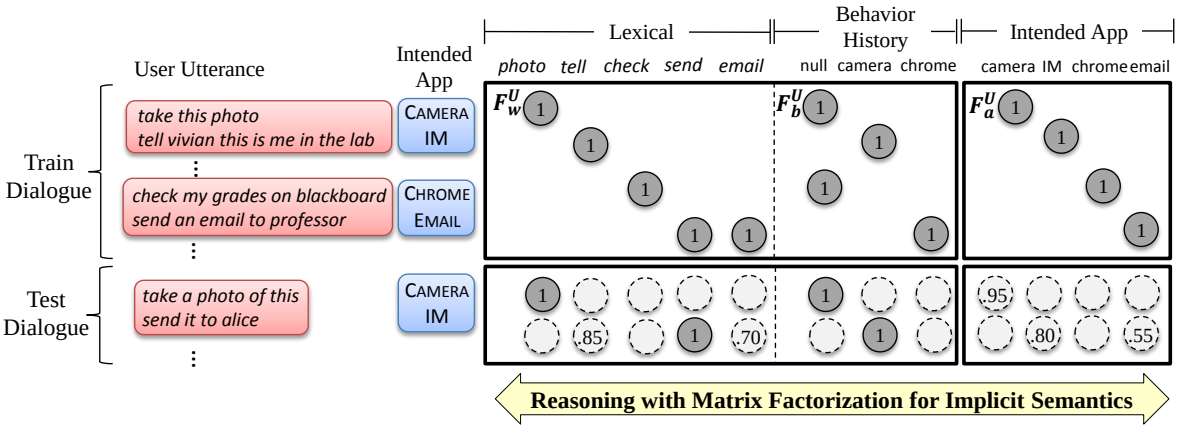
$$P(M_{x,y} = 1 \mid \theta_{x,y}) = \sigma(\theta_{x,y}) = \frac{1}{1 + \exp(-\theta_{x,y})} \tag{7.1}$$

For single-turn requests and multi-turn interactions, we construct a matrix M as observed facts using different types of enriched features, and the matrix can then be factorized by a matrix completion technique with the assumption that the matrix is low-rank.

[5]The dataset is available at `http://AppDialogue.com`.



(a) The feature matrix for single-turn requests incorporates app descriptions, spoken contents, automatically acquired knowledge, and pseudo relevant intents in a joint fashion.



(b) The feature matrix for multi-turn interactions incorporates lexical and behavioral patterns to build the personalized model.

Figure 7.3: The feature-enriched MF method completes a partially-missing matrix to factorize the low-rank matrix for implicit information modeling. Dark circles are observed facts, and shaded circles are latent and inferred facts. Reasoning with MF considers latent semantics to predict intents based on rich features corresponding to the current user utterance.

Algorithm 2: Semantics Enrichment Procedure

Data: a word observation set W in the utterance; a vocabulary V; the word relatedness function $f_s(\cdot)$
Result: a set of enriched semantics S
initializating $S^* = \{\}$;
for $w \in W$ **do**
    Extracting the words with similarity higher than a threshold from the vocabulary, $V^* = \{v \mid f_s(w, v) \geq \delta, v \in V\}$;
    Enrich the semantic set $S^* \leftarrow S^* \cup V^*$;
end
$S \leftarrow S^*$;

7.3.1 Feature-Enriched Matrix Construction

For each of single-turn request and multi-turn interaction tasks, we construct a feature-enriched matrix below. The illustration of two matrices is shown in Figure 7.3. They are enriched with various modalities. For unsupervised single-turn requests, the matrix in Figure 7.3a incorporates word observations, enriched semantics, and pseudo relevant apps for intent modeling. For supervised multi-turn interactions, the matrix in Figure 7.3b models word observations and contextual behavior for intent prediction. Below we use a similar MF framework, which contains three sets of information, lexical features (word observation matrix), low-level semantic features (enriched semantics matrix), and high-level intent results (intent matrix), to model user intents [42].

7.3.1.1 Word Observation Matrix

A word observation matrix features with binary values based on n-gram word patterns. For single-turn requests, two word observation matrices are built, where F_w^A is for textual app descriptions and F_w^U is for spoken utterances. Each row in the matrix represents an app/utterance and each column refers to an observed word pattern. In other words, F_w^A and F_w^U carry basic word vectors for all apps and all utterances respectively. Similarly, for multi-turn interactions, a word observation matrix, F_w^U, is constructed for spoken utterances. The left-most column set in Figure 7.3 illustrates lexical features for the given utterances.

7.3.1.2 Enriched Semantics Matrix

For single-turn requests, considering to include open domain knowledge based on user utterances, we utilize distributed word representations to capture syntactic and semantic relationship for acquiring domain knowledge [28, 33].

- Embedding-based semantics
  We enrich original utterances with semantically similar words, where the similarity is measured by word embeddings trained on all app descriptions [28, 118]. Algorithm 2 shows the procedure of acquiring domain knowledge for semantics enrichment. This procedure is to obtain semantically related knowledge for enriching original utterances. For example, "*compose an email to alex*" focuses on the email writing domain, and generated semantic seeds may include the tokens "*text*" and "*contacting*". Then word embeddings can help provide additional words with similar concepts. For example, the nearest vectors of "*text*" include "*message*", "*msg*", etc. in the continuous space. This procedure leverages rich semantics by incorporating conceptually related knowledge, so that the system can provide proper apps for supporting open domain requests.

- Type-embedding-based semantics
  In addition to semantically similar words, types of concepts are included to further expand semantic information. For example, "*play lady gaga's bad romance*" may contain the types "singer" and "song" to improve semantic inference (domain-related cues about music playing), where we detect all entity mention candidates in the given utterances and use entity linking with Freebase and Wikipedia to mine entity types [28].

  – Wikipedia page linking
    For an entity mention set in the given utterance, we output a set of linked Wikipedia pages, where an integer linear programming (ILP) formulation to generate the mapping from mentions to Wikipedia pages [43, 133]. For each entity, we extract the definition sentence from the linked page, and then all words parsed into adjectives or nouns in the noun phrase just following the part-of-speech pattern (VBZ) (DT) such as "*is a/an/the*" are extracted as semantic concepts. For example, the sentence about the entity "*lady gaga*" is "*Stefani Joanne Angelina Germanotta, better known by her stage name Lady Gaga, is an American singer and songwriter.*", and the entity types, "*American singer*" and "*songwriter*", are extracted.

  – Freebase list linking
    Each mention can be linked to a ranked list of Freebase nodes by Freebase API[6], and we extract the top K notable types for each entity as the acquired knowledge.

  Then an enriched semantics matrix can be built as F_s^U, where each row is a utterance and each column corresponds a semantic element shown in Figure 7.3.

For multi-turn interactions, we enrich utterances with contextual behaviors to incorporate behavioral information into personalized and context-aware intent modeling. Figure 7.3b

[6] `https://developers.google.com/freebase/`

illustrates the enriched behavioral features as F_b^U, where the second utterance "*tell vivian this is me in the lab*" involves "CAMERA" acquired from the previous turn "*take this photo*". The behavioral history at turn t, h_t, can be formulated as $\{a_1, ..., a_{t-1}\}$, which is a set of apps that were previously launched in the ongoing dialogue.

7.3.1.3 Intent Matrix

To link word patterns with corresponding intents, an intended app matrix F_a^A is constructed, where each column corresponds to launching a specific app. Hence, the entry is 1 when the app and the intent correspond to each other, and 0 otherwise,

For unsupervised single-turn requests, to induce user intents, we use a basic retrieval model for returning top K relevant apps for each utterance u, and treat them as pseudo intended apps [28], which is detailed in Section 7.4.1. Figure 7.3a includes an example of utterance "*i would like to contact alex*", where the utterance is treated as a request to search for relevant apps such as "OUTLOOK" and "SKYPE". Then we build an app matrix F_a^U with binary values based on the top relevant apps, which also denotes intent features for utterances. Note that we do not use any annotations, the app-related intents are returned by a retrieval model and may contain some noises.

For personalized intent prediction on multi-turn interactions, the intent matrix can be directly acquired from users' app usage logs. F_a^U can be built and illustrated in the right part of matrix from Figure 7.3b.

7.3.1.4 Integrated Model

As shown in Figure 7.3, we integrate word matrices, an enriched semantics matrix, and intent matrices from both apps and utterances together for training an MF-SLU model. The integrated model for single-turn requests can be formulate as

$$M = [\ \begin{matrix} F_w^A & 0 & F_a^A \\ F_w^U & F_s^U & F_a^U \end{matrix}\ ]. \tag{7.2}$$

Similarly, the integrated matrix for multi-turn interactions can be built as

$$M = [\ F_w^U \quad F_s^U \quad F_a^U\ ]. \tag{7.3}$$

Hence, the relations among word patterns, domain knowledge, and behaviors can be automatically inferred from the integrated model in a joint fashion. The goal of the feature-enriched MF-SLU model is, for a given user utterance, to predict the probability that the user intents

to launch each app.

7.3.2 Optimization Procedure

With the built matrix, M, we can learn a model θ^* that can best estimate the observed patterns by parametrizing the matrix through weights and latent component vectors, where the parameters are estimated by maximizing the log likelihood of observed data similar to (6.5) [46].

$$\begin{aligned}\theta^* &= \arg\max_{\theta} \prod_{x \in X} p(\theta \mid M_x) \\ &= \arg\max_{\theta} \prod_{x \in X} p(M_x \mid \theta) \cdot p(\theta) \\ &= \arg\max_{\theta} \sum_{x \in X} \ln p(M_x \mid \theta) - \lambda_\theta,\end{aligned} \tag{7.4}$$

where X is a set indicating row information. For single-turn requests, M_x is a row vector corresponding either an app or an utterance; for multi-turn interactions, M_x corresponds to an utterance. Here the assumption is that each row (app/utterance) is independent of others.

Similar to Chapter 6, we apply BPR to parameterize the integrated model, and create a dataset by sampling from M. In single-turn requests, for each app/utterance x and each observed fact $f^+ = \langle x, y^+ \rangle$, we choose each feature y^- referring to the word/semantics that does not correspond to x or the app that is not returned as by the basic retrieval model according to the utterance x. Also, in multi-turn interactions, for each utterance x (e.g. "*take this photo*" in Figure 7.3b) we created pairs of observed and unobserved facts: $f^+ = \langle x, y^+ \rangle$ and $f^- = \langle x, y^- \rangle$, where y^+ corresponds to an observed lexical/behavioral/intent feature (e.g., "*photo*" for lexical, "*null*" for behavior, "*camera*" for intended app) and y^- corresponds to an unobserved feature (e.g. "*tell*", "*camera*", "*email*"). Then for each pair of facts f^+ and f^-, we maximize the margin between $p(f^+)$ and $p(f^-)$ with the objective:

$$\sum_{x \in X} \ln p(M_x \mid \theta) = \sum_{f^+ \in \mathcal{O}} \sum_{f^- \notin \mathcal{O}} \ln \sigma(\theta_{f^+} - \theta_{f^-}). \tag{7.5}$$

The BPR objective is an approximation to the per utterance AUC, which correlates with well-ranked apps per utterance. The fact pairs constructed from sampled observed facts $\langle x, y^+ \rangle$ and unobserved facts $\langle x, y^- \rangle$ form $|\mathcal{O}|$, and an SGD update is applied to maximize the objective for MF [68, 134].

Finally we can obtain estimated probabilities of various features given the current utterance, which corresponds to probabilities of intended apps given the utterance, $P(a \mid u)$. For single-turn requests, Figure 7.3a shows that hidden semantics, "*message*" and "*email*", are inferred from "*i would like to contact alex*" because relations between features are captured by the

model based on app descriptions and previous user utterances. For multi-turn interactions, as shown in Figure 7.3b, hidden semantics (e.g. "*tell*", "*email*") can also be inferred from "*send it to alice*" in this model based on both lexical and behavioral features.

7.4 User Intent Prediction for Mobile App

For each test utterance u, with the trained MF model, we can predict the probability of each intended app a based on the observed features corresponding to the current utterance by taking into account two models, a baseline model for explicit semantics and a feature-enriched MF-SLU model for implicit semantics:

$$\begin{aligned} P(a \mid u) &= P_{\text{exp}}(a \mid u) \times P_{\text{imp}}(a \mid u) \\ &= P_{\text{exp}}(a \mid u) \times P(M_{u,a} = 1 \mid \theta), \end{aligned} \tag{7.6}$$

where $P(a \mid u)$ is an integrated probability for ranking apps, $P_{\text{exp}}(a \mid u)$ is the probability outputted by the baseline model that considers explicit semantics, and $P_{\text{imp}}(a \mid u)$ is the probability estimated by the MF-based model. The fused probabilities are able to consider hidden intents by learning latent semantics from the enriched features. The baselines for modeling explicit semantics $P_{\text{exp}}(a \mid u)$ for single-turn requests and multi-turn interactions are described below.

7.4.1 Baseline Model for Single-Turn Requests

Considering an unsupervised task of ranking apps based on user spoken requests, the baselines for modeling explicit semantics $P_{\text{exp}}(a \mid u)$ for single-turn requests apply a language modeling retrieval technique for query likelihood estimation, and app-related behaviors are ranked by

$$\begin{aligned} P_{\text{exp}}(a \mid u) &= \frac{P(u \mid a)P(a)}{P(u)} \\ &\propto P(u \mid a) = \frac{1}{|u|} \sum_{w \in u} \log P(w \mid a), \end{aligned} \tag{7.7}$$

where u is a user's query, a is an app-related intent, w represents a token in the query, and $P(u \mid a)$ represents the probability that user speaks the utterance u to make the request for launching the app a[7] [28, 130]. For example, in order to use the app Gmail, a user is more likely to say "*compose an email to alex*", while the same utterance should correspond to a lower probability when launching the app Maps. To estimate the likelihood by the language modeling approach, we use the description content of this app with assumption that it carries

[7]Here we assume that the priors for apps/utterances are the same.

semantically related information. For example, the description of GMAIL includes a text segment "*read and respond to your conversations*", and MAPS includes texts "*navigating*" and "*spots*" in its description.

7.4.2 Baseline Model for Multi-Turn Interactions

In multi-turn interactions, the goal is to predict the apps that are more likely to be used to handle user requests given input utterances and behavioral contexts, considering not only the desired functionality but also user preference. Considering that multi-app tasks are often user-specific, we want to build a personalized SLU component for each user to model his/her intents; we frame the task as a multi-class classification problem, and perform a standard MLR to model explicit semantics, where the model uses a standard maximum likelihood estimation approach by the gradient ascent to estimate the likelihood $P_{\text{exp}}(a \mid u)$.

7.5 Experimental Setup

In single-turn requests, we train a word embedding model to enrich utterances with domain knowledge for semantics enrichment, which is described in Section 7.5.1. To build an intent matrix, a retrieval model is applied to acquire pseudo intended apps. This procedure is detailed in Section 7.5.2. In multi-turn interactions, we group 70% of each user's multi-app dialogues (chronologically ordered) into a training set, and use the rest 30% as a testing set. In the experiments, we then build a user-independent and user-dependent (personalized) SLU model to estimate the probability distribution of all apps. All experiments are performed on both ASR and manual transcripts as follows.

7.5.1 Word Embedding

To include distributional semantics for SLU, we use description contents of all apps to train word embeddings using CBOW, which predicts the current word based on the context[8]. The resulting vectors have dimensionality 300, and vocabulary size is 8×10^5 [118].

7.5.2 Retrieval Setup

Lemur toolkit[9] is used to perform our retrieval model for ranking apps. For the retrieval setting, word stemming[10] and stopword removal[11] are applied, and we assign an equal weight

[8] https://code.google.com/p/word2vec/
[9] http://www.lemurproject.org/
[10] http://tartarus.org/martin/PorterStemmer/
[11] http://www.lextek.com/manuals/onix/stopwords1.html

to each term in the query to eliminate the influence of weighting [130]. To further consider popularity of apps, for each returned list, we rerank "popular" apps to the top of this list, where "popular" means the apps with more than ten million downloads, because we assume that users are more willing to use/install popular apps, and also our ground truth is based on subjects' annotations, where most reference apps belong to the set of popular apps we define.

7.6 Results

Under the app-oriented SDS, the main idea is to model users' intents; therefore, we evaluate the model performance by measuring whether the predicted apps satisfy users' need. For single-turn requests, we use subject-labeled apps as our ground truth for evaluating our returned apps, where we use standard metrics of information retrieval, MAP and P@10 in the experiments. Similarly, for multi-turn interactions, we also evaluate the performance by MAP to consider the whole ranking list of all apps corresponding to each utterance. Considering that the multi-turn interaction task is supervised, we additionally report turn accuracy (ACC) as the percentage of our top-1 predictions that match the correct apps, which is exactly the same as P@1.

7.6.1 Results for Single-Turn Requests

We evaluated the proposed feature-enriched MF approach for single-turn requests in Table 7.2 and Table 7.3, which present the results using different features before and after integrating with the feature-enriched MF-SLU model for ASR and manual transcripts.

Table 7.2 shows that almost all results are improved after combining with the MF model, indicating that hidden semantics modeled by MF techniques helps estimate intent probabilities. For ASR results, enriching semantics using embedding-based (row (b)) and type-embedding-based semantics (row (c)) significantly improve the baseline performance (row (a)) using a basic retrieval model, where the MAP performance is increased from 25.1% to 31.5%. Then the performance can be further improved by integrating MF to additionally model hidden semantics, where row (b) achieves 34.2% on MAP. The reason why type-embedding-based semantics (row (c)) does not perform better compared with embedding-based semantics (row (b)) is that the automatically acquired type information appears to introduce noises, and row (c) is slightly worse than row (b) for ASR results.

For manually transcribed speech in Table 7.2, the semantic enrichment procedure and MF-SLU models also improve the performance. Different from ASR results, the best result for user intent prediction is based on the features enriched with type-embedding-based semantics (row (c)), achieving 34.0% on MAP. The reason may be that manual transcripts are more likely to

Table 7.2: User intent prediction for single-turn requests on MAP using different training features (%). LM is a baseline language modeling approach that models explicit semantics.

