Distribution Agreement

In presenting this thesis or dissertation as a partial fulfillment of the requirements for an advanced degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis or dissertation in whole or in part in all forms of media, now or hereafter known, including display on the world wide web. I understand that I may select some access restrictions as part of the online submission of this thesis or dissertation. I retain all ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Signature:

_____ _____

Ali Ahmadvand Date

Knowledge-Aware User Intent Inference for Web Search and Conversational Agents

By

Ali Ahmadvand
Doctor of Philosophy

Computer Science and Informatics

Eugene Agichtein, Ph.D.
Advisor

Jinho Choi, Ph.D.
Committee Member

Carl Yang, Ph.D.
Committee Member

Surya Kallumadi, Ph.D.
Committee Member

Accepted:

Kimberly Jacob Arriola, Ph.D.
Dean of the James T. Laney School of Graduate Studies

Date

Knowledge-Aware User Intent Inference for Web Search and Conversational Agents

By

Ali Ahmadvand
M.Sc., Emory University, GA, 2019

Advisor: Eugene Agichtein, Ph.D.

An abstract of
A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2021

Abstract

Knowledge-Aware User Intent Inference for Web Search and Conversational Agents
By Ali Ahmadvand

User intent inference is a critical step in designing intelligent information systems (e.g., conversational agents and e-commerce search engines). Accurate user intent inference improves user experience and satisfaction, but is a challenging task since user utterances or queries can be short, ambiguous, and contextually dependent. Moreover, in an e-commerce setting, the collected datasets are often labeled by weak supervision (e.g., click-through data), resulting in an imbalanced and sparse dataset. To address these problems, my dissertation proposes integrating entity knowledge-bases, conversation context, and user profile information to improve user intent inference for conversational agents. Additionally, I investigate joint learning, product taxonomies, and unlabeled domain-specific corpora (e.g., catalog) to improve query intent inference in e-commerce search.

To evaluate the proposed models, I examine the user intent inference for two main settings: 1) open-domain conversational agents and 2) e-commerce search engines. The conversational agent research is evaluated on conversations collected from real users as part of Amazon Alexa Prize competitions, and the e-commerce efforts use real query logs collected from The Home Depot's search engine. My dissertation shows that leveraging entity knowledge-base, conversation context, and user profile information accounts for most improvements for the conversational setting. The results demonstrate that the proposed models significantly enhance topic classification accuracy by 15% and dialogue act accuracy by 8% for conversational agents. For e-commerce search, the dissertation shows that joint-learning, product taxonomies, and unlabeled domain-specific corpora can significantly improve intent inference accuracy. The proposed models improve the performance of the top-1 retrieved documents by 6%-8% on standard metrics for e-commerce search. The results in both settings offer a significant improvement over state-of-the-art deep learning methods. The insights and findings in this dissertation suggest a promising direction for developing the user intent inference in both open-domain conversational agents and e-commerce search.

Knowledge-Aware User Intent Inference for Web Search and Conversational Agents

By

Ali Ahmadvand
M.Sc., Emory University, GA, 2019

Advisor: Eugene Agichtein, Ph.D.

A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2021

Acknowledgments

This dissertation was shaped by the support of multiple professionals and organizations who helped hone my skills as a researcher. Our collaborations allowed me to grow in the machine learning, natural language processing, search, and conversational AI fields and the work required towards reaching this milestone. It is impossible to convey my appreciation in only a few words, and I want to honor each of you here.

First, I want to express my gratitude to my dissertation committee for their dedication and leadership: Dr. Eugene Agichtein, Dr. Jinho Choi, Dr. Surya Kallumadi, and Dr. Carl Yang. Your guidance made the difference. Dr. Agichtein, as my lab advisor and mentor, you shaped me into a stronger scholar. You supported me throughout the Ph.D. program to cross the finish line. You will always be an inspiration as I seek to shape the next steps in my career. Dr. Choi, your advice and wisdom are always a source of motivation for me. It was an honor working with you and seeing our efforts pay off by winning the Alexa Prize. Dr. Kallumadi, your friendship and support throughout our collaborations with The Home Depot inspired me and made me a better researcher.

Next, a special thank you to Emory University and the Computer Science Department for providing the grant and access to this life-changing opportunity as part of their Ph.D. program. It afforded an excellent environment to grow as a professional and scholar. Furthermore, this dissertation would not be possible without the collaborations formed with The Home Depot and Amazon Alexa. My deep appreciation for the financial and computing support from the Amazon Alexa Prizes 2017, 2018, and 2019 groups and The Home Depot search and NLP team.

Also, I am grateful to my other mentors and supporters for their insight, friendship, and belief in me: Dr. Elizabeth Tamez Méndez, Dr. Julia Kiseleva, and Dr. Michael Gamon. You shared your knowledge, affirmed my career, and our collaborations enriched this process. To my esteemed labmates and Emory colleagues, your

friendship and companionship made the journey more exciting. I learned so much from you: Zihao Wang, Jason Choi, Harshita Sahijwani, Sergay Volokhin, Jianghong Zhou, Sayyed M. Zahiri, Farnaz Tahmasebian, Mani Sotoodeh, Alessandro Barone, Payam Karisani, Reza Karimi, Denis Sevankov, and Tomasz Jurczyk.

Finally, I am immensely grateful for the relentless love and sacrifices from my parents Reza and Fatemeh. Your belief in my dreams and support have kept me going even when the miles separated us all these years. Thank you to my siblings, Rahim, Leila, Maryam, Fereshteh, Zahra, Mahboobeh, and Pariya, for all their love and kindness. You have made it possible for me to be the first to earn a doctoral degree in our family. To my nieces and nephews, thank you for keeping me company and making me laugh. I hope this accomplishment inspires your own sense of purpose, dreams, and goals. I am very proud of you.

To my loving mother and father,

who believed in me and taught me the value of education.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction and Motivation

Artificial intelligence systems that have real-time interactions with humans are ubiquitous in our modern society. Two main paradigms of intelligent information systems are 1) conversational agents (e.g., Amazon Alexa, Apple Siri, and Microsoft Cortana), and 2) search engines such as those in e-commerce (e.g., The Home Depot, Amazon, and eBay). Although intelligent information systems under each of these paradigms support different modalities such as speech and query, both require efficiently understanding user to provide an accurate response for the user's request. Since user intent understanding is a preliminary and fundamental step in the user journey while interacting with intelligent information systems. A misunderstanding of the user intent can cause a cascading effect throughout the system and degrade the overall performance. The new problems towards user intent understanding raised by these new applications open opportunities for researchers to develop new ideas in these fields. To keep bridging these gaps, my Ph.D. dissertation focuses on user intent inference for both conversational agents and e-commerce search engines. I discuss the different methodologies to improve user intent inference throughout this dissertation. Each model proposed in this dissertation focus on addressing a specific element of user intent inference (e.g., topic classification, dialogue act prediction, query catego-

rization, etc.) depending on the application. The ultimate aim is to improve the user intent inference by improving these underlying components. My dissertation consists of two phases of research development. The first phase is my work on the Amazon Alexa Prize competitions[1] and my contributions within them [138, 1, 43]. In the second section, my research on query understanding for e-commerce search engines is discussed.

The following section discusses the open-domain conversational agents, e-commerce search, and the challenges for an accurate user intent inference. Finally, I summarize research contributions to address some of the challenges in user intent inference.

1.1 User Intent Inference in Open-Domain Conversational Agents

In the first phase of my dissertation, I plan to address specific issues related to knowledge-aware user intent inference and classification for open-domain conversational agents. In this dissertation phase, I tested my proposed models on the Amazon Alexa Prize projects developed as part of the annual Amazon Alexa Prize competitions. Alexa Prize is a global grand challenge that Amazon has held since 2017, and it seeks to advance conversational AI. In this competition, users were asked to talk to our conversational agent and give it a rating from 1.0 to 5.0 (inclusive) based on their experience.

Current conversational agents use a component-based architecture for designing an open-domain conversational agent [138, 1, 43]. In these systems, domain-specific components such as *Movie* Bot[2] are responsible to generate an answer for assigned utterance. To assign the incoming utterances to proper components, an

[1] *https://developer.amazon.com/alexaprize*

[2] *https://www.amazon.com/Amazon-MovieBot/dp/B01MRKGF5W*

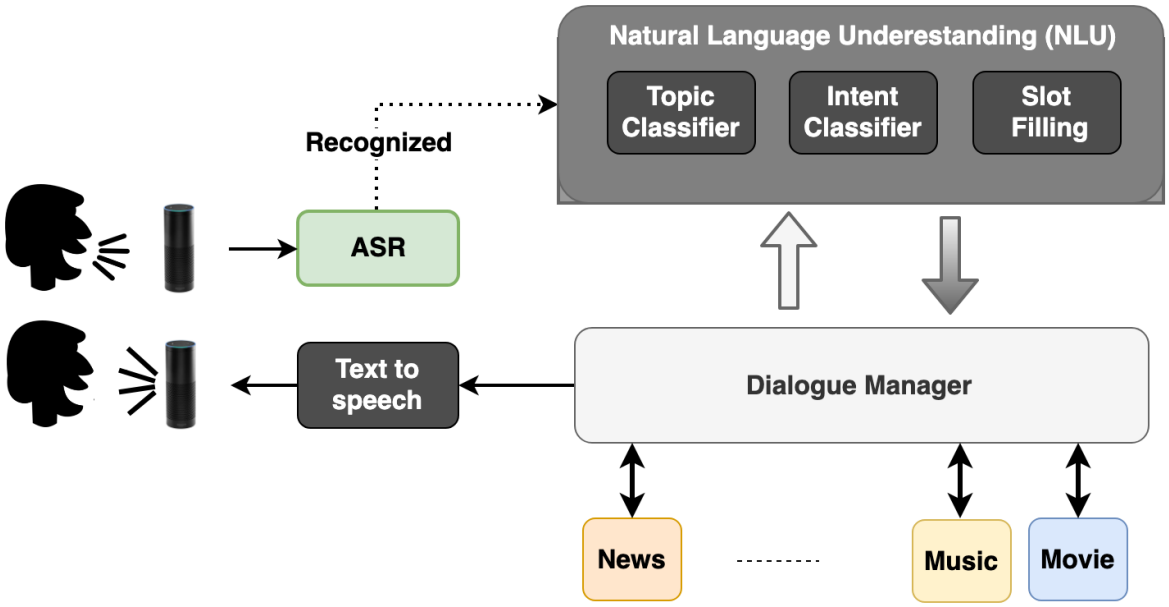

Figure 1.1: Conversational agent flowchart, where ASR stands for Automatic Speech Recognition.

open-domain conversational agent requires an accurate user intent inference module to determine the conversation, which consists of three major parts, including *topic*, *intent*, and *slot filling*. To understand user intent, I need to identify that each utterance follows a specific structure and training phrases (slot filling). For example, when a user says "who won The Hawks and Boston Celtics game," they want to talk about *Sports*, with a *Information-Request* intent, where it follows the slot filling pattern of "who/OTHER, won/OTHER, the/SPORTTEAM, Hawks/SPORTTEAM, and/OTHER, Boston/SPORTTEAM, Celtics/SPORTTEAM, game/OTHER ." As a result, an accurate intent inference results in a coherent and engaging conversation, which spurs dramatic improvements of conversational agents. Figure. 1.1 shows an open-domain conversational agent flowchart with associate components.

In the next section, I discuss the challenges for the user intent inference in open-domain conversational agents.

1.1.1 Challenges for User Intent Inference in Conversational Agents

User intent inference for open-domain conversations is a more challenging task compared to a regular text classification due to different factors. 1) Human-machine utterances are often short and ambiguous and suffer from the lack of textual information. 2) conversations contain contextual evidence between the preceding utterances which need to be addressed. 3) the quality of current Automatic Speech Recognition (ASR) is not in a human-level performance, where they often suffer from recognizing specific types of entities that do not appear much on trained corpus such as names, companies, or locations [1, 2, 3, 31, 138, 43].

Open-domain conversational agents increasingly require accurate identification of the utterance's topic (domain) and intent. Often, domain classification is one of the first steps, and an error in this step can cause a cascading effect throughout the system and degrade the overall performance. Most current conversational systems use a component architecture [114], where each user utterance is assigned to a domain-specific component such as *Movie* Bot[3]. Mis-classifying the intent of an utterance and assigning it to the wrong component can produce erroneous responses and degrade the user experience.

An important challenge for conversational domain classification is that keyword-based classification is not sufficient. Domain-specific keywords or triggers might help for queries like "Let's talk about my dog", since the word "dog" frequently appears in utterances from the *Pets_Animals* domain. However, they do not enable us to correctly classify utterances containing ambiguous keywords that can refer to multiple entities. For example, to correctly classify utterances like "When is the next Hawks game?", I need to take into account all the possible types of entities that the word "Hawks" might be referring to, i.e., the bird hawk and the sports team Atlanta Hawks,

[3] *https://www.amazon.com/Amazon-MovieBot/dp/B01MRKGF5W*

and the context, which mentions "game."

Moreover, creating new entities, like recent movies, makes the model obsolete with time. To fix this problem, it would be necessary to constantly update the model by incorporating new information about people, organizations, movies, and other entities, which can cause unintended effects and be inefficient.

In order for a conversational system to respond properly, it must first understand the intent of the user utterance. Dialogue Act (DA) identification is a traditional approach in dialogue system research, which aims to predict the goal of each utterance, such as information request, statement of opinion, greeting, opinion request, and others. [109] categorized DAs as having two main categories: 1) primary intents; 2) secondary intents. Each of these intents can further be divided into implicit or explicit. Since identifying these intents correctly is crucial for dialogue systems, this problem has been studied extensively for decades for human-human conversations. Recently, this idea was also extended to human-machine conversations [120, 89, 74].

Utterances in natural conversations are contextually dependent, making the DA prediction challenging. For example, an utterance like "Oh, yeah" can be interpreted as "Yes-Answer," "Accept-Agree," or "Backchannel", which requires the previous context to disambiguate [89, 18]. Therefore, I propose a context-aware model for this task. As mentioned above, utterance understanding is heavily dependent on context. Considering utterances independently from contextual information can cause ambiguity and errors. For example, voice-based conversational agents rely on ASR models, which, despite significant progress, still tend to introduce mistakes. For instance, the utterance "hmm tell me about the new Mercedes" can be mistakenly transcribed as "hmm tell me about the new Sadie's." In these instances, contextual information is helpful, as I show in our empirical results. Even without ASR errors, contextual information is crucial. Users commonly say short utterances without explicitly specifying the domain, for instance, by saying, "Let's talk about eagles." This query would be

ambiguous even to humans since it is unclear whether the person wants to talk about the bird eagle or the Eagles' sports team. Contextual information, like the previous utterances and the system's state, would help identify the domain of this utterance.

In the next section, I explain user intent inference for Web search engines.

1.2 User Intent Inference in Web Search Engines

This part of my research focuses on the user intent inference in web search engines, focusing on e-commerce search engines. All the models were developed during my collaboration with The Home Depot[4]. The Home Depot receives billions of search queries every year and collects terabytes of data logs from the user experience during their interaction with the website. My research focused on enhancing their high-level user intent understanding module. To this end, I proposed a hierarchical architecture for the user intent classification, where in the first layer, the intent of the users in purchasing a product or seeking information (product vs. informational) is discovered. Then, in the next layer, if the user intent is purchasing, another intent classifier is applied to determine whether the query is either broad or specific. Otherwise, if the user's intent is information seeking, the type of information the user is looking for is determined using another classifier. As a result, the search engine can guide a user to an appropriate web page to handle the user's request. Figure 1.2 shows how an intent hierarchy looks like in a general e-commerce search engine.

Query intent understanding and ranking are the building blocks of advanced retrieval systems (e.g., e-commerce search engines) [36]. Large commercial websites collect billions of product information in a hierarchical structure across various granularities called taxonomy or catalog. The taxonomy is prudently developed by human supervision based on the product document descriptions. Query understanding can be defined as mapping search queries to at least one of the fine-grained product cat-

[4] *https://www.homedepot.com/*

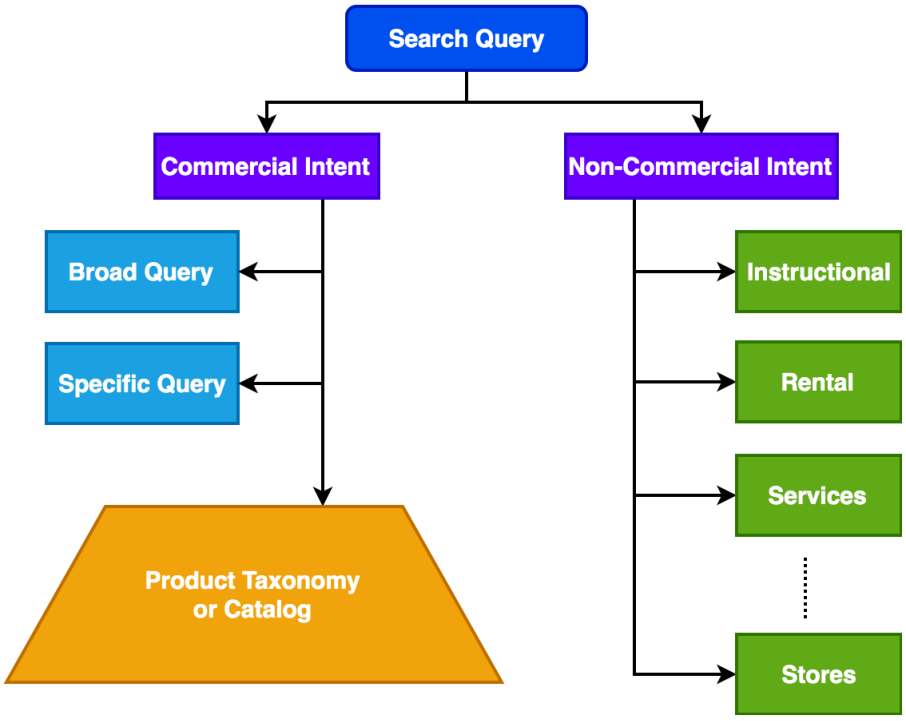

Figure 1.2: Intent hierarchy in an e-commerce search engine. Each color shows an intent hierarchy level.

egories that exist in the taxonomy [156] (except non-commercial queries, which are rare and only cover 1.5% of our traffic). In other words, the goal is returning the most semantically relevant fine-grained categories to a query. Query understanding is just the beginning of the search process. Later, the output of this step can provide reliable signals for a ranker to improve the final ranking results [42]. For instance, when a customer inputs a query such as "grass cutter with 18-Volt Lithium-Ion battery, " an e-commerce search engine should suggest the relevant product categories such as (e.g., *lawn mower, string trimmer, brush cutter, and hedge trimmer*). These categories satisfy the customer's intent and more relevant ranking results, resulting in a successful shopping experience. Therefore, an accurate user intent understanding profoundly impacts the overall system performance and, consequently, the company's revenue from online sales.

In the next section, I discuss the challenges for the user intent inference in e-

commerce search engines.

1.2.1 Challenges for User Intent Inference in E-commerce Search

User intent inference in e-commerce search is a non-trivial task. 1) queries are short, vague, and ambiguous. Utilizing only textual information from the queries is not sufficient for an accurate product category mapping [49]. 2) queries with similar textual information and only slight variations such as "3-cup white rice cooker" and "3-cup white pressure cooker" belong to different fine-grained categories. However, queries with no term overlap like "French door 32-inch refrigerator" and "white fridge" are semantically matched and may belong to similar categories. 3) customers have different ways to express the same intent. For example, "blocktile white-colored - 12 in. x 12 in." and "block tile white 12x12 inches" are the same queries while their word-level representations are far away from each other. Fourth, product category mapping is a non-exclusive problem. When a customer inputs a broad concept like "Kitchenware" or a brand name like "ryobi," results could cover a wide range of correct product categories. In other words, a search engine must include a broader range of possible correct categories while simultaneously keeping precision as high as possible [156]. 4) not only is query understanding an extreme multi-label text classification problem (XMTC), which raises challenges such as data sparsity and scalability [86], it also represents a soft distribution over relevant categories. Concerning this issue, a knowledge-aware model needs to develop to extract the unbiased relevance score for each query and product category pair. 5) the severe data imbalance problem resulted from customer *bias* towards some specific products in general or at a particular time. Also, the product categories' correlations directly impact customer click behavior, where some of them received more click rates than usual, and others get fewer click rates, and 6) queries with low customer behavior feedback (e.g., *tail*

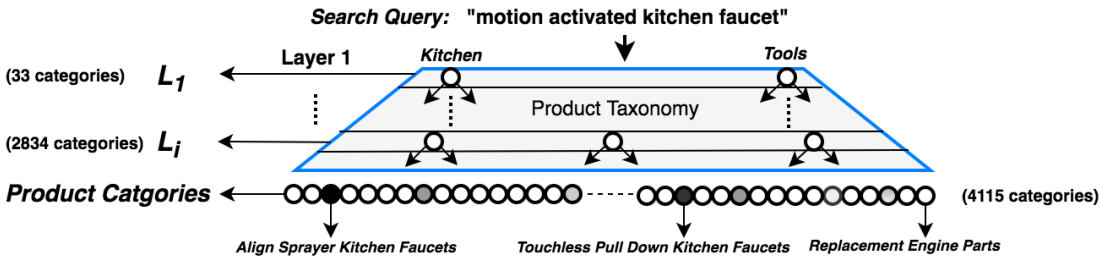

Figure 1.3: Query understanding procedure.

and *torso*) are more challenging to classify as they have a high signal-to-noise ratio. Current neural models achieve a *softer representation* with *richer compositionality* of the queries compared to conventional term-based models [154, 4, 6, 7].

Query understanding is an essential step in developing advanced retrieval systems (e.g., e-commerce search engines) [36]. In an e-commerce setting, one aspect of query understanding is achieved by mapping a query to a set of relevant product categories [156]. For example, for the query " motion activated kitchen faucet", an e-commerce search engine should return products from relevant categories like *bath, plumbing, kitchen*. These categories match the customer's intent and provide signals for downstream tasks such as retrieval and ranking. In this dissertation, I develop a new model for query understanding in an e-commerce search engine, depicted in Fig. 1.3. Fig. 1.3 shows the query understanding procedure where a search query like "motion activated kitchen faucet" is mapped to a set of relevant product categories in a hierarchical product taxonomy.

In summary, in this phase of my dissertation, I addressed specific issues related to user intent inference for e-commerce search engines. The next section describes my contributions in knowledge-aware user intent inference for open-domain conversational agents and e-commerce search engines.

1.3 Dissertation Research Questions

In summary, the main research objectives for advancing user intent inference that this dissertation plan to address are as follows:

1. **RQ1:** What external source of knowledge could be leveraged to effectively improve user intent inference in intelligent information systems (e.g., conversational agents and e-commerce search engines)?