Feature for Single-Turn Request		ASR		Transcripts	
		LM	w/ MF-SLU	LM	w/ MF-SLU
(a)	Word Observation	25.1	29.2 (+16.2%)	26.1	30.4 (+16.4%)
(b)	Word + Embedding Semantics	32.0	**34.2** (+6.8%)	33.3	33.3 (-0.2%)
(c)	Word + Type-Embedding Semantics	31.5	32.2 (+2.1%)	32.9	**34.0** (+3.4%)

Table 7.3: User intent prediction for single-turn requests on P@10 using different training features (%). LM is a baseline language modeling approach that models explicit semantics.

Feature for Single-Turn Request		ASR		Transcripts	
		LM	w/ MF-SLU	LM	w/ MF-SLU
(d)	Word Observation	28.6	29.5 (+3.4%)	29.2	30.1 (+2.8%)
(e)	Word + Embedding Semantics	31.2	**32.5** (+4.3%)	32.0	33.0 (+3.4%)
(f)	Word + Type-Embedding Semantics	31.3	30.6 (-2.3%)	32.5	**34.7** (+6.8%)

capture the correct semantic information by word embeddings and have more consistent type information, allowing the MF technique to model user intents better and more accurate.

Also, Table 7.3 shows the similar trends on P@10, where row (e) with MF is best result for ASR, and row (f) is best results for manual transcripts. In sum, the results show that rich features carried by app descriptions and utterance-related contents can help intent prediction in single-turn requests using the proposed MF-SLU model for most cases. The evaluation results also prove the effectiveness of our feature-enriched MF-SLU models, which incorporate enriched semantics and model implicit semantics along with explicit semantics in a joint fashion, demonstrating promising performance.

7.6.2 Results for Multi-Turn Interactions

To analyze whether contextual behaviors convey informative cues for personalized intent prediction, Table 7.4 and Table 7.5 show the personalized SLU performance using lexical and behavioral features individually on both ASR and manual transcripts, where we build an SLU model for each subject and then predict intents of given utterances using the corresponding user-dependent models.

For the baseline MLR that models explicit semantics, we can see that lexical features alone (row (a)) achieve better performance than behavioral features alone (row (b)), indicating that the majority of utterances contains explicit expressions that are predictable. In addition, combining behavior history patterns with lexical features (row (c)) performs best in terms of the two measures. This implies that users' personal app usage patterns can improve prediction

Table 7.4: User intent prediction for multi-turn interactions on MAP (%). MLR is a multi-class baseline for modeling explicit semantics. † means that all features perform significantly better than lexical/behavioral features alone; § means that integrating using MF can significantly improve the MLR model (t-test with $p < 0.05$).

Feature for Multi-Turn Interaction		ASR		Transcripts	
		MLR	w/ MF-SLU	MLR	w/ MF-SLU
(a)	Word Observation	52.1	52.7 (+1.2%)	55.5	55.4 (-0.2%)
(b)	Behavioral Pattern	25.2	26.7 (+6.0%§)	25.2	26.7 (+6.0%§)
(c)	Word + Behavioral Features	53.9†	**55.7**† (+3.3%§)	56.6	**57.7**† (+1.9%§)

Table 7.5: User intent prediction for multi-turn interactions on turn accuracy (%). MLR is a multi-class baseline for modeling explicit semantics. † means that all features perform significantly better than lexical/behavioral features alone; § means that integrating using MF can significantly improve the MLR model (t-test with $p < 0.05$).

Feature for Multi-Turn Interaction		ASR		Transcripts	
		MLR	w/ MF-SLU	MLR	w/ MF-SLU
(d)	Word Observation	48.2	48.3 (+0.2%)	51.6	51.3 (-0.6%)
(e)	Behavioral Pattern	19.3	20.6 (+6.7%)	19.3	20.6 (+6.7%)
(f)	Word + Behavioral Features	50.1†	**51.9**† (+3.6%§)	52.8	**54.0**† (+2.3%)

by allowing the model to capture dependencies between lexical and behavioral features.

To evaluate the effectiveness of modeling latent semantics via MF, we can find that the performance of the MLR model integrated with MF-SLU estimation. For ASR transcripts, combining MLR with MF-SLU outperforms all MLR models (significant improvement with $p < 0.05$ in t-test), because MF is able to model latent semantics under lexical and behavioral patterns. For manual results, integrating with the MF-SLU model does not perform better when using only lexical features (row (a)), possibly because manually transcribed utterances contain more clear and explicit semantics, so that learning latent semantics does not improve the performance. However, using behavioral features or using all features show significantly better performance when combining with the proposed MF-based method.

Table 7.5 shows the similar trend as Table 7.4, where the improvement of ACC is more than the improvement of MAP, showing that the model produces accurate prediction of the top returned intent. Comparing results for ASR and manual transcripts, we observed that the performance is worse when user utterances contain recognition errors, possibly because the explicit expression contains more noises and thus results in worse prediction. The benefit of MF techniques on ASR results is more pronounced (3.3% and 3.6% relative improvement of MAP and ACC respectively) than for manual transcripts, showing the effectiveness of MF in modeling latent semantics in noisy data.

Finally, our experiments show the feasibility of disambiguating spoken language inputs for better intent prediction using behavioral patterns. The best prediction on ASR reaches 55.7% of MAP and 51.9% of ACC.

7.6.3 Comparing between User-Dependent and User-Independent Models

To deeply investigate the performance for personalized models, we conduct experiments to compare the difference between user-independent models and user-dependent models in Table 7.6 and Table 7.7.

The first two rows are baseline results using maximum likelihood estimation (MLE), which predict the intended apps based on the observed app distribution. Here it can be found that the user-dependent model (row (b)) performs better than the user-independent model (row (a)). Note that MLE only considers the observed frequency of app usage, so there is no difference between ASR and manual results. The rows (c) and (d) use MLR to model explicit semantics, where the performance is significantly better than MLE, and user-dependent models (row (d)) still outperform user-independent models (row (c)) using lexical alone, behavioral alone, or combined features for both ASR and manual transcripts.

To analyze the capability of modeling hidden semantics, the rows (e) and (f) apply MF to model the implicit semantics before integrating with models about explicit semantics. Using an MF model alone does not perform better compared to MLR, because MF takes latent information into account and has weaker capability of modeling explicit observations. However, the performance of the user-independent model through behavioral patterns achieves 39.9% for both ASR and manual transcripts, but combining with lexical features only performs 21.2% and 19.8% for ASR and manual results. The results may be unreliable due to data sparsity in MF. Personalized results (row (f)) perform much better than user-independent models (row (e)), indicating that inference relations between apps are more obvious for individual users.

Then we investigate the performance using different combinations in rows (g) and (h), where row (g) combines user-independent MLR (row (c)) and personalized MF (row (e)), and row (h) combines personalized MLR (row (d)) and personalized MF (row (e)). By integrating personalized MF, both models are improved for ASR and manual transcripts. Between these two combinations, it can be found that utilizing user-specific data to model both explicit and implicit semantics (row (h)) is able to achieve the best performance, where the best results are 55.7% and 57.5% on ASR and manual transcripts respectively.

Table 7.7 shows the performance of turn accuracy, and almost all trends are the same, except for the user-independent MF model using only behavioral patterns (row (e)), which produces more reasonable results, 7.6% for both ASR and manual transcripts. The poor performance

Table 7.6: User intent prediction for multi-turn interactions on MAP for ASR and manual transcripts (%). † means that all features perform significantly better than lexical/behavioral features alone; § means that integrating with MF significantly improves the MLR model (t-test with $p < 0.05$).

Approach			ASR			Transcripts		
			Lex.	Behav.	All	Lex.	Behav.	All
(a)	MLE	User-Independent	19.6					
(b)		Personalized	27.9					
(c)	MLR	User-Independent	46.4	18.7	50.1†	51.3	18.7	53.1
(d)		Personalized	52.1	25.2	53.9†	55.5	25.2	56.6
(e)	MF-SLU	User-Independent	19.4	39.9	21.2	16.8	39.9	19.8
(f)		Personalized	29.8	43.8	29.8	30.8	43.8	31.5
(g)	(c) + Personalized MF-SLU		51.1	20.3	54.2†§	53.3	20.3	57.6†§
(h)	(d) + Personalized MF-SLU		52.7	26.7	**55.7**†§	55.4	26.7	**57.7**†§

Table 7.7: User intent prediction for multi-turn interaction on ACC for ASR and manual transcripts (%). † means that all features perform significantly better than lexical/behavioral features alone; § means that integrating with MF significantly improves the MLR model (t-test with $p < 0.05$).

Approach			ASR			Transcripts		
			Lex.	Behav.	All	Lex.	Behav.	All
(a)	MLE	User-Independent	13.5					
(b)		Personalized	20.2					
(c)	MLR	User-Independent	42.8	14.9	46.2†	47.7	14.9	48.8
(d)		Personalized	48.2	19.3	50.1†	51.6	19.3	52.8
(e)	MF-SLU	User-Independent	13.4	7.6	14.8	10.1	7.6	13.6
(f)		Personalized	21.7	16.5	21.8	23.3	16.5	24.2
(g)	(c) + Personalized MF-SLU		47.6	16.4	50.3†§	49.1	16.4	53.5†§
(h)	(d) + Personalized MF-SLU		48.3	20.6	**51.9**†§	51.3	20.6	**54.0**†

of the user-independent model is expectable, because different users usually have different preference and history usage. In terms of ACC, the best performance is 51.9% and 54.0% on ASR and manual results respectively, concluding that applying personalized SLU on explicit and implicit semantics is better.

7.7 Summary

This chapter presents a feature-enriched MF-SLU approach to learn user intents based on the automatically acquired rich features, in one case taking account into domain knowledge and in another case incorporating behavioral patterns along with user utterances. In a smart-phone intelligent assistant setting (e.g. requesting an app launch), the proposed model considers

implicit semantics to enhance intent inference given the noisy ASR inputs for single-turn request dialogues. The model is also able to incorporate users' behavioral patterns and their app preferences to better predict user intents in multi-turn interactions. We believe that the proposed approach allows systems to handle users' open domain intents when retrieving relevant apps that provide desired functionality either locally available or by suggesting installation of suitable apps and doing so in an unsupervised way. The framework can be extended to incorporate personal behavior history and use it to improve a system's ability to assist users to pursue multi-app activities. In sum, the effectiveness of the feature-enriched MF model can be shown in different domains, indicating good generality and providing a reasonable direction for future work.

8 SLU in Human-Human Conversations

> *We have already discovered how quickly we become dependent on the Internet and its applications for business, government and research, so it is not surprising that we are finding that we can apply this technology to enable or facilitate our social interactions as well.*

Vint Cerf, *"The father of the Internet" and Turing Award winner*

The recent successes of voice interactions with smart devices (human-machine genre) and improvements in speech recognition for conversational speech show the possibility of conversation-related applications. Previous chapters focus on supporting human-machine interactions by SDS. After building SDS to handle open-domain dialogues, with available human-machine data from IAs, this chapter investigates the task of SLU in human-human conversations (human-human genre). Specifically, a system is proposed to detect actionable items in conversations and dynamically provide participants access to information (e.g. scheduling a meeting, taking notes) without interrupting the dialogues. Furthermore, domain ontology induction can be improved by filtering out the utterances without actionable items, and then the refined ontology provides better domain knowledge for SLU modeling. The iterative framework shows the feasibility of continuously improving system performance from a practical perspective.

8.1 Introduction

Meetings pose unique knowledge sharing opportunities, and have been a commonly accepted practice to coordinate work of multiple parties in organizations. With the surge of smart phones, computing devices have been easily accessible and real-time information search has been a common part of regular conversations [20]. Furthermore, recent improvements in conversational speech recognition suggest the possibility of speech recognition and understanding on continual and background audio recording of conversations [112]. In meetings, discussions

me018: Have <from_contact_name>they</from_contact_name> ever responded to <contact_name>you</contact_name>? me011: Nope.	**find_email** action: check emails of me011, search for any emails from them
mn015: Yeah it's - or - or just - Yeah. It's also all on my - my home page at E_M_L. It's called "An Anatomy of afind Spatial Description". But I'll send <email_content>that link</email_content>.	**send_email** action: email all participants, "link to An Anatomy of Spatial Description"
mn015: I suggest w- to - for - to proceed with this in - in the sense that maybe, <date>throughout this week</date>, the <contact_name>three of us</contact_name> will - will talk some more about maybe segmenting off different regions, and we make up some - some toy a- observable "nodes" - is that what th-	**create_calendar_entry** action: open calendars of participants, marking times free for the three participants and schedule an event

Figure 8.1: The ICSI meeting segments annotated with actionable items. The triggered intents are at the right part along with descriptions. The intent-associated arguments are labeled within texts.

could be a rich resource for identifying participants' next actions and then assist with the action executions.

In this chapter, we investigate a novel task of actionable item detection in multi-party meetings, with the goal of providing participants easy access to information and performing actions that a personal assistant would handle without interrupting the meeting discussions. Actionable items in meetings would include discussions on scheduling, emails, action items, and search. Figure 8.1 shows some meeting segments from the ICSI meeting corpus, where actionable items and their associated arguments are annotated [92]. A meeting assistant would then take an appropriate action, such as opening the calendars of the involved participants for the dates being discussed, finding the emails and documents being discussed, or initiating a new one.

Most of the previous work on language understanding of human-human conversations focuses on analyzing task-oriented dialogues such as customer service conversations, and aims to infer semantic representations and bootstrap language understanding models [6, 31, 38, 39, 151]. These would then be used in human-machine dialogue systems that automate the targeted task, such as travel arrangements. In this work, we assume presence of task-oriented dialogue systems (human-machine genre), such as personal assistants that can schedule meetings and send emails, and focus on adapting such systems to aid users in multi-party meetings (human-human genre). Previous work on meeting understanding investigated detection of decisions [22, 62], action items [167], agreement and disagreements [67, 88], and summarization [27, 136, 165]. Our task is closest to detection of action items, where action items are considered as a subgroup of actionable items.

Utterances in the human-human genre are more casual and include conversational terms, but

Human-Machine Genre

create_calendar_entry schedule a meeting with <contact_name>John</contact_name> <start_time>this afternoon</start_time>

Human-Human Genre

create_calendar_entry how about the <contact_name>three of us</contact_name> discuss this later <start_time>this afternoon</start_time>?

Figure 8.2: The genre mismatched examples with the same action.

the terms related to the actionable item (arguments), such as dates, times, and participants are similar. Figure 8.2 shows genre-mismatched examples (human-machine v.s. human-human), where both utterances have the same action create_calendar_entry. The similarity between two genres suggests that the data available from human-machine interactions (source genre) can be useful in recognizing actionable items in human-human interactions (target genre). Furthermore, due to the mentioned differences, the use of adaptation methods could be promising.

In this work, we treat actionable item detection in meetings as a meeting utterance classification task, where each user utterance can trigger an actionable item. Recent studies used CDSSM to map questions into relation-entity triples for question answering, which motivates us to use CDSSM for learning relations between actions and their triggering utterances [169, 170]. Also, several studies investigated embedding vectors as features for training task-specific models, which can incorporate more informative cues from large data [8, 28, 33, 41, 69]. Hence, for utterance classification, this chapter first focuses on taking CDSSM features to help detect triggered actions. Second, embedding adaptation has been studied using different languages and external knowledge [59, 60]. Due to the genre mismatch, embedding adaptation is proposed to fit the target genre and provide additional improvement. Third, we investigate whether actionable cues help improve ontology induction, and propose an iterative framework for automating system learning. It can benefit both human-machine and human-human interactions.

8.2 Convolutional Deep Structured Semantic Models (CDSSM)

We describe how to train CDSSM for actionable item detection, whose architecture is described by Section 2.4.4.1.

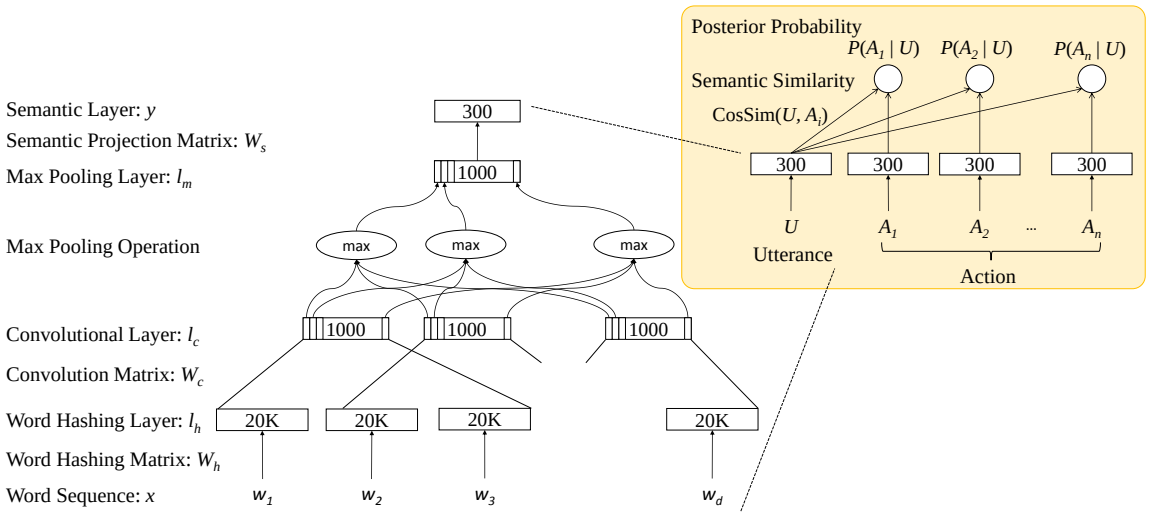


Figure 8.3: Illustration of the CDSSM architecture for the predictive model.

8.2.1 Architecture

The model is a deep neural network with a convolutional layer illustrated in Figure 8.3 [69, 90, 139, 140]. The model contains: 1) a word hashing layer obtained by converting one-hot word representations into tri-letter vectors, where the total number of tri-letters in our experiments is about 20.6K, 2) a convolutional layer that extracts contextual features for each word with its neighboring words defined by a window, 3) a max-pooling layer that discovers and combines salient features to form a fixed-length sentence-level feature vector, and 4) a semantic layer that further transform the max-pooling layer to a low-dimensional semantic vector for the input sentence. The final output semantic vector, y, can be either utterance embeddings y_U or action embeddings y_A.

8.2.2 Training Procedure

The meeting data contains utterances and associated actions. The idea of this model is to learn embeddings for utterances and actions such that utterances with same actions can be close to each other in the continuous space. Below we define the semantic score between an utterance U and an action A using the cosine similarity between their embeddings:

$$\text{CosSim}(U, A) = \frac{y_U \cdot y_A}{|y_U||y_A|}. \tag{8.1}$$

8.2.2.1 Predictive Model

The posterior probability of a possible action given an utterance is computed based on the semantic score through a softmax function,

$$P(A \mid U) = \frac{\exp(\text{CosSim}(U, A))}{\sum_{A'} \exp(\text{CosSim}(U, A'))}, \tag{8.2}$$

where A' is an action candidate.

For model training, we maximize the likelihood of the correctly associated actions given the utterances across the training set. The parameters of the model $\theta_1 = \{W_c, W_s\}$ is optimized by an objective:

$$\Lambda(\theta_1) = \log \prod_{(U,A^+)} P(A^+ \mid U). \tag{8.3}$$

The model is optimized using mini-batch SGD [90]. Then we can transform test utterances into vector representations.

8.2.2.2 Generative Model

Similarly, we can estimate the posterior probability of an utterance given an action using the reversed setting,

$$P(U \mid A) = \frac{\exp(\text{CosSim}(U, A))}{\sum_{U'} \exp(\text{CosSim}(U', A))}, \tag{8.4}$$

which is a generative model that emits utterances for each given action. Also, the model parameters θ_2 is optimized by an objective:

$$\Lambda(\theta_2) = \log \prod_{(U^+,A)} P(U^+ \mid A). \tag{8.5}$$

The model can be obtained similarly and performs a reversed estimation for the relation between utterances and actions.

8.3 Adaptation

Practically the data for the target genre may be unavailable or insufficient to train CDSSM, so there may be a mismatch between source and target genres. Based on the model trained on the source genre (θ_1 or θ_2), each utterance and action from the target genre can be transformed into a vector. Then it is possible that embeddings of the target data cannot accurately estimate the score $\text{CosSim}(U, A)$ due to the mismatch. Below we focus on adaptation approaches

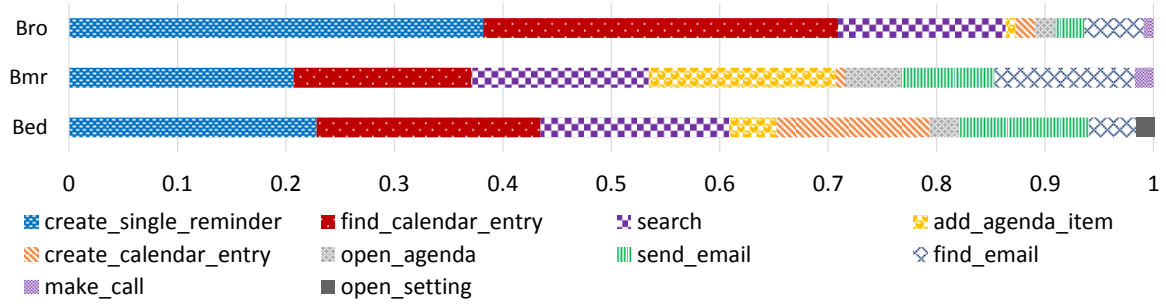


Figure 8.4: Action distribution for different types of meetings.

that adjust embeddings generated by the source genre to fit the target genre, where two adaptation approaches are proposed.

8.3.1 Adapting CDSSM

Because CDSSM is trained on a mismatched genre (human-machine genre), the CDSSM can be adapted by continually training the model using the data from the target genre (human-human genre) for several epochs (usually stop early before fully converged). Then the final CDSSM contains information about both genres, so it can be robust because of data from different genres and specific to the target genre.

8.3.2 Adapting Action Embeddings

Instead of adapting the whole CDSSM, this section applies an adaptation technique to directly learn adapted action embeddings that may be proper for the target genre. After converting actions and utterances from the target genre into vectors using CDSSM trained on the source genre, the idea here is to move the action embeddings based on the distribution of corresponding utterance embeddings, and then adjusted action embeddings can fit to the target genre better. A similar idea was used to adapt embeddings based on the predefined ontology [60, 173].

Here we define Q as a set of action embeddings and R as a set of utterance embeddings obtained from the trained model (θ_1 or θ_2). Then we define two objectives, Φ_{act} and Φ_{utt}, to model action and utterance embeddings respectively.

$$\Phi_{\text{act}}(\hat{Q}, \hat{R}) = \sum_{i=1}^{n} \left[\alpha_i |\hat{q}_i - q_i|^2 + \sum_{l(r_j)=i} \beta_{ij} |\hat{q}_i - \hat{r_j}|^2 \right], \tag{8.6}$$

$$\Phi_{\text{utt}}(\hat{R}) = \sum_{i:l(r_i)=1}^{n} \left[\alpha_i |\hat{r_i} - r_i|^2 + \sum_{l(r_j)=l(r_i)} \beta_{ij} |\hat{r_i} - \hat{r_j}|^2 \right], \tag{8.7}$$

where $q_i \in Q$ is the original action embedding of the i-th action, $r_i \in R$ is the original utterance embedding of the i-th utterance, and $l(\cdot)$ indicates the action label for an utterance. The idea here is to learn new action embeddings $\hat{q_i}$ that are close to q_i and the utterances labeled with the action i, $\hat{r_j}$. Also, Φ_{utt} suggests to learn new utterance embeddings $\hat{r_i}$ close to r_i and other utterances with the same action label. Here α and β control relative strengths of associations. An objective $\Phi(\hat{Q}, \hat{R})$ combines them together:

$$\Phi(\hat{Q}, \hat{R}) = \Phi_{\text{act}}(\hat{Q}, \hat{R}) + \Phi_{\text{utt}}(\hat{R}). \tag{8.8}$$

With an integrated objective $\Phi(\hat{Q}, \hat{R})$, the sets of adapted action embeddings and adapted utterance embeddings ($\hat{Q}$ and $\hat{R}$ respectively) can be obtained simultaneously by an efficient iterative updating method [9, 60]. The updates for $\hat{q_i}$ and $\hat{r_i}$ are:

$$\Delta\hat{q_i} = \frac{\alpha q_i + \sum \beta_{ij} \hat{r_j}}{\alpha + \sum \beta}, \Delta\hat{r_i} = \frac{\alpha r_i + \sum \beta_{ij} \hat{r_j}}{\alpha + \sum \beta}. \tag{8.9}$$

Then the adapted action embeedings Q are obtained in order to estimate better scores for the target domain. Below we use the notation $\hat{y_A}$ to refer to the adapted action embeddings.

8.4 Actionable Item Detection

In order to predict possible actions given utterances, for each utterance U, we transform it into a vector y_U, and then estimate the semantic similarity with vectors for all actions. For the utterance U, the estimated semantic score of the k-th action is defined as:

$$\widehat{\text{CosSim}}(U, A_k) = \frac{y_U \cdot \hat{y_{A_k}}}{|y_U||\hat{y_{A_k}}|}, \tag{8.10}$$

which is similar to (8.1), but replaces the original action embeddings y_A with the adapted embeddings $\hat{y_A}$. Note that the utterance embeddings are the original ones, so they can match the embeddings of test utterances.

The estimated semantic scores can be used in two ways [139]:

1. As final prediction scores: $\widehat{\text{CosSim}}(U, A)$ is directly treated as the prediction score of an actionable item detector.

2. As features of a classifier: $\widehat{\text{CosSim}}(U, A)$ is an input feature of a classifier and then a multi-class classifier can be trained as an actionable item detector. Then the trained

classifier outputs final prediction scores of actions given each test utterance for the detection task.

8.4.1 Unidirectional Estimation

With predictive and generative models from Section 8.2.2.1 and Section 8.2.2.2, here for the utterance U_i, we define the final prediction score of the action A_j using the predictive model as $S_P(i,j)$ and using the generative model as $S_G(i,j)$, where the prediction score can be obtained via above two ways.

8.4.2 Bidirectional Estimation

Considering that the estimation from two directions may model the similarity in different ways, we can incorporate bidirectional estimation by fusing prediction scores, $S_P(i,j)$ and $S_G(i,j)$, to balance the effectiveness between predictive and generative models.

$$S_{\text{Bi}}(i,j) = \gamma \cdot S_P(i,j) + (1-\gamma) \cdot S_G(i,j), \tag{8.11}$$

where γ is a weight to control contributions from both sides.

8.5 Iterative Ontology Refinement

We assume that utterances corresponding to the action create_single_reminder contain core information in the conversations, so inducing a domain ontology while considering these utterances in the mean time can focus more on important concepts and may benefit the performance of ontology induction. The ontology induction approach proposed by Chapter 3 is based on $w(s)$ in (2.7), where $w(s)$ is an importance measurement of the slot s. Therefore, we utilize $S_{\text{Bi}}(i,j)$, which estimates the probability of an actionable item as shown in (8.11), to adjust original frequency scores into $f'(s)$ for all slots in the conversation, and then update ranking scores based on weighted frequencies,

$$w'(s) = (1-\alpha) \cdot \log f'(s) + \alpha \cdot h(s), \tag{8.12}$$

$$f'(s) = \sum_i \Big(tf_s(i) \cdot S_{\text{Bi}}(i) \Big), \tag{8.13}$$

where $f'(s)$ is the weighted frequency of a slot s, $tf_s(i)$ is the term frequency of the slot s in the utterance U_i, and $S_{\text{Bi}}(i)$ can be viewed as the probability that the utterance U_i refers to create_single_reminder via bidirectional estimation. Specifically, $S_{\text{Bi}}(i)$ is equal to $S_{\text{Bi}}(i,j)$ in

(8.11), which estimates the probability that the utterance U_i contains the action A_j (A_j is create_single_reminder). Below we use $S_{\text{Bi}}(i)$ for simplification.

In order to avoid decreasing weights for slots in actionable utterances, we proposed two types of estimation for $S_{\text{Bi}}(i)$.

- Original estimation: $S_{\text{Bi}}(i)$
- Highly confident adjustment: $\hat{S}_{\text{Bi}}(i)$

$$\hat{S}_{\text{Bi}}(i) = \begin{cases} 1 & \text{, if } S_{\text{Bi}}(i) > \delta. \\ S_{\text{Bi}}(i) & \text{, otherwise.} \end{cases} \tag{8.14}$$

  The adjustment is to retain weights for slots in highly confident actionable utterances, and it is also to emphasize the difference of carried information between actionable utterances and others.

The refined ontology may produce a better SLU model, which improves intent prediction performance. Then the ontology can be further improved if actionable scores are more accurate due to better intent prediction. Therefore it forms an iterative framework and may benefit SLU in both human-machine and human-human conversations.

8.6 Experiments

8.6.1 Experimental Setup

The dataset is from the ICSI meeting corpus[1], where 22 meetings previously used as test and developmentsets are included for the actionable item detection task [2, 92]. These include three types of meetings, Bed, Bmr, and Bro, which include regular project discussions between colleagues and conversations between students and their advisors[2]. The total numbers of utterances are 4544, 9227, and 7264 for Bed, Bmr, and Bro respectively.

Actionable items were manually annotated, where the annotation schema was designed based on the Microsoft Cortana conversational agent schema. There are in total 42 actions in Cortana data, and we identified 10 actions that are relevant to meeting scenarios: find_calendar_entry, create_calendar_entry, open_agenda, add_agenda_item, create_single_reminder, send_email, find_email, make_call, search, and open_setting, where 2 actions

[1] http://www1.icsi.berkeley.edu/Speech/mr/

[2] Bed (003, 006, 010, 012), Bmr (001, 005, 010, 014, 019, 022, 024, 028,030), Bro (004, 008, 011, 014, 018, 021, 024, 027)

Table 8.1: Actionable item detection performance on the average AUC using bidirectional estimation (%).

Approach			#dim	Mismatch-CDSSM	Adapt-CDSSM	Match-CDSSM
(a)	w/o	Sim (CosSim(U, A))		49.10	50.36	50.57
(b)		AdaptSim ($\widehat{\text{CosSim}}(U, A)$)		**55.82**	**60.08**	**62.34**
(c)	SVM	Embeddings	300	55.71	63.95	**69.27**
(d)		(c) + Sim	311	59.09	65.08	68.86
(e)		(c) + AdaptSim	311	**59.23**	**65.71**	69.08

(find_email and open_agenda) do not occur in Cortana data. There are total 318 utterances annotated with actionable items, which accounts for about 2% of all utterances. Figure 8.4 shows actionable item distribution in the meeting corpus, where it can be found that different types of meetings contain slightly different distribution of actionable items, but some actions frequently occur in all meetings, such as create_single_reminder and find_calendar_entry.

Two meetings were annotated by two annotators, and we test the agreement for two settings using Cohen's Kappa coefficient [45]. First, the average agreement about whether an utterance includes an actionable item is 0.64; second, the average agreement about annotated actions (including others) is 0.67, showing that the actionable items are consistent across persons. The detail of the dataset can be found in Appendix A.

Due to imbalanced classes (number of non-actionable utterances is larger than number of actionable ones), the evaluation focuses on detection performance for each action. Here we compute AUC for each action as the metric to evaluate whether the detector is able to effectively detect an action, and then report average AUC over all classes (10 actions plus others).

8.6.2 CDSSM Training

To test the effect of CDSSM training data, we conduct experiments using following models:

- Mismatch-CDSSM: a CDSSM trained on conversational agent data, which mismatches with the target genre.
- Adapt-CDSSM: a CDSSM pretrained on conversational agent data and then continually trained on meeting data.
- Match-CDSSM: a CDSSM trained on meeting data, which matches with the target genre.

For all experiments, the total number of training iterations is set to 300, the dimension of the convolutional layer is 1000, and the dimension of the semantic layer is 300, where Adapt-CDSSM is trained on two datasets with 150 iterations for each.

8.6.3 Implementation Details

Considering that individuals may have consistent ways of referring to actionable items, to show the applicability of our approach to different speakers and meeting types, we take one of meeting types as training data and test on each of remaining two. Hence, we have 6 sets of experiments and report the average of AUC scores for evaluation, which is similar to 6-fold cross-validation. Note that the meeting data used in Match-CDSSM and Adapt-CDSSM is the training set of meeting data. The multi-class classifier we apply for actionable item detection in Section 8.4 is SVM with RBF kernel using a default setting [25]. The parameters, α and β in (8.6), are set to 1 in order to balance the effectiveness of original embeddings and utterance embeddings with the same action. The parameter γ in (8.11) is set as 0.5 to allow predictive and generative models contribute equally.

8.7 Evaluation Results

Experimental results of bidirectional estimation with different CDSSMs are shown in Table 8.1. Rows (a) and (b) use the semantic similarity as final prediction scores, where Sim (row (a)) uses CosSim(U_i, A_j) and AdaptSim (row (b)) uses $\widehat{\text{CosSim}}(U_i, A_j)$ as $S_P(i,j)$ or $S_G(i,j)$. Rows (c)-(e) use the similarity as features and then train an SVM classifier to estimate final prediction scores, where row (c) takes utterance embedding vectors as features, and rows (d) and (e) include the semantic similarity as additional features for the classifier. Hence the dimension of features is 311, including 300 values of utterance embeddings and 11 similarity scores for all actions.

When we treat the semantic similarity as final prediction scores, adapted embeddings (Adapt-Sim) perform better, achieving 55.8%, 60.1%, and 62.3% for Mismatch-CDSSM, Adapt-CDSSM, and Match-CDSSM respectively. Due to the fact that the learned embeddings do not fit the target genre well, the similarity treated as features of a classifier can be combined with other features to automatically adapt the reliability of similarity features. Row (c) shows the performance using only utterance embeddings, and including similarity scores as additional features can improve the performance for Mismatch-CDSSM (from 55.7% to 59.1%) and Adapt-CDSSM (from 64.0% to 65.1%). The action embedding adaptation further adjusts embeddings to the target genre based on Section 8.3.2, and row (c) shows that the performance can be further improved (59.2% and 65.7%). Below we discuss results in

Table 8.2: Actionable item detection performance on AUC (%).

Approach				#dim	$P(A \mid U)$	$P(U \mid A)$	Bidir
(a)	Mismatch	w/o	Sim ($\text{CosSim}(U, A)$)		47.45	48.17	49.10
(b)			AdaptSim ($\widehat{\text{CosSim}}(U, A)$)		54.00	53.89	**55.82**
(c)		SVM	Embeddings	300	53.07	48.07	55.71
(d)			(c) + Sim	311	52.80	54.95	59.09
(e)			(c) + AdaptSim	311	52.75	55.22	**59.23**
(a)	Adapt	w/o	Sim ($\text{CosSim}(U, A)$)		48.67	50.09	50.36
(b)			AdaptSim ($\widehat{\text{CosSim}}(U, A)$)		59.46	56.96	**60.08**
(c)		SVM	Embeddings	300	60.06	59.03	63.95
(d)			(c) + Sim	311	60.78	60.29	65.08
(e)			(c) + AdaptSim	311	61.60	61.13	**65.71**
(a)	Match	w/o	Sim ($\text{CosSim}(U, A)$)		56.33	43.39	50.57
(b)			AdaptSim ($\widehat{\text{CosSim}}(U, A)$)		64.19	60.36	**62.34**
(c)		SVM	Embeddings	300	64.33	65.58	**69.27**
(d)			(c) + Sim	311	64.52	64.81	68.86
(e)			(c) + AdaptSim	311	64.72	65.39	69.08

different aspects.

8.7.1 Comparing Different CDSSM Training Data

Because the target genre is not always available or not enough for training CDSSM, we compare results using CDSSM trained on different data. From Table 8.2, model adaptation (Adapt-CDSSM) improves the performance of Mismatch-CDSSM in all cases, showing that embeddings pre-trained on the mismatched data are successfully adapted to the target genre and then results in better performance. Although Adapt-CDSSM takes more data than Match-CDSSM, Match-CDSSM performs better. However, for row (a), we can see that Match-CDSSM is not robust enough, because the generative model ($P(U \mid A)$) performs 43.4% on AUC, even worse than Mismatch-CDSSM. It shows that the bidirectional model, the embedding adaptation, and additional classifiers help improve the robustness so that Match-CDSSM achieves better performance compared to Adapt-CDSSM.