2. **RQ2:** What are efficient approaches to represent and incorporate the external source of knowledge for both applications?

3. **RQ3:** In what ways can external sources mitigate the data imbalance problem and data sparsity for user intent inference?

In the next section, I describe my contributions to the field and the methods that I developed to address the research questions in this dissertation.

1.4 Contributions and Dissertation Structure

This section introduces my research questions and contributions to the user intent inference for both conversational agents and e-commerce search engines. To address the challenges that I discussed in sections 1.1.1 and 1.2.1, I proposed to leverage knowledge from relevant resources to enhance the quality of intent understanding. These external sources of information (e.g., entity type, product documents, and user clicks) are utilized to enrich the representation of utterance or search queries.

1.4.1 Contributions to Open-Domain Conversational Agents

The first part of my dissertation focuses on developing multiple machine learning models to advance user intent inference for conversational agents. During Amazon Alexa Prize 2018 [1], we developed an open-domain social bot named IrisBot that can converse in multiple domains with real users. The real conversation logs during these interactions are used to evaluate our models. I developed mainly three different knowledge-aware user intent understanding models.

First, I introduced the Concurrent Conversational Entity-aware Topic classifier (ConCET), which incorporates entity-type information and the utterance content features for topic classification. Specifically, ConCET utilizes entity information to enrich the utterance representation, combining character, word, and entity-type embeddings into a single representation [3]. Next, a version of the ConCET was used in a mixture of experts to make it more stable and accurate [1]. Second, I proposed a novel method, CDAC (Contextual Dialogue Act Classifier), a simple yet effective deep learning approach for contextual dialogue act classification. Specifically, I used transfer learning to adapt models trained on human-human conversations to predict dialogue acts in human-machine dialogues [2]. To investigate the effectiveness of our method, I train our model on the well-known Switchboard[5] human-human dialogue dataset and fine-tune it for predicting dialogue acts in human-machine conversation data. Finally, current intent inference models have trouble understanding particular user intents (e.g., topic switching intent). To improve this capability of IrisBot, I developed a knowledge-aware recommendation component to propose the next most exciting macro-topic to the user [1, 5]. Since the right topic to recommend user depends on both prior user interests and the conversation context, I leverage knowledge from user profile information to incorporate into the prediction process.

[5] `https://catalog.ldc.upenn.edu/LDC97S62`

Finally, I complimented our quantitative results with a detailed analysis of system performance, which could be used to improve conversational agents.

1.4.2 Contributions to E-commerce Search

The second part of my dissertation focuses on developing multiple machine learning models to improve the quality of user intent inference for conversational agents. The real search logs during user interactions with the Home Depot Search engine are used to evaluate the proposed models. The dissertation focused on multiple knowledge-aware models to generate the datasets and intent inference.

First, I proposed an active learning model in conjunction with distant supervision to generate a labeled dataset for the users' intents. This model utilizes user behavior data (e.g., click rate) to create the category labels associated with each commercial query. To generate the labels for non-commercial queries, I incorporate weak human supervision signals combined with a machine learning model. This section is followed by implementing different individual classifiers for each intent hierarchy. Later, my main focus was implementing a joint and multi-task learning model named JointMAP to study the search query's commercial category simultaneously. JointMap works by leveraging the transfer bias that exists between these two related tasks through a joint-learning process. Then, I proposed a new label representation to incorporate knowledge from category interactions into user intent understanding. In this model, category-category concurrences are leveraged to enrich the textual query representations. Finally, I proposed a deep learning model to incorporate pseudo-relevance feedback from product documents to enhance the query representation in the latent space. The proposed model leverages a hierarchical attention mechanism on each particular product document field.

1.5 Summary and Dissertation Structure

In summary, modern artificial intelligence systems have a long way to go before reaching a level of behavior as intelligently as humans. One of the fundamental steps in designing intelligent information systems is understanding what the real user (humans) is looking for while interacting with them. I focus on this critical step for two types of intelligent information systems throughout my dissertation: conversational agents and e-commerce search engines. The research questions guiding my dissertation focus on improving the quality of current user intent inference methods. A better understanding of user intents can be leveraged to enhance the performance of various downstream tasks such as ranking, resulting in an enriched user experience for both conversational agents and e-commerce search. The insights from my study can enable more intelligent conversational systems and e-commerce search qualities and can be used for further improvements of conversational agents and e-commerce search.

In the next chapter, I present related work to place my contributions in the context. It follows by chapter 3 where I explain the different sources of external knowledge that I leveraged to improve user intent inference in both conversational agents and e-commerce search. Chapter 4 describes my contributions in knowledge-aware user intent inference for open-domain conversational agents. In chapter 5, I explain my contributions in knowledge-aware user intent inference for e-commerce search. I discuss the strengths of the proposed approaches and then present some potential limitations in chapter 6. Finally, in chapter 7, I summarize what I have done so far and present the potential future research directions for user intent inference, which focuses on user intent inference for both conversational agents and e-commerce search engines.

Chapter 2

Related Work

In this chapter, I present related work to place my contributions in context. I briefly review the published research on general user intent inference, then discuss the published work on open-domain conversational agents, critical techniques used for user intent inference in the Web search engine, focusing on e-commerce search, and finally provide a summary for the section. The chapter's content related to conversational agents is based on references [3, 2, 5], published in CIKM'2019, SIGIR'2019, and CHIIR'2020. Also, the content related to e-commerce search is based on references [4, 6, 7], published in SIGIR'2020, SIGIR e-com 2021, and SIGIR'2021.

2.1 User Intent Inference

User intent inference has been a fundamental topic in Web search since the first search engines like AliWeb, and later Google and Yahoo emerged [72]. In commercial companies like Amazon and The Home Depot, user intent inference has a direct impact on the overall revenue of the company [156]. Developing an effective conversational search that can interact with users in a conversation-like manner has been a long-term goal in designing a search engine. Companies like Google began developing conversation-wise search methods for users in an attempt to make the experience as

close as possible to the way people interact in daily life [52]. Thus, unlike traditional Web search, which typically consists of one-sided interactions from a user making a request or question, the search session can be designed in a two-sided style. In other words, a conversational search system should include at least two capabilities: 1) being able to retrieve information instantly upon users' requests and proactively recommend information if the user does not know what to ask; and 2) maintaining the state and flow of conversation to allow the user to naturally refine or participate in the search process [62]. Recently, with the new developments in speech recognition technology [146], companies like Amazon have gone further by investing heavily in developing their social bots, which can keep a coherent conversation in an open-domain and human-like manner with their customers [138, 1]. As a result, developing an accurate and efficient Natural Language Understanding (NLU) unit for these social bots has been in the spotlight in recent years [48]. In this dissertation, I developed several models to improve the quality of user intent inference in both conversational agents and e-commerce search engines.

2.2 User Intent Inference in Open-Domain Conversational Agents

The concept of conversation has been considered as a human trait for a long time [126]. In the early '50s, Alan Turing [50] connected the concept of "true intelligence" to the capability of maintaining a human-like conversation; where not only is a machine required to think, it also needs to converse indistinguishably like humans. In the '60s, Joseph Weizenbaum developed ELIZA [140], the first conversational bot that can use similar patterns to carry human-like conversations. Recently, with the advancements in ASR systems, conversational AI agents have been gaining traction. The Amazon Alexa Prize annual challenges are one of the platforms that provide monetary

incentives for researchers to advance conversational AI [114, 64]. Currently, several conversational systems have been developed for application in a variety of fields, such as mental well-being [104] and therapy [44]. Conversational agents [45] are also applied to educational purposes such as tutoring. Moreover, unlike traditional Web search or automatic question answering, conversational agents could enable users to formulate their information needs better and even help them ask suitable topics.

Conversational agents such as Amazon's Alexa, Apple's Siri, and Microsoft's Cortana are becoming increasingly popular. Yet, there is yet much room for improvement, and this has inspired extensive recent research, such as references [141, 142, 138, 1]. Su et al. [121] categorized dialogue systems into two main paradigms: task-based and chatbot style. Chatbot-style models are designed for chit-chatting with the user, while task-based models follow a goal-oriented conversation [113]. One of the early models for chatbot-style is Artificial Intelligence Markup Language (AIML). Richard Wallace developed this template-based model. Likewise, the Ravenclaw system is an example of early task-based conversational agents [14]. This model was built for an online room reservation service. Papaioannou et al. [108] developed a system that is trained by reinforcement-learning techniques, and it combined both the AIML template-based chatbot and a task-specific dialogue management system.

Currently, most of the conversational agents are designed for a single domain like *Movie* or *Music*. Having an open-domain conversational agent that coherently and engagingly converses with humans on a variety of topics is a long-term goal for dialogue systems [48, 129]. Deploying a user intent inference module to understand the conversational topics is crucial for the success of open-domain conversational systems. Despite current advancements in designing the open-domain conversational agent, generating coherent and engaging conversations is challenging. To help address some challenges in current conversational systems, the first part of my Ph.D. dissertation focuses on user intent understanding for open-domain conversational agents.

2.2.1 Topic Classification: NLU

Topic classification, which determines the conversation topic, is the first stage in an NLU component to infer the user's intent. Topic classification in open-domain dialogue systems can be treated as a text classification problem, although utterance classification is a much more challenging task compared to general text classification due to four main factors: 1) Human utterances are often short; 2) Errors in Automatic Speech Recognition (ASR); 3) Users frequently mention out-of-vocabulary words and entities; 4) Lack of available labeled open-domain human-machine conversation data [3, 63]. Text classification models have traditionally used handcrafted features like bag-of-words, $tf * idf$, part-of-speech tagging, and tree kernels [136, 112]. However, current models put more focus on the semantic and implicit information in the text using features like the word to vector representation described in [98] and universal sentence encoder proposed in [23]. Wang et al. in [136] classified text classification algorithms into two main categories: implicit representation based models and explicit representation based models. Both types of representations have their advantages. Therefore, the method proposed in this dissertation uses both handcrafted and semantic features of the utterance.

Entity-based text representation: Entity linkers identify entity mentions in a text and resolve and map the mentions to a unique identifier in an associated knowledge base. A common choice is Wikipedia. Belfry [103] uses a graph-based approach for jointly performing word sense disambiguation and entity linking. DBpedia Spotlight [96] links entity mentions to their DBpedia [10] URIs. The SMAPH [35] system for linking web-search queries piggybacks on a web search engine to put the query into a larger context and further uses a supervised ranking model to predict the joint annotation of the full query. Entity-based text representation has been studied in different fields such as information retrieval [145], question answering [147], and co-

herence modeling [58, 59]. Yamada et al. [147] proposed a model to encode entity information in a corpus such as Wikipedia into a continuous vector space. This model jointly learns word and entity representations from Wikipedia and DBpedia. [133] proposed a CNN-based model for merging the text and entities extracted from a large taxonomy knowledge base for short-text classification.

Moreover, despite these efforts, I am not aware of a published result evaluating entity linking for conversational topic classification beyond the utterance content itself[63]. In this dissertation, I propose a neural network architecture and processing pipeline for conversational topic classification where the entities, their significance, and positional order are taken into account.

Deep learning approaches: Both CNN [66, 34] and RNN [87] models show promising results for text classification. Lecun et al. proposed VDCNN [34] for text classification based on a popular model in computer vision, which they redesigned for text classification. FastText [16] is another character-embedding-based text classification model that Facebook released for efficient learning of word representations and text classification. Character-embedding-based models have shown higher robustness in representing misspellings and out-of-domain words. My approach in this dissertation builds on the success of deep learning models for classification, and I explore both CNN and RNN models in my implementation. Moreover, I employ character-based modeling to make further the classification robust to ASR errors and out-of-domain words, which are frequent in conversations.

Conversational topic classification: Contextual evidence extracted from the preceding utterances could provide essential clues to predict the user intent in each stage of conversations. Incorporating contextual features has long been studied in human-human conversations [90][17][61]. In contrast, open-domain human-machine conversations is a relatively new field of research [114][2] in which topic classification

plays an important role in having a coherent and engaging conversation [48][129]. Khatri et al. propose one of the first contextual topic classification methods for open-domain conversational bots [63]. This method is the contextual version of ADAN [48], in which they demonstrated that using both dialogue acts and context of the previous utterances improves topic classification accuracy in open-domain dialogue agents. In this dissertation, we developed multiple models to incorporate the knowledge from the previous utterances into user intent inference.

2.2.2 Dialogue Act Classification: NLU

For a conversational system to respond appropriately, it must first understand the intent of the user utterance. Dialogue Act (DA) identification is a traditional approach to intent classification in dialogue system research, which aims to predict the goal of each utterance, such as information request, statement of opinion, greeting, opinion request, and others. [109] categorized DAs as having two main categories: 1) primary intents; 2) secondary intents. Each of these intents can further be divided into implicit or explicit. Since correctly identifying these intents is crucial for dialogue systems, this problem has been studied extensively for decades for human-human conversations. Recently, this idea was also extended to human-machine conversations [120, 89, 74].

Utterances in natural conversations are contextually dependent, which makes the DA prediction challenging. For example, the utterance like "Oh, yeah" can be interpreted as "Yes-Answer," "Accept-Agree," or "Backchannel," which requires the previous context to disambiguate [89, 18]. Therefore, I propose a context-aware model for this task.

For human-machine conversations, DA classification is more challenging due to three additional factors: 1) Often, human utterances are short (only 2.8 words on average in this data); 2) Automatic Speech Recognition (ASR) is still not quite a human-level performance; 3) Lack of available open-domain labeled human-machine

conversation data. To address these challenges, I propose a deep learning-based model, which utilizes contextual evidence, such as preceding utterances alongside the system state information, for more accurate predictions. To reduce requirements for labeled training data, I follow the approach of pre-training the model followed by fine-tuning, which has proven its effectiveness on various natural language processing tasks such as question answering [105, 33].

To represent the utterance and system state for each conversation turn, I built on previous studies which identified effective features for human-human DA classification, including syntax, prosody, and lexical cues (e.g., [120, 47]. I integrate many of these ideas into the proposed lexical and syntactic features in my CDAC system and augment these with representation learning.

Recently, deep learning and representation learning approaches have shown promising results on many tasks, including text classification and DA classification (e.g., [13]). For instance, reference [13] proposed a context-based RNN model for dialogue act classification for the human-human Switchboard dataset, while reference [89] proposed a hierarchical CNN and RNN model for this task.

Reference [63] demonstrated the benefits of accurate DA classification for topic classification in open-domain dialogue systems. Inspired by the promising results of [63], [134], and [28], I propose a novel, yet relatively simple Contextual Dialogue Act Classifier (CDAC) model, which incorporates lexical, syntactic, semantic, and contextual evidence into DA classification. To my knowledge, CDAC is the first to extend and adapt the ideas [63] for contextual DA classification in open-domain conversational systems, such as those fielded in the Amazon Alexa 2018 Challenge. Interestingly, previous state-of-the-art DA models (e.g., References [89], and [74]), rely on complete conversations for context, including future utterances. At the same time, CDAC uses only the utterances from earlier in the conversation, which makes CDAC feasible for online (real-time) conversational DA classification. As far as I

know, CDAC is also the first successful attempt to fine-tune a DA classification model trained on a dataset of *human-human conversations*, to predict DAs in open domain *human-machine* conversations.

In summary, my contributions in this dissertation for intent classification are twofold: (1) development of a novel context-aware Dialogue Classification model, CDAC, for open-domain human-machine conversations; (2) demonstrating promising results after fine-tuning CDAC trained on human-human conversations to human-machine conversations, which is a necessary step for intelligent open-domain conversational agents.

2.2.3 Smart Topic Suggestion for Open-Domain Conversational Agents

I briefly review the published research on open-domain conversational agents, key techniques used for general recommender systems, utterance suggestion approaches used in conversational agents, and recent research on the single-domain conversational recommendation.

General chatbots and conversational agents: Recent years witnessed significant research activity in developing a coherent and engaging conversational agent [142, 125, 143, 150]. Conversational agents have been generally categorized [121] into two main categories, namely task-oriented and general chat. Chatbots are traditionally aimed primarily at *small talk*, while task-oriented models are designed to carry out information-oriented and transactional tasks [8, 113]. Recently, several shared tasks and challenges have been proposed to push the boundaries of conversational AI to develop more intelligent chatbots to carry on in-depth conversations about several topics, not just *small talk*. This research has been evaluated by both crowd workers [41] and live users as part of the Alexa Prize Conversational AI challenge [114, 64].

General Recommender Systems Research: Recommender systems have been studied for decades and are now pervasive [115]. Traditional recommendation algorithms have been classified as primarily *model-based* or *content-based*, where a classifier model is trained for each user's profile, and *collaborative filtering*, where a user's unknown preferences are estimated based on the neighborhood of similar users [53]. More effective methods have been shown to be a hybrid of the two approaches [95], with increasingly sophisticated methods reported for movie [70, 71] and news recommendation [39]. However, there are significant challenges that still remain active topics for research in all cases. The most closely related issue is the *cold start* problem, i.e., recommending an item for a new user with no existing profile or recommending new (or dynamic or changing) items with no prior likes from any users. Model-based or content-based recommendations have been shown to perform better in such scenarios[132, 92]. This is the primary approach I attempt to adopt here for the conversational topic recommendation. Other approaches have explored online experimentation (e.g., [82]), and using social media or other metadata (e.g., [26, 111] for recommendation. Unfortunately, these signals are not easily available in the conversational setting. All the attributes of users and their preferences need to be inferred from their interaction with the conversational agent.

Utterance Suggestion in conversational agents: Yan et al. [148] describe an end-to-end generative model, which gives a user query, generates a response, and a proactive suggestion to continue the conversation. However, generative models like this still strictly rely on training corpora or restricted information without the ability to query external data sources, thus limiting their capacity for an informative conversation. In the other work, Yan et al. [149] describe a next-utterance suggestion approach for retrieving utterances from a conversational dataset to use as suggestions, along with the response. The proposed model learns to give suggestions related to the

response, to continue the conversation on the same topic. In practice, due to the vast number of possible utterances coming into a social bot, many conversational systems rely on multiple response modules where each response module would be responsible for a particular domain or set of domains [65]. Fine-grained utterance suggestions would be applicable to the implementation of each domain-specific module. However, when the user is passive or gets fatigued with a particular topic, the system needs to switch to a different component with domain-specific capabilities to keep the user engaged. In this dissertation, I attempt to formalize the problem of suggesting the best next interesting topic.

Conversational Recommendation: Recently, the idea of conversational recommendation was introduced [32], primarily as a way to elicit the user's interest in item recommendation. For example, Sun et al. [122] introduced an end-to-end reinforcement learning framework for a personalized conversational sales bot, and in [83], a combination of deep learning-based models is used for conversational movie recommendation. Currently, most existing conversational agents are designed for a single domain, such as *Movies* or *Music*. An open-domain conversational agent that coherently and engagingly converses with humans on a variety of *topics* remains an aspirational goal for dialogue systems [129, 114]. To address this problem, I propose a conversational *topic* suggestion method for open-domain conversational agents in which the conversational agent proactively suggests the following best topic to discuss based on the conversation so far, as I describe next.

This dissertation introduced a smart topic suggestion with three main contributions: 1) formalizing the conversational topic suggestion problem for open-domain conversational agents; 2) development of a sequential contextual topic suggestion model for this task; 3) empirical exploration of the effectiveness of model-based, collaborative filtering-based, and hybrid approaches to topic suggestions in the con-

versational setting. The experiments indicate the value of the proposed hybrid recommendation solution, highlight the challenges and opportunities inherent in the conversational topic recommendation and suggest promising directions for future work.

2.3 User Intent Inference for Web Search

User intent inference for general Web search engines has been studied for decades[21, 56]. Five different research tracks are: 1) defining a predefined set of categories, then map queries into this categories (e.g., informational, navigational, and transactional) [21] or (e.g., commercial or noncommercial, navigational or informational) [9], 2) Using graph click and utilizing semi-supervised learning for prediction [84], 3) considering temporal query intent modeling [51], 4) understanding word-level user intent [116], and 5) leveraging relevance feedback alongside user behavior for a better understanding of the user intents [117]. Despite recent advances in understanding user intents while interacting with a search engine, it is still not in human-level performance. In an e-commerce setting [118], this problem is even more challenging since enough research has not been done. To keep filling this gap, I propose several models for user intent inference for e-commerce search engines in this dissertation.

An effective query representation is crucial for an accurate user intent inference. Traditional models are generally based on bag-of-words techniques, $tf * idf$, and so forth. The enhanced version of these models with a combination of machine learning models has been the backbone of many advanced search engines and retrieval systems [154]. For example, utilizing query term weighting methods [94], extracting n-gram term dependencies [97], and particularly expanding query with relevant terms [37, 77]. Unfortunately, term-based query representations have difficulties modeling queries with similar textual information and different intents [156]. In this dissertation, I developed multiple models for query understanding to link customers' search queries

to a set of product categories in the product taxonomy. In this dissertation, I consider all types of traditional features like lexical information as well as dense representation of the query and relevant documents.

User queries are often short and suffer from a lack of textual information. Utilizing external information and expanding search queries are two groups of methods to alleviate this problem [77, 40]. Relevance feedback [77] is a part of external methods that incorporates user's knowledge and other resources such as Wikipedia into the user intent inference process [55]. To expand query with relevant terms, recently, two groups of methods based on the word representation are developed, which attract much attention, such as local embedding [40] and global embedding[75]. This dissertation will leverage pseudo-relevance feedback and incorporate external knowledge from product documents into the user intent inference process.

Deep neural networks become favored in various search applications by providing softer matches [99], richer compositionality [110], and the more excellent capability of transferring knowledge [88]. Deep networks utilize the distributed representation of rich semantics of term proximity in a large corpus. For information retrieval tasks, it has been shown that enriching word embedding is effective [54]. Zamani and Croft [153] proposed the relevance likelihood maximization model (RLM) and relevance posterior estimation model (RPE) to compute the relevance between the document's words to a particular query. Ha et al. [49] proposed the DeepCN method to incorporate metadata information to predict the appropriate category to place a new productID. However, word-level query representations cannot always capture users' intents in search queries because typos and misspellings frequently create out-of-vocabulary words [67, 154]. To this end, there have been different studies conducted to represent characters and subwords using deep neural networks [34, 128]. Utilizing both representations has been useful for both the search engine and the conversational agents. Zhang et al. [154] proposed GEneric iNtent Encoder (GEN) for user intent predic-

tion in Microsoft Bing, and Kim et al. [68] proposed the ONESET model to learn user topic and intent for Amazon Alexa jointly. According to the promising results reported in recent years that show the effectiveness of the deep learning models for user intent inference, this dissertation mainly focuses on deep neural networks while utilizing the strength of popular features like lexical information.