The best result from matched features is one using only embeddings features (69.27% in row (c)), and a possible reason is that embeddings fit well to the target genre, so adding similarity cannot provide additional information to improve the performance.

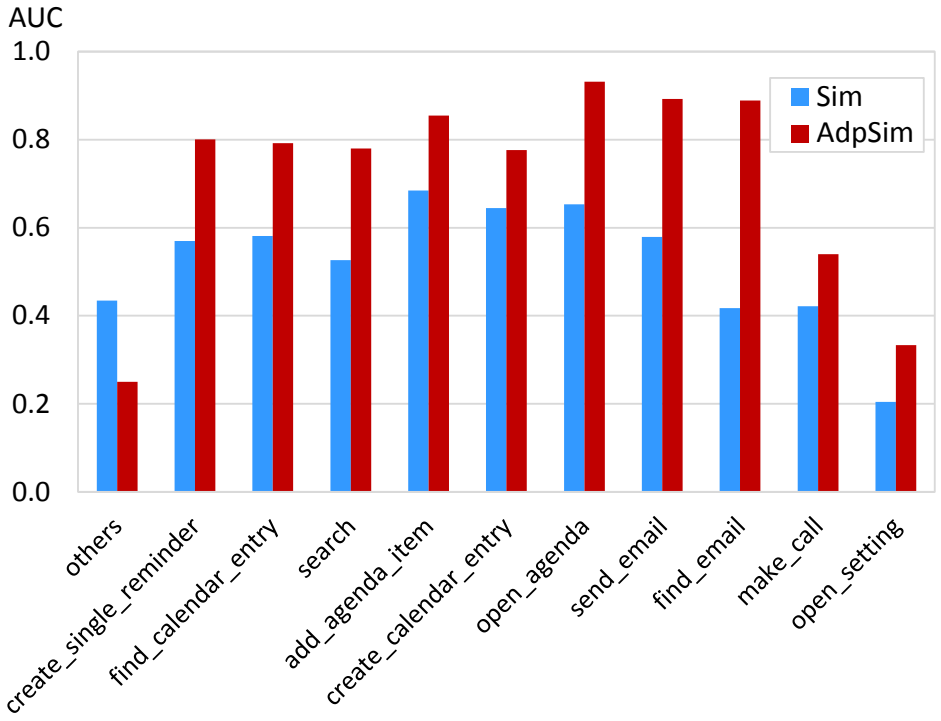


Figure 8.5: The average AUC distribution over all actions in the training set before and after action embedding adaptation using Match-CDSSM.

8.7.2 Effectiveness of Bidirectional Estimation

From Table 8.2, it is shown that all results from the bidirectional estimation significantly outperform results using unidirectional estimation across all CDSSMs and all methods except for rows (a) and (b) from Match-CDSSM. Comparing between the predictive model ($P(A \mid U)$) and the generative model ($P(U \mid A)$), the performance is similar and does not show that a certain direction is better in most cases. The improvement of bidirectional estimation suggests that the predictive model and the generative model can compensate each other, and then provides more robust estimated scores.

8.7.3 Effectiveness of Adaptation Techniques

Two adaptation approaches, CDSSM adaptation and action embedding adaptation, are useful when the features do not perfectly fit to the target genre. When without SVM, model adaptation and action embedding adaptation improve the performance from 49.1% to 50.4% and to 55.8% respectively. Applying both adaptation techniques achieves 60.1% on average AUC. After we use similarity scores as additional features of SVM, using individual adaptation improves the performance, and applying both techniques achieves further improvement. Therefore, it is shown that the proposed CDSSM and adaptation approaches can be applied

when the data for the target genre is unavailable or scarce.

On the other hand, when using matched data for CDSSM training (Match-CDSSM), action embedding adaptation still improves the performance before SVM (from 50.6% to 62.3%). Figure 8.5 shows the performance distribution over all actions in the training set before and after action embedding adaptation, where we find that all AUC scores are increased except for others so that overall performance is improved. The reason why matched data cannot induce good enough embeddings is that there are much more utterances belonging to others in the meetings, so CDSSM is more sensitive to the action others. However, the adaptation adjusts all action embeddings equally, forcing to increase the reliability of other action embeddings. Therefore, although the adapted result of others drops, the performance of all other actions is improved, resulting in better overall performance.

8.7.4 Effectiveness of CDSSM

To evaluate whether CDSSM provides better features for actionable item detection, we compare the performance with three baselines trained on the meeting corpus:

- AdaBoost with ngram features
  A boosting classifier is trained using unigram, bigram and trigram features [61].

- SVM with ngram features
  An SVM classifier is trained using unigram, bigram and trigram features [25].

- SVM with paragraph embeddings `doc2vec`
  An SVM classifier is trained using paragraph vectors[3] introduced in Section 2.4.3, where the training set of paragraph vectors is the same as one CDSSM takes, the vector dimension is set to 300, and the window size is 3 [104].

First two baselines use lexical features while the third one uses semantic features. Table 8.3 shows that two lexical baselines perform similarly, and AdaBoost is slightly better than SVM. Semantic embeddings trained on the meeting data as features perform better than lexical features, where `doc2vec` obtains 59.8% on AUC [104]. For the proposed approaches, both unidirectional CDSSMs outperform three baselines, achieving 64.3% for the predictive model and 65.6% for the generative model. In addition, bidirectional CDSSM improves the performance to 69.3%, showing a promising result and proving the effectiveness of CDSSM features.

[3] `https://radimrehurek.com/gensim/index.html`

Table 8.3: Actionable item detection performance on AUC (%).

Approach			AUC
Baseline	AdaBoost	ngram ($n = 1, 2, 3$)	54.31
	SVM	ngram ($n = 1, 2, 3$)	52.84
	SVM	`doc2vec`	59.79
Proposed	SVM	CDSSM: $P(A \mid U)$	64.33
	SVM	CDSSM: $P(U \mid A)$	65.58
	SVM	CDSSM: Bidirectional	**69.27**

8.7.5 Discussion

In addition to the power of CDSSM features, another advantage of CDSSM is the ability of generating more flexible action embeddings. For example, the actions open_agenda and find_email in the meeting data do not have corresponding predefined intents in the Cortana data; however, CDSSM is still able to generate the action embedding for find_email by incorporating the semantics from find_message and send_email. The flexibility may fill the gap between mismatched annotations. In the future work, we plan to investigate the ability of generating unseen action embeddings in order to remove the domain constraint for practical usage.

8.8 Extensive Experiments

To investigate the feasibility of applying proposed ontology induction and iterative ontology refinement to real-world human-human interactions, below we briefly introduce a real-world dialogue data and discuss results produced by the proposed approaches. The detail of the dataset is presented in Appendix B.

8.8.1 Dataset

The dataset is a set of insurance-related dialogues, where each conversation is a phone call between a customer and an agent collected by MetLife[4]. There are total 155 conversations and the number of utterances is 5,229 segmented by different speakers. WER is reported as 31.8% using end-to-end deep recurrent neural network models trained on SwitchBoard [113]. In the dataset, we consider all utterances annotated with either customer intents or agent actions as actionable utterances, and therefore the number of reference actionable utterances is 753, accounting for only 14% of total 5,229 utterances.

[4] `https://www.metlife.com/`

Table 8.4: The learned ontology performance with $\alpha = 0.8$ tuned from Chapter 3 (%)

Approach	α	ASR				Manual			
		Slot		Structure		Slot		Structure	
		AP	AUC	AP	AUC	AP	AUC	AP	AUC
Baseline: Frequency	.0	**47.36**	**43.37**	**13.48**	**11.36**	**61.29**	59.46	26.13	25.94
+ OriginAct S_{Bi}		45.19	43.14	13.36	11.31	59.98	58.97	26.35	26.11
+ AdjustAct $\hat{S}_{\text{Bi}}$		44.97	42.92	13.36	11.31	60.83	**59.83**	**26.85**	**26.60**
+ OracleAct S^*_{Bi}		47.15	44.28	14.47	12.20	70.59	66.65	37.88	37.77
External Word Vec.	.8	**51.83**	**49.61**	**15.25**	**12.82**	65.48	64.67	40.47	40.20
+ OriginAct S_{Bi}		51.27	49.17	15.17	12.75	**66.45**	**65.66**	40.44	40.19
+ AdjustAct $\hat{S}_{\text{Bi}}$		51.33	49.24	15.21	12.79	65.77	64.96	**40.72**	**40.45**
+ OracleAct S^*_{Bi}		50.46	48.37	15.31	12.87	82.66	82.43	57.04	56.86
Max RI (%)	-	+9.4	+14.4	+13.1	+12.8	+8.4	+10.4	+55.8	+56.0

Considering the performance of an induced ontology, we evaluate induced slots and their structure individually. For slot evaluation, there are total 31 human-annotated slots: Statement, Transfer, Law, Request, Inspecting, Verification, Sending, Getting, Receiving, Quantity, Commerce_pay, Contacting, Information, Grant_permission, Needing, Motion, Perception_experience, Awareness, Have_as_requirement, Sent_items, Locative_relation, Desiring, Assistance, Processing_materials, Capability, Undergo_change, Vehicle, Temporal_collocation, Evidence, Visiting, and Intentionally_act. The associated slot fillers labeled by annotators are shown in Table B.4. The uncovered slots contain Cancel, Refund, Delete, Discount, Benefit, Person, Status and Care and the detail is shown in Table B.5. Therefore, FrameNet coverage in this dataset is about 79.5%. Below evaluation metrics of induced slots are AP and AUC by comparing induced slots and reference ones covered by FrameNet. For structure evaluation, there are 212 reference slot pairs connected with typed dependency relations shown in Table B.6-B.11. AP and AUC are also computed as evaluation metrics in the experiments.

8.8.2 Ontology Induction

To evaluate the performance of ontology induction on the real-world data, we perform ontology induction proposed by Chapter 3 on the insurance-related dataset. That is, the ranking weight $w(s)$ in (2.7) is computed for each slot candidate to induce a domain ontology. The results of ontology induction are shown in Table 8.4, where the weight α is empirically set based on the value tuned from Chapter 3 ($\alpha = 0.8$). For results before adding actionable item information, it can be found that the proposed ontology induction using external word vectors performs better than only using frequency for both ASR and manual transcripts, which aligns well with our findings in Chapter 3.

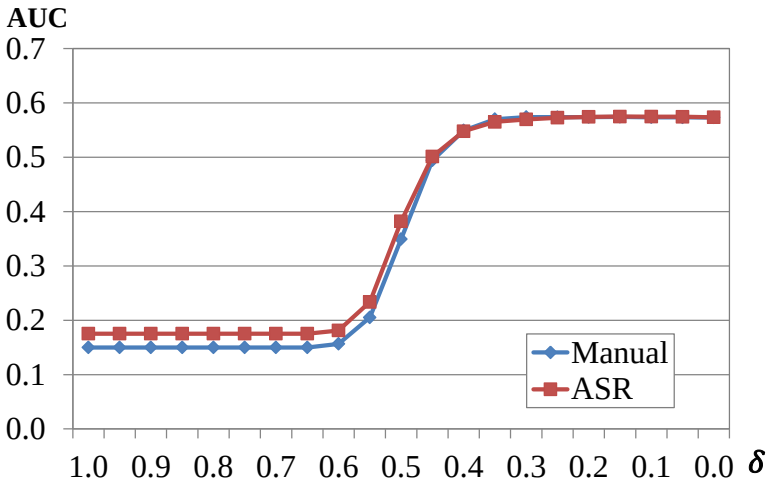


Figure 8.6: The AUC trend of actionable item detection with different thresholds δ.

8.8.3 Iterative Ontology Refinement

Because utterances related to the intent create_single_reminder indicate that agents may need to take notes when listening to them, we utilize the CDSSM trained on Cortana data to detect actionable utterances corresponding to create_single_reminder, where the bidirectional estimation $S_{\text{Bi}}(i)$ denotes the actionable score of the utterance U_i. Then the adjusted importance of each slot $w'(s)$ is estimated in (8.12) based on detected actionable utterances. The threshold δ in (8.14) is decided according to the trend of actionable item performance shown in Figure 8.6, where we remain original weights for more highly confident actionable utterances when δ decreases ($\delta = 1$ indicates that we use original estimation $S_{\text{Bi}}(i)$). Because $\delta = 0.5 - 0.55$ is most efficient to retain higher weights for actionable utterances according to the largest slope ($\max \frac{\partial AUC}{\partial \delta}$), we set the thresholds to 0.55 and 0.50 to compute $\hat{S}_{\text{Bi}}(i)$ for ASR and manual transcripts respectively. The decided threshold also aligns well as our intuition, where the estimated probability higher than about 0.5 suggests actionable utterances. Table 8.4 also presents the results of integrating actionable scores based on original estimation (OriginAct) S_{Bi}, adjusted estimation (AdjustAct) $\hat{S}_{\text{Bi}}$, and oracle estimation (OracleAct) S^*_{Bi}. The oracle estimation of actionable utterances is to show the upper bound for the performance of iterative ontology refinement.

For ASR transcripts, additionally integrating actionable scores does not show any improvement for both baseline and external word vector results. However, for manual transcripts, leveraging estimated actionable scores (OriginAct and AdjustAct) slightly improves the performance of the induced slots and the learned structure. Also, oracle scores (OracleAct) significantly improve the performance for both baseline and external word vector results, suggesting that there is still a large room for improvement. The experiment demonstrates the

effectiveness of iterative ontology refinement using actionable information. Due to the similar performance of actionable item detection for ASR and manual transcripts, the incapability of refining the ontology for ASR results is probably resulted from recognition errors.

8.8.4 Influence of Recognition Errors

Due to higher WER of ASR results in the dataset (31%), recognition errors may degrade the performance of ontology induction. In terms of induced slots, the ASR baseline achieves 47.36% of AP and 43.37% of AUC, while the manual baseline performs 61.29% of AP and 59.46% of AUC. The external word vector approach for ASR performs 51.83% of AP and 49.61% of AUC, and for manual one is 65.48% of AP and 64.67% of AUC. The performance difference between ASR and manual results is more than 10% due to recognition errors.

On the other hand, in terms of the induced ontology structure, the ASR baseline performs only 13.48% of AP and 11.36% of AUC, while manual baseline is 26.13% and 25.94%. Performing the external word vector approach improves ASR results to 15.25% of AP and 12.82% of AUC, where the improvement is about 2%. However, the external word vector approach is able to improve manual results to 40.47% of AP and 40.20% of AUC, where the improvement is more than 10%. The difference of the induced structure is due to the poor performance of the originally induced ontology on ASR transcripts. Therefore, unlike the results on human-machine dialogues shown in Table 3.2, robustness is an important issue for ontology induction of human-human conversations.

8.8.5 Balance between Frequency and Coherence

To analyze the sensitivity of α (tradeoff between the frequency and the coherence), we show the performance of ontology induction in terms of individual slots with different α in Figure 8.7 and Figure 8.8 for ASR and manual transcripts respectively[5]. The best performance is $\alpha = 0.7$, which is close to the tuned value based on our previous experiments ($\alpha = 0.8$) in Chapter 3. In terms of structure performance, Figure 8.9 and Figure 8.10 show the trends of performance over different α for ASR and manual results respectively. The optimal value of α is 0.7 for ASR and $\alpha = 0.8$ for manual transcripts.

In sum, the value of α tuned from human-machine conversations can be also applied to human-human dialogues. In addition, the balance between the frequency and the coherence in the insurance data for both slot and structure performance has similar trends when comparing with Figure 3.4, demonstrating the effectiveness of the proposed ontology induction

[5]The left figure contains the baseline, proposed approaches, and oracle results; the right one only contains all but the oracle one.

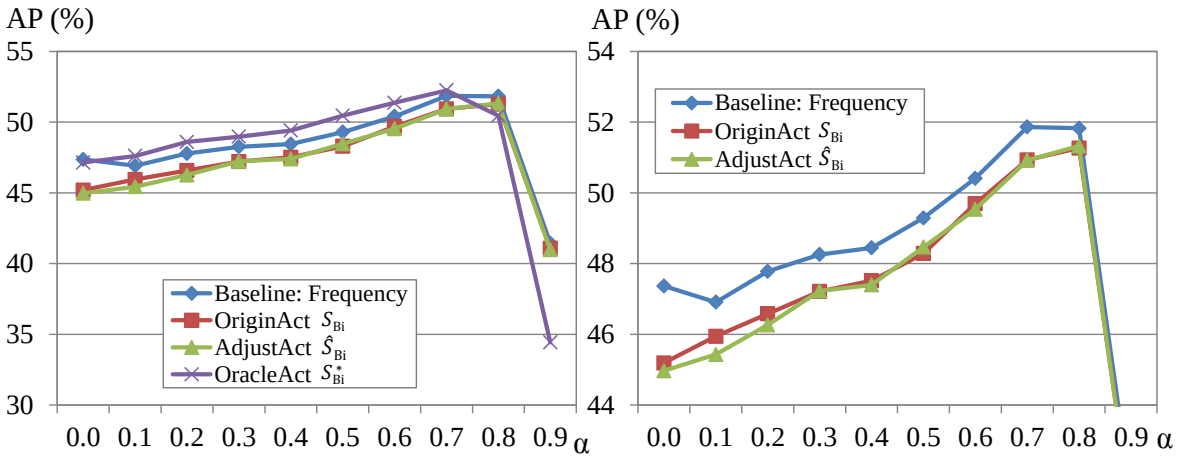


Figure 8.7: The performance of the induced slots with different α values for ASR transcripts; the right one shows the detailed trends about the baseline and the proposed ones.

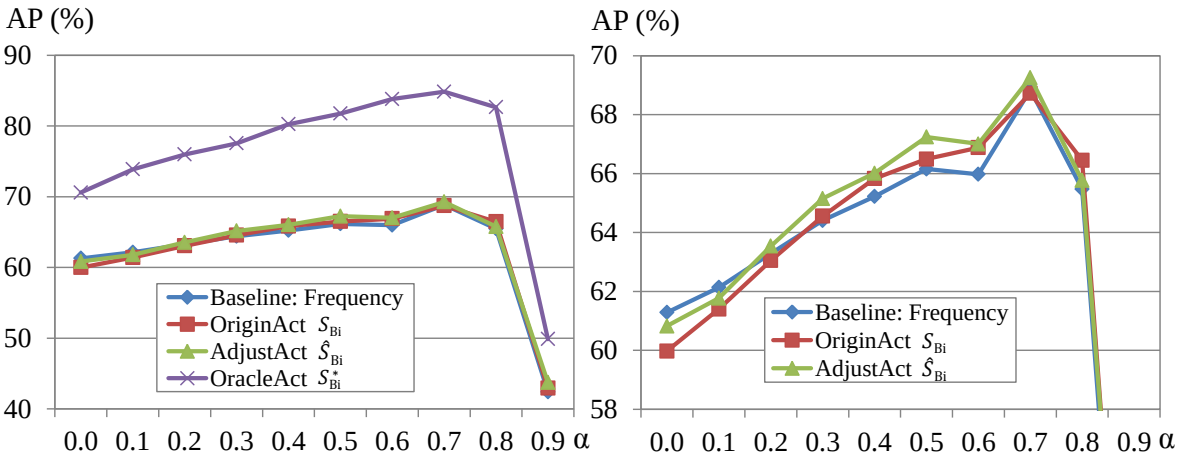


Figure 8.8: The performance of the induced slots with different α values for manual transcripts; the right one shows the detailed trends about the baseline and the proposed ones.

approaches for human-human conversations.

8.8.6 Effectiveness of Actionable Item Information

To eliminate the influence brought by recognition errors, we analyze the results for manual transcripts after leveraging actionable item information. From Table 8.4, the original actionable scores (OriginAct) slightly improve ontology induction using the external word vector approach, probably because the difference between actionable scores from different utterances is too subtle to significantly change the final ranking list. However, when we remains weights for slots in highly confident actionable utterances, leveraging $\hat{S}_{\text{Bi}}(i)$ (AdjustAct) significantly

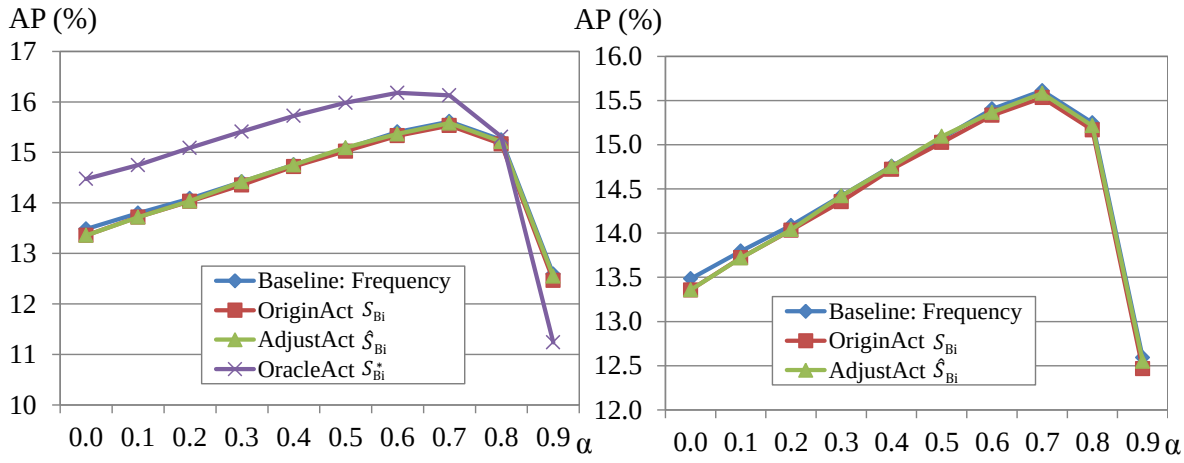


Figure 8.9: The performance of the learned structure with different α values for ASR transcripts; the right one shows the detailed trends about the baseline and the proposed ones.

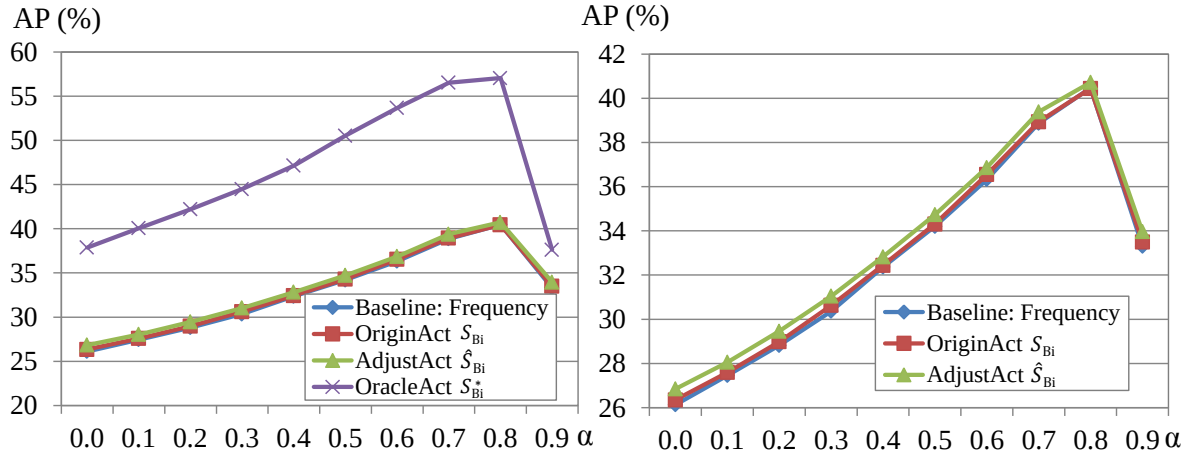


Figure 8.10: The performance of the learned structure with different α values for manual transcripts; the right one shows the detailed trends about the baseline and the proposed ones.

improves the induced ontology, especially for the structure performance. Figure 8.8 and Figure 8.10 show that the oracle actionable scores are able to significantly improve the ontology performance in terms of both individual slots and the structure. Due to the performance of actionable item detection lower than 60% for the mismatched genre, there is a large room for potential improvement. Nevertheless, we can conclude that actionable item information provides additional cues to help induce the domain ontology for human-human dialogues and demonstrate the feasibility of the proposed iterative framework.

8.9 Summary

This chapter focuses on applying SLU techniques developed for human-machine interactions to human-human interactions, specifically, detecting actionable items in human-human conversations, where a CDSSM is built to learn both utterance and action embeddings. Then the latent semantic features generated by CDSSM show the effectiveness of detecting actions in meetings compared to lexical features, and also outperform the state-of-the-art semantic paragraph vectors. The adaptation techniques are proposed to adjust the learned embeddings to fit the target genre when the source genre does not match well with target genre, improving detection performance. The actionable item information is also demonstrated to be useful for iterative ontology refinement in human-human dialogues. This chapter highlights a future research direction by transferring knowledge across genres and shows the feasibility and potential for applying the proposed approaches on the real world data.

9 Conclusions and Future Work

> *Mobile communications and pervasive computing technologies, together with social contracts that were never possible before, are already beginning to change the way people meet, mate, work, war, buy, sell, govern and create.*
>
> Howard Rheingold, *critic, writer, and teacher*

9.1 Conclusions

This dissertation consists in automatically acquiring domain knowledge from available semantic resources and then understanding both semantics of individual utterances and user intents. Owing to the increasing amount of spoken interactions from different domains, automating the system development is a trend considering efficiency and maintenance. The thesis addresses two challenges of SDS: the lack of a predefined structured ontology and shallow understanding, and proposes knowledge acquisition and SLU modeling techniques to show potential solutions.

For knowledge acquisition, semantic concepts, the structure, and surface forms are automatically learned. It is shown that such information is helpful to better understand semantics. For SLU modeling, we consider a structured ontology to predict semantics of individual utterances, because knowledge acquisition demonstrates the effectiveness of the automatically learned ontology for an understanding task. The proposed MF-SLU framework is useful and easy to expand for incorporating more features and predicted elements for both low-level semantic decoding and high-level intent prediction.

To further investigate the performance of SLU techniques applied to human-human dialogues, this dissertation exploits SLU techniques for human-machine conversations to detect actionable items in human-human conversations through the proposed CDSSM. The learned action embeddings are useful to detect actionable utterances, and further refine the induced ontology

in the real world human-human conversations. The iterative framework presents the feasibility of continuously improving understanding performance for human-human dialogues, and ultimately benefiting human-machine dialogue systems.

9.2 Future Work

This dissertation presents instantiations of automating the development process for SDS based on unlabeled conversations, and points to some potential research directions discussed below.

9.2.1 Domain Discovery

The proposed approach can be directly applied to utterances that cannot be handled by the current systems (e.g. Siri, Cortana) in order to discover new domains. The discovered domains can guide system developers for handling the domains users are more interested in. Supporting proper and important domains is the most efficient way to improve current system performance [78].

9.2.2 Large-Scaled Active Learning of SLU

Considering the rapidly increasing amount of data, it is essential to acquire knowledge from large data effectively. However, noisy labels may degrade the performance of SLU modeling. To ensure the quality of the SLU component, distinguishing highly-confident acquired knowledge from noisy knowledge can improve the learned model. Different active learning methods can be applied when the data has diverse qualities in order to achieve better performance.

- Active learning without labels
  Fang and Zhu proposed an "uncertain labeling knowledge" based active learning paradigm to characterize the knowledge set of each labelers and filter uncertain instances for better learning performance [58]. From a practical perspective, training data selection is critical for large-scaled data, where *submodular subset selection* can be considered to properly select data for SLU training in order to address the issue about large-scaled data [162, 161].

- Active learning with labels
  Crowdsourcing techniques are usually applied to filer noisy labels for bootstrapping the performance in the most efficient way [157, 126]. The explicit labels are more accurate but require additional cost.

The above approaches can be applied based on different engineering situations (e.g. different amount of available labels, different quality requirements, etc.), and it can also be fused to improve the flexibility.

9.2.3 Iterative Learning through Dialogues

The conversational interactions between users and systems can be utilized to refine the performance of current systems [123]. The performed interactions that fulfill user requests can be treated as verified positive samples for retraining the SLU model so that final performance can be iteratively improved when users interact with the systems more. That also indicates that the system keeps learning and refining the applied knowledge to better understand users and then provide proper responses. This can also benefit active learning using implicitly inferred labels.

9.2.4 Error Recovery for System Robustness

Due to multiple components in an SDS, error propagation is the main issue for user-aware performance. The errors may come from the recognition component, the imperfect domain ontology, or incorrectly decoded semantic representations. For each component, some research directions are presented below.

- Recognition
  The unlabeled dialogues for knowledge acquisition may contain recognition errors, which make the automatically acquired knowledge unreliable. Controlling the focused topic of a single dialogue may decrease probabilities of out-of-topic words and help learn out-of-vocabulary (OOV) words so that the ASR performance can be improved [147].

- Domain ontology
  The automatically induced domain ontology may not be perfect, and the SLU model mostly relies on the uncertain knowledge to decode semantics of utterances. Therefore, quality control could be handled by the approaches mentioned in Section 9.2.2.

- Semantic representation
  The incorrect semantic decoding greatly affects understanding performance. The reliable semantic forms can be controlled based on some strategies of belief tracking in the dialogue systems [132, 96, 121, 105, 87].

Each types of errors should be controlled and recovered afterwards in order to improve system robustness. To sum up, the dissertation presents the feasibility of automating the SLU

development process with little manual effort, demonstrating *effectiveness*, *scalability*, and *efficiency* of the proposed framework for building dialogue systems.

Bibliography

[1] James F Allen, Donna K Byron, Myroslava Dzikovska, George Ferguson, Lucian Galescu, and Amanda Stent. Toward conversational human-computer interaction. *AI magazine*, 22(4):27, 2001.

[2] Jeremy Ang, Yang Liu, and Elizabeth Shriberg. Automatic dialog act segmentation and classification in multiparty meetings. In *Proceedings of The 30th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1061–1064, 2005.

[3] Collin Baker. Framenet, present and future. *Programme Committee 7*, 2008.

[4] Collin F Baker, Charles J Fillmore, and John B Lowe. The berkeley framenet project. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING)*, pages 86–90. ACL, 1998.

[5] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract meaning representation for sembanking. In *Proceedings of Language Annotation Workshop*, 2013.

[6] Srinivas Bangalore, Giuseppe Di Fabbrizio, and Amanda Stent. Learning the structure of task-driven human–human dialogs. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(7):1249–1259, 2008.

[7] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, 2014.

[8] Yonatan Belinkov, Mitra Mohtarami, Scott Cyphers, and James Glass. Vectorslu: A continuous word vector approach to answer selection in community question answering systems. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval)*, volume 15, 2015.

[9] Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. Label propagation and quadratic criterion. *Semi-supervised learning*, 10, 2006.

[10] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference of the European Chapter of the Association for Computational Linguistics (EMNLP)*, pages 1533–1544, 2013.

[11] Anshuman Bhargava, Asli Celikyilmaz, Dilek Hakkani-Tur, and Ruhi Sarikaya. Easy contextual intent prediction and slot detection. In *Proceedings of 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8337–8341. IEEE, 2013.

[12] Dan Bohus. *Error Awareness and Recovery in Coversational Spoken Language Interfaces*. PhD thesis, Carnegie Mellon University, 2007.

[13] Dan Bohus and Alexander I Rudnicky. LARRI: A language-based maintenance and repair assistant. In *Spoken Multimodal Human-Computer Dialogue in Mobile Environments*, pages 203–218. Springer, 2005.

[14] Dan Bohus and Alexander I Rudnicky. The RavenClaw dialog management framework: Architecture and systems. *Computer Speech & Language*, 23(3):332–361, 2009.

[15] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of International Conference on Management of Data*, 2008.

[16] Antoine Bordes, Jason Weston, Ronan Collobert, Yoshua Bengio, et al. Learning structured embeddings of knowledge bases. In *Proceedings of AAAI*, 2011.

[17] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2013.

[18] Kendrick Boyd, Vitor Santos Costa, Jesse Davis, and C David Page. Unachievable region in precision-recall space and its effect on empirical evaluation. In *Proceedings of the International Conference on Machine Learning (ICML)*, page 349. NIH Public Access, 2012.

[19] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN systems*, 30(1):107–117, 1998.

[20] Barry Brown, Moira McGregor, and Donald McMillan. Searchable objects: Search in everyday conversation. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, pages 508–517. ACM, 2015.

[21] Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4): 467–479, 1992.

[22] Trung H Bui and Stanley Peters. Decision detection using hierarchical graphical models. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 307–312. ACL, 2010.

[23] Asli Celikyilmaz, Dilek Hakkani-Tür, and Gokhan Tür. Leveraging web query logs to learn user intent via bayesian discrete latent variable model. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2011.

[24] Asli Celikyilmaz, Zhaleh Feizollahi, Dilek Hakkani-Tür, and Ruhi Sarikaya. Resolving referring expressions in conversational dialogs for natural user interfaces. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2094–2104. ACL, 2014.

[25] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[26] Yun-Nung Chen and Florian Metze. Two-layer mutually reinforced random walk for improved multi-party meeting summarization. In *Proceedings of The 4th IEEE Workshop on Spoken Language Tachnology (SLT)*, pages 461–466, 2012.

[27] Yun-Nung Chen and Florian Metze. Multi-layer mutually reinforced random walk with hidden parameters for improved multi-party meeting summarization. In *Proceedings of The 14th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 485–489, 2013.

[28] Yun-Nung Chen and Alexander I. Rudnicky. Dynamically supporting unexplored domains in conversational interactions by enriching semantics with neural word embeddings. In *Proceedings of 2014 IEEE Spoken Language Technology Workshop (SLT)*, 2014.

[29] Yun-Nung Chen and Alexander I. Rudnicky. Two-stage stochastic natural language generation for email synthesis by modeling sender style and topic structure. In *Proceedings of The 8th International Natural Language Generation Conference (INLG)*, pages 152–156, 2014.

[30] Yun-Nung Chen, William Yang Wang, and Alexander I. Rudnicky. An empirical investigation of sparse log-linear models for improved dialogue act classification. In *Proceedings of 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8317–8321, 2013.

[31] Yun-Nung Chen, William Yang Wang, and Alexander I Rudnicky. Unsupervised induction and filling of semantic slots for spoken dialogue systems using frame-semantic parsing. In *Proceedings of IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*, pages 120–125, 2013.

[32] Yun-Nung Chen, Dilek Hakkani-Tür, and Gokhan Tur. Deriving local relational surface forms from dependency-based entity embeddings for unsupervised spoken language understanding. In *Proceedings of 2014 IEEE Spoken Language Technology Workshop (SLT)*, 2014.

[33] Yun-Nung Chen, William Yang Wang, and Alexander I Rudnicky. Leveraging frame semantics and distributional semantics for unsupervised semantic slot induction in spoken dialogue systems. In *Proceedings of 2014 IEEE Spoken Language Technology Workshop (SLT)*, 2014.

[34] Yun-Nung Chen, Dilek Hakkani-Tur, and Xiaodong He. Detecting actionable items in meetings by convolutional deep structured semantic models. In *Proceedings of 2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2015.

[35] Yun-Nung Chen, Dilek Hakkani-Tur, and Xiaodong He. Learning bidirectional intent embeddings by convolutional deep structred semantic models for spoken language understanding. In *Extended Abstract of The 29th Annual Conference on Neural Information Processing Systems – Machine Learning for Spoken Language Understanding and Interactions Workshop (NIPS-SLU)*, 2015.

[36] Yun-Nung Chen, Ming Sun, and Alexander I. Rudnicky. Matrix factorization with domain knowledge and behavioral patterns for intent modeling. In *Extended Abstract of The 29th Annual Conference on Neural Information Processing Systems – Machine Learning for Spoken Language Understanding and Interactions Workshop (NIPS-SLU)*, 2015.