Finally, with a drastic increase in online shopping and e-commerce sales growth in recent years, there is a high demand for designing practical user intent and understanding models to optimize search results [127, 20]. Moreover, there is a crucial difference between web search and e-commerce search engines, that is, the immediate impact of the query understanding step on different business metrics such as click-through rate, search conversion, and gross demand [38, 156]. Due to the growing demand to improve the current query understanding models for the e-commerce domain, the second part of my dissertation focuses on user intent inference for e-commerce search engines. To keep filling this gap, in this dissertation, I implemented several deep neural network models for different tasks such as product category mapping and intent classification.

In the following sections, I discuss the related works for three main approaches I proposed in this dissertation to address user intent inference for e-commerce search.

2.3.1 Joint Learning for User Intent Inference

Query intent understanding is a key step in designing advanced retrieval systems like e-commerce search engines [36]. Various approaches have been proposed to address query understanding, such as 1) considering predefined high-level categories (i.e., informational, navigational, and transactional), 2) deploying semi-supervised learning with click graphs, 3) considering temporal query intent modeling, 4) understanding word-level user intent and 5) applying relevance feedback and user behaviors. Although there has been a significant improvement in user intent inference, query

understanding remains a significant challenge [154].

E-commerce search queries have multiple intents associated with them. Ashkan et al. [9] categorized search queries for e-commerce websites into commercial and non-commercial intents. However, Zhao et al. [156] ignore the non-commercial queries due to a small percentage of the search traffic. Commercial queries are queries with purchasing intent, while non-commercial queries cover a wide range of customer services (e.g., "military discounts" and "installation guides"), as shown in Table. 5.1.

Query understanding in e-commerce search is challenging: 1) queries are often short, vague, and suffer from the lack of textual evidence [49], 2) small variation in textual evidence causes a drastic change in query intent; for example, "30 in. 5.8 cu. ft. gas range installation kit" has commercial intent but "30 in. 5.8 cu. ft. gas range installation", has non-commercial intent, 3) product category mapping is a multi-label and non-exclusive problem. A practical solution must include a broader possible set of correct categories while simultaneously keeping precision as high as possible [156], 4) there is a class imbalance in both commercial vs. non-commercial and product category mapping tasks because only a small fraction of data (1.5% in this domain) has a non-commercial intent, and within the commercial queries, some product categories contain significantly more samples compared to others, and 5) commercial queries are easy to identify using user behavior information like click rates; however, that is not the case for non-commercial queries.

To address these problems, I introduce a new method of jointly learning query intent and category mapping, which allows transferring the inductive bias between these two relevant tasks. Also, I leverage label representation, which provides a richer representation to model the product categories. Finally, I propose an active learning algorithm to generate data for commercial vs. non-commercial intent. To address the class imbalance problem, I deploy focal loss borrowed from computer vision.

Joint learning has been proposed as a practical approach to simultaneously learn

relevant tasks due to the transfer of the inductive bias among them. Joint-learning finds applications in computer vision and natural language understanding [63]. Joint-learning improves the regularization and generalization of the learning models by utilizing the domain information [22]. In addition, with a joint model that addresses multiple tasks, only one model needs to be deployed; this contributes to reducing overhead and facilitates the maintenance of the system [135]. In this dissertation, I propose a joint-learning model that simultaneously learns both commercial and non-commercial query intents and maps the incoming commercial query to a set of relevant product categories.

This dissertation introduces a data-driven approach called Joint Query Intent Mapping (JointMap). JointMap leverages the label representation proposed by Guoyin et al. [130] and modifies it to be applicable for a joint-learning task. JointMap also utilizes a self-attention mechanism to improve the quality of the joint word-label attention vectors. For product category mapping, JointMap handles the imbalanced class problem using focal loss [85] which has been well-studied in the computer vision field to control the sparse set of candidate object locations. Finally, I propose an approach based on distant supervision in the combination of active learning to generate both commercial and non-commercial queries.

In summary, my contributions in this dissertation are: 1) proposing a deep learning model to jointly learn product category mapping as well as users' non-commercial intents, 2) developing an active learning algorithm in conjunction with distant supervision to generate a user intent dataset from e-commerce data logs, and 3) modifying the joint word-category representation for query intent mapping tasks in e-commerce, as described in detail next.

2.3.2 Label Representation for User Intent Inference

Query understanding is an essential step in developing advanced retrieval systems (e.g., e-commerce search engines) [36]. In an e-commerce setting, one aspect of query understanding is achieved by mapping a query to a set of relevant product categories [156]. For example, for the query "motion activated kitchen faucet," an e-commerce search engine should return products from relevant categories like *bath, plumbing, kitchen*. These categories match the customer's intent and provide signals for downstream tasks such as retrieval and ranking. In this dissertation, I develop a new model for query understanding in an e-commerce search engine where a search query like "motion activated kitchen faucet" is mapped to a set of relevant product categories in a hierarchical product taxonomy.

Query understanding is a challenging task since 1) queries are often short, vague, and suffer from the lack of textual evidence [49], 2) queries with similar textual information with slight variations such as "9 cu. ft. chest freezer in white" and "9 cu. ft. upright white freezer" belong to different categories. However, queries with no term overlap like "french door 32 inch. refrigerator" and "black fridge with glass panes" are semantically similar and may belong to related categories, 3) The severe data imbalance problem resulted from customer *bias* towards some specific products in general or at a particular time. Also, the product categories' correlations directly impact customer click behavior, where some of them received more click rates than usual, and others get fewer click rates, and 4) queries with low customer behavior feedback (e.g., *tail* and *torso*) are more challenging to classify as they have a high signal-to-noise ratio. Current neural models achieve a *softer representation* with *richer compositionality* of the queries compared to conventional term-based models [154].

There have been numerous studies in neural models for text representation in different levels, such as characters, subwords, words [34, 86, 128]. These models utilize

distributed representation by transferring knowledge from other resources to enrich the query representation [15, 66]. However, they still have difficulty addressing challenges (3) and (4) for query understanding. To alleviate these problems, inspired by work in information networks [124], I propose a joint word-category (label) representation to provide both word and category embeddings. Then, category representations are leveraged to boost the model's efficiency on both *tail* queries and the *minority* classes.

Tang et al. [123] introduce the idea of heterogeneous text network embedding to model the word and label interactions. Guoyin et al. [130] expand the concept to extract the relative spatial information among consecutive terms with their associated labels. Although these models leveraged the joint word-label interactions, they still lose the knowledge in label-label correlations. Extracting category (label) co-occurrence information is essential for query understanding, where product categories inherent this correlation during taxonomy formation. Product categories are not mutually exclusive and are semantically related to each other. This correlation between product categories impacts the customer click behavior, which utilizes as supervision signals in dataset generation. Thus, category co-occurrences can be used to improve the quality of *minority* classes and *tail* queries, where there is less customer feedback available. To this end, I consider the product categories as an undirected *homogeneous* graph, where the edges represent category correlations.

In this dissertation, I introduce a data-driven approach named DeepCAT for query understanding. My model consists of a pipeline of deep learning models that utilize both word-category and category-category interactions. In summary, my contributions are: (1) proposing a novel deep learning model for joint word-category representation and (2) introducing a new loss function to incorporate pairwise category information into the query understanding process.

2.3.3 Pseudo-relevance feedback for User Intent Inference

In the realm of Web and e-commerce search, query categorization is an important step in query intent understanding, which in e-commerce can be viewed as mapping search queries to relevant fine-grained product categories [156]. It has been shown that query categorization could play a pivotal role in increasing user satisfaction by returning more relevant products in e-commerce search [154]. Query categorization is challenging due to multiple reasons: (i) queries are generally short and might suffer from lack of concrete evidence of customers' intents [102], (ii) customers have different ways to express the same intent. For example, "blocktile white-colored - 12 in. x 12 in." and "block tile white 12x12 inches" (iii) query categorization is an extreme multi-label text classification problem (XMTC), which raises challenges such as data sparsity and scalability [86, 131, 15], and (iv) the most challenging case is handling rare queries (a.k.a. tail queries). Tail queries suffer from the lack of customer behavior signals (e.g., click-through data), leading to poor representation of rare queries. Typos, synonyms, morphological variants generally cause rare queries, and, more importantly, when customers express their intent in a unique fashion [152]. Different methodologies have been proposed to deal with tail queries in retrieval systems, such as corpus-based and knowledge-based query expansion methods [77] and [102]. Although standard query expansion models are effective, they might not be sufficient since even perfect expansion terms for a tail query might produce another tail query [119]. Also, adding new terms in query expansion may result in a higher recall but sometimes a lower precision, and (ii) the suggested terms are not ordered, which could be problematic for text embedding models trained on natural language text. Unlike standard query expansion techniques, my model is trained to attend across document fields based on their semantic similarities with a query, which, as will show, improves the performance on the query categorization task.

In this dissertation, inspired by [80], I propose a deep learning-based PRF model,

named Attentive Pseudo-Relevance Feedback Network (APRF-Net). Unlike [80] that was developed to boost ranking performance, APRF-Net is an extreme multi-label classifier that is specifically developed for e-commerce query categorization. APRF-Net utilizes PRF to enrich the tail queries' representations with the information in the retrieved product documents to mitigate tail queries' sparsity. The product documents in e-commerce websites are well-structured and contain fields like (e.g., title, short description, color). In my setting, fields form product documents, and top-ranked documents create a corpus. APRF-Net jointly generates query and product field-level embeddings to capture this existing hierarchical structure and utilizes hierarchical attention to prioritize the top-k retrieved product documents at three levels of abstractions (field, document, and corpus). In summary, my main contribution is proposing a new model to address the customer signal gap between frequent and rare queries by leveraging the informative signals from PRF for query categorization in e-commerce.

2.4 Summary

In summary, my Ph.D. dissertation focuses on addressing the questions relevant to natural language and query understanding in both conversational agents and search engines, respectively. My dissertation aims to develop advanced models to aid in inferring what the user is looking for while interacting with these intelligent information systems. Based on my research, I proposed several models to solve these problems, and I evaluated them on real data collected from Amazon Alexa Prize and The Home Depot search engine.

In the next section, I present my research on user intent inference in conversational agents. I explain my contributions in this field and place them in context.

Chapter 3

Integrating knowledge in User Intent Inference

This chapter describes different resources that this dissertation utilizes to incorporate into the intent inference process for conversational agents and e-commerce search. In section 3.1, I present the external resources that I used for conversational agent setting, including entity knowledge-base, conversation context, and user profile information. Section 3.2 describes resources, including inductive bias from relevant tasks, product taxonomy structure, and unlabeled domain-specific corpora for e-commerce search. The chapter's content related to conversational agents is based on references [3, 2, 5], published in CIKM'2019, SIGIR'2019, and CHIIR'2020. Also, the content related to e-commerce search is based on references [4, 6, 7], published in SIGIR'2020, SIGIR e-com 2021, and SIGIR'2021.

3.1 Integrating Knowledge for Conversational Agents

My dissertation claims that integrating external knowledge into the user intent inference for conversational agents is effective. This section explicitly describes three different resources this dissertation studies for conversational agent settings.

Entity Knowledge-base: An essential challenge for conversational domain classification is that human users tend to use many entities in their conversations. Moreover, the creation of new entities, like recent movies, makes the model obsolete with time. It would be necessary to constantly update the model by incorporating new information about people, organizations, movies, and other entities to fix this problem. Updating the training set might cause unintended effects in the model and be inefficient. Also, constantly retraining the NLU is not a good answer for such scenarios as many downstream components are tuned on different parameters of the previously trained model.

An answer for this issue is using an entity knowledge-base where instead of updating the training set and frequently retraining the topic classifier on the new training set, we can only update the entity knowledge-base. As a result, an entity inference step is needed to assist the topic classifier for an accurate prediction. According to this idea, the model will depend on the entity types rather than the actual entity text. Two different entity-knowledge-based are used in this section, such as DBpedia Spotlight and PMI-based Entity Linker. DBpedia Spotlight annotates DBpedia resources mentioned in the text as described in reference [96]. It annotates DBpedia resources of any 272 classes (more than 30 top-level ones) in the DBpedia Ontology. Moreover, PMI-based Entity Linker that annotates a customized entity knowledge-based for IrisBot. PMI-based Entity Linker created a domain-specific entity linker called PMI-EL for IrisBot, annotating the 20 entity-types most relevant to the IrisBot design.

Conversation Context: I utilized conversation context information for both topic classification and dialogue act classifications. The proposed model for contextual topic classification, which I deployed in our Amazon Alexa Prize, consists of three phases: Independent Topic Classification, Topic Merging, and Contextual Topic Merging.

Individual Topic Classification: The individual topic classifier is the classifier that only uses the current utterance for topic prediction. The result of this step is a nonexclusive classification vector. During Amazon Alexa Prizes, we developed a mixture of the expert model, which was a combination of three different classifiers trained on different datasets and with different parameters and architecture (Section 4.2.4). This mixture of the expert model includes a customized classifier, an entity-aware classifier, and an Amazon annotator.

Contextual Topic Merging: In this step, the contextual information is included for final classification. For this purpose, the topic distribution for the previous utterances is used as input to inform the final topic classification results. To apply contextual merging, a transition matrix between topics is computed based on the conversation logs. This transition matrix represents the transition probability between topic i and j through the conversation log. As a result, in turn, i, this value indicates the transition probability from the previous topic $i-1$ to the current topic.

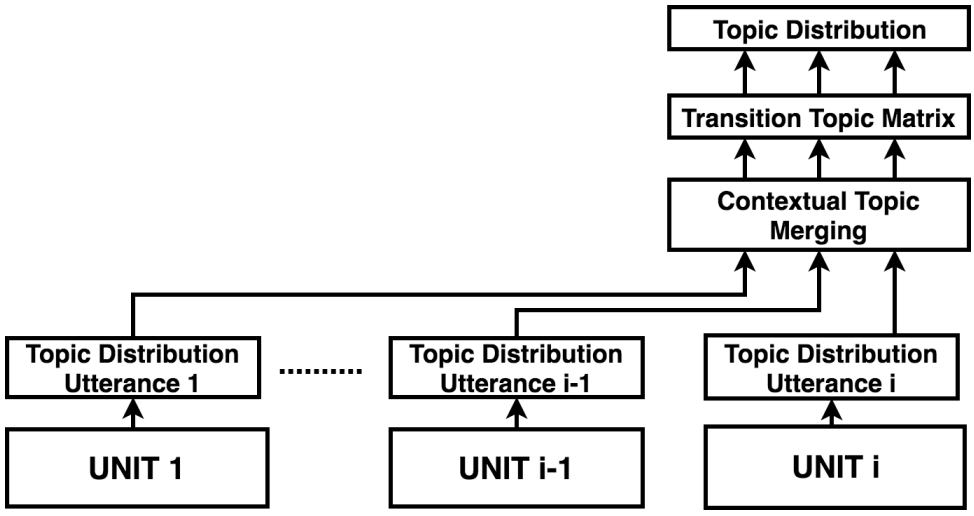

Figure 3.1: Contextual Topic Merging [1].

For dialogue act classification, we leverage the different types of conversation context. This dissertation proposes a deep learning model that utilizes contextual

evidence, such as preceding utterances alongside the system state information. To represent the utterance and system state for each conversation turn, I built on previous studies which identified effective features for human-human DA classification, including syntax, prosody, and lexical cues (e.g., [120, 47]. I integrate many of these ideas into the proposed lexical and syntactic features in the context-aware deep learning model and augment these with representation learning. An exciting idea for end-to-end context learning is to let deep models learn the transitions between two consecutive turns instead of using the transition matrix idea that I develop for contextual topic classification.

User Profile Information: swhile keeping the actual features constant.

I explore a list of users' profile information to improve the user intent inference process, which can be categorized into three types: 1) user information; 2) general behavioral user features; 3) system features; 4) topic preference features. These features are concatenated to produce one feature vector per turn. The first group of user information is user information. It contains the inferred gender of the user [-1,1] based on the provided name and whether they gave their name at the start of the conversation or not (a weak indicator of the user's openness to sharing information with the bot). Other features like age and location, often used for user profiling, are usually not available in the conversational setting. General behavioral features encode user behaviors in various dimensions, including lexical, semantics, and conversational. First, we define engagements as subsets of conversation that have 4+ conversational depth on the same topic. Count of engagements and max length of engagements are derived, respectively. Sentiment analysis using Valence Aware Dictionary for sEntiment Reasoning (VADER) [46] on utterances is applied to obtain positive and negative sentiment scores. To capture how much topic transition occurs, the state change ratio is derived by dividing total transitions by the current turn

index. Similarly, agreement and disagreement ratios are derived based on intent classification results. To measure the repetition between (U_i, R_i), (R_{i-1}, R_i) and (U_{i-1}, U_i), counts of token overlaps are computed. Lastly, the average and total word count of user utterances and system responses are extracted.

Local Features	**Short Description**
NumEngagements	#Engagements
MaxEngagements	Max engagement in # of turns
UtterancePos	Positive sentiment in U_i
UtteranceNeg	Negative sentiment in U_i
AvgPos	Sum of pos sentiment counts / i
AvgNeg	Sum of neg sentiment counts / i
StateChangeRatio	#Topic Transitions / i
YesRatio	#Yes Responses/Agreements / i
NoRatio	#No Responses/Disagreements / i
$TokenOverlap_U$	Token overlap in U_i, U_{i-1}
$TokenOverlap_R$	Token overlap in R_i, R_{i-1}
$TokenOverlap_{UR}$	Token overlap in U_i, R_i
$TotalWord_U$	Total #Words in U_i
$TotalWord_R$	Total #Words in R_i
$AvgWord_U$	Average #Words in U_1 ... U_i
$AvgWord_R$	Average #Words in R_1 ... R_i
$Word_U$	#Words only in U_i
$Word_R$	#Words only in R_i

System features are directly related to systematic aspects of our conversational agent. Two binary session-level features capture if a user agreed to provide his name or if he is a returning user. We define two types of latency: system latency and user latency, both measured in seconds. System latency measures how long a user had to wait to hear the system response; user latency measures how long a user had to think before issuing an utterance. Lastly, every token in our utterances was annotated with an ASR confidence value ranging from 0.0 to 1.0. Using these values, minimum, maximum, and average token confidence on each U_i are added.

Session-level Features	**Short Description**
NameProvided	Name provided or not
ReturningUser	Returning user or not
Local Features	**Short Description**
Latency	System latency on U_i
$Latency_{avg}$	Average system latency
$Latency_{max}$	Max system latency
UserLatency	User latency on R_i
$UserLatency_{avg}$	Average user latency
$UserLatency_{max}$	Max user latency
ASR_{min}	Min token confidence on U_i
ASR_{max}	Max token confidence on U_i
ASR_{avg}	Average token confidence on U_i

The next section will discuss the external resources used to improve the user intent inference for e-commerce search.

3.2 Integrating Knowledge for E-commerce Search

This dissertation also investigates integrating external knowledge into the user intent inference for e-commerce search. In this section, I discuss three different resources that are studied.

Inductive Bias from Relevant Tasks: Joint learning is a practical approach to learning relevant tasks due to the transfer of the inductive bias among them. Joint learning has shown its efficiency in different fields, such as computer vision. It mainly attracts much attention in designing the natural language understanding unit of spoken dialogue systems. There have been recent efforts to learn different utterance intents simultaneously. Unfortunately, it has not been explored enough for e-commerce settings. Moreover, user intent understanding is challenging, as there are complex relationships among different user intents. Implementing independent and individual classifiers loses the transferable inductive bias. We thus, in this dissertation, propose

a joint learning model to improve the quality of intent understanding of two relevant user intent tasks on e-commerce search data.

Furthermore, in the e-commerce domain, accurate query classification will help identify the correct product groupings from which relevant products can be retrieved. For product search, it might be necessary to identify intents associated with a query across various granularities (e.g., category intent, accessory intent, vertical intent). As these intent classification tasks are related, knowledge from one task might help improve the performance of other tasks. Joint learning has been shown to improve the performance of related tasks due to the transfer of the inductive bias across tasks. In this dissertation, we then leverage a joint learning model of high-level intent prediction and query categorization to improve the performance of these two individual intent classification tasks.

Product Taxonomy: Many query understanding models solely focus on query textual and session representation. Although these models extract valuable information for user intent inference, they are still missing some incorporated knowledge in the taxonomy or product catalog. For the e-commerce setting, extracting category (label) co-occurrence information is essential for query understanding, where product categories inherent this correlation during taxonomy formation. Product categories are not mutually exclusive and are semantically related to each other. This correlation between product categories impacts the customer click behavior, which utilizes supervision signals in dataset generation.

Node representation is helpful to model the interaction between the related nodes in knowledge graphs. Researchers such as Jian et al. [124] proposed a local pairwise proximity embedding to model large-scale information networks. In this network, a node represents an entity, and the edges represent the relations among the entities. In this dissertation, the product categories are considered as nodes of a knowledge

graph, and the connection between them is computed based on the user click rate. As a result, an undirected graph can be formed to represent the co-occurrence of each product category pair. As a result, a local pairwise proximity model between the vertices (labels) can be created in the form of a Co-occurrence Matrix (CM).

Unlabeled Domain-Specific Corpora (Product Catalog): The last source of knowledge this dissertation utilizes for user intent inference in e-commerce search is unlabeled domain-specific corpora or product catalog. Unlike most web documents, the product documents in e-commerce websites are well-structured and contain fields like title, short description, color, and so forth. In our setting, fields form product documents, and top-ranked documents create a corpus. APRF-Net jointly generates a query and product field-level embeddings to capture this existing hierarchical structure and utilizes hierarchical attention to prioritize the top-k retrieved product documents at three levels of abstractions (field, document, and corpus). The table below summarizes the fields included in a product document.

Fields	**Short Description**
Title	Product title
Description	Product descriptions
Taxonomy path	Product taxonomy path from root to leaf
Color/material	Color and material information
Numerical	Numeric values (e.g., height, width)
Brand	Brand information (brand name)

3.3 Summary

In this chapter, I discussed different sources of knowledge used in this dissertation to improve the user intent inference process for conversational agents and e-commerce search. I presented different resources, including entity knowledge-base, contextual

information, and user profile information. Moreover, I described other resources for e-commerce search, including the inductive bias in the relevant tasks, product taxonomy structure, and unlabeled domain-specific corpora.

The next chapter presents three knowledge-aware models that incorporate entity knowledge base, conversation context, and user profile information into user intent inference in the conversational agents.