[37] Yun-Nung Chen, Ming Sun, I. Alexander Rudnicky, and Anatole Gershman. Leveraging behavioral patterns of mobile applications for personalized spoken language understanding. In *Proceedings of the 17th International Conference on Multimodal Interaction (ICMI)*, pages 83–86. ACM, 2015.

[38] Yun-Nung Chen, William Yang Wang, Anatole Gershman, and Alexander I. Rudnicky. Matrix factorization with knowledge graph propagation for unsupervised spoken language understanding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and The 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP)*. ACL, 2015.

[39] Yun-Nung Chen, William Yang Wang, and Alexander I. Rudnicky. Jointly modeling inter-slot relations by random walk on knowledge graphs for unsupervised spoken language understanding. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 619–629, 2015.

[40] Yun-Nung Chen, William Yang Wang, and Alexander I. Rudnicky. Learning semantic hierarchy with distributional representations for unsupervised spoken language understanding. In *Proceedings of The 16th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 1869–1873. ISCA, 2015.

[41] Yun-Nung Chen, Dilek Hakkani-Tür, and Xiaodong He. Zero-shot learning of intent embeddings for expansion by convolutional deep structured semantic models. In *Proceedings of The 41st IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2016.

[42] Yun-Nung Chen, Ming Sun, Alexander I. Rudnicky, and Anatole Gershman. Unsupervised user intent modeling by feature-enriched matrix factorization. In *Proceedings of The 41st IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2016.

[43] X. Cheng and D. Roth. Relational inference for wikification. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2013.

[44] Ananlada Chotimongkol. *Learning the structure of task-oriented conversations from the corpus of in-domain dialogs*. PhD thesis, Carnegie Mellon University, 2008.

[45] Jacob Cohen et al. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

[46] Michael Collins, Sanjoy Dasgupta, and Robert E Schapire. A generalization of principal components analysis to the exponential family. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 617–624, 2001.

[47] Bob Coyne, Daniel Bauer, and Owen Rambow. VigNet: Grounding language in graphics using frame semantics. In *Proceedings of the ACL 2011 Workshop on Relational Models of Semantics*, pages 28–36, 2011.

[48] Dipanjan Das, Nathan Schneider, Desai Chen, and Noah A Smith. Probabilistic frame-semantic parsing. In *Proceedings of 2010 Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT)*, pages 948–956, 2010.

[49] Dipanjan Das, Desai Chen, André F. T. Martins, Nathan Schneider, and Noah A. Smith. Frame-semantic parsing. *Computational Linguistics*, 2013.

[50] Dipanjan Das, Desai Chen, André FT Martins, Nathan Schneider, and Noah A Smith. Frame-semantic parsing. *Computational Linguistics*, 40(1):9–56, 2014.

[51] Marie-Catherine De Marneffe and Christopher D Manning. The Stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8. ACL, 2008.

[52] Renato De Mori, Frédéric Bechet, Dilek Hakkani-Tur, Michael McTear, Giuseppe Riccardi, and Gokhan Tur. Spoken language understanding. *IEEE Signal Processing Magazine*, 25(3), 2008.

[53] Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407, 1990.

[54] Li Deng, Gokhan Tur, Xiaodong He, and Dilek Hakkani-Tür. Use of kernel deep convex networks and end-to-end learning for spoken language understanding. In *Proceedings of 2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 210–215, 2012.

[55] Marco Dinarelli, Silvia Quarteroni, Sara Tonelli, Alessandro Moschitti, and Giuseppe Riccardi. Annotating spoken dialogs: from speech segments to dialog acts and frame semantics. In *Proceedings of the 2nd Workshop on Semantic Representation of Spoken Language*, pages 34–41. ACL, 2009.

[56] John Dowding, Jean Mark Gawron, Doug Appelt, John Bear, Lynn Cherny, Robert Moore, and Douglas Moran. Gemini: A natural language system for spoken-language understanding. In *Proceedings of The 31st Annual Meeting on Association for Computational Linguistics (ACL)*, pages 54–61. ACL, 1993.

[57] Ali El-Kahky, Derek Liu, Ruhi Sarikaya, Gokhan Tur, Dilek Hakkani-Tür, and Larry Heck. Extending domain coverage of language understanding systems via intent transfer between domains using knowledge graphs and search query click logs. In *Proceedings of The 39th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2014.

[58] Meng Fang and Xingquan Zhu. Active learning with uncertain labeling knowledge. *Pattern Recognition Letters*, 43:98–108, 2014.

[59] Manaal Faruqui and Chris Dyer. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 2014.

[60] Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. Retrofitting word vectors to semantic lexicons. In *Proceedings of 2015 Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT)*, 2015.

[61] Benoit Favre, Dilek Hakkani-Tür, and Sebastien Cuendet. ICSIBoost. 2007. URL `http://code.google.come/p/icsiboost`.

[62] Raquel Fernández, Matthew Frampton, Patrick Ehlen, Matthew Purver, and Stanley Peters. Modelling and detecting decisions in multi-party dialogue. In *Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue*, pages 156–163. ACL, 2008.

[63] Charles Fillmore. Frame semantics. *Linguistics in the morning calm*, pages 111–137, 1982.

[64] Charles J Fillmore. Frame semantics and the nature of language. *Annals of the NYAS*, 280(1):20–32, 1976.

[65] Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A Smith. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. Citeseer, 2014.

[66] GW Furnas, TK Landauer, LM Gomez, and ST Dumais. Statistical semantics: Analysis of the potential performance of keyword information systems. In *Proceedings of Human Factors in Computer Systems*, pages 187–242, 1984.

[67] Michel Galley, Kathleen McKeown, Julia Hirschberg, and Elizabeth Shriberg. Identifying agreement and disagreement in conversational speech: Use of bayesian networks to model pragmatic dependencies. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (ACL)*, page 669. ACL, 2004.

[68] Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Mymedialite: A free recommender system library. In *Proceedings of the fifth ACM Conference on Recommender Systems*, pages 305–308. ACM, 2011.

[69] Jianfeng Gao, Patrick Pantel, Michael Gamon, Xiaodong He, Li Deng, and Yelong Shen. Modeling interestingness with deep neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[70] M Gašić, Pirros Tsiakoulis, Matthew Henderson, Blaise Thomson, Kai Yu, Eli Tzirkel, and Steve Young. The effect of cognitive load on a statistical dialogue system. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 74–78. ACL, 2013.

[71] Milica Gašic, Catherine Breslin, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve Young. POMDP-based dialogue manager

adaptation to extended domains. In *Proceedings of 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 214–222, 2013.

[72] Narendra Gupta, Gokhan Tur, Dilek Hakkani-Tur, Srinivas Bangalore, Giuseppe Riccardi, and Mazin Gilbert. The AT&T spoken language understanding system. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(1):213–222, 2006.

[73] Dilek Hakkani-Tür, Frédéric Béchet, Giuseppe Riccardi, and Gokhan Tur. Beyond ASR 1-best: Using word confusion networks in spoken language understanding. *Computer Speech & Language*, 20(4):495–514, 2006.

[74] Dilek Hakkani-Tür, Asli Celikyilmaz, Larry P Heck, and Gokhan Tur. A weakly-supervised approach for discovering new user intents from search query logs. In *Proceedings of The 14th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2013.

[75] Dilek Hakkani-Tür, Larry Heck, and Gokhan Tur. Using a knowledge graph and query click logs for unsupervised learning of relation detection. In *Proceedings of The 38th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 8327–8331, 2013.

[76] Dilek Hakkani-Tur, Asli Celikyilmaz, Larry Heck, Gokhan Tur, and Geoff Zweig. Probabilistic enrichment of knowledge graph entities for relation detection in conversational understanding. In *Proceedings of The 15th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2014.

[77] Dilek Hakkani-Tür, Malcolm Slaney, Asli Celikyilmaz, and Larry Heck. Eye gaze for spoken language understanding in multi-modal conversational interactions. In *Proceedings of the 16th International Conference on Multimodal Interaction*, pages 263–266. ACM, 2014.

[78] Dilek Hakkani-Tür, Yun-Cheng Ju, Geoffrey Zweig, and Gokhan Tur. Clustering novel intents in a conversational interaction system with semantic parsing. In *Proceedings of the 16th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2015.

[79] Zellig S Harris. Distributional structure. *Word*, 1954.

[80] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Applied statistics*, pages 100–108, 1979.

[81] Kazi Saidul Hasan and Vincent Ng. Frame semantics for stance classification. *Proceedings of CoNLL-2013*, page 124, 2013.

[82] Larry Heck and Dilek Hakkani-Tür. Exploiting the semantic web for unsupervised spoken language understanding. In *Proceedings of 2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 228–233, 2012.

[83] Larry P Heck, Dilek Hakkani-Tür, and Gokhan Tur. Leveraging knowledge graphs for web-scale unsupervised semantic parsing. In *Proceedings of the 14th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2013.

[84] Steffen Hedegaard and Jakob Grue Simonsen. Lost in translation: authorship attribution using frame semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers (HLT)*, pages 65–70. ACL, 2011.

[85] Matthew Henderson. *Discriminative Methods for Statistical Spoken Dialogue Systems.* PhD thesis, University of Cambridge, 2015.

[86] Matthew Henderson, Milica Gasic, Blaise Thomson, Pirros Tsiakoulis, Kai Yu, and Steve Young. Discriminative spoken language understanding using word confusion networks. In *Proceedings of 2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 176–181, 2012.

[87] Matthew Henderson, Blaise Thomson, and Steve Young. Deep neural network approach for the dialog state tracking challenge. In *Proceedings of 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 467–471, 2013.

[88] Dustin Hillard, Mari Ostendorf, and Elizabeth Shriberg. Detection of agreement vs. disagreement in meetings: Training with unlabeled data. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 34–36. ACL, 2003.

[89] Dustin Hillard, Asli Celikyilmaz, Dilek Z Hakkani-Tür, and Gokhan Tur. Learning weighted entity lists from web click logs for spoken language understanding. In *Proceedings of The 12th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2011.

[90] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management (CIKM)*, pages 2333–2338. ACM, 2013.

[91] Diana Inkpen and Graeme Hirst. Building and using a lexical knowledge base of near-synonym differences. *Computational Linguistics*, 32(2):223–262, 2006.

[92] Adam Janin, Don Baron, Jane Edwards, Dan Ellis, Davis Gelbart, Nelson Morgan, Barbara Peskin, Thilo Pfau, Elizabeth Shriberg, Andreas Stolcke, and Chuck Wooters. The ICSI meeting corpus. In *Proceedings of 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2003.

[93] Sujay Kumar Jauhar, Yun-Nung Chen, and Florian Metze. Prosody-based unsupervised speech summarization with two-layer mutually reinforced random walk. In *Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP)*, pages 648–654, 2013.

[94] Frederick Jelinek. *Statistical methods for speech recognition.* MIT press, 1997.

[95] Richard Johansson and Pierre Nugues. Extended constituent-to-dependency conversion for english. In *Proceedings of The 16th Nordic Conference of Computational Linguistics.* University of Tartu, 2007.

[96] Rudolf Kadlec, Jindrich Libovickỳ, Jan Macek, and Jan Kleindienst. IBM's belief tracker: Results on dialog state tracking challenge datasets. page 10, 2014.

[97] Raghunandan H Keshavan, Andrea Montanari, and Sewoong Oh. Matrix completion from a few entries. *IEEE Transactions on Information Theory*, 56(6):2980–2998, 2010.

[98] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[99] Spyros Kousidis, Casey Kennington, Timo Baumann, Hendrik Buschmeier, Stefan Kopp, and David Schlangen. A multimodal in-car dialogue system that tracks the driver's attention. In *Proceedings of the 16th International Conference on Multimodal Interaction (ICMI)*, pages 26–33. ACM, 2014.

[100] Amy N Langville and Carl D Meyer. A survey of eigenvector methods for web information retrieval. *SIAM review*, 47(1):135–161, 2005.

[101] Ni Lao, Tom Mitchell, and William W Cohen. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 529–539. ACL, 2011.

[102] Ni Lao, Amarnag Subramanya, Fernando Pereira, and William W Cohen. Reading the web with learned syntactic-semantic inference rules. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP)*, pages 1017–1026. ACL, 2012.

[103] Staffan Larsson and David R Traum. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural language engineering*, 6(3&4):323–340, 2000.

[104] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 1188–1196, 2014.

[105] Sungjin Lee and Maxine Eskenazi. POMDP-based Let's Go system for spoken dialog challenge. In *Proceedings of 2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 61–66. IEEE, 2012.

[106] Fabrice Lefevre, François Mairesse, and Steve Young. Cross-lingual spoken language understanding from unaligned data using discriminative classification models and machine translation. In *Proceedings of The 11th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2010.

[107] Oliver Lemon, Kallirroi Georgila, James Henderson, and Matthew Stuttle. An ISU dialogue system exhibiting reinforcement learning of dialogue policies: generic slot-filling in the talk in-car system. In *Proceedings of the Eleventh Conference of the European Chapter of the Association for Computational Linguistics: Posters & Demonstrations*, pages 119–122. ACL, 2006.

[108] Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of The 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2014.

[109] Peipei Li, Haixun Wang, Hongsong Li, and Xindong Wu. Assessing sparse information extraction using semantic contexts. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management (CIKM)*, pages 1709–1714. ACM, 2013.

[110] Peipei Li, Haixun Wang, Kenny Q Zhu, Zhongyuan Wang, and Xindong Wu. Computing term similarity by large probabilistic isa knowledge. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management (CIKM)*, pages 1401–1410. ACM, 2013.

[111] Percy Liang. *Semi-supervised learning for natural language*. PhD thesis, Massachusetts Institute of Technology, 2005.

[112] Donald McMillan, Antoine Loriette, and Barry Brown. Repurposing conversation: Experiments with the continuous speech stream. In *Proceedings of the 33rd annual ACM conference on Human factors in computing systems*, pages 3953–3962. ACM, 2015.

[113] Yajie Miao, Mohammad Gowayyed, and Florian Metze. EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding. In *Proceedings of 2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, 2015.

[114] Tomáš Mikolov. *Statistical language models based on neural networks*. PhD thesis, Brno University of Technology, 2012.

[115] Tomáš Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Proceedings of The 11th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 1045–1048, 2010.

[116] Tomas Mikolov, Stefan Kombrink, Anoop Deoras, Lukar Burget, and Jan Cernocky. RNNLM-recurrent neural network language modeling toolkit. In *Proceedings of the 2011 ASRU Workshop*, pages 196–201, 2011.

[117] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at International Conference on Learning Representations*, 2013.

[118] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2013.

[119] Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of 2013 Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT)*, pages 746–751, 2013.

[120] Scott Miller, Robert Bobrow, Robert Ingria, and Richard Schwartz. Hidden understanding models of natural language. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 25–32. ACM, 1994.

[121] Nikola Mrkšić, Diarmuid O Séaghdha, Blaise Thomson, Milica Gašić, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. Multi-domain dialog state tracking using recurrent neural networks. *arXiv preprint arXiv:1506.07190*, 2015.

[122] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2:849–856, 2002.

[123] Aasish Pappu and Alexander I Rudnicky. Learning situated knowledge bases through dialog. *Proceedings of the 15th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2014.

[124] J. Pasternack and D. Roth. The wikipedia corpus, 2008.

[125] Panupong Pasupat and Dilek Hakkani-Tür. Unsupervised relation detection using automatic alignment of query patterns extracted from knowledge graphs and query click logs. In *Proceedings of Sixteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2015.

[126] Genevieve Patterson, Grant Van, Horn Serge, Belongie Pietro, and Perona James Hays. Bootstrapping fine-grained classifiers: Active learning with a crowd in the loop, 2013.

[127] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

[128] Slav Petrov and Dan Klein. Learning and inference for hierarchically split pcfgs. In *Proceedings of the National Conference on Artificial Intelligence*, 2007.

[129] Roberto Pieraccini, Evelyne Tzoukermann, Zakhar Gorelov, J Gauvain, Esther Levin, Chin-Hui Lee, and Jay G Wilpon. A speech understanding system based on statistical representation of semantics. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 193–196. IEEE, 1992.

[130] Jay M Ponte and W Bruce Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR conference on Research and Development in Information Retrieval*, 1998.

[131] Patti Price. Evaluation of spoken language systems: The ATIS domain. In *Proceedings of the Third DARPA Speech and Natural Language Workshop*, pages 91–95. Morgan Kaufmann, 1990.

[132] Deepak Ramachandran and Adwait Ratnaparkhi. Belief tracking with stacked relational trees. In *Proceedings of 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 68–76. ACL, 2015.

[133] L. Ratinov, D. Roth, D. Downey, and M. Anderson. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2011.

[134] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 452–461. AUAI Press, 2009.

[135] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. Relation extraction with matrix factorization and universal schemas. 2013.

[136] Korbinian Riedhammer, Benoit Favre, and Dilek Hakkani-Tür. Long story short–global unsupervised models for keyphrase based meeting summarization. *Speech Communication*, 52(10):801–815, 2010.

[137] Alexander I Rudnicky, Eric H Thayer, Paul C Constantinides, Chris Tchou, R Shern, Kevin A Lenzo, Wei Xu, and Alice Oh. Creating natural dialogs in the Carnegie Mellon communicator system. In *Eurospeech*, 1999.

[138] Stephanie Seneff. TINA: A natural language system for spoken language applications. *Computational linguistics*, 18(1):61–86, 1992.

[139] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Gregoire Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM)*, pages 101–110. ACM, 2014.

[140] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web Companion (WWW)*, pages 373–374, 2014.

[141] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[142] Choonsung Shin, Jin-Hyuk Hong, and Anind K Dey. Understanding and prediction of mobile application usage for smart phones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 83–86. ACM, 2012.

[143] Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, 2013.

[144] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1631–1642, 2013.

[145] Yangqiu Song, Haixun Wang, Zhongyuan Wang, Hongsong Li, and Weizhu Chen. Short text conceptualization using a probabilistic knowledgebase. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2330–2336. AAAI Press, 2011.

[146] Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 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 (EACL)*, pages 102–107. ACL, 2012.

[147] Ming Sun, Yun-Nung Chen, and Alexander I Rudnicky. Learning oov through semantic relatedness in spoken dialog systems. In *Proceedings of the 16th Annual Conference of the International Speech Communication Association*, pages 1453–1457, 2015.