Chapter 4

User Intent inference in Conversational Agent

This chapter describes my previous research on Conversational AI by focusing on user intent inference. I studied three external sources of information, including entity knowledge base, conversation context, and user profile information, to improve user intent inference. I adopt four papers published in [1, 3, 2, 5]. I start the chapter by explaining the details of IrisBot in section 4.1, which is an open-domain conversational agent that our team at Emory developed for the Amazon Alexa Prize 2018. It was published in the 2nd Proceedings of Alexa Prize titled "Emory IrisBot: An open-domain conversational bot for personalized information access." The remaining sections cover my specific contributions in user intent inference for conversational agents working on the real data during these three global challenges. In section 4.3, I discuss my paper about the entity-aware topic classification, which was published in CIKM 2019 titled "ConCET: Entity-Aware Topic Classification for Open-Domain Conversational Agents." This paper examines how entity-type information should be utilized as an external source of information to improve the topic prediction. It follows by section 4.4, where I describe my work on contextual-based dialogue act

classification, which was published in SIGIR 2018 titled "Contextual Dialogue Act Classification for Open-Domain Conversational Agent." This paper investigates the effective way to represent and integrate conversation context information into user intent inference. In section 4.5, I explain my research on the smart topic suggestion to improve user topic switching intent, which was published in CHIIR 2020 titled "Would you Like to Talk about Sports Now? Towards Contextual Topic Suggestion for Open-Domain Conversational Agents." This paper investigates a method to leverage user profile information to improve user intent and engagement in a conversational agent.

4.1 Emory IrisBot: An Open-Domain Conversational Bot for Personalized Information Access

In this section, I describe IrisBot, a conversational agent that aims to help a customer be informed about the world around them, while being entertained and engaged. Our bot attempts to incorporate real-time search, informed advice, and latest news recommendation into a coherent conversation. IrisBot can already track information on the latest topics and opinions from News, Sports, and Entertainment and some specialized domains. The key technical innovations of IrisBot are novel algorithms for contextualized classification of the topic and intent of the user's utterances, modular ranking of potential responses, and personalized topic suggestions. Our preliminary experimental results based on overall customer experience ratings and A/B testing analysis, focus on understanding the contribution of both algorithmic and surface presentation features. Finally, promising directions for continued research are suggested, primarily focusing on increasing the coverage of topics for in-depth domain understanding, further personalizing the conversation experience, and making the conversation interesting and novel for repeating customers.

IrisBot is designed through loose coupling of domain-specific components that interact through the Dialogue Manager. Each utterance is processed to identify the key entities, topics, and intents (as described below), and is heavily annotated with NLP information helpful for the domain specific components to respond to the query.

it was chosen to do "lazy" response evaluation in that the final response ranking is performed after each component returns a candidate response, at which point the responses are ranked and selected based on the combination of the utterance interpretation and the estimated response coherence and relevance. Thus, each query is processed by every component, which has a computational drawback of significant processing for each utterance, and the benefit of redundancy and multiple available fallback responses. It was chosen to optimize the expected response quality from having redundant options available. Each domain component implements a common set of interfaces, and is expected to return a score of the response, the response type and topic, and follow-up suggestion (which could be the same component or a switch to another topic). As a result, adding new components turned out to be quite easy with the main challenge being to expand the topic classifier to identify when the new component is relevant.

4.1.1 Language Understanding and Entity Recognition pipeline

IrisBot's NLP pipeline is executed during the pre-processing stage in the Dialogue Manager. Many different functions, including both heavy and light NLP modules are executed in parallel to reduce latency. Once each thread is finished, its output is stored in the utterance context maintained by the Dialogue Manager, enabling easy backtracking for components to make smarter decisions.

Knowledge-Based Named Entity Recognition: To recognized key entities we created an index of 3.5 million entities and their types. That includes all the entities

that are present in our relational ontology, and all the entities that our domain-specific components can currently handle. Given an utterance, we use it to query our index and and retrieve all the entities present in it and their types.

Relational Ontology: IrisBot has its own knowledge-base that is modeled as a graph, and stores attributes of and relations between millions of entities in the index. It has been initially derived from DBPedia [79] and extended with the entities important to our components. Given an entity, the ontology server allows us to look up entities related to it, for example for a movie it might a directory and cast list, or for a person it might be a spouse or a place of birth, and many other relations as defined by DBPedia.

Co-reference resolution: Co-reference resolution in a dialogue setting becomes challenging because our system expects continuous influx of many different types of entities. It also remains uncertain whether we should prioritize entities appearing on our responses or on user's utterance. For IrisBot, co-reference resolution depends on knowledge-based named entity recognition, along with two main parameters: half-life and bias. Based on half-life, co-reference module continuously discounts the confidence score as time passes, prioritizing newly appeared entity to selection process. Bias is used to control weights between entities appeared on utterances and responses. We tuned these parameters over our development period, and they are incorporated with handcrafted tie-breaking features to assist selection process.

4.1.2 Domain-specific Components:

IrisBot currently has a total of 12 domain-specific components, listed below. This is a fluid list as components are added and occasionally retired (e.g., the WorldCup component was retired after the conclusion of the 2018 FIFA World Cup).

Movies and Shows	Music and Concerts
News	Travel and Activities
Animals	Video Games
Cars	Sports
Opinions	Wikipedia
World Cup	Emotional Support

Table 4.1: Domain-specific retrieval components

4.2 Topic and Intent Classification

We chose to follow the current state of the art and the Amazon practice, and to identify both *topic* and *intent*. For example, a customer may request an opinion or recommendation (intent) for a particular movie (topic). We developed our own state of the art classifiers for this task, as described next, as our component capabilities and boundaries did not easily map to available classes provided by existing Amazon Comprehend services and similar. However, we do use these services as input to our own classification.

4.2.1 Contextual Topic Classification

Our proposed model for contextual topic classification consists of three phases: Independent Topic Classification, Topic Merging, and Contextual Topic Merging.

Phase 1: Independent Topic Classification: For this step, the current utterance is only used. The result of this step is a nonexclusive classification vector. A Mixture of Experts Model is applied for topic classification, consisting of three different classifiers trained on different datasets and with different parameters and architecture. Topic classifier includes a customized classifier, an entity classifier, and an amazon annotator.

Phase 2: Topic Merging from Mixture of Experts: In this step, the outputs of three classifiers are merged to produce a single topic distribution based on evidence from all of the "expert" base classifiers.

Phase 3: Contextual Topic Resolution: The third step is Contextual Topic Merging, where the topic distribution for the previous utterances is used as input to inform the final topic classification results. To apply contextual merging, a transition matrix between topics is computed based on the conversation logs, which indicates the transition probability from the previous topic to the current one.

The logic and motivation for using the mixture of experts model for this task come from the "No Free Lunch Theorem" in machine learning, which suggests multiple classifiers for decision making instead of a single powerful model[107]. Moreover, according to the promising results for Convolutional Neural Networks (CNN) models in text classification, the CNN model is used as the base classifiers [144][69]. Finally, a mixture of entity-aware models and the Amazon Annotator (another expert) is deployed for the Topic Classification step. The three different models are 1. a customized classifier based on the CNN, an entity-aware classifier, and an Amazon Annotator, all of which are used in a mixture of expert models. The customized classifier is trained on a customized dataset, and the word embeddings are tuned on this data. The entity classifier was trained on the DBPedia and a customized entity knowledge-base that simultaneously train on both textual and entity-type information of the utterances. Consequently, the topic merging step merges all the results from these three classifiers to make the final prediction. Finally, a topic distribution across all possible covered topics are generated.

4.2.2 Language Understanding and Entity Recognition Pipeline

IrisBot's NLP pipeline is executed during the pre-processing stage in the Dialogue Manager. Many different functions, including both heavy and light NLP modules are executed in parallel to reduce latency. Once each thread is finished, its output is stored in the utterance context maintained by the Dialogue Manager, enabling easy backtracking for components to make smarter decisions.

Knowledge-Based Named Entity Recognition: To recognized key entities, an index of 3.5 million entities and their types are created. That includes all the entities that are present in our relational ontology, and all the entities that our domain-specific components can currently handle. Given an utterance, we use it to query our index and and retrieve all the entities present in it and their types.

Relational Ontology: IrisBot has its own knowledge-base that is modeled as a graph, and stores attributes of and relations between millions of entities in the index. It has been initially derived from DBPedia [79] and extended with the entities important to our components. Given an entity, the ontology server allows us to look up entities related to it, for example for a movie it might a directory and cast list, or for a person it might be a spouse or a place of birth, and many other relations as defined by DBPedia.

Co-reference resolution: Co-reference resolution in a dialogue setting becomes challenging because our system expects continuous influx of many different types of entities. It also remains uncertain whether we should prioritize entities appearing on our responses or on user's utterance. For IrisBot, co-reference resolution depends on knowledge-based named entity recognition, along with two main parameters: half-life and bias. Based on half-life, co-reference module continuously discounts the confidence score as time passes, prioritizing newly appeared entity to selection process.

Bias is used to control weights between entities appeared on utterances and responses. These parameters over our development period were tuned, and they are incorporated with handcrafted tie-breaking features to assist selection process.

4.2.3 Domain-specific Components

IrisBot currently has a total of 12 domain-specific components, listed below. This is a fluid list as components are added and occasionally retired (e.g., the WorldCup component was retired after the conclusion of the 2018 FIFA World Cup).

Movies and Shows	Music and Concerts
News	Travel and Activities
Animals	Video Games
Cars	Sports
Opinions	Wikipedia
World Cup	Emotional Support

Table 4.2: Domain-specific retrieval components

4.2.4 Topic and Intent Classification

Irisbot chose to follow the current state of the art and the Amazon practice, and to identify both *topic* and *intent*. For example, a customer may request an opinion or recommendation (intent) for a particular movie (topic). Developed classifiers for this task were developed, as described next, as our component capabilities and boundaries did not easily map to available classes provided by existing Amazon Comprehend services and similar. However, we do use these services as input to our own classification.

Semantic and Lexical Classification using Convolutional Neural Networks (CNN)

CNN models are used as basic classifier for both Customized and Entity classifiers, because of promising results in text classifications in the recent years. The basic classifier is a CNN model consisting of 4 layers, an embedding layer of size 300 followed by 3 Convolution layers, any of which consists of a convolution and a max pooling layer. A Fully Connected Neural Networks (FCNN) was used for the final classification. In each Convolution layer, 256 filters of size 2, 3 and 4 were used, respectively. We used l2-regularization of 0.01 and a dropout of 0.5 to avoid over fitting during training. Lexical and unsupervised features were extracted from text utterances and concatenated with CNN features to utilize the CNN features. The final vector was used as input of the FCNN model for classification.

Customized Classifier: We generated more that 50,000 utterances which are customized to our bot, and the components that bot supported. Google Word2Vec is used for initialization of each word embedding. Then, each word has been trained during training to be more customized to our bot. To use maximum possible number of vocabularies for conversation data 4 different popular dialogue or conversation datasets such as Cornell, ubuntu and scenarios and reddit, in addition to our dataset. Finally, more than 25,000 frequent words were extracted from these datasets and were been used as lexical features. To leverage training, we also extracted different unsupervised features like LDA and cosine similarity between LDA topics, to extract high level clustering features from the utterances.

Entity classifier: The architecture for this classifier is the same as the previous classifier, but this classifier uses one more Convolution layer; and 256 filters in each Convolution layer. Finally, a FCNN was applied for final classification. External

features such as cosine-similarity and LDA and LSI features had been used as unsupervised features [137][25]. DBpedia entities were used to generate the dataset. Some general rules were applied to assign every entity to a special topic. E.g. University Names can be categorized in both News and Tech topics. Each entity is considered as one word and average Google Word2Vector is used as word embeddings. This dataset contains millions of samples, as a result the generated dataset is much larger than the customized dataset. We used 3M Google word2vector as vocabularies to train this model. The word vectors are not retrained during training to make model more generalized.

Contextual topic post-processing: Contextual information plays an important role in conversational AI. To leverage our model using contextual information in a real conversation, contextual Topic Merging phase is proposed. This phase contains two sub steps. First, specific contextual topic merging and second global contextual topic merging. In the specific contextual topic merging, different heuristics based on our domain knowledge about a specific topic are used to refine the probabilities for a special topic. For example, specific regular expressions were used to capture some corner cases, which the proposed Topic Classifier could not classify them correctly. In the global step, a transition matrix was used which had been computed based on the conversation logs. This matrix represents the transition probabilities between different topics.

4.2.5 Intent Classification

Building on the Amazon intent ontology, we considered 11 different intent classes, such as Opinion-request, opinion-delivery, info-request, user-instruction, personal-delivery, topic-switch, topic-acceptance, DontKnow, repetition, self-harm, clarification and user-correction (to support ASR) as our intent classes. The classes of

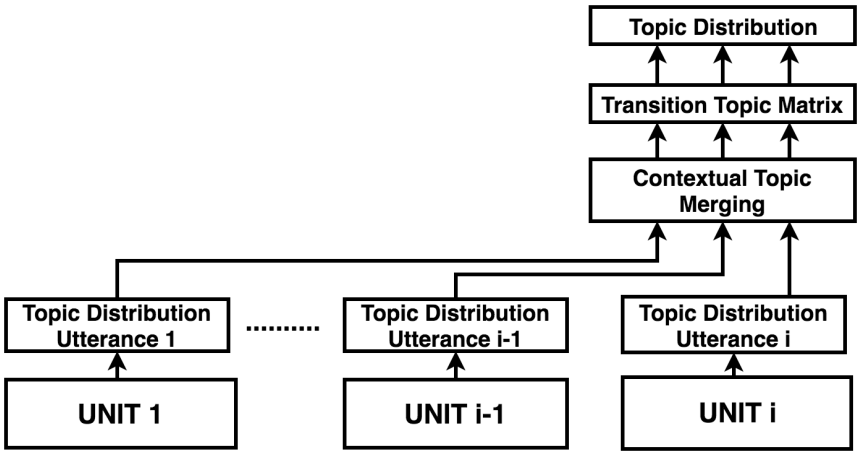

Figure 4.1: Contextual Topic Merging [1].

TopicAcceptance and UserCorrection have been added to the original classes from Amazon. Also, the information-delivery intent is the main (default) intent supported by domain-specific components, thus did not need to be predicted with the intent classifier.

Similar to the Topic Classifier architecture, for the intent classifier a 4-layer CNN model was used. For word embeddings, Google Word2Vec for representing the words, and 512 filters in each step are used. Whole 3M vocabularies are used and word vectors did not retrain during training. We also used transfer learning to improve the generalization of the model. For this purpose, TensorflowHub features are used as an external feature [23]. TensorflowHub represents a sentence in the vector space. Those vectors are extracted based on the pre-trained models that had been trained on different domains and large corpus of data. Moreover, another embedding layer of size 16 was added to train POS features, and they have been trained during training. Finally, 37 different POS tags from NLTK have been trained.

The dataset for intent classifier is highly unbalanced, which makes training challenging and might cause over-fitting on the more popular classes such as Information-Request. To solve this problem, we created balanced training dataset, and used the output of the topic classifier as an input to provide additional contextual features for

intent classifier. Thus, pipeline of Topic and Intent classifiers is ultimately used for intent classification. For some classes, such as user-correction and repetition (user asking for repetition), which related to previous utterances, we designed a Post Intent Merging phase, similar to the Contextual Topic Merging described above. In this phase, contextual features are taken into account to decide and potentially modify the intent decision to a more likely one given the conversation context. This is done primarily to deal with important special cases such as recovering from an error, or remaining in a component for following up to a question asked to a customer.

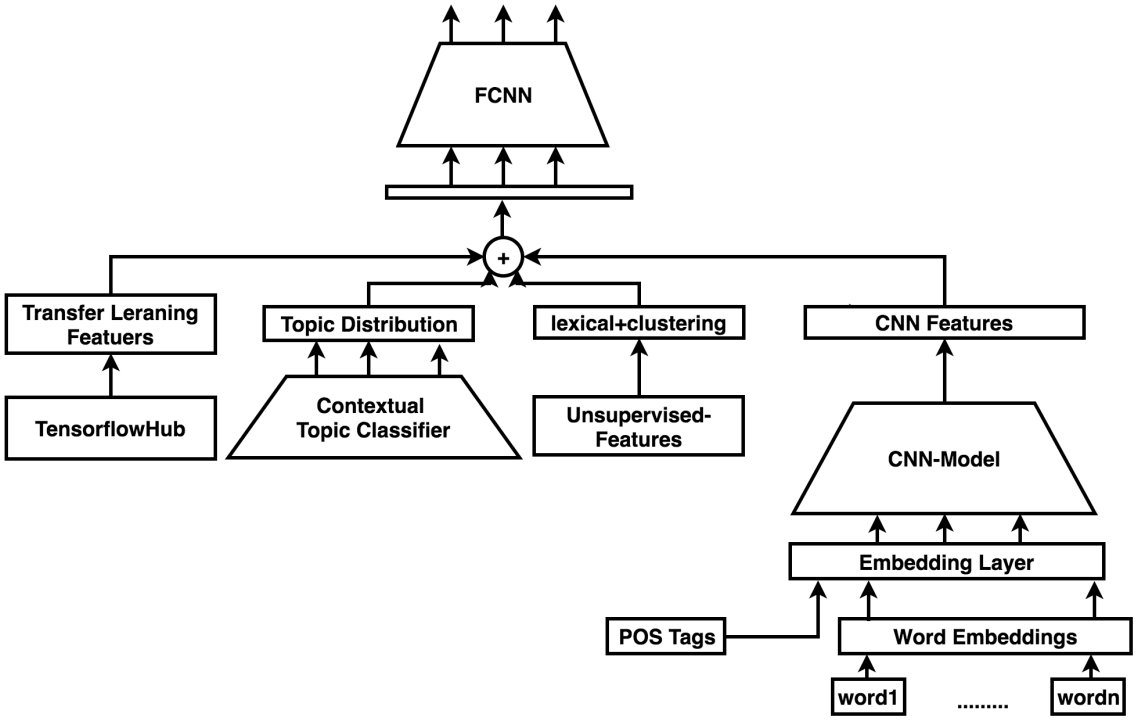

Figure 4.2: Intent Classifier [1].

4.2.6 Dialogue Manager and Response Ranking

The Dialogue Manager is at the heart of the IrisBot engine. It manages the conversation context, which stores all the NLP annotations, classification results, and is passed to each domain component to record their responses. The context object is used for

ranking and selecting the responses and post-processing them before returning to the customers.

4.2.7 Dialogue Manager: Implementation

The dialogue manager is a pipeline of utterance processing for NLU, component task execution, response ranking and filtering, and relevant information pre-fetching.

The utterance processing includes utterance ASR thresholding, profanity checking, and multi-threaded NLP processing. The ASR thresholding checks on the ASR probability input on each possible speech transcription and compares the largest probability with an empirical threshold. The conversation goes to an utterance ambiguity state and the user would be requested to repeat the utterance if the probability is less than the threshold. Profanity check is applied on the utterance and a default answer and a deflection suggestion would be returned if the utterance is detected profane. If the utterance passes the previous two checkpoints, a multi-processing NLP processing is executed to extract features in context for later decision makings. For each NLP function, a thread is designated with a processing timeout limit of 1.5 seconds. The total processing time limit of all threads is 2 seconds.

The features extracted by NLP processes are maintained in the conversation context. Based on context information, the dialogue manager then selects components to take in context and retrieve responses. The are 6 scenarios, as shown in Table 4.3, for the dialogue manager to make different decisions. For example, if the user intent is Repetition, only the Special State Component will be selected to repeat the last response. If the user intent is Information Request, all domain components would have an equal chance to retrieve responses. Domain components retrieve responses in parallel threads sharing context information with a timeout limit of 2 seconds for each thread and 2.5 seconds in total.

Assigned by the dialogue manager, components retrieve responses, assign them

Scenario	Decision
Instruction Request	Special State Component
Pause and Hesitation	Special State Component
SelfHarm	Special State Component
Stop	Special State Component
Opening and Greeting	Conversation Opening Component
Repetition	Special State Component
Information Request	Domain Components

Table 4.3: Conversation States and Component Assignment for Retrieving Responses.

with relevance scores and follow up scores, and pass them to the response ranking function. The ranking function assigns each response with a final ranking score. The final step is profanity check on the candidate responses and the best response satisfying all requirements will be returned.

Pre-fetching is a separate thread during the period between returning the response back to a user and waiting for the next input from the user. It extracts information from the responses, such as entities, and pre-fetches relevant information for the next turn of conversation.

4.2.8 Personalized Topic Suggestion

One of the key aspects of a successful conversation is topic transitions and recommending the next topic of conversation. IrisBot initially attempted to recommend topics randomly, which turned out to be disastrous for cohesive conversation flow. Instead, our next version pre-defined reasonable topic sequences, or scripts for conversations, and suggested topics following a predefined sequence if a customer did not express a preference or interest in any specific topic. However, we learned that a small number of default topic sequences does not work for all customers, and instead developed a *personalized* topic suggestion module, trained on past conversations, that attempts to predict which topic a customer would be interested in discussing next. We believe this is an important innovation as it allows our bot to tailor the topic

order to each customer based on a variety of traits and characteristics.

4.2.9 Personalized Sequential Topic Suggestion Model

To propose the next interesting topic for each customer, I introduce a Conditional Random Field (CRF)-based sequence model. To make the suggestion personalized, I investigated two approaches, outlined below. First, I introduce the CRF model for Topic suggestion, and then discuss personalization later in this section.

First, every conversation is divided into different turns, then three different feature groups were extracted from each turn. The first group is Accepted and Reject suggestions, this features vectors has the value of 1 for the accepted topics and -1 for the rejected topic and 0 for the topics that has not been proposed. This features help our model to be more eager to pick the 0 and 1 topics in the future, and consequently more reluctant to suggest ones rejected from similar conversation states. The second group is topic classification features, which represents the the current conversation topic. This feature could indicate the historical probability that a current state is a potential topic-switching point, or whether it should be a follow-up for the previous state. The third group is a set of contextual features, which represents the previous visited states, and the previous suggested topic also if it had been accepted or not. The fourth group is high level features such as the Infer-gender of user [-1,1], and friendly users [GIVENAME or NOT].

To evaluate our approach I first experimented with the topic suggestion model in simulation mode on off-line data. To label the data, two different scenarios have been followed for training and test data, for training data if topic X was suggested in turn i, and user starts talking about topic X in turn $i + 1$, the the label of "X-ACC" is assigned to turn i. If the customer rejects the suggestion and asked for something else, the label was "X-REJ", otherwise the label is a "FOLLOW-UP" for one of the components. There is a small difference between labeling of the test data compared

to training data, in which, if a customer in turn i, rejects the suggested topic and in turn $i + n$ wanted to talk about topic "X", the the label for turn i is modified from "X-REJ" to "X-ACC", because it ultimately matched the customer interests.

Personalized topic suggestion: I developed a Mixture of Experts model, in which each model had been trained on a specific group of customers. Two different ways to cluster customers into different groups had been attempted. The first one is clustering customers based on their past visiting frequency with the bot, resulting in two groups: returning, and new customers. The second way for clustering is based on time of day information, which is expected to correlate with customer demographics: dividing every day into into 4 different time buckets such as, morning, afternoon, evening and midnight. then assign every customer to one of these time buckets.