[148] Pirros Tsiakoulis, Milica Gašic, Matthew Henderson, Joaquin Planells-Lerma, Jorge Prombonas, Blaise Thomson, Kai Yu, Steve Young, and Eli Tzirkel. Statistical methods for building robust spoken dialogue systems in an automobile. *Proceedings of the 4th Applied Human Factors and Ergonomics*, 2012.

[149] Gokhan Tur. Multitask learning for spoken language understanding. In *Proceedings of The 31st IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2006.

[150] Gokhan Tur and Renato De Mori. *Spoken language understanding: Systems for extracting semantic information from speech.* John Wiley & Sons, 2011.

[151] Gokhan Tur, Andreas Stolcke, L Lynn Voss, John Dowding, Benoît Favre, Raquel Fernández, Matthew Frampton, Michael W Frandsen, Clint Frederickson, Martin Graciarena, et al. The CALO meeting speech recognition and understanding system. In *Proceedings of 2008 IEEE Spoken Language Technology Workshop (SLT)*, pages 69–72, 2008.

[152] Gokhan Tur, Dilek Z Hakkani-Tür, Dustin Hillard, and Asli Celikyilmaz. Towards unsupervised spoken language understanding: Exploiting query click logs for slot filling. In *Proceedings of The 12th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2011.

[153] Gokhan Tur, Li Deng, Dilek Hakkani-Tür, and Xiaodong He. Towards deeper understanding: deep convex networks for semantic utterance classification. In *Proceedings of The 37th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 5045–5048, 2012.

[154] Gokhan Tur, Minwoo Jeong, Ye-Yi Wang, Dilek Hakkani-Tür, and Larry P Heck. Exploiting the semantic web for unsupervised natural language semantic parsing. In *Proceedings of The 13th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2012.

[155] Gokhan Tur, Asli Celikyilmaz, and Dilek Hakkani-Tür. Latent semantic modeling for slot filling in conversational understanding. In *Proceedings of 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8307–8311. IEEE, 2013.

[156] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 384–394, 2010.

[157] Sudheendra Vijayanarasimhan and Kristen Grauman. Large-scale live active learning: Training object detectors with crawled data and crowds. *International Journal of Computer Vision*, 108(1-2):97–114, 2014.

[158] Fang Wang, Zhongyuan Wang, Zhoujun Li, and Ji-Rong Wen. Concept-based short text classification and ranking. In *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1069–1078. ACM, 2014.

[159] Lu Wang, Dilek Hakkani-Tür, and Larry Heck. Leveraging semantic web search and browse sessions for multi-turn spoken dialog systems. In *Proceedings of The 39th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2014.

[160] Wayne Ward and Sunil Issar. Recent improvements in the CMU spoken language understanding system. In *Proceedings of the Workshop on Human Language Technology*, pages 213–216, 1994.

[161] Kai Wei, Yuzong Liu, Katrin Kirchhoff, and Jeff Bilmes. Using document summarization techniques for speech data subset selection. In *Proceedings of 2013 Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT)*, pages 721–726, 2013.

[162] Kai Wei, Yuzong Liu, Katrin Kirchhoff, Christopher Bartels, and Jeff Bilmes. Submodular subset selection for large-scale speech training data. In *Proceedings of the 39th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3311–3315. IEEE, 2014.

[163] Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015.

[164] Jason D Williams and Steve Young. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422, 2007.

[165] Shasha Xie, Dilek Hakkani-Tür, Benoit Favre, and Yang Liu. Integrating prosodic features in extractive meeting summarization. In *Proceedings of IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*, pages 387–391. IEEE, 2009.

[166] Wei Xu and Alexander I Rudnicky. Task-based dialog management using an agenda. In *Proceedings of the 2000 ANLP/NAACL Workshop on Conversational Systems*, pages 42–47, 2000.

[167] Fan Yang, Gokhan Tur, and Elizabeth Shriberg. Exploiting dialogue act tagging and prosodic information for action item identification. In *Proceedings of the 33rd IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4941–4944. IEEE, 2008.

[168] Nicole Yankelovich. Using natural dialogs as the basis for speech interface design. In *Proceedings of Human Factors and Voice Interactive Systems*, pages 255–290. Springer, 2008.

[169] Wen-tau Yih, Xiaodong He, and Christopher Meek. Semantic parsing for single-relation question answering. In *Proceedings of the 52nd Annual Meeting on Association for Computational Linguistics (ACL)*, 2014.

[170] Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*. ACL, 2015.

[171] Steve Young. CUED standard dialogue acts. Technical report, Cambridge University Engineering Department, 2007.

[172] Steve Young, Milica Gasic, Blaise Thomson, and Jason D Williams. POMDP-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179, 2013.

[173] Mo Yu and Mark Dredze. Improving lexical embeddings with semantic knowledge. In *Proceedings of Association for Computational Linguistics (ACL)*, pages 545–550, 2014.

[174] Ke Zhai and Jason D Williams. Discovering latent structure in task-oriented dialogues. In *Proceedings of the Association for Computational Linguistics (ACL)*, 2014.

[175] Victor Zue, Stephanie Seneff, James R Glass, Joseph Polifroni, Christine Pao, Timothy J Hazen, and Lee Hetherington. JUPITER: a telephone-based conversational interface for weather information. *IEEE Transactions on Speech and Audio Processing*, 8(1):85–96, 2000.

Appendices

AIMU: Actionable Items in Meeting Understanding

A.1 AIMU Dataset

The actionable items annotations are performed on a subset of the ICSI meeting corpus[1] [92], where 22 meetings that were used as test and development data sets in previous work [2] are included for the actionable item detection task. These include three types of weekly meetings, Bed, Bmr, and Bro, which include regular project discussions between colleagues and conversations between students and their advisers[2]. The meeting types and associated data sizes are shown in Table A.1, and the meeting types are detailed as below as also described in the ICSI meeting corpus data collection documentation [92].

Table A.1: The data set description

Name	Type	#Utt
Bed	Even Deeper Understanding	4,544
Bmr	Meeting Recorder	9,227
Bro	Robustness	7,264

- Even Deeper Understanding meetings focus mainly on issues in natural language understanding and neural theories of language.
- Meeting Recorder meetings are concerned mostly with the ICSI meeting corpus data collection project, but include some discussions of more general speech research.
- Robustness meetings focus on signal processing techniques to compensate for noise, reverberation and other environmental issues in speech recognition.

A.2 Semantic Intent Schema

To collect actionable resources in meetings, we annotate each utterance that can trigger an actionable item with the corresponding intent and associated arguments (i.e., slot-fillers).

Although utterances in the human-human genre are more casual and include conversational terms, some intents and the terms related to the actionable item, such as dates, times, and

[1] `http://www.icsi.berkeley.edu/Speech/mr/`

[2] Bed (003, 006, 010, 012), Bmr (001, 005, 010, 014, 019, 022, 024, 028,030), Bro (004, 008, 011, 014, 018, 021, 024, 027)

participants are similar in terms of form and content to the ones in human-machine genre. Figure 8.2 shows utterance examples with the same intents, create_calendar_entry, from different genres (human-machine v.s. human-human). Therefore, we apply the semantic intent schema for an intelligent assistant to meeting conversations.

A.2.1 Domain, Intent, and Argument Definition

We chose five domains that may contain actions triggered by utterances in meetings, where there are total 10 intents in these domains. Table A.2 shows the detailed schema and description.

Table A.2: The description of the semantic intent schema for meetings

Domain	Action		Description
Calendar	Intents	find_calendar_entry	check the calendar of participants to find a specific event
		create_calendar_entry	create a new event in the opening period of participants' calendars
		open_agenda	check the meeting agenda
		add_agenda_item	create a new entry for the meeting agenda
	Argument	contact_name	owners of the targeted calendars
		start_date, end_date	exact date or generic descriptions like "tomorrow" or "yesterday"
		start_time, end_time	exact time or generic descriptions like "afternoon"
		entry_type	e.g. "meeting", "talk", "discussion"
		title	the meeting goal, e.g. "discussion on finite state automaton"
		absolute_location	exact location
		implicit_location	implicitly described location
		agenda_item	the content of the agenda item
Reminders	Intent	create_reminder	create a reminder of participants
	Argument	contact_name	the person who should get the reminder or the speaker when note taking
		reminder_text	the reminder content, also could be a referral
		start_date	targeted date/meeting reference; exact time or generic descriptions
		start_time	exact time or generic descriptions
Communication	Intents	send_email	initialize an email to someone
		find_email	search the specific content from email
		make_call	dial a phone call to someone
	Argument	contact_name	person/people who will be contacted
		email_subject	what the email is about
		email_content	what to include in the email (could also be a description, such as "the paper")
		from_contact_name	the email sender (could be the speaker of the utterance)
OnDevice	Intent	open_setting	launch the setting of devices (e.g. computer, projector)
	Argument	setting_type	the type to modify
Search	Intent	search	retrieve information through the search engine
	Argument	query_term	word sequence to search for

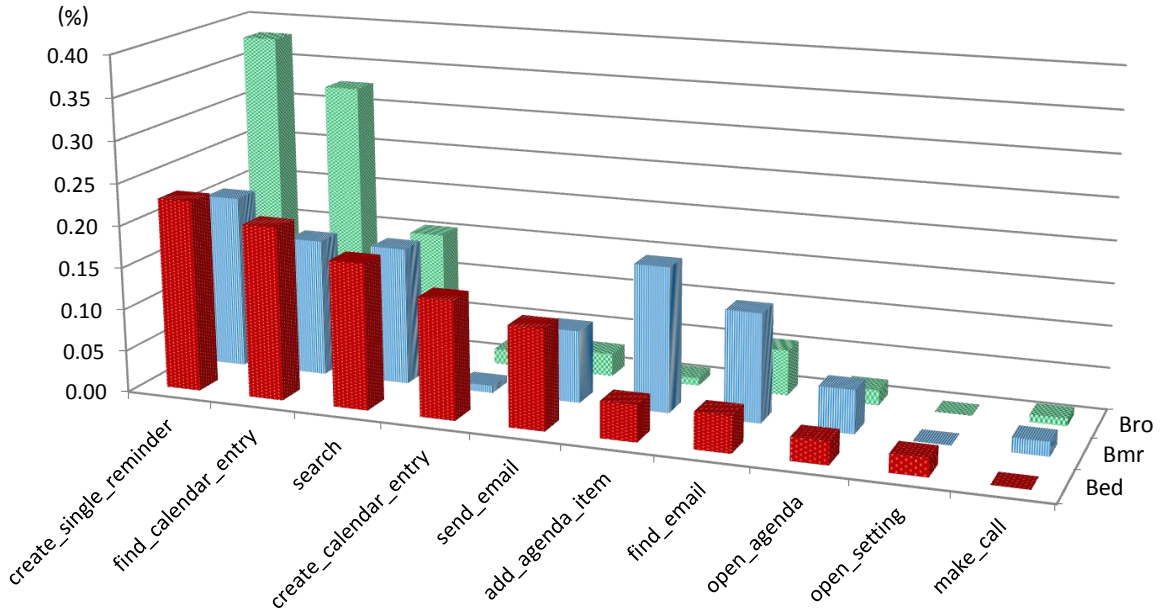


Figure A.1: Action distribution for different types of meetings.

A.3 Annotation Agreement

Randomly selected two meetings were annotated by two annotators, and we tested the agreement for three meeting types using Cohen's Kappa coefficient [45]. The agreements are shown in Table A.3.

Agreement	Bed003	Bed010	Average
Actionable	0.699	0.642	0.644
Type	1.000	1.000	1.000
Overall	0.70	0.646	0.673

Table A.3: Annotation agreement during different settings.

- Actionable Utterance Agreement
  We treat 10 defined actions as positive and others as negative (binary) to compute the actionable utterance agreement. The average agreement about whether an utterance includes an actionable item is 0.644.

- Action Type Agreement
  To deeply analyze action types cross annotators, we compute the agreement on the actionable utterances that both annotators agree with (positive). The average agreement is 1.000, indicating that both annotators always decide on the same action if they both agree that there is an action in this utterance. It also suggests that most actions may not be ambiguous.

- Overall Agreement
  We treat 10 actions and others as total 11 considered labels, and the average agreement about the annotations is 0.673, showing that the actionable items are consistent across human annotators.

A.4 Statistical Analysis

Actionable items were manually annotated based on the designed schema. There are total 318 turns annotated with actionable items, which account for about 1.5% of all of the turns. The detailed numbers are shown in Table A.4, where the number of actionable utterances in Bed meetings are much more than ones in Bmr and Bro meetings. It suggests that the number of actionable utterances may depend on the meeting type. For example, a project status update meeting usually contains more actionable items about reminders, so there is a difference across meeting types and across meeting groups.

Meeting	#Utt	#Utt w/ Actions	Percentage
Bed	4,544	192	4.2%
Bmr	9,227	116	1.3%
Bro	7,264	110	1.5%
Total	21,035	318	1.5%

Table A.4: The annotation statistics

Figure A.1 shows actionable item distribution in the meeting corpus, where it can be found that different types of meetings contain slightly different distribution of actionable items, but some actions frequently occur in all meetings, such as create_single_reminder and find_calendar_entry. Some actions such as open_setting and make_call rarely appear in all meetings.

B Insurance Dataset

B.1 Dataset

The dataset is a set of insurance-related dialogues, where each conversation is a phone call between a customer and an agent collected by MetLife[1]. There are total 155 conversations and the number of utterances is 5,229 segmented by different speakers. The WER is reported as 31.8% using end-to-end deep recurrent neural network models trained on SwitchBoard [113].

Table B.1: The data statistics.

	#Utt	#Word (ASR)	#Word (Manual)
Mean (μ)	33.74	388.64	462.72
S.D. (σ)	28.37	368.41	435.63
Total	5,229	60,239	71,722

B.2 Annotation Procedure

To obtain the reference domain ontology and actionable utterances, annotators were asked to rate the degree about actionable items for all utterances and to annotate semantic concepts. Figure B.1 shows the annotation interface, where following information is asked for annotating.

- The customer intent degree (1 to 5)
  The value indicates whether the annotator thinks the customer intent can be inferred from the given utterance.
  (1: strongly disagree, 2: disagree, 3: maybe, 4: agree, 5: strongly agree)
- The agent action degree (1 to 5)
  The value indicates whether the annotator thinks the agent action can be inferred from the given utterance.
  (1: strongly disagree, 2: disagree, 3: maybe, 4: agree, 5: strongly agree)
- The semantically important frames associated with arguments in the utterances (checked/unchecked)
  The checked FrameNet based frames are salient concepts for understanding the given utterances, which correspond to domain-specific knowledge.

[1] `https://www.metlife.com/`

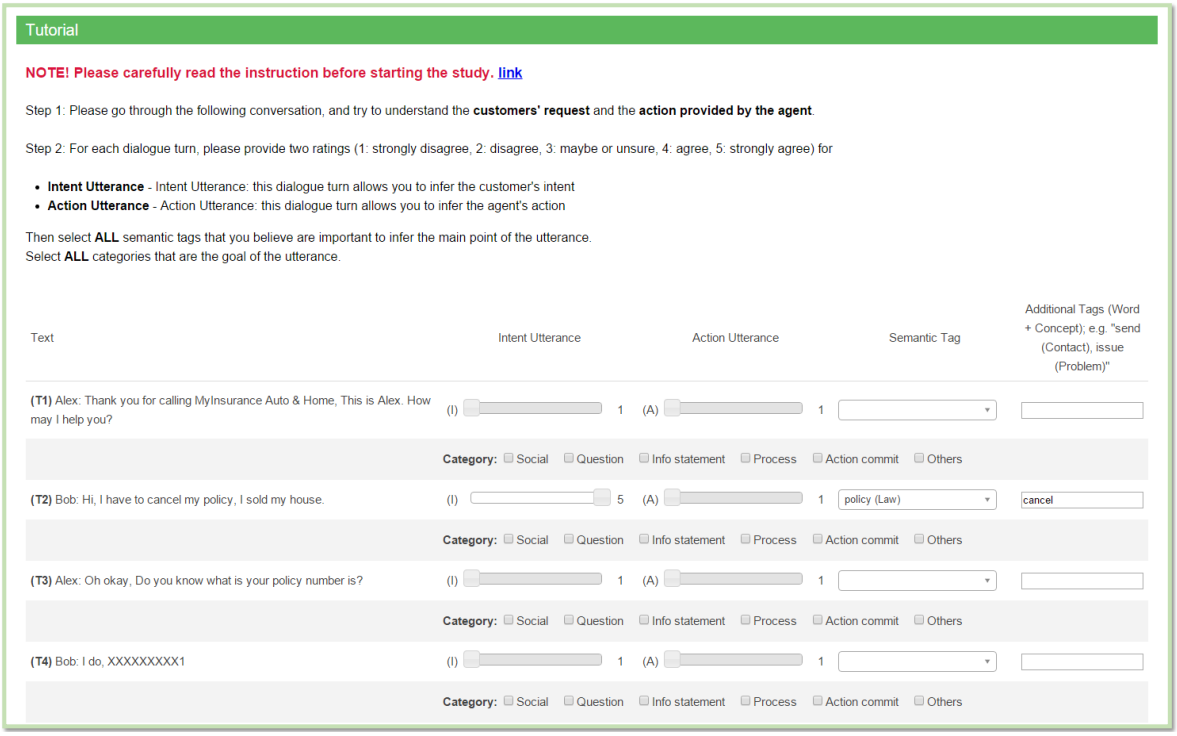


Figure B.1: The annotation interface.

- Additional slots and fillers that are not covered by FrameNet
  The additional concepts are manually created when they are important for domain knowledge but are not produced by the frame semantic parser. These concepts are used to estimate the FrameNet coverage in the dataset.

- Dialogue Act (checked/unchecked)
  The selected acts correspond categories of the goals in the given utterance, where there are six types for selection: Social, Question, Info Statement, Process, Action Commit, and Others.

To obtain more accurate ground truths, we take the degree labels into account when annotators *agree* (degree = 4) or *strongly agree* (degree = 5) for intent utterances and action utterances (threshold is set as 4). The number of recruited annotators is 7, and all of them were asked to start with the same annotation exercise[2] and then allowed them to see the suggested labels. Therefore, annotators can understand requirements better and provide more consistent annotations for quality management. Also, the exercise can be used to compute the rater agreement, where the inter-rater agreement is reported in Table B.2.