For the first experiment, I used a history field in context data which keeps track of accepted and rejected topics for all customers. In this model, the customers that previously talked to the bot, a history was recorded. For feature extraction, a customer that previously accepted and n times rejected our suggestions on topic X, a history vector like history-X [n,m] is extracted and added to original feature vector. Finally, two different models are trained on each cluster and each customer in the test is assigned for one of these clusters then the corresponding model is used for prediction. In other words a separate model is trained for repeat customers from the model used to predict topics for new customers who have never spoken to IrisBot before.

In the second experiment, 4 different models had been train on 4 different time buckets, and the corresponding model has been used for personalized suggestion for each customer based on the time of the days that they talk to the bot. A potential problem with this model is time zone difference, which make prediction less effective in our experiments described below.

4.2.10 Cross-Component topic suggestion

In addition to reactive topic suggestion to move the conversation forward if the customer does not express an interest, Irisbot also propose proactive cross-component topic suggestion, whereas a component may recommend a another component as part of its response. For example, a Movie recommendation may include a famous actor, and IrisBot would pro-actively suggest to the customer to look up the latest news on that actor. Another scenario Irisbot explored centers around locations: if a location is mentioned in the news or in a movie plot, that location would be recommended as potential point of exploration for the customer and then offered to explore the Travel and Attractions component. These two initial ideas are implemented in IrisBot and other cross-component connections and suggestions are being explored in current extensions to IrisBot.

4.2.11 Results and Discussion

First, we report the overall performance of topic and intent classifiers. Then, we dive into specific technical component evaluation to analyze how our proposed techniques work in isolation and together to improve the system performance. Specifically, we first report results of topic and intent classifier, which exceed the current state of the art as represented by the Amazon Annotate system (Section 4.2.12). Then, we report results on personalized topic suggestion (Section 4.2.13). We conclude this section by analyzing topic suggestion performance in live A/B testing for different groups of customers (Section 4.2.13), and other effects of personalization on customer behavior and satisfaction.

4.2.12 Topic and Intent Classifier: Internal Evaluation

To evaluate the topic classifier, I developed a test dataset containing 1,121 utterances of different conversations from logs. These utterances were manually labeled by three different human annotators, whom I call annotator A and B. The matching score between the annotators is 1.0, as both annotators reach an agreement on a label. The results for the classifier on 6 major topics that has intersection with the Amazon topics is reported in Table 4.4. The average (accuracy and Micro F1-score) for the Customized Classifier (CC), Entity Classifier (EC) and Contextual Topic Merging (CTM) on all covered topics in our bot are (0.669, 0.605), (0.676, 0.616) and (0.766, 0.719), respectively.

Topic	CC-accuracy	EC-accuracy	AA-accuracy	CTM-accuracy
Movie	0.693	0.851	0.840	0.912
Music	0.836	0.916	0.875	0.891
Sports	0.923	0.693	0.693	0.851
Celebrities	0.969	0.866	0.814	0.903
Politics	0.770	0.854	0.915	0.861
Fashion	0.823	0.95	0.76	0.851
Average	0.835	0.855	0.816	0.878

Table 4.4: Classification accuracy for CC: Customized Classifier, EC: Entity Classifier, AA: Amazon Annotator, CTM: Contextual Topic Merging Classifier .

training set containing 20K samples generated from logs and other resources, where 80% is used for training and 20% for validation. To evaluate the intent classifier, I developed a test sets of 350 utterances based on our manual labels, and Amazon annotation labels. The average Micro F1 and Macro F1 scores for this dataset is 0.79 and 0.83 respectively. The detailed results for different intent classes are reported in Table 4.5.

Suggested Topic	Accuracy
user instruction	0.727
Opinion Request	0.980
Opinion delivery	0.560
Personal Delivery	0.880
Topic Switch	0.901
Topic acceptance	0.956
unknown	0.752
Repetition	0.941
Info Request	0.951
Dont know	0.833
Self harm	0.98
Average	**0.831**

Table 4.5: Intent classification accuracy.

4.2.13 Topic Suggestion Results

I evaluated our proposed CRF-based topic suggestion model on 3,606 different conversations from August 1 to August 15, the first 10 days of conversations are used for training and validation, and the rest for testing. The off-line results for comparisons two methods of Heuristic suggestion and CRF model are reported on Table. 4.25. In the Heuristic method, the method always suggests the next topic based on their overall popularity.

Suggested Topic	Heuristic Accuracy	CRF Accuracy
Movie	0.508	0.800
Music	0.648	0.811
Attraction	0.498	0.778
Pets-animals	0.486	0.775
news	0.397	0.796
politics	0.389	0.569
sports	0.270	0.6125
Scitech	0.417	0.639
cars	0.397	0.756
games	0.473	0.698
Average	**0.476**	**0.772**

Table 4.6: CRF for Topic suggestion.

If customers are clustered into returning and new cohorts, as I described above,

two different models can be trained on each group. If I train a model on returning customers and try to predict the suggested topic for just the returning customers, the average acceptance rate is expected to be 0.629; However, the regular model that has been trained on all users predicts 0.655. the results shows, first, the returning customers are more difficult to predict, and second using specialized classifier to model just returning customers is not more effective than one general model. I also experimented with clustering customers into 4 different groups by conversation time. However, preliminary results show that using all data is still more accurate than having four different models for each special time period.

Preliminary A/B testing results: Due to lack of time I were not able to fully test the personalized predictions in the live setting. One preliminary result on 36 conversations on live data in A/B testing is inconclusive: it shows only slight improvement from 0.460 to 0.461 for returning users, which is insignificant; therefore, more data is needed to decide whether a new model is needed or a different algorithm for using the suggested predictions. I plan to explore this in our immediate current work over the next several weeks, combined with other improvements to the experience of returning customers.

4.2.14 Effects of Personalization on conversation behavior and ratings

In order to compare the effect of personalization suggestion feature, a controlled experiment was conducted: identical system with and without personalization suggestion feature was ran side-by-side for one day. The results are promising, indicating that personalization could play a significant role on increasing customer satisfaction.

One bucket which had personalization suggestion received an average of 4.02 rating from 360 returning users and 3.22 rating from 2,161 new users. Another bucket which

didn't have personalization suggestion scored the average of 3.65 rating from 178 new users, while receiving only 2.80 average rating from 52 returning users.

As a result, it is clear that treating returning customers as new, is devastating to their conversation experience. It seems natural for customers to remember their previous experience regardless of anonymous setting of this competition. Recovering previous context, and providing personalized suggestions or diversifying the topics of conversation could potentially improve experience of the customers dramatically, and is a central focus of our ongoing work.

4.3 ConCET: Entity-Aware Topic Classification for Open-Domain Conversational Agents

Identifying the topic (domain) of each user's utterance in open-domain conversational systems is a crucial step for all subsequent language understanding and response tasks. In particular, for complex domains, an utterance is often routed to a single component responsible for that domain. Thus, correctly mapping a user utterance to the right domain is critical. To address this problem, I introduce ConCET: a Concurrent Entity-aware conversational Topic classifier, which incorporates entity-type information together with the utterance content features. Specifically, ConCET utilizes entity information to enrich the utterance representation, combining character, word, and entity-type embeddings into a single representation. However, for rich domains with millions of available entities, unrealistic amounts of labeled training data would be required. To complement our model, I propose a simple and effective method for generating synthetic training data, to augment the typically limited amounts of labeled training data, using commonly available knowledge bases as to generate additional labeled utterances. I extensively evaluate ConCET and our proposed training method first on an openly available human-human conversational dataset called Self-

Dialogue, to calibrate our approach against previous deep learning-based methods; second, I evaluate ConCET on a large dataset of human-machine conversations with real users, collected as part of the Amazon Alexa Prize. Our results show that ConCET significantly improves topic classification performance on both datasets, including 8-10% improvements over deep learning methods. I complement our quantitative results with detailed analysis of system performance, which could be used for further improvements of conversational agents.

In summary, our contributions are: (1) The development of ConCET, a novel entity-aware topic classifier by combining implicit and explicit representations of an utterance and fusing them with handcrafted features; (2) Incorporating external knowledge about entities retrieved from a knowledge base; and (3) creation of a new large-scale synthetic yet realistic dataset for training topic classification systems designed for open-domain conversational agents. Next, I present related work to place our contributions in context.

4.3.1 ConCET System Overview

I now introduce our ConCET system at a high level, before diving into implementation details. Our proposed ConCET model is illustrated in Figure 4.3.

ConCET utilizes both textual and entity information from an utterance. To represent textual and entity information, ConCET extracts both sparse and dense representations. To this end, a pipeline of deep neural networks and handcrafted feature extraction modules is designed. This pipeline consists of four components namely Utterance-to-Vector (Utt2Vec) network, feature engineering module, Entity-to-Vector (Ent2Vec) network, and the Entity-type distribution generator. The Entity-type distribution generator module uses an entity linker to get the entity-type distribution corresponding to each entity in the utterance.

Utt2Vec and the feature engineering module extract the textual representation.

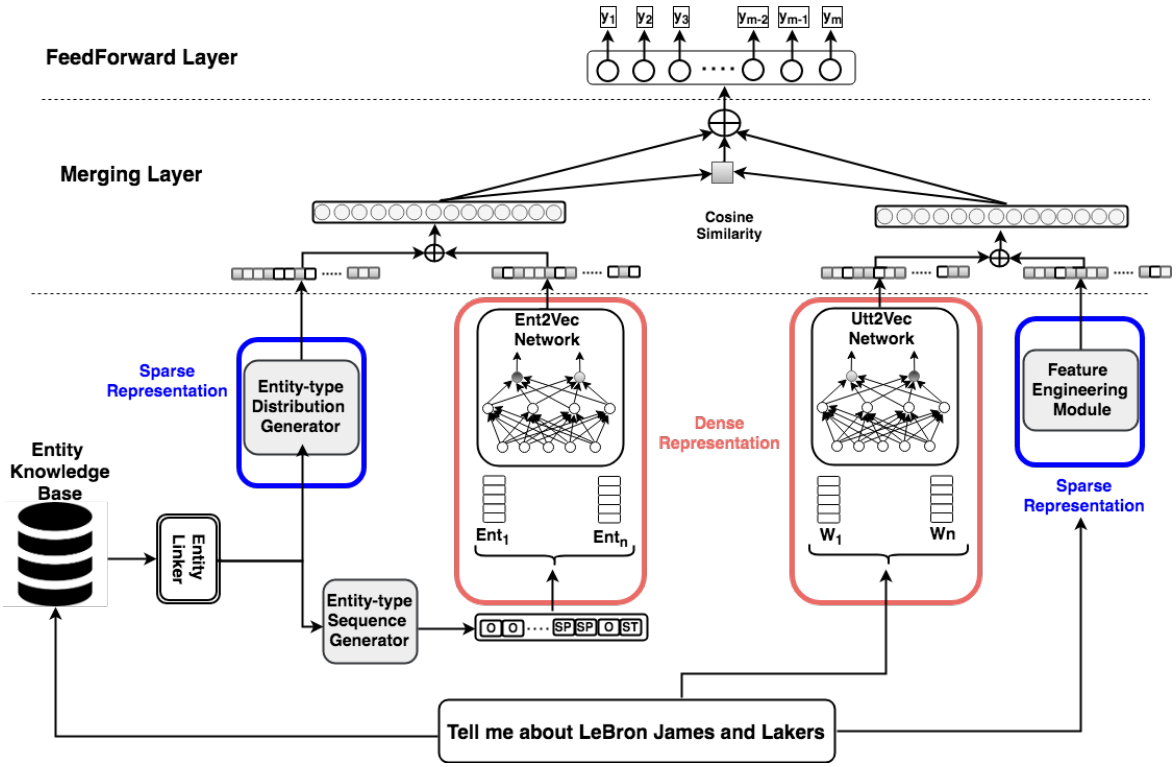

Figure 4.3: The overall network architecture for Entity-Aware Topic Classifier (ConCET) model, where "SP" and "ST" stand for *Sports_Player* and *Sports_Team* entity-types [3].

Utt2Vec is a deep neural network model which utilizes character, word, and POS tags for utterance representation. Feature engineering module extracts handcrafted features such as LDA and LSA topical distribution from an utterance. Finally, they are combined through a fully-connected neural network.

To model the entity information, ConCET utilizes both the entity-type distribution and the order of entity-types appearing in the utterances.

Ent2Vec network is responsible for mapping this entity sequence representation to a high dimensional vector. Entity-type distribution features and the output of the Ent2Vec network are combined through a fully-connected neural network.

Next, the cosine similarity[1] between textual and entity representations is computed. This similarity value, concatenated with the textual and entity representations, is fed to a feed-forward layer to compute the final softmax distribution of topics.

To summarize, ConCET proposes an entity-aware text representation model that learns a ternary representation of character, word and entity information. In the next section, I introduce the entity linking methods used to derive entity-based information. I conducted our experiments using two different entity linkers to measure the sensitivity of the ConCET model to the entity linking step. Then, in Section 4.3.5 I explain the details of the ConCET model.

4.3.2 Conversational Entity Linking

In this section, I describe the two entity linkers that were used for detecting entities and their type distributions. The type information is used for semantic representation in the ConCET model.

I emphasize that the focus of this work is *not* on developing a novel entity linker, which is an important area of research on its own. Rather, I experiment with an off-the-shelf entity linker, DBPedia Spotlight[2], and our own PMI-based domain-specific

[1]Dot product also can be used. In this case, the entity vector should be normalized to unit length.
[2]https://github.com/dbpedia-spotlight/spotlight-docker

entity linker (PMI-EL), designed to cover in more depth some of the conversation domains and entity-types most relevant to our conversational agent. Our experiments with different off-the-shelf entity linkers during the development of our conversational agent showed these two linkers are the most effective for topics that our bot supported. I describe both entity linkers in depth in the next section, here I want to emphasize that the proposed classifier model can incorporate the output of any available entity tagger or linker.

4.3.3 DBpedia Spotlight

DBpedia Spotlight annotates DBpedia resources mentioned in the text as described in reference [96]. It annotates DBpedia resources of any of the 272 classes (more than 30 top-level ones) in the DBpedia Ontology. It performs entity annotations in 3 steps, 1) spotting, 2) candidate selection, and 3) disambiguation. It uses the Aho-Corasick string matching algorithm for finding all the phrases which could potentially be entity mentions or surface forms. It then finds candidate entities for each surface form using the DBpedia Lexical Dataset. For disambiguation, each candidate DBpedia resource is first modeled in a Vector Space Model (VSM) as the aggregation of all paragraphs mentioning that concept in Wikipedia. The candidates are then ranked by their $tf*icf$ cosine similarity score with respect to the context, where the icf score estimates how discriminating a word is, which is assumed to be inversely proportional to the number of DBpedia resources it is associated with.

4.3.4 PMI-based Entity Linker (PMI-EL)

I created a domain-specific entity linker called PMI-EL for our conversational system for the Alexa Prize, which annotates the 20 entity-types most relevant to our system. It links entities to an associated knowledge base containing all the entities supported by our conversational agent. PMI-EL follows similar steps to DBpedia

Spotlight. However, it does not use the utterance context in the disambiguation step and relies solely on an estimated prior distribution of types for a given entity for disambiguation. The main reason was that most of the user utterances were short (average utterance length of 3.07 words), and sometimes consisted of just the entity name. Thus, the context was often not helpful or present, and type inference based on prior probabilities may be sufficient for this setting. I next describe the process by which the knowledge base was constructed, and how the prior type probabilities were estimated for entity-type inference.

Movie_Name	Celebrities	Authors	Bands
Sports_Team	Sportname	Companies	Food
Organization	Politicians	Universities	Singers
Songname	Animal	Country	Actors
Hotels_Foodchains	Tourist_points	Genre_Books	City

Table 4.7: Entity-types recognized by PMI-EL.

PMI-EL knowledge base construction

Our knowledge base starts with entities from a snapshot of DBpedia from 2016. Additionally, to provide coverage of current entities of potential interest to the user, I augment the knowledge base by adding entities that our open-domain conversational agent supports. I periodically retrieve entities from the following sources and domains:

- Persons, Organizations and Locations: from news provided by Washington Post[3]
- Cities and Tourist Attractions: from Google Places API[4]
- Bands and Artists: from Spotify[5] and Billboard[6]

[3] *https://www.washingtonpost.com/*
[4] *https://developers.google.com/places/web-service/search*
[5] *https://www.spotify.com/us/*
[6] *https://www.billboard.com/*

- Books and Authors: from Goodreads[7] and Google Books[8]
- Actors and Movies: from IMDb[9] and Rotten Tomatoes[10]

I maintain an index of all the entities and their corresponding types using ElasticSearch[11], which is used in the online entity linking step.

PMI-based type distribution

For entities with more than one type, I also index the estimated pointwise mutual information (PMI) [19] of the entity with all its types. PMI is a measure of how much the actual probability of a particular co-occurrence of events p(x, y) differs from what I would expect it to be on the basis of the probabilities of the individual events and the assumption of independence of events x and y, and is calculated as:

$$PMI(x, y) = ln\left(\frac{p(x, y)}{p(x)p(y)}\right) \tag{4.1}$$

To predict the most likely type for entities with multiple types, I estimate the point-wise mutual information (PMI) of the entity with each type by counting the co-occurrences of the entity and the type's name in a large corpus, which has been shown to correlate with the probability of association [19]. More formally, the entity-type PMI score is computed as:

$$PMI(m, t_i) = \frac{|(Docs(m, C) \cap Docs(t_i, C))|}{|Docs(m, C)|} \tag{4.2}$$

[7] *https://www.goodreads.com/*
[8] *https://developers.google.com/books/*
[9] *https://www.imdb.com/*
[10] *https://www.rottentomatoes.com/*
[11] *https://www.elastic.co/products/elasticsearch*

where m is an entity mention, t_i is a type, C is a corpus and $Docs(phrase, C)$ is a set of documents in C containing a given phrase. For our experiments, as C I used a publicly available corpus of 46 million social media posts from a snapshot of Reddit.[12] For example, to disambiguate the mention "Kings" in a user's utterance, I compute the number of times each type name co-occurred with the word "Kings" in the corpus, normalized by the total number of occurrences of the word "Kings" itself.

In this example, the type distribution for a string "Kings" is: $[Sports\_Team : 0.54, Movie\_Name : 0.44, City : 0.02]$. Because of the large size and diversity of the corpus, PMI is expected to be a good estimate of type distribution. Despite potential noise in estimating type distribution for some polysemous entities, the ConCET model is able to use the type distribution, as I demonstrate empirically under a variety of conditions.

PMI-EL entity detection in utterances

To support efficient entity linking at run-time, an inverted n-gram entity index was constructed for all entities in the knowledge base. At runtime, entities are detected via n-gram matching against an entity index. For example, if the utterance is "who won the Hawks and Kings game", I query the index for "the Hawks", "Kings", "Hawks" and every other possible n-gram with less than 6 words. For this utterance, the response from the entity index would be the entities and the type distributions associated with them, e.g. "Hawks": $[Sports\_Team : 0.88, Animal : 0.11, City : 0.01]$ and "Kings": $[Sports\_Team : 0.54, Movie\_Name : 0.44, City : 0.02]$.

The entity detection step has time complexity $O(n^2)$ in the number of words in the utterance since I perform $O(1)$ look-ups for $O(n^2)$ n-grams for each utterance. The running time for entity linking is 16 ms on an average for utterances with 4 words which were common, and 100 ms for utterances with 32 words, which were among

[12] *https://files.pushshift.io/reddit/submissions/*

the longest utterances I encountered. However, PMI-EL would not be efficient if used on very long text.

The output from the entity linker is passed to the Entity Representation Model, described in 4.3.7, which converts it into a suitable representation for the ConCET model.

4.3.5 ConCET: Concurrent Entity-Aware Topic Classifier

In this section, I present the details of ConCET model. First, Section 4.3.6, describes our model for the textual representation of the utterance. Then, Section 4.3.7, presents the proposed entity representation model. Finally, Section 4.5.6 discusses the merging and decision layer of the ConCET model.

4.3.6 Textual Representation

I use character, word, and POS tagging to model the textual representation. Then, I enrich the representation with the unsupervised topic distribution, as described in detail next.

Utterance to vector (Utt2Vec) network

Utt2Vec network takes word tokens Utt_w, characters Utt_c and POS tags Utt_p of an utterance Utt as inputs:

$$Utt_w = [w_1; w_2; w_3 \ ... \ w_n] \tag{4.3}$$

$$Utt_c = [[c_{11}...c_{1k}]; [c_{21}...c_{2k}]; \ ... \ [c_{n1}...c_{nk}]] \tag{4.4}$$

$$Utt_p = [p_1; p_2; p_3 \ ... \ p_n] \tag{4.5}$$

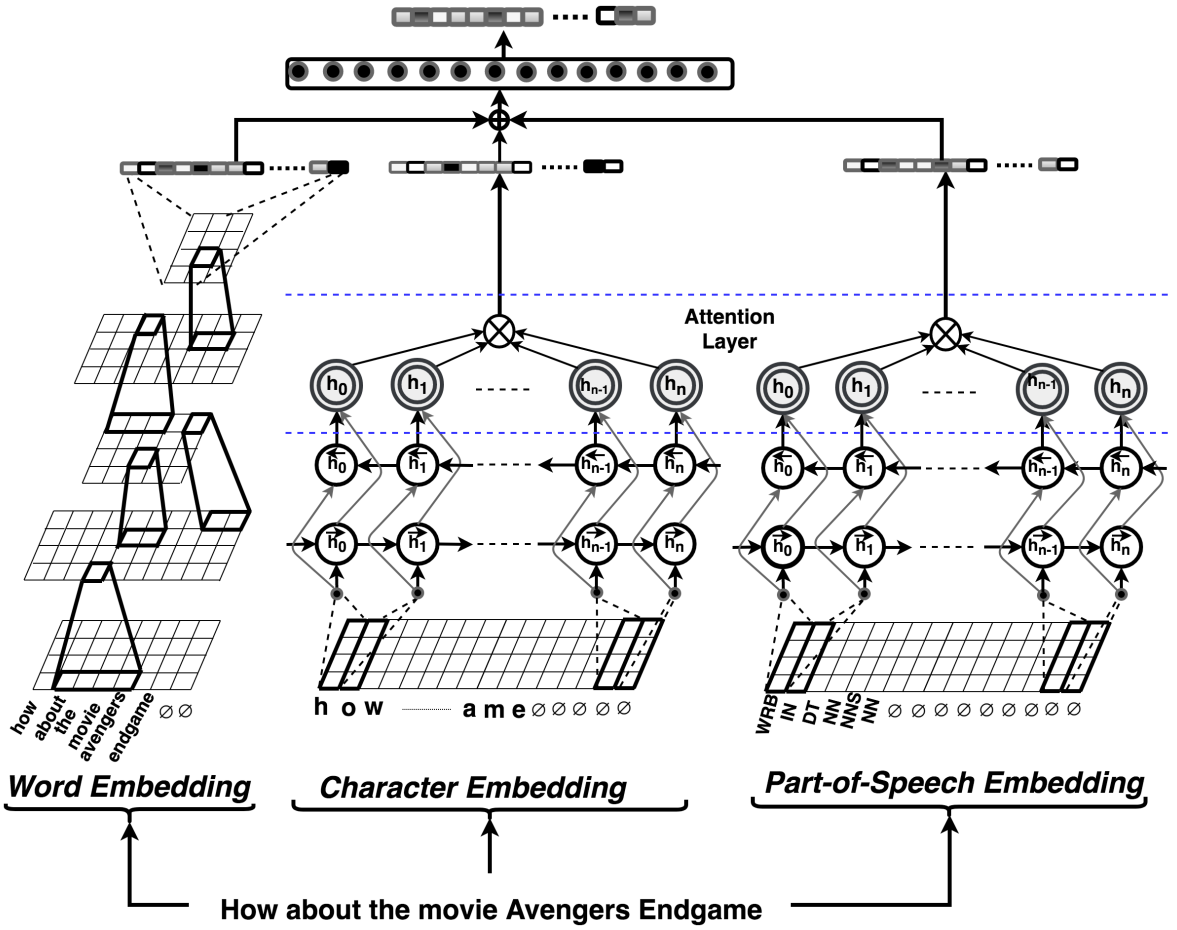

Figure 4.4: Utt2Vec network [3].

I use the NLTK[13] library for extracting POS tags. Utt2Vec network allows freedom of combining different deep learning architectures such as CNN and RNN to extract features. I define three functions f_w, f_c, and f_p that each take these inputs and output learned hidden representations (h):

$$h_w = f_w(Utt_w) \tag{4.6}$$

$$h_c = f_c(Utt_c) \tag{4.7}$$

$$h_p = f_p(Utt_p) \tag{4.8}$$

[13] *http://www.nltk.org*

For our implementation, f_{w} is a 3-layered CNN with max pooling. For f_{c} and f_{p}, I use 1-layered BiLSTM network with global attention. For the word embedding layer, I pre-initialize the weights using Word2Vec vectors with size 300. The weights on the word embedding layer are tuned during training. For character and POS embeddings, I randomly initialize the embedding layer with size 16. Given the hidden representations of each timestamp h_{i} in LSTM cells, dot product similarity score $\mathrm{s_i}$ is computed based on a shared trainable matrix M, context vector c and a bias term b_i. Softmax activation is applied on similarity scores to obtain attention weights α. Lastly, using learned α, weighted sum on BiLSTM hidden representations is applied to obtain the output $\hat{h}$ as follows:

$$s_i = tanh(M^{\mathrm{T}} h_i + b_i) \tag{4.9}$$

$$\alpha_i = \frac{exp({s_{\mathrm{i}}}^{\mathrm{T}} c)}{\sum_{i=1}^{n} exp({s_{\mathrm{i}}}^{\mathrm{T}} c)} \tag{4.10}$$

$$\hat{h} = \sum_{i=1}^{n} \alpha_{\mathrm{i}} h_{\mathrm{i}} \tag{4.11}$$

M, c, and b are randomly initialized and jointly learned during training. The three outputs from word-CNN (h_{w}), char-BiLSTM ($\hat{h}_{\mathrm{c}}$), and POS-BiLSTM ($\hat{h}_{\mathrm{c}}$) are concatenated to produce Utt2Vec output:

$$Utt2Vec_{\mathrm{out}} = [h_{\mathrm{w}}; \hat{h_{\mathrm{c}}}; \hat{h_{\mathrm{p}}}] \tag{4.12}$$

This final output is fed to a linear layer of size 256 with ReLU activation and a dropout rate of 0.5 to obtain the utterance vector.

Feature engineering module

The goal of this module is to provide the flexibility of incorporating various external features in ConCET. Since I are focusing on domain classification, I extract unsu-

pervised topic modeling features. However, depending on the data and the task, any type of feature extraction pipeline can be incorporated here. I combined two different topic modeling algorithms, LDA and LSA, and implemented models using the Gensim library[14]. Given hyperparameter n, these models output the unsupervised topic distribution of size n. By concatenating the two outputs described in the table below, I obtain a topic distribution vector of size $2n$.

Features	**Short Description**
F_{LDA}	LDA topic distribution
F_{LSA}	LSA topic distribution

The outputs of these two vectors are concatenated to produce F_{out}:

$$F_{\text{out}} = [F_{\text{LDA}}; F_{\text{LSA}}] \tag{4.13}$$

4.3.7 Entity Representation

I now describe how I encode the entity information from the linker as input to our model. I have two modules to do this encoding, 1) Entity-type sequence generator and 2) Entity-type distribution generator. Entity-type sequence generator converts the input word sequence to an entity-type sequence so that the model can learn to predict the topic based on the order in which different entity-types appeared in the utterance. This sequence is fed into the Ent2Vec network, which creates a high-dimensional vector representation for the sequence. The Entity-type distribution generator constructs an overall entity-type distribution for the utterance by aggregating type distributions for all the entities. Finally, the output of Ent2Vec is concatenated with the entity-type distribution to generate the final entity representation. I now describe these modules in detail.

[14] *https://radimrehurek.com/gensim/*

Entity-type sequence generator

The input of this module is the list of entities and their type distributions derived from the entity linker. To generate this entity sequence, I need to assign the best type corresponding to each entity. The words that are not a part of an entity are assigned *Other* or *O*. For example, for "who won the Hawks and Kings game", a possible entity sequence vector would be ["who"/*O*, "won"/*O*, "the"/*ST*, "Hawks"/*ST*, "and"/*O*, "Kings"/*ST*, "game"/*O*]. However, different entity linkers can differently assign entity-types to each word. Consequently, the resulting entity vector has the exact length of the utterance.

$$Utt_{\text{ent}} = [e_1; e_2; e_3 \ ... \ e_n] \tag{4.14}$$

Entity-type distribution generator

For this module, I first have to determine the total number of entity-types that I want the model to support. For example, for the PMI-based linker, I support 20 types, and for DBpedia Spotlight, I support the 1000 most frequent entity-types from the training set. After determining the size, the distribution value for each entity-type is either 0, or the maximum value for that type in the list of entity-type distributions. For the example from the previous section, "who won the Hawks and Kings game", the type distributions for the two entities from the PMI-based linker are, respectively, $[Sports\_Team : 0.88, Animal : 0.11, City : 0.01]$ for "Hawks", and $[Sports\_Team : 0.54, Movie\_Name : 0.44, City : 0.02]$ for "Kings". In that case, if the entity linker identifies 20 types in total, the final entity-type distribution is $[Sports\_team : 0.88, Movie\_Name : 0.44, Animal : 0.11, City : 0.02]$. The value corresponding to the remaining types in Table 4.7 is 0.0 in the final output vector of length 20.

Entity to vector (Ent2Vec) network

The input to Ent2Vec network is a list of resolved entity-types per word for Utt_{ent} from entity-type sequence generator:

$$Utt_w = [w_1; w_2; w_3 \ ... \ w_n] \tag{4.15}$$

$$Utt_{\text{ent}} = [e_1; e_2; e_3 \ ... \ e_n] \tag{4.16}$$

I define a function f_e that takes Utt_{ent} and outputs learned hidden representations as follows:

$$h_e = f_e(Utt_{\text{ent}}) \tag{4.17}$$

I also use 1-layered BiLSTM network as our f_e function. I randomly initialize an entity embedding layer that has 16 trainable weights per each entity-type. Then, the same attention mechanism as in Section 4.3.6 is applied to h_e to obtain $\hat{h}_e$ or $\text{Ent2Vec}_{\text{out}}$. Lastly, entity-type distribution Ent_{dist} is concatenated with $\hat{h}_e$ to obtain the final Entity output:

$$Ent_{\text{out}} = [Ent2Vec_{\text{out}}; Ent_{dist}] \tag{4.18}$$

This output is fed to a linear layer of size 100 with ReLU activation and a dropout rate of 0.5 to obtain the final entity vector.

4.3.8 Merging and FeedForward Layer

I obtained the three different outputs each from Utt2Vec network, feature engineering module and Ent2Vec network. $Utt2Vec_{out}$ is first concatenated with F_{out} to obtain the following final textual representation $Text_{out}$ of an utterance:

$$Text_{out} = [Utt2Vec_{out}; F_{out}] \tag{4.19}$$

I feed $Text_{out}$ to a linear layer of size 100 with ReLU activation to obtain vector of the same length as Ent_{out}. Cosine similarity between these two vectors are computed and concatenated to obtain 201-dimensional $ConCET_{out}$:

$$ConCET_{out} = [Ent_{out}; Text_{out}; Cos(Ent_{out}, Text_{out})] \tag{4.20}$$

According to [147], cosine similarity represents the normalized likelihood that entity-type Ent_{out} appears in $Text_{out}$. Finally, softmax activation is applied to generate a probability distribution over n possible domains.

4.3.9 Conversational Dataset Overview

In this section, I describe the conversational data collected during the 2018 Alexa Prize and another publicly available dataset called Self-Dialogue. I also describe the algorithm I designed to generate synthetic training samples, which will be used to augment the original data.

4.3.10 Amazon Alexa Prize 2018

The data for evaluation of the proposed models is collected from the 2018 Alexa Prize, a competition held by Amazon every year since 2017 to advance conversational AI. Our team was one of the 8 semi-finalist teams funded by Amazon for the competition. Users were asked to talk to our conversational bot and give a rating from 1.0 to 5.0 (inclusive) based on their experience.

4.3.11 Obtaining True Labels for Alexa Data

Two hundred conversations from the Alexa Prize data were randomly chosen, which consist of 3,000 utterances and responses. These utterances were manually labeled by three different human annotators, whom I call annotator A, B, and C. The matching and kappa scores between the annotator pairs (A, B), (A, C), and (B, C) are (0.82, 0.78), (0.72, 0.65), and (0.80, 0.75), respectively. Overall, these metrics indicate *substantial agreement* between all annotators. The final true labels were selected by majority voting. When there was no majority, one of the labels was randomly selected. The final distribution of annotated topics is shown in Table 4.24.

Movie	31%	Music	20%	News	16%
Pets_Animal	6%	Sci_Tech	6%	Sports	6%
Travel_Geo	2.5%	Celebrities	2.5%	Weather	1.5%
Literature	1.5%	Food_Drinks	1.5%	Other	1.5%
Joke	1%	Fashion	1%	Fitness	1%
Games	1%				

Table 4.8: Topics distribution in Alexa Data.

I randomly selected 90 conversations for training and 10 conversations for validation. The remaining 100 conversations were reserved for evaluation.

4.3.12 Self-Dialogue Dataset

Self-Dialogue dataset[15] released by one of the Alexa Prize teams [73] is a human-human conversational dataset collected by using Amazon Mechanical Turk. Given a predefined topic, two workers talked about anything related to this topic for 5 to 10 turns. Although this dataset is not comprised of human-machine conversations, it is one of the few publicly available datasets which has a very similar structure to real human-machine conversations, except that the utterances are syntactically richer. This dataset contains 24,165 conversations from 23 sub-topics and 4 major topics:

[15] *https://github.com/jfainberg/self_dialogue_corpus*

Movie, *Music*, *Sports*, and *Fashion*. The topic distribution for the Self-Dialogue dataset is 41.6%, 35.1%, 22.2%, and 1.1% for *Movie*, *Music*, *Sports*, and *Fashion*, respectively.

For training, all subtopics are merged into the 4 major topics. I also filtered 198 conversations that were designed only for transitions from *Movie* to *Music* topics and 216 conversations with mixed *Movie* and *Music* labels because I could not assign a unique label. In addition, some of the utterances in the dataset are non-topical chit-chat utterances. They are mostly used for conversational follow-ups such as *Yes-Answers*, *Backchannel*, and *Conventional-opening*. Since these utterances are unrelated to domain classification, I removed these types in both the training and the test set. To do this, I annotated all the utterances using pre-trained ADAN [63] classifier, which supports 25 topical domains and one *Phatic* domain. The *Phatic* domain represents all chit-chat and non-topical utterances and any utterance annotated as *Phatic* is removed from both the training set and the test set. To verify the accuracy of ADAN classifier, I randomly selected 20 conversations and asked one human annotator to label each utterance as *Phatic* or *Non-Phatic*. Based on this setup, inter-annotator agreement of 0.87 and Kappa score of 0.82 were achieved, indicating substantial agreement. The final processed dataset consists of 23,751 conversations (363,003 utterances) on 4 main topics. Finally, I divided the dataset into 70%, 10% and 20% for training, validation, and evaluation, respectively.

A summary of the Alexa data and Self-Dialogue dataset statistics is reported in Table 4.9. Utterances from the Alexa data are significantly shorter (3.07 words on average compared to 9.79 in Self-Dialogue), indicating that often entities may be mentioned without extensive context, e.g., as a response to a system question.

Dataset	Words per Utterance	Turns per Conversation	Vocabulary Size
Alexa	3.07	16.49	16,331
Self-Dialogue	9.79	5.84	117,068

Table 4.9: Alexa and Self-Dialogue data statistics.

4.3.13 Synthetic Training Data Generation

I propose a simple yet effective approach to generate many synthetic utterances for training topic classification models. As I will show, this ability can be particularly useful for augmenting real data when limited manual labels are available, to train deep neural network models which require large amounts of labeled training data. The approach is summarized in Algorithm 1.

For each topic, a small number of predefined intent templates are created. These templates are designed by engineers who developed each domain-specific module. The rules described in Amazon Alexa developers' guide[16] were applied in order to capture the most common topic-specific intents and accommodate enough lexical and syntactic variations in the text. The templates contain slots to be filled with either entities or keywords, for example, "Play a *KEYWORD_MUSICGENRE* music from *NER_SINGER*" and "tell me some *KEYWORD_MOVIEGENRE* films played by *NER_ACTOR*".

Each slot starting with NER is filled by an entity from the knowledge base, and each slot starting with KEYWORD is filled using a predefined list of intent-oriented keywords. For instance, the slot *KEYWORD_MUSICGENRE* is randomly filled using a list of popular music genres like *rock*, *pop* and *rap*. I first generated these predefined keywords manually and expanded the lists with the 10 most similar words from WordNet[17] for each keyword. To fill in the entity slots, I used the corresponding lists from our knowledge base (described above), prioritizing the most popular entities, and the

[16] *https://developer.amazon.com/docs/custom-skills/best-practices-for-sample-utterances-and-custom-slot-type-values.html*

[17] *https://wordnet.princeton.edu*

most common templates according to domain knowledge and most frequent utterance statistics. While the possible number of generated utterances is the direct product of the number of templates, keyword values, and entity-values, the process ends after a predefined number of synthetic utterances is reached. For our experiments, I control the size of the synthetic dataset with a parameter named ρ. This value is determined based on the number of available templates for a topic, importance of a topic, and the overall number of covered topics. I conducted an experiment on this value described in Section 4.3.20. I decided to choose $400K$ to make a trade-off between time and accuracy and to make the experiments manageable.

Movie	28%	Music	15%	Pets_Animal	13%
Travel_Geo	12%	News	10%	Games	10%
Sports	5%	Sci_Tech	3%	Celebrities	2.5%
Fashion	1%	Weather	1%	Literature	1%
Food_Drinks	0.9%	Other	0.1%		

Table 4.10: Topics distribution in Synthetic Dataset.

Any other external dataset can be incorporated into the synthetic generator above to enrich classes lacking sufficient samples. In our experiments, I did not have as many entities from *Technology* and *Sports* domain compared to *Movies* and *Music* domains. Hence, I used an open-source Yahoo-Answers question-answer corpus to add questions for these classes. Since human-machine utterances tend to be short, as reported in Table 4.9, I only added questions shorter than 10 words. The final topic distribution of the synthetic dataset is shown in Table 4.10.

4.3.14 Experimental Setup

In this section, I first describe baseline methods in Section 4.3.15. Experimental metrics and procedures are described in Section 4.3.17.

All experiments were implemented in Python 2.7 using TensorFlow 1.12.0[18] li-

[18] *https://www.tensorflow.org*

Template and Entity-based Synthetic Utterance Generator
for *topic in topic_list* **do**
  for *template in common_topical_templates* **do**
    tmp = read(template);
    $\rho = SYNTHETIC\_DATASET\_SIZE$;
    e.g. tmp = "Fun facts for NER_ANIMALS"
    e.g. tmp = "The best KEYWORD_LEAGUE team"
    slot_list = find(slots);
    for *entity_type* ***and*** *keyword_type in slot_list* **do**
      for *entity in entity_type* ***and*** *keyword in keyword_type* **do**
        if *entity* ***in*** *common_entity_list* ***and*** *keyword* ***in*** *common_keyword_list* **then**
          generate_utterance(temp);
          generate_label();
          if *len(dataset) ¿ ρ* **then**
            return ***dataset;***
          else
          end
        else
          continue;
        end
      end
    end
  end
end

Algorithm 1: Algorithm to generate the synthetic template-, keyword-, and entity-oriented utterances.

brary.

4.3.15 Baselines

Three deep learning based methods were used as baselines:

- **ADAN** [63]: ADAN was proposed by Amazon for conversational topic classification, and it was trained on over 750K utterances from internal Alexa user data for 26 topics.
- **FastText**[16]: FastText is a text classification model from Facebook Research. FastText operates on character n-grams and uses a hierarchical softmax for prediction, where word vectors are created from the sum of the substring character n-grams.
- **VDCNN** [34]: This model was proposed as a character-based text classification model. VDCNN, like FastText, can model misspelled words (potentially mitigating ASR problems in human-machine conversations) more robustly than word-embedding based models.

4.3.16 Training Parameters

To train the ConCET model, the parameters for CNN and BiLSTM described in Figure 4.4 were chosen based on our experience and previous literature. Finally, I trained the overall model with an Adams optimizer and a learning rate of 0.001. All experiments for ADAN were conducted using the topic classifier API made available to the teams by the Amazon Alexa Prize [63]. To train the FastText model[19], character 5-grams with word embedding of size 300 were used. Finally, VDCNN results are reported based on a publicly available implementation.[20]. The results are reported

[19] *https://fasttext.cc*
[20] *https://github.com/zonetrooper32/VDCNN*

for a 29-layer VDCNN, based on the original paper.

4.3.17 Evaluation metrics

I used two standard classification metrics, **Micro-Averaged Accuracy** and **Micro-Averaged F1** [100], to evaluate our approach.

4.3.18 Results and Discussion

I begin this section by reporting the performance of ConCET in comparison to the baseline models described in Section 4.3.15. Then, I illustrate the impact of the entity, external, and utterance features through a feature ablation study.

4.3.19 Main Results

Table 4.11 summarizes the performance of the models on Alexa and Self-Dialogue datasets.

The results show that both variations of ConCET outperform the baselines Fastext, VDCNN, and ADAN on Alexa dataset by large margins of 13%, 23%, and 10%, respectively in terms of Micro-Averaged F1 score. Among the baselines, ADAN has the best results on the Alexa dataset, while VDCNN achieves the best results on the Self-Dialogue dataset. All the improvements are statistically significant using one-tailed Student's t-test with p-value ¡ 0.05.

Interestingly, the performance of the VDCNN and ADAN methods switches for the human-machine and human-human datasets, as ADAN relies only on keywords, which is not sufficient for complex human-human utterances, while VDCNN exhibits the worst performance for short human-machine utterances. In contrast, ConCET exhibits robust and consistently high performance on both human-human and human-machine conversations.

Method	Dataset			
	Alexa		Self-Dialogue	
	Accuracy	F1	Accuracy	F1
FastText [16]	54.54	58.34	79.21	79.32
ADAN [63]	62.01	66.10	46.64	59.66
VDCNN [34]	46.48	48.56	79.98	80.61
ConCET **(S)**	68.75 (+10.9%)	68.73 (+4.0%)	**84.58 (+5.7%)**	**84.71 (+5.1%)**
ConCET **(P)**	**71.46 (+15.2%)**	**71.72 (+8.5%)**	84.59 (+5.7%)	84.66 (+5.0%)

Table 4.11: Topic classification on Alexa and Self-Dialogue datasets, where (S) stands for Spotlight entity linker and (P) stands for the domain-specific PMI-EL entity linker. The relative improvements over ADAN and VDCNN are shown on the Alexa and Self-Dialogue datasets, respectively.

4.3.20 Detailed Performance Analysis

ConCET is a complex model consisting of different steps built based on deep learning models like CNN and RNN. I performed a comprehensive feature ablation analysis to evaluate the effect of each subsection on the overall performance of the system.

Entity linker evaluation

While entity linking is not the focus of this dissertation, since entities and their types play a central role in our approach, entity linking performance could have a significant effect on the overall classifier performance. To quantify the downstream effects of the entity linking accuracy, and to understand whether ConCET can operate with inaccurate entity linkers, I manually annotated entity-types for 350 utterances, which contained entities spotted by at least one entity linker. The distribution over classes is similar to that indicated in Table 4.24, with a higher number of utterances from *Movies*, *Music*, and *Travel_Geo* compared to the other classes. Table 4.12 presents the accuracy and F1 values of PMI-EL and Spotlight on different classes of utterances.

The two entity linkers exhibit comparable performance, with PMI-EL showing higher Accuracy on the *Movies*, *Music*, *Travel_Geo*, and *News* topics, but DBpedia Spotlight exhibiting higher overall F1 scores. As I will show later in this section,

Class	Entity Linker			
	PMI-EL		Spotlight	
	Accuracy	F1	Accuracy	F1
Movie	**80.00**	77.19	71.83	78.46
Travel_Geo	**80.77**	**87.50**	75.47	82.47
Music	65.51	59.37	64.44	**72.5**
Sports	70.56	63.16	**84.00**	**91.30**
News	**76.47**	**78.78**	66.66	70.59
Others	**53.68**	54.84	50.69	**62.12**
Overall	**68.30**	68.18	63.48	**72.67**

Table 4.12: Accuracy and F1 scores of entity detection by PMI-EL and DBPedia Spotlight entity linkers.

ConCET can perform well with either entity linker.

Impact of textual representation

To evaluate the impact of the textual representation choices, I conducted a feature ablation study. Table 4.13 summarizes the results, which indicate that all of the implemented components are significantly contributing to the final performance. Both Utt2Vec and TopicDist representations contribute to the classification performance, but the contributions are greater in Alexa dataset, due to a stronger correlation between the keywords with the user topics.