Table B.2 shows that the agreement is 0.71 when we consider only utterances annotated with customer intents, and the agreement is 0.66 when we consider only utterances annotated with agent intents. The difference indicates annotators' confusion between intents and actions, because they sometimes refer to the same one but sometimes refer to different ones. The

[2]The exercise human-human dialogue contains 38 utterances and is included in 155 conversations.

Table B.2: The inter-rater agreements for different annotated types (%).

Annotation Type	Cohen's Kappa
Customer Intent	0.7149
Agent Action	0.6610
Customer Intent or Agent Action	0.7567

Kappa agreements higher than 0.65 suggest that action utterances and intent utterances in the human-human dialogues are generally consistent across annotators. In addition, the inter-rater agreement is about 0.76 when we take into account the utterances labeled with either customer intents or agent actions. Then we can conclude that the agreement can be higher if we remove the confusion between intents and actions. Table B.3 shows the detail of pair-wised agreements when we consider utterances annotated with either customer intents or agent actions as actionable utterances. In sum, the number of actionable utterances is 753, accounting for only 14% of total 5,229 utterances.

Table B.3: The detail of inter-rater agreements for actionable utterances (%).

	Rater 2	Rater 3	Rater 4	Rater 5	Rater 6	Rater 7
Rater 1	0.7226	0.8430	0.7226	0.8430	0.3596	0.7226
Rater 2		0.8742	1.0000	0.8742	0.5366	0.7697
Rater 3			0.8742	1.0000	0.6415	0.8742
Rater 4				0.8742	0.5366	0.7697
Rater 5					0.6415	0.8742
Rater 6						0.5366

B.3 Reference Ontology

The reference ontology includes a set of slots and their inter-slot relations as the structure.

B.3.1 Ontology Slots

We compute the frequency of each checked slots and keep ones with the total frequency higher than 10 in total 155 dialogues. Then there are total 31 reference slots based on FrameNet. The slots and associated slot fillers are detailed in Table B.4. To measure the coverage of FrameNet, we analyze the human-labeled additional slots and words, where there are 8 concepts uncovered by FrameNet. The uncovered slots and their fillers are is shown in Table B.5. Therefore, the FrameNet coverage in this dataset is about 79.5% (31 covered and 8 uncovered).

Table B.4: The reference FrameNet slots associated with fillers and corresponding frequency.

Slot (Frequency)	Slot Filler (Frequency)
Statement (45)	claim (21), adding (5), claims (4), note (3), add (2), speak (2), said (1), notes (1), reported (1), added (1), statement (1), remarks (1), stating (1), talk (1)
Transfer (39)	transfer (30), transferring (4), transferred (3), transfered (1), transfering (1)
Law (38)	policy (38)
Request (36)	call (14), calling (11), request (7), called (2), requested (1), requesting (1)
Inspecting (35)	check (26), checking (3), inspection (3), checked (2), inspect (1)
Verification (32)	confirmation (10), verify (8), make sure (5), confirm (5), verifying (2), confirmed (1), verified (1)
Sending (29)	send (21), sending (3), sent (3), mailed (2)
Getting (27)	get (23), got (3), gotten (1)
Receiving (23)	received (10), receive (8), receiving (5)
Quantity (22)	number (14), both (3), all (1), few (1), amount (1), touch (1), any (1)
Commerce_pay (21)	pay (7), paid (6), payment (6), payments (2)
Contacting (20)	contact (6), call (5), email (5), emailed (1), calls (1), phone (1), contacting (1)
Information (20)	information (19), informations (1)
Grant_permission (17)	let (10), approval (3), permit (2), lets (1), approved (1)
Needing (16)	need (14), needed (1), needs (1)
Motion (13)	go (6), going (5), move (2)
Perception_experience (12)	see (11), heard (1)
Awareness (12)	know (10), aware (2)
Have_as_requirement (11)	take (9), need (2)
Sent_items (11)	mail (11)
Locative_relation (11)	up (6), out (4), off (1)
Desiring (11)	wanted (6), want (5)
Assistance (11)	help (6), assist (3), helping (1), assisting (1)
Processing_materials (11)	processing (6), process (3), processed (2)
Capability (11)	can (8), ahead (2), could (1)
Undergo_change (11)	change (8), changes (2), changed (1)
Vehicle (10)	vehicle (8), car (2)
Temporal_collocation (10)	now (6), todays (1), when (1), in-depth (1), currently (1)
Evidence (10)	showing (6), show (2), verify (1), confirming (1)
Visiting (10)	call (10)
Intentionally_act (10)	do (6), did (3), step (1)

Table B.5: The labeled slots uncovered by FrameNet.

Additional Slot	Slot Filler
Cancel	cancellation, canceled, cancel, cancelled
Refund	refund, deduct, debiting
Delete	delete, deleting
Discount	discount, discounts
Benefit	beneficiary, beneficiaries, benefit
Person	whom, who, someone
Status	status, eligibility
Care	care

B.3.2 Ontology Structure

The reference slot pairs connected with typed dependencies are verified by an annotator to produce a list of slot pairs for structure evaluation. The reference ontology structure composed of inter-slot typed dependencies is shown in Table B.6–B.11, where there are 212 slot pairs.

Table B.6: The reference ontology structure composed of inter-slot relations (part 1).

Slot Pair	Dependency
⟨Assistance, Capability⟩	DOBJ-1, AUX
⟨Assistance, Desiring⟩	DEP-1
⟨Assistance, Grant_permission⟩	COMP-1
⟨Assistance, Perception_experience⟩	COMP-1
⟨Assistance, Temporal_collocation⟩	TMOD
⟨Assistance, Transfer⟩	COMP
⟨Awareness, Capability⟩	AUX
⟨Awareness, Commerce_pay⟩	COMP-1
⟨Awareness, Desiring⟩	COMP-1, COMP
⟨Awareness, Getting⟩	COMP-1, DEP
⟨Awareness, Grant_permission⟩	COMP-1
⟨Awareness, Information⟩	PREP
⟨Awareness, Intentionally_act⟩	COMP-1, AUX
⟨Awareness, Needing⟩	COMP, DEP-1
⟨Awareness, Perception_experience⟩	COMP
⟨Awareness, Quantity⟩	DOBJ
⟨Awareness, Request⟩	DEP, RCMOD-1
⟨Awareness, Sending⟩	COMP
⟨Awareness, Statement⟩	DEP
⟨Awareness, Temporal_collocation⟩	ADVMOD
⟨Capability, Commerce_pay⟩	AUX-1
⟨Capability, Contacting⟩	AUX-1
⟨Capability, Evidence⟩	AUX-1
⟨Capability, Getting⟩	AUX-1
⟨Capability, Grant_permission⟩	AUX-1
⟨Capability, Have_as_requirement⟩	COMP, AUX-1
⟨Capability, Inspecting⟩	AUX-1
⟨Capability, Intentionally_act⟩	AUX-1
⟨Capability, Motion⟩	AUX-1
⟨Capability, Perception_experience⟩	AUX-1
⟨Capability, Receiving⟩	AUX-1
⟨Capability, Request⟩	AUX-1
⟨Capability, Sending⟩	COMP, AUX-1
⟨Capability, Sent_items⟩	PRT
⟨Capability, Statement⟩	AUX-1
⟨Capability, Transfer⟩	AUX-1
⟨Capability, Verification⟩	AUX-1

Table B.7: The reference ontology structure composed of inter-slot relations (part 2).

Slot Pair	Dependency
⟨Commerce_pay, Desiring⟩	COMP-1
⟨Commerce_pay, Grant_permission⟩	RCMOD
⟨Commerce_pay, Inspecting⟩	COMP-1
⟨Commerce_pay, Intentionally_act⟩	AUX
⟨Commerce_pay, Law⟩	NSUBJ-1, NSUBJPASS
⟨Commerce_pay, Locative_relation⟩	PRT
⟨Commerce_pay, Motion⟩	NSUBJ-1, COMP
⟨Commerce_pay, Perception_experience⟩	COMP-1
⟨Commerce_pay, Processing_materials⟩	PREP-1
⟨Commerce_pay, Quantity⟩	NSUBJPASS
⟨Commerce_pay, Receiving⟩	DOBJ-1
⟨Commerce_pay, Sending⟩	NSUBJPASS-1
⟨Commerce_pay, Statement⟩	COMP-1
⟨Commerce_pay, Temporal_collocation⟩	ADVMOD
⟨Commerce_pay, Verification⟩	COMP-1
⟨Contacting, Desiring⟩	COMP-1
⟨Contacting, Getting⟩	PREP-1
⟨Contacting, Grant_permission⟩	COMP-1
⟨Contacting, Information⟩	DOBJ
⟨Contacting, Inspecting⟩	DOBJ-1
⟨Contacting, Intentionally_act⟩	COMP-1
⟨Contacting, Motion⟩	COMP-1
⟨Contacting, Needing⟩	COMP-1
⟨Contacting, Quantity⟩	NN-1, DEP
⟨Contacting, Receiving⟩	DOBJ-1
⟨Contacting, Request⟩	COMP-1
⟨Contacting, Sending⟩	DOBJ-1
⟨Contacting, Sent_items⟩	CONJ_OR-1
⟨Contacting, Statement⟩	PREP
⟨Contacting, Temporal_collocation⟩	ADVMOD
⟨Desiring, Evidence⟩	DOBJ
⟨Desiring, Getting⟩	COMP-1, COMP
⟨Desiring, Grant_permission⟩	COMP-1, COMP
⟨Desiring, Have_as_requirement⟩	DOBJ
⟨Desiring, Information⟩	NSUBJ
⟨Desiring, Inspecting⟩	NSUBJ-1, COMP
⟨Desiring, Intentionally_act⟩	AUX, COMP-1

Table B.8: The reference ontology structure composed of inter-slot relations (part 3).

Slot Pair	Dependency
⟨Desiring, Motion⟩	COMP
⟨Desiring, Needing⟩	COMP
⟨Desiring, Quantity⟩	XSUBJ
⟨Desiring, Request⟩	NSUBJ
⟨Desiring, Sending⟩	COMP, ADVCL-1
⟨Desiring, Statement⟩	COMP, COMP-1
⟨Desiring, Temporal_collocation⟩	ADVCL-1
⟨Desiring, Transfer⟩	COMP
⟨Desiring, Vehicle⟩	DOBJ
⟨Desiring, Verification⟩	COMP
⟨Evidence, Law⟩	DOBJ
⟨Evidence, Quantity⟩	DET-1
⟨Evidence, Request⟩	DOBJ-1
⟨Getting, Grant_permission⟩	AUX
⟨Getting, Information⟩	DOBJ
⟨Getting, Intentionally_act⟩	AUX, COMP-1
⟨Getting, Law⟩	DOBJ
⟨Getting, Locative_relation⟩	ADVMOD
⟨Getting, Motion⟩	COMP-1
⟨Getting, Needing⟩	COMP-1
⟨Getting, Perception_experience⟩	ADVCL-1
⟨Getting, Quantity⟩	DOBJ
⟨Getting, Request⟩	COMP
⟨Getting, Sent_items⟩	PREP
⟨Getting, Statement⟩	NSUBJ
⟨Getting, Temporal_collocation⟩	ADVMOD, PRT-1
⟨Getting, Vehicle⟩	VMOD-1, PREP
⟨Getting, Verification⟩	DOBJ
⟨Grant_permission, Have_as_requirement⟩	COMP
⟨Grant_permission, Inspecting⟩	DISCOURSE-1, COMP
⟨Grant_permission, Intentionally_act⟩	COMP, ADVCL-1
⟨Grant_permission, Law⟩	DEP
⟨Grant_permission, Motion⟩	COMP
⟨Grant_permission, Perception_experience⟩	COMP, COMP-1
⟨Grant_permission, Quantity⟩	NSUBJ
⟨Grant_permission, Statement⟩	COMP
⟨Grant_permission, Temporal_collocation⟩	ADVMOD

Table B.9: The reference ontology structure composed of inter-slot relations (part 4).

Slot Pair	Dependency
⟨Grant_permission, Transfer⟩	COMP
⟨Grant_permission, Undergo_change⟩	COMP
⟨Grant_permission, Verification⟩	DEP
⟨Have_as_requirement, Intentionally_act⟩	COMP-1
⟨Have_as_requirement, Law⟩	PREP
⟨Have_as_requirement, Motion⟩	COMP-1
⟨Have_as_requirement, Needing⟩	COMP-1
⟨Have_as_requirement, Perception_experience⟩	COMP
⟨Have_as_requirement, Quantity⟩	NSUBJ
⟨Have_as_requirement, Statement⟩	COMP-1
⟨Have_as_requirement, Vehicle⟩	DOBJ
⟨Information, Inspecting⟩	DOBJ-1
⟨Information, Locative_relation⟩	ADVMOD
⟨Information, Motion⟩	PREP-1
⟨Information, Needing⟩	DEP, NSUBJ-1
⟨Information, Perception_experience⟩	DOBJ-1
⟨Information, Processing_materials⟩	VMOD
⟨Information, Quantity⟩	DET, DOBJ-1
⟨Information, Receiving⟩	DOBJ-1, RCMOD
⟨Information, Sending⟩	DOBJ-1
⟨Information, Verification⟩	DEP, XSUBJ-1
⟨Inspecting, Motion⟩	COMP-1
⟨Inspecting, Needing⟩	COMP-1
⟨Inspecting, Perception_experience⟩	CONJ_AND
⟨Inspecting, Request⟩	COMP-1
⟨Inspecting, Statement⟩	PREP
⟨Inspecting, Temporal_collocation⟩	ADVMOD
⟨Intentionally_act, Law⟩	PREP
⟨Intentionally_act, Motion⟩	COMP-1, PARATAXIS
⟨Intentionally_act, Needing⟩	COMP, COMP-1
⟨Intentionally_act, Perception_experience⟩	DEP, DEP-1
⟨Intentionally_act, Receiving⟩	AUX-1
⟨Intentionally_act, Request⟩	PREP
⟨Intentionally_act, Sending⟩	AUX-1
⟨Intentionally_act, Statement⟩	COMP-1
⟨Intentionally_act, Temporal_collocation⟩	ADVMOD-1, ADVMOD
⟨Intentionally_act, Transfer⟩	NSUBJ

Table B.10: The reference ontology structure composed of inter-slot relations (part 5).

Slot Pair	Dependency
⟨Intentionally_act, Undergo_change⟩	AUX-1, DEP
⟨Intentionally_act, Verification⟩	DEP-1
⟨Law, Motion⟩	NSUBJ-1
⟨Law, Needing⟩	DOBJ-1
⟨Law, Perception_experience⟩	DOBJ-1
⟨Law, Quantity⟩	NN-1
⟨Law, Request⟩	COMP, PREP-1
⟨Law, Sending⟩	RCMOD, PREP-1
⟨Law, Statement⟩	PREP-1
⟨Law, Temporal_collocation⟩	ADVMOD
⟨Law, Undergo_change⟩	COMP-1
⟨Law, Vehicle⟩	PREP
⟨Locative_relation, Motion⟩	ADVCL, PRT-1
⟨Locative_relation, Perception_experience⟩	ADVMOD-1
⟨Locative_relation, Sending⟩	PRT-1
⟨Locative_relation, Sent_items⟩	PRT-1
⟨Locative_relation, Temporal_collocation⟩	CONJ_AND, ADVMOD-1
⟨Motion, Needing⟩	COMP, COMP-1
⟨Motion, Perception_experience⟩	COMP, DEP-1
⟨Motion, Quantity⟩	XSUBJ
⟨Motion, Receiving⟩	COMP
⟨Motion, Sending⟩	COMP
⟨Motion, Statement⟩	COMP, COMP-1
⟨Motion, Temporal_collocation⟩	ADVMOD
⟨Motion, Transfer⟩	COMP
⟨Motion, Vehicle⟩	DOBJ
⟨Motion, Visiting⟩	DOBJ
⟨Needing, Perception_experience⟩	COMP
⟨Needing, Quantity⟩	DOBJ
⟨Needing, Request⟩	COMP
⟨Needing, Statement⟩	COMP-1, COMP
⟨Needing, Temporal_collocation⟩	DEP-1, ADVMOD
⟨Needing, Transfer⟩	COMP
⟨Needing, Verification⟩	COMP
⟨Perception_experience, Quantity⟩	DOBJ
⟨Perception_experience, Receiving⟩	ADVCL
⟨Perception_experience, Request⟩	CONJ_AND-1

Table B.11: The reference ontology structure composed of inter-slot relations (part 6).

Slot Pair	Dependency
⟨Perception_experience, Sending⟩	CONJ_AND
⟨Perception_experience, Statement⟩	COMP
⟨Perception_experience, Temporal_collocation⟩	PREP-1, ADVMOD
⟨Processing_materials, Statement⟩	DOBJ
⟨Processing_materials, Verification⟩	PREP-1
⟨Quantity, Statement⟩	NN, DET-1
⟨Quantity, Vehicle⟩	NN
⟨Receiving, Request⟩	COMP
⟨Receiving, Sent_items⟩	PREP
⟨Receiving, Statement⟩	COMP, COMP-1
⟨Receiving, Temporal_collocation⟩	ADVMOD
⟨Receiving, Visiting⟩	COMP
⟨Request, Statement⟩	PREP
⟨Request, Temporal_collocation⟩	DEP-1, ADVMOD
⟨Request, Visiting⟩	DEP
⟨Sending, Sent_items⟩	DOBJ
⟨Sending, Statement⟩	IOBJ, DET-1
⟨Sending, Temporal_collocation⟩	ADVMOD
⟨Sending, Verification⟩	DOBJ
⟨Sent_items, Temporal_collocation⟩	NN-1
⟨Statement, Temporal_collocation⟩	ADVMOD
⟨Statement, Undergo_change⟩	COMP
⟨Statement, Vehicle⟩	DOBJ
⟨Statement, Verification⟩	VMOD
⟨Temporal_collocation, Undergo_change⟩	ADVMOD-1
⟨Temporal_collocation, Verification⟩	NSUBJ, ADVMOD-1
⟨Transfer, Visiting⟩	DOBJ