Method	Dataset			
	Alexa		Self-Dialogue	
	Accuracy	F1	Accuracy	F1
CNN	47.59	42.93	79.61	79.73
CNN+$BiLSTM_{pos}$	51.60	48.14	82.82	82.75
	(+8.4%)	(+12.1%)	(+4.0%)	(+3.8%)
CNN+$BiLSTM_{char}$	52.40	48.65	83.12	83.01
	(+10.1%)	(+13.3%)	(+4.4%)	(+4.1%)
Utt2Vec	54.27	50.84	83.33	83.35
	(+14.0%)	(+18.4%)	(+4.6%)	(+4.5%)
Utt2Vec+TopicDist	55.88	53.09	83.45	83.75
	(+17.4%)	(+23.6%)	(+4.8%)	(+5.0%)

Table 4.13: Topic classification Accuracy and F1 for different textual representations Alexa and Self-Dialogue datasets.

Impact of entity-type representation

Our model utilizes two variants of entity-type representations, namely entity-type distribution (TypeDist) and entity-type sequence modeling (Ent2Vec). I evaluate both entity representation vectors separately on both Alexa and Self-Dialogue datasets. Moreover, I report the result when different combinations of the entity representations are joined with the Utt2Vec network. Table 4.14 reports the contribution of each entity representation to the final performance. While both representations contribute greatly to the classifier performance, the effects are greater in the Alexa dataset, due to the strong correlation between the entity-types and the user topics of interest.

Method	**Dataset**			
	Alexa		Self-Dialogue	
	Accuracy	F1	Accuracy	F1
Utt2Vec	54.27	50.84	83.33	83.35
Ent2Vec	26.93	19.93	52.45	50.32
TypeDist	33.73	25.54	58.95	57.00
Ent2Vec+TypeDist	35.66	26.33	60.22	57.91
Utt2Vec+Ent2Vec	60.26	57.93	84.48	84.83
	(+11.3%)	(+14.6%)	(+1.4%)	(+1.5%)
Utt2Vec+TypeDist	63.46	61.03	84.43	84.71
	(+17.0%)	(+20.0%)	(+1.4%)	(+1.6%)
Utt2Vec+TypeDist+Ent2Vec	64.80	61.59	84.51	84.86
	(+17.4%)	(+23.6%)	(+1.4%)	(+1.8%)

Table 4.14: Ablation study for different entity representations.

Impact of synthetic dataset on ConCET

To evaluate the effectiveness of the synthetic dataset, I augmented the Alexa and Self-Dialogue datasets using the synthetic data described above and re-trained the models. The results are reported in Table 4.15. Even though the synthetic dataset is effective in the real human-machine conversations with Alexa, it has a negligible impact on the Self-Dialogue dataset. I attribute this effect to the large size of the Self-Dialogue dataset. I argue that even a portion of this dataset is enough for a

Train On	Dataset			
	Alexa		Self-Dialogue	
	Accuracy	F1	Accuracy	F1
Synthetic **(S)**	61.60	57.44	75.62	75.52
Synthetic **(P)**	62.93	63.83	58.73	59.03
Alexa data **(S)**	64.81	61.92	-	-
Alexa data **(P)**	62.93	60.24	-	-
Alexa data+Synthetic **(S)**	68.75	68.73	-	-
	(+6.1%)	(+10.7%)	-	-
Alexa data+Synthetic **(P)**	**71.46**	**71.72**	-	-
	(+13.5%)	(+19.0%)	-	-
Self-Dialogue **(S)**	-	-	**84.61**	**85.86**
Self-Dialogue **(P)**	-	-	84.55	84.71
Self-Dialogue+Synthetic **(S)**	-	-	84.58	84.71
	-	-	(-0.0%)	(-1.3%)
Self-Dialogue+Synthetic **(P)**	-	-	84.59	84.66
	-	-	(-0.0%)	(-1.4%)

Table 4.15: Performance of ConCET with and without training on the synthetic dataset, where "S" stands for the Spotlight entity linker and "P" stands for domain-specific PMI-EL entity linker.

model to reach its asymptotic performance. To evaluate this hypothesis, I re-trained ConCET in two different settings. First, I randomly sampled 1% of Self-Dialogue dataset and used it as the training set. Then, I added the synthetic dataset to the sampled portion and trained the model again. In the former case, ConCET reached the Accuracy of (72.01 ± 0.1), while in the latter case it reached the Accuracy of (73.12 ± 0.09). I performed each experiment 5 times. This confirms that the size of the labeled dataset is indeed affecting the extent to which the synthetic data can be helpful. I conducted an experiment to determine an estimate for the value of ρ using DBPedia Spotlight as the entity linker. The results are shown in Figure 4.5, which indicate that a value of 400K samples is appropriate for ρ in Algorithm 1, due to the classifier peaking at this point with more than 61% Accuracy.

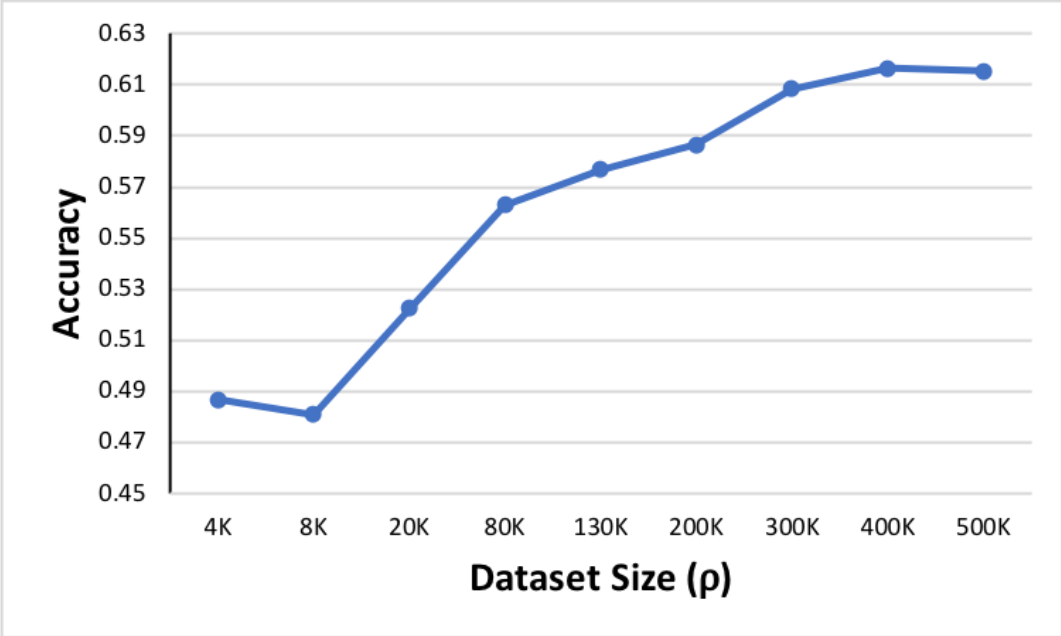

Figure 4.5: ConCET Accuracy on Alexa Prize dataset for varying ρ values in Algorithm 1 [3].

4.4 Contextual Dialogue Act Classification for Open-Domain Conversational Agents

Classifying the general intent of the user utterance in a conversation, also known as Dialogue Act (DA), e.g., open-ended question, statement of opinion, or request for an opinion, is a key step in NLU for conversational agents. While DA classification has been extensively studied in human-human conversations, it has not been sufficiently explored for the emerging open-domain automated conversational agents. Moreover, despite significant advances in utterance-level DA classification, full understanding of dialogue utterances requires conversational context. Another challenge is the lack of available labeled data for open-domain human-machine conversations. To address these problems, I propose a novel method, CDAC (Contextual Dialogue Act Classifier), a simple yet effective deep learning approach for contextual dialogue act classification. Specifically, I use transfer learning to adapt models trained on

human-human conversations to predict dialogue acts in human-machine dialogues. To investigate the effectiveness of our method, I train our model on the well-known Switchboard human-human dialogue dataset, and fine-tune it for predicting dialogue acts in human-machine conversation data, collected as part of the Amazon Alexa Prize 2018 competition. The results show that the CDAC model outperforms an utterance-level state of the art baseline by 8.0% on the Switchboard dataset, and is comparable to the latest reported contextual DA classification results. Furthermore, our results show that fine-tuning the CDAC model on a small sample of manually labeled human-machine conversations allows CDAC to more accurately predict dialogue acts in real users' conversations, suggesting a promising direction for future improvements

In summary, our contributions are twofold: (1) development of a novel context-aware Dialogue Classification model, CDAC, for open domain human-machine conversations; (2) demonstrating promising results after fine-tuning CDAC trained on human-human conversations to human-machine conversations, which is a necessary step for intelligent open-domain conversational agents.

4.4.1 Contextual Dialogue Act Classifier (CDAC) Model

In this section, I describe our proposed method, CDAC, for contextual dialogue act classification. First, I describe the features used to represent user utterances, and the conversation and system context. Then, I present the CDAC model architecture, implementation, and training details.

Content and Context Representation

For individual utterance representation, I use both word embedding, and surface (lexical and syntactic) features, described next. The word representation weights are initialized using pre-trained Word2Vec embeddings (300 dimensions), and updated

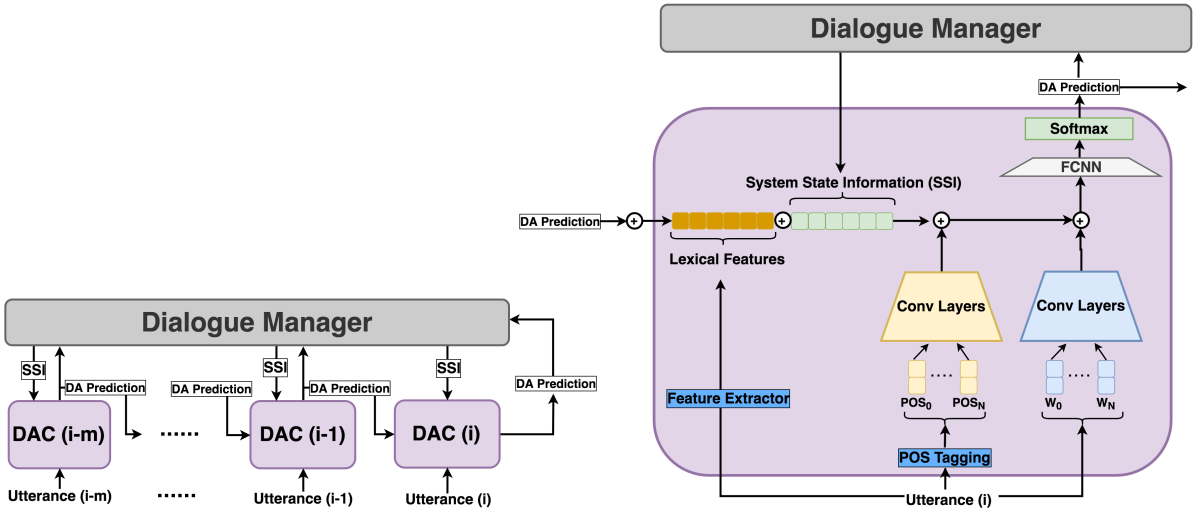

Figure 4.6: Contextual Dialogue Act Classifier (CDAC) architecture overview (left), where "DAC" is the dialogue act classifier for individual utterances (shown in detail on right). The extracted features, system state features, and details of the convolution (Conv) layers are described in detail in (Section 4.4.1) [31].

during training. The embedding-based utterance representation is augmented using three types of features: 1) lexical; 2) syntactic; 3) system state information (SSI), summarized below.

Lexical Features	**Short Description**
F_1 - *Word Count*	$\text{Word}_{\text{count}}$ in utterance
F_2 - *Char Count*	$\text{Char}_{\text{count}}$ in utterance
F_3 - *Sentence Count*	$\text{Sentence}_{\text{count}}$ in utterance
F_4 - *Average Word Count*	Average $\text{Word}_{\text{count}}$ in utterances
F_5 - *Average Char Count*	Average $\text{Char}_{\text{count}}$ in utterances
F_6 - *IsQuestion*	Binary feature to check for "?"
Syntactic & SSI Features	**Short Description**
F_7 - *POS Tagging*	Part of speech tags
F_8 - *Topic Distribution*	Topic distribution vector
F_9 - *Suggested Topic*	Suggested topic to user
F_{10} - *Suggested Item*	Suggested entity to user
F_{11} - *Speaker Id*	ID assigned to each user

The CDAC model combines individual Dialogue Act Classifiers (DACs) for each utterance. Each DAC uses textual features as well as syntactic features, i.e., part of speech tags, are modeled using a convolutional layer pipeline, as shown in Figure 4.6

(right). Each "Conv Layers" in Figure 4.6 is a 3-layer CNN with kernels of size 1, 2, and 3, with 100 filter maps for each kernel. Each layer is implemented using a 2D-convolution followed by a max-pooling, batch normalization, and relu activation function. To implement the batch normalization, I used a momentum of $M = 0.997$ and an epsilon of $\epsilon = 1e - 5$. Then, the output of both pipelines (syntactic and word embeddings) for each utterance representation, and the lexical features, as well as SSI features, are combined through a Fully Connected Neural Network layer (FCNN) with the size of 100. A dropout rate of 0.5 is applied at the FCNN layer to prevent the model from overfitting to the limited training data. Finally, softmax activation is used to obtain the final multi-class DA distribution. For training, categorical cross-entropy loss is minimized using Adam optimizer, with a $\alpha = 1e - 3$ learning rate and mini-batch size of 64.

The full Contextual DA Classifier model, CDAC, uses the DAC predictions from up to m previous turns in the conversation, where m effectively controls the scope of the conversation context. These DA predictions for the previous turns are appended to the current utterance representation vector through the FCNN layer, as shown in Figure 4.6 on right.

The window size m and other hyper-parameters above, largely follow the previous literature. Overall, our CDAC architecture and implementation choices builds on the ideas from the recent state of the art models for DA classification on the Switchboard dataset, while keeping the model simple and inexpensive to train.

Utterance	**Response**	**Topic**	**PS**	**ST**	**DA**
Alexa, let's chat	Nice to meet you! Do you want to talk about music?	*Music*	*NONE*	*Music*	**fp:** Opening
I want some recent music from Drake	How about God's plan by Drake?	*Music*	*Open*	*NONE*	**sv:** Statement-opinion
I really like that song!	You're welcome. Do you want some more information about Drake?	*Music*	*Music*	*NONE*	**sv:** Statement-opinion
Sure, sounds good	Drake is a Canadian rapper with 30 million followers on Spotify. Do you want some news about him?	*Music*	*Music*	*News*	**aa:** Accept
I'm good, tell me some fun dog facts	Sure, do you want to know why dogs have wet noses?	*Animal*	*Music*	*NONE*	**qo:** Open-question
Yes please	Dog's wet noses are for absorbing scent chemicals! Do you want another one?	*Animal*	*Animal*	*NONE*	**aa:** Accept
Do you have a dog?	Unfortunately, I do not have one. What do you want to chat next?	*Phatic*	*Animal*	*NONE*	**qy:** Yes-No-Question
Can you tell me about penguins?	Sure, do you want to hear about the fastest penguin species?	*Animal*	*Animal*	*NONE*	**qo:** Open-Question
Okay, that sounds interesting	Gentoo penguins can reach swimming speed up to 22 mph. Do you want more about penguins?	*Animal*	*Animal*	*NONE*	**aa:** Accept
No, I'm done chatting for today	Nice to talk to you, good bye!	*Phatic*	*Animal*	*NONE*	**fc:** Closing

Table 4.16: An example human-machine conversation, where "Topic" is current system state (topic), "PS" is the previous system topic, "ST" is the system-initiated topic suggestion, if any, and "DA" is the manually chosen Dialogue Act label.

System State Information (SSI) Features

I hypothesized that modeling context in human-machine conversations is similar to human-human conversations, but with a major difference that the system state information (SSI), unlike the state of a human, can be directly captured and represented as features. I incorporate SSI features alongside the utterance and context representation features for each *turn* of the conversation. The SSI features include system topic distribution, the suggested topics (e.g., "Music"), and suggested items (e.g., specific artists). To encode topic distribution features, I used one-hot encoding, while specific items are represented using Word2Vec word embeddings of the words in the item names.

Note that the SSI features were not used or available for human-human conversations.

Transfer Learning from Human-Human Conversations: To fine-tune the CDAC model from human-human to human-machine conversations, all the weights in the CDAC model are first trained on the human-human Switchboard dataset. Then, all of the network weights are tuned using the Alexa Prize data, but with a smaller learning rate of $\alpha = 1e - 4$.

4.4.2 Experimental Setup

I now introduce the human-human (Switchboard) and human-machine (Alexa Prize) conversation datasets used for training and evaluating CDAC. Then, I explain the human annotation procedure to obtain ground truth labels for the human-machine conversations, followed by the experimental design and metrics.

Switchboard Dataset (Human-Human Conversations)

Switchboard DA corpus [60] is a well-known telephone speech corpus, which contains 42 main DA labels[21]. The Switchboard corpus contains two official splits: the training split with 1,115 conversations and 196,258 utterances, and the test split, with 40 conversations and 7535 utterances.

Alexa Prize 2018 Dataset (Human-Machine Conversations)

I collected the human-machine conversation data during the Amazon Alexa Prize 2018. 200 conversations of real users with our open-domain conversational agent, containing more than 3,000 utterances were randomly selected. Table 4.21 shows an example conversation[22] of a hypothetical (not real) user with the actual system responses. Two different human annotators were asked to manually label two hundred conversations in the human-machine Alexa prize data. The inter-annotator agreement was 0.790, and Kappa was 0.755, indicating strong agreement between the annotators. For the final ground truth label values, in case of disagreements, the label was randomly chosen between the two annotator labels. The distribution of annotated DAs[23] is reported in Table 4.17. The top four most frequent dialogue acts observed are Agree/Accept (aa), Conventional Opening (fp), Reject (ar), and Statement Opinion (sv), accounting for over 68.2% of the user utterances.

Switchboard Experimental Design

For CDAC model training, the training conversations were split into 1,000 for training and 115 for validation, leaving the test split untouched during training. For testing, some of the previous studies [120, 13, 78] used only 19 conversations of the available

[21] *https://github.com/cgpotts/swda*

[22] This is a representative conversation between the authors and the real system, since user utterances from live system deployment cannot be reported to protect user privacy.

[23] See *http://compprag.christopherpotts.net/swda.html for full description of DA labels.*

DA	Frequency	DA	Frequency
aa	655 (21.7%)	*fp*	501 (16.6%)
ar	478 (15.8%)	*sv*	425 (14.1%)
qo	227 (7.5%)	*fc*	198 (6.6%)
sd	154 (5.1%)	*b^m*	114 (3.8%)
no	107 (3.5%)	*qw*	80 (2.7%)
qy	48 (1.6%)	%	24 (0.8%)
ft	7 (0.2%)		

Table 4.17: Dialogue Acts (DA) frequency distribution in user utterances in the Alexa Prize dataset.

40 test conversations. Instead, I follow the convention of Liu et al. [89] and use all 40 test conversations for evaluation. The main baseline model for this experiment is [120], based on a hidden Markov model, as it remained an effective method for more than 10 years. Other reported results are from the three recent DA classifiers described in references [18, 13, 78] respectively.

Alexa Prize Experimental Design

For the Alexa prize dataset experiment, Support Vector Machines (SVM) and Multinomial Bayes models are selected as baseline models, using lexical features (words) with tf-idf term weights due to simplicity and low requirements for labeled training data. Contextual features such as previous utterances and system state features are appended to the bag of words feature vector for each utterance. 5-fold cross-validation was used, where 4 folds were used for tuning the weights, and the last fold for the prediction. Finally, following the conventions of the DA classification literature, the main evaluation metric was **overall (micro-averaged) Accuracy**.

4.4.3 Results and Discussion

In this section, I report the overall accuracy of CDAC in comparison to previous baselines on Switchboard and Alexa data. Feature ablation and error analysis are also reported to provide insights into CDAC system performance.

DA Prediction Results on Switchboard Data

Our main results on the Switchboard dataset are summarized in Table 4.18. CDAC improves the baseline model [120] by 8.0%. Moreover, compared to the best known contextual model, I reach comparable results with a more general and simple model. Context window of size 3 yields the strongest performance. However, our results are on all 40 conversations in Switchboard dataset, while [18] used only 19 of the text conversations. I were unable to replicate the model and results reported in reference [18], due to the required model complexity and not having access to a working implementation or sufficient details to reproduce their exact system. In contrast, our proposed model is simpler, while producing state-of-the-art DA classification accuracy on the standard benchmark of human-human conversations.

Methods	Accuracy
Baseline Stolcke et al. [120]	71.00
Previous state of the art methods	
Kalchbrenner et al. [13]	73.90
Young et al. [78]	73.10
Bothe et al. [18]*	**77.34 (+8.9%)**
CDAC-2	76.40
CDAC-3	**76.70 (+8.0%)**
CDAC-4	76.51
Annotator Agreement	84.00

Table 4.18: DA classification Accuracy (micro-averaged) on Switchboard dataset, where (*) represents the latest reported contextual DA classification[18]. CDAC-2,-3, and -4 stands for the model using on contexts of size 2, 3 and 4 turns, respectively.

DA Prediction Results on Alexa Prize dataset

Table 4.19 summarizes the baseline and CDAC performances on the human-machine Alexa Prize conversation data. CDAC outperforms traditional classification models such as SVM and Multinomial Bayes (without context) by about 22.7%. Furthermore, by adding context to SVM and Multinomial Bayes, there are 1.8% and

4.5% improvements over the respective baselines. Encouragingly, by pre-training the CDAC model on human-human Switchboard data, and then fine-tuning on the (limited) labeled human-machine data, CDAC achieves an additional 2.7% improvement. It is important to note that despite annotating for only 13 most common DA classes observed in the human-machine conversation data, compared to the 42 classes in Switchboard human-human data, DA performance degrades on human-machine conversations. This confirms our observation that human-machine DA classification is a more challenging task than for human-human conversations.

Methods	Accuracy
Without context	
Multinomial Bayes	58.21
SVM	65.73
With context	
Multinomial Bayes	59.26
SVM	68.70
CDAC	**73.25*(+6.6%)**
CDAC + Transfer Learning	**75.34*(+9.6%)**

Table 4.19: DA prediction micro-averaged Accuracy on Alexa dataset with and without context information (size 3 turns), where (*) represents significance levels of $p < 0.05$.

Feature Ablation and Error Analysis

Table 5.9 summarizes the change in accuracy by systematically removing feature sets on the Alexa prize human-machine conversation data. Both lexical and syntactic features are important for DA classification since removing either group decreased the accuracy. Interestingly, the most common error is distinguishing between statement-opinion and open-question labels. For instance, the utterance "I like to talk about animals" is challenging to classify, since without context, it is difficult to determine whether a user expressed an opinion, or requested information from the system. Knowing the context and the system state can enable such disambiguation.

Syntactic Features	Lexical Features	Accuracy
-	-	73.94 (-1.64%)
-	✓	74.80 (-0.50%)
✓	-	74.91 (-0.35%)
✓	✓	**75.18**

Table 4.20: Feature ablation on CDAC with context window size 1. - and ✓indicates features removed and added respectively.

4.5 Knowledge-Aware Contextual Topic Suggestion for Open-Domain Conversational Agents

To hold a true conversation, an intelligent agent should be able to occasionally take initiative and recommend the next natural conversation topic. This is a challenging task. A topic suggested by the agent should be relevant to the person, appropriate for the conversation context, and the agent should have something interesting to say about it. Thus, a scripted, or one-size-fits-all, popularity-based topic suggestion is doomed to fail. Instead, I explore different methods for a personalized, contextual topic suggestion for open-domain conversations. I formalize the Conversational Topic Suggestion problem (CTS) to more clearly identify the assumptions and requirements. I also explore three possible approaches to solve this problem: (1) model-based sequential topic suggestion to capture the conversation context (CTS-Seq), (2) Collaborative Filtering-based suggestion to capture previous successful conversations from similar users (CTS-CF), and (3) a hybrid approach combining both conversation context and collaborative filtering. To evaluate the effectiveness of these methods, I use real conversations collected as part of the Amazon Alexa Prize 2018 Conversational AI challenge. The results are promising: the CTS-Seq model suggests topics with 23% higher accuracy than the baseline, and incorporating collaborative filtering signals into a hybrid CTS-Seq-CF model further improves recommendation accuracy by 12%. Together, our proposed models, experiments, and analysis significantly advance

the study of open-domain conversational agents, and suggest promising directions for future improvements.

While the art of conversation can be considered a uniquely human trait [126], "artificial" conversational intelligence (Conversational AI) agents have been gaining traction, especially with the recent series of Amazon Alexa Prize Challenges providing a competition platform and monetary incentives to spur development [114, 64]. Many practical applications of conversational agents have been proposed, e.g., for companionship to improve mental well-being, (e.g., [104]) and for therapy (e.g., [44]). Open-domain conversational agents can also be used in educational settings as tutors and evaluators, as proposed in [45]. While much room for improvement remains in the current implementations of the conversational AI systems, the potential for intelligent, empathic, and broad-coverage conversational agents is widely recognized.

However, for an open-domain conversational agent to be coherent and engaging, it must be able to drive the conversation to the next topic, and in a way that does not appear scripted. This task is complicated. As for many realistic and complex tasks, extensive knowledge engineering is needed for in-depth domain-specific capabilities, usually handled by specialized components. For a user to remain engaged, the overall conversational AI system should be able to recommend the next conversation topic (or component) in a natural and coherent fashion. Appropriate topic recommendations are also critical to expose the capabilities of the system to the user, who otherwise may not know that a conversational agent is an expert in particular topics like sports, cars, or video games.

Yet, the right topic to recommend depends on both prior user interests and the conversation context. Extensive work has been done in topic and content recommendation using content-based [91] and collaborative filtering methods [101, 71]. However, it is unclear just *how* to adapt recommender system techniques to the conversational setting. In open-domain conversational agents, the recommended items might be

agent's sub-components with dynamically changing content and interactions underneath, specific items, general suggestions, or even clarification questions. In this paper, I extend model-based and collaborative filtering recommendation algorithms for topic and content recommendation in the conversational setting. Our contributions are threefold: (1) Formalizing the conversational topic suggestion problem for open-domain conversational agents; (2) Development of a sequential contextual topic suggestion model for this task; (3) Empirical exploration of the effectiveness of model-based, collaborative filtering-based, and hybrid approaches to topic suggestions in the conversational setting. Our experiments indicate the value of our proposed hybrid recommendation solution, highlight the challenges and opportunities inherent in the conversational topic recommendation, and suggest promising directions for research.

Turn	User Utterance	System Response	Topic	PS	PST	Labels
1	Alexa, let's chat.	Hi! What would you like to talk about?	*Music*	None	*None*	*chat*
2	Tell me recent songs.	Circles by Post Malone, Talk by Khalid, and Bad Guy by Billy Eilish are currently popular. Would you like to talk about Billy?	*Music*	*Opening*	*Music*	*Music_accept*
3	No I do not.	Alright. How about some info on Khalid?	*Music*	*Music*	*Music*	$follow-up$
4	Oh, no.	Ok, do you want to hear about some places to travel?	*Phatic*	*Music*	*Music*	*chat*
5	I love traveling.	Cool! Which country do you want to visit?	*Travel*	*Music*	*Travel*	*Travel_accept*
6	Somewhere in Australia.	Australia is beautiful, especially the beaches. Do you like beaches?	*Travel*	*Travel*	*Travel*	$follow-up$
7	Yes.	I recommend Bondi Beach, it is amazing. Wanna here more about it?	*Travel*	*Travel*	*Travel*	$follow-up$
8	No thanks, let's talk about something else.	Sure, would you like to discuss the news?	*Phatic*	*Travel*	*Travel*	*chat*
9	No, news is boring.	Sure, Do you want to talk about recent movies?	*Phatic*	*Phatic*	*News*	*News_reject*
10	Okay, that sounds interesting.	Cool, which genre do you prefer, comedy or drama?	*Movies*	*Phatic*	*News*	*Movie_accept*
11	I like both.	How about The Favourite by Yorgos Lanthimos?	*Movies*	*Movies*	*Movies*	$follow-up$
12	I have to go Alexa, bye!	Nice talking to you, good bye!	*Phatic*	*Movies*	*Movies*	*chat*

Table 4.21: A Conversation example, where PS and PST refer to Previous State and Previous Suggested Topic, respectively.

Features	***Description***	***Example Values***
Topic and Behavior		
F_1 – F_8	1-hot encoding for user response for each topic on previous turn, where 1=Accepted, 0=not suggested, and -1=rejected	[+1,0,-1,0,0,0,0,0]
F_9 – F_{10} - Two previous topics	Two previous components that user engaged with	Movies, Music
F_{11} – Previous accepted topic	Previous suggestion that was accepted by the user	Music
F_{12} – Previous rejected topic	Previous topic that was rejected by users	Pets_Animals
User Profile		
F_{13} – Name	Does user give his/her name	True/False
F_{14} – Gender	What is the user's gender	M/F
F_{15} – Time	Time of the day during the conversation	Morning/Day/Evening/Night

Table 4.22: Dialogue manager state information features used for CTS recommendation. The values are computed up to turn i in the conversation so far.

4.5.1 Conversational Topic Suggestion (CTS): Problem Definition

I now define the conversational topic suggestion problem and introduce our proposed solutions in the following section.

Consider the example conversation in Table 4.21. While this is not a real user[24], the conversation is typical of those observed with our system during the Alexa Prize challenge. In a regular Alexa conversation, a user may have an initial interest or information need (e.g., "recent songs") which is handled by a particular system component (in this case, the *Music* component); however, the user might quickly lose his/her interest, and the system (conversational agent) must take the initiative to find the next topic of conversation that this user is likely to be interested in, for example, *Travel*. In the example conversation, the user accepts the suggestion to talk about the topic *Travel*, and a different system component starts interacting with the user to drive the conversation. The next suggested topic *News*, however, is not accepted by the user, and the system has to make another recommendation, which would degrade the user experience.

I define Conversational Topic Suggestion (CTS) as follows:

Setting:	Open-Domain mixed initiative conversation with a multi-component conversational agent.
Given:	A conversation C, consisting of a sequence of user utterances $U_{0..i}$, a sequence of system states $S_{0..i}$, and a set of possible conversation topics $t \in T$, (e.g., system components or **mini-skills**).
Problem:	At conversation turn i, select a topic t_i to suggest for the current user u, to maximize the likelihood of *acceptance* (i.e., the probability that user u would like to talk about the topic t_i next).

Figure 4.7: Definition 1: Conversational Topic Suggestion (CTS) Problem Statement.

[24]Exact user conversations cannot be reproduced due to Alexa Prize terms.

Note that this definition focuses on the acceptance of the topic suggestion, and does not explicitly consider the user's future satisfaction or engagement with the selected topic (e.g., as measured in reference [30]). I also emphasize that I formulate CTS based only on short-term history (conversation or session-level), and not on long-term user interests. Still, it is worth noting that in many practical situations, a conversational agent must make coherent topic suggestions for new (cold-start) or inactive users. Secondly, note that I do not require a set of other users (e.g., as would be required for collaborative filtering approaches for recommendation), which might be an attractive setting for privacy and security considerations. Finally, unlike in a traditional recommendation setting, a suggested topic represents a system component which, if accepted, begins interacting with the user dynamically, and thus cannot be easily mapped to a single *item* that can be easily represented and compared across users. Our models, described next, attempt to capture both the conversation context and the internal system information for this task.

4.5.2 CTS-Seq Approach

For relevant and coherent topic suggestions, it is necessary to consider the conversation context, e.g., the sequence of previous user utterances and system states. For example, if a user is talking about *Movies*, it might be more natural to suggest *Music* as the next topic, as opposed to *Cars*. Also, if a user declined to talk about *Movies* in the past, the system should not suggest this topic or a related topic like *Television* unless explicitly requested. For this reason, I propose to use a sequence modeling approach for the conversational topic suggestion. I will further show how this approach can be combined with more traditional collaborative filtering-based methods.

Before I describe the specific sequence models, I first discuss the conversation *features*, used for both sequence modeling and collaborative filtering-based methods.

4.5.3 System State and User Profile Features

To represent the conversation context, two different groups of features are extracted for each conversation turn, as summarized in Table 5.1. The first group is *Topic and Behavior* features, which represents the user's previous responses, i.e., the accepted and rejected topic suggestions. These features have the values of **1** for accepted topics, $-\mathbf{1}$ for rejected topics, and **0** for the topics that have not been proposed yet. This group of features is designed to prioritize the topics that have been accepted **1** or unexplored **0**. *Topic and Behavior* features also model topic classification features and the current conversation context and system state. These features could indicate the historical probability that the current state is a potential topic-switching point, or whether it should be a *follow-up* for the previous topic. The second group of features is *User Profile* features. They contain the inferred gender of the user [-1,1] based on the provided name, and whether they gave their name at the start of the conversation or not (a weak indicator of the user's openness to sharing information with the bot). Other features like age and location, which are often used for user profiling, are usually not available in the conversational setting. Table 5.1 shows different categories of features that are used in all the CTS-CRF, CTS-CNN, and CTS-RNN models. The values of these feature groups are computed for each conversation turn and stored in separate vectors, which are then concatenated to produce the full conversational state representation, fv, specifically:

$$fv = [F_1; F_2; ...; F_{15}] \tag{4.21}$$

I emphasize that these features will be used for all sequence and CF-based model variations, to explore the trade-offs in modeling, while keeping the actual features constant.

4.5.4 CTS-Seq: Models

In this section, I list three different implementations of the proposed CTS-Seq method. First, I describe the Conditional Random Fields (CRF) implementation. Then, I describe the CNN-based followed by the RNN-based implementation.

4.5.5 CRF Implementation of CTS-Seq: CTS-CRF

As the first and most straightforward implementation of CTS, I use the well-known and robust CRF model. CRF is an undirected graphical model, which estimates the conditional probability of a sequence of labels (tags) with respect to the observed features, and requires relatively small amounts of training data [76, 151].

Each conversation is represented as a sequence of turns, with observable features extracted from each utterance and system state. Recall that I represent a conversation j as a sequence of turns $Conv_{(j)} = [utt_{(1)}, ..., utt_{(i)}, ..., utt_{(n)}]$. Then, for each sequence of utterances, a sequence of labels (topics) $[t_1, ..., t_i, ..., t_n]$ is generated. The recommended topic t is modeled as the CRF hidden state, and X is the observed variable represented by the features described above. Thus, the CTS-CRF model aims to predict the most likely *next* topic $t_{(i+1)}$ after observing the first i conversation turns and system states.

More formally, Eq. 4.22 computes the probability of a topic t given the sequence of previous turns and topic decisions, where $Z(X)$ indicates the normalization factor and θ and η are weights that can be tuned using maximum likelihood estimation. Moreover, $f(t_i; X_t)$ and $g(t_i; t_{i-1}; X_t)$ jointly represent the next topic to predict, the context (previous topic) and the features for the current turn x.

$$p(t|x) \propto \frac{1}{Z(X)} \prod_{i=1}^{m} \exp^{\left\{ \sum_{j=1}^{m} \theta_j f_j(t_i; X_t) + \sum_{k=1}^{m} \eta_k g_k(t_i; t_{i-1}; X_t) \right\}} \tag{4.22}$$

The CRF-based implementation of CTS, CTS-CRF, is illustrated in Figure 4.8.

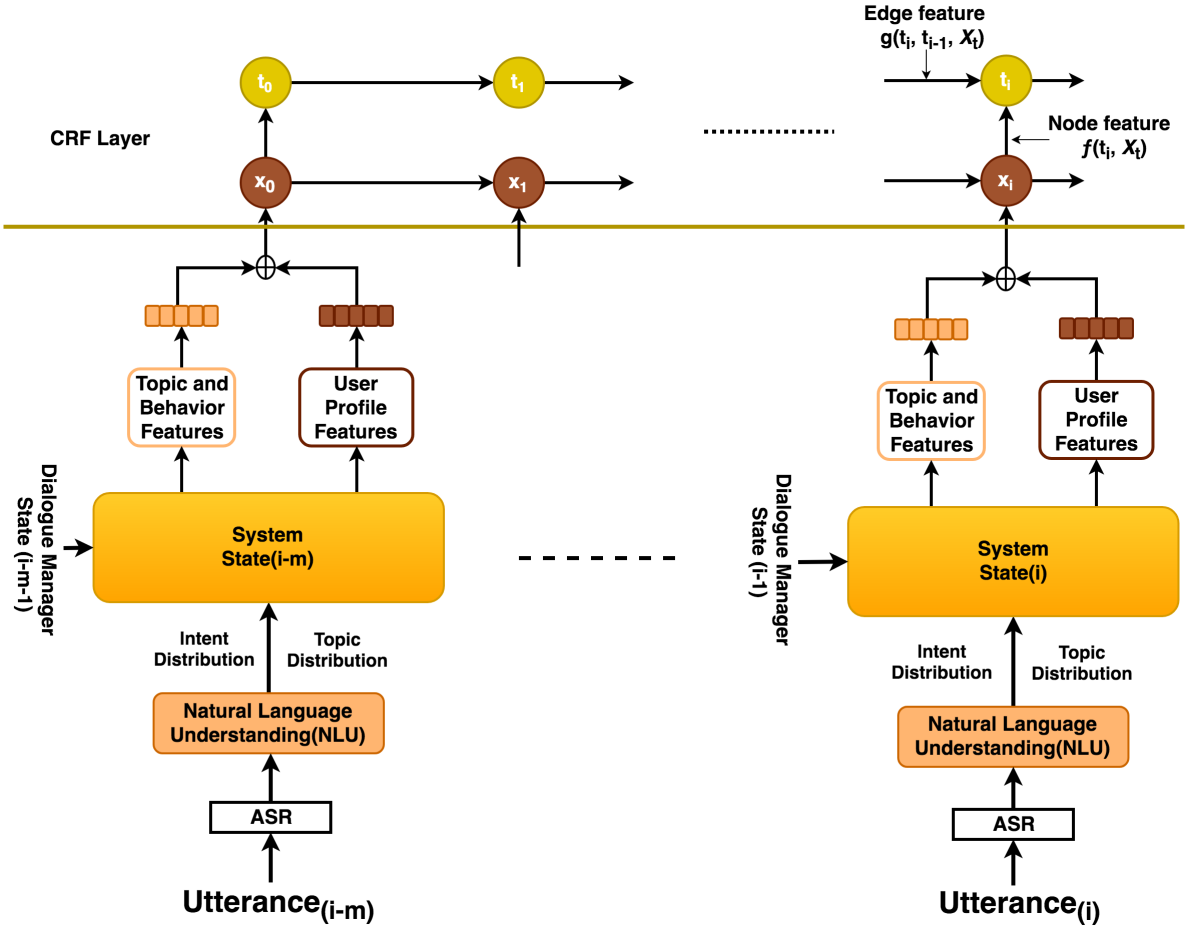

Figure 4.8: CTS-CRF topic suggestion model for conversation turn i. Feature details are reported in Table 5.1 and Section 4.5.3. ASR stands for authomatic speech recognition [5].

4.5.6 Deep-learning based implementation of CTS-Seq: CTS-CNN and CTS-RNN

Deep learning approaches such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have shown promising results for different natural language processing tasks, from text classification to dialogue act classification (e.g., [12, 155, 34, 23, 66, 13, 78]). Lee et al. [78] proposed a pipeline of deep learning methods to model a *sequence* of short texts. Inspired by [78], I propose two deep learning

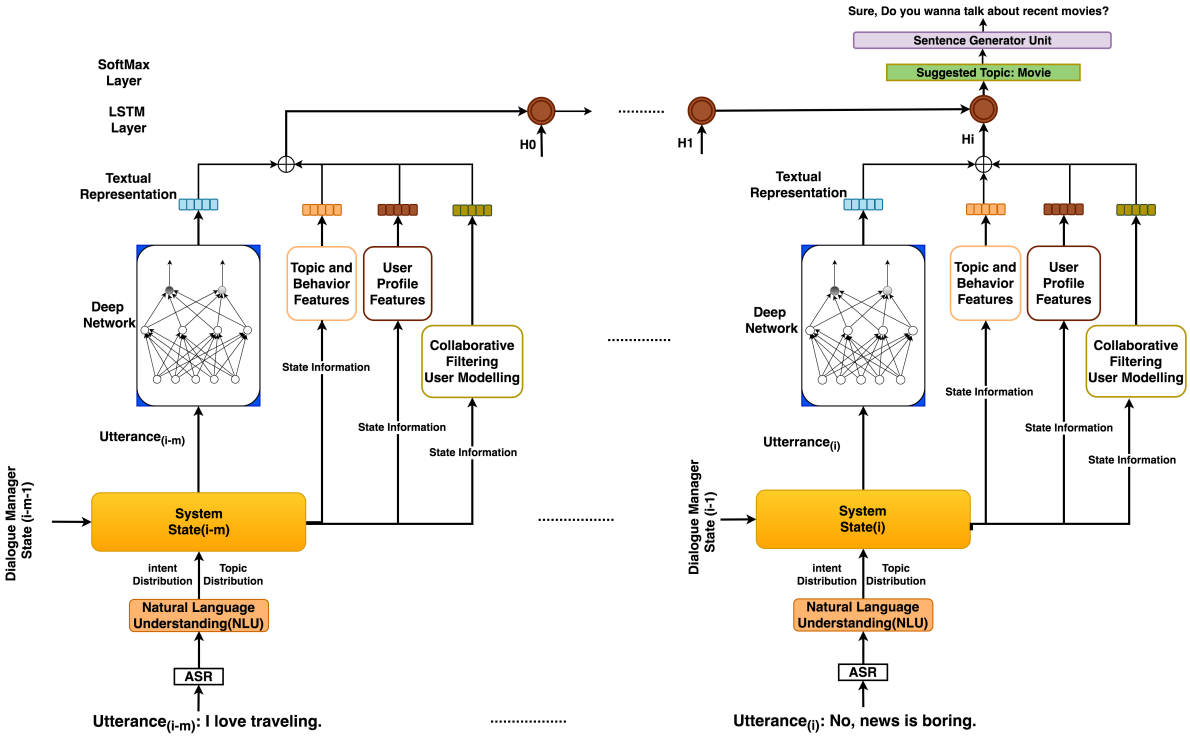

Figure 4.9: CTS-RNN-CF or CTS-CNN-CF model architecture, where *Topic and Behavior* features include the list of all previously suggested, accepted and rejected topics from the beginning of the conversation. *User Profile* features contain the list of Name, predicted Gender, and time of the day. CF features also include the suggested topic distribution extracted from the collaborative filtering model. Feature details are reported in Table 5.1. ASR stands for automatic speech recognition [5].

models for implementing CTS, namely CTS-CNN and CTS-RNN. CTS-CNN and CTS-RNN respectively use a CNN and a BiLSTM to incorporate textual and contextual evidence gathered so far from the conversation to recommend (predict) the next conversation topic.

CTS-CNN Implementation.

Here, I walk through different steps in the CTS-CNN model. CTS-CNN-CF network takes word tokens, from m consecutive utterances. $utt_{(i)}$ stands for the words in the

i-th utterance, where w_{ij} stands for j-th word in i-th utterance.

$$utt_{(i-m)} = [w_{(i-m)1}; w_{(i-m)2}; w_{(i-m)3} \ ... \ w_{(i-m)n}] \tag{4.23}$$

....

$$utt_{(i-1)} = [w_{(i-1)1}; w_{(i-1)2}; w_{(i-1)3} \ ... \ w_{(i-1)n}] \tag{4.24}$$

$$utt_{(i)} = [w_{i1}; w_{i2}; w_{i3} \ ... \ w_{in}] \tag{4.25}$$

I define a function f_c that takes an utterance as input and outputs the learned utterance representation y_{cnn}:

$$y_{(cnn_{(i)})} = f_c(utt_{(i)}) \tag{4.26}$$

f_c is a 3-layered CNN with max pooling, which is applied in parallel on all the utterances in a window of size m. The first layer is a word embedding layer with pre-initialized weights from Word2Vec vectors of size 300. The weights on the embedding layer are tuned during training using the cross-entropy loss function.

CTS-RNN Implementation.

CTS-RNN uses a BiLSTM network followed by an attention layer to model the utterance representation. In CTS-RNN, a function f_r is defined that takes an utterance as input and outputs a hidden representation $h_{(i)}$ for each utterance:

$$h_{(i)} = f_r(utt_{(i)}) \tag{4.27}$$

where f_r is a BiLSTM model with 256 hidden layers. It is also applied in parallel on a window of size m in the same way as for f_c. Then, the hidden representation for the i-th utterance is passed to an attention layer to generate the final representation

$y_{(rnn_{(i)})}$. Given the hidden representations of each timestamp of j in $LSTM_{(i)}$ is h_{ij}, dot product similarity score s_{ij} is computed based on a shared trainable matrix M_i, context vector c_i and a bias term b_{ij}. M_i, c_i and b_{ij} are initialized randomly and jointly learned during training. Softmax activation is applied on similarity scores to obtain attention weights α_{ij}. Lastly, using learned α_{ij}, a weighted sum on BiLSTM hidden representations is applied to obtain the output $y_{rnn_{(i)}}$ for the i-th utterance as follows:

$$s_{ij} = tanh\left((M_i)^{\mathrm{T}} h_{ij} + b_{ij}\right) \tag{4.28}$$

$$\alpha_{ij} = \frac{e^{(s_{\mathrm{ij}})^{\mathrm{T}} c_i}}{\sum_{j=1}^{n} e^{(s_{\mathrm{ij}})^{\mathrm{T}} c_i}} \rightarrow y_{(rnn_{(i)})} = \sum_{j=1}^{n} \alpha_{\mathrm{j}} h_{\mathrm{ij}} \tag{4.29}$$

Finally $y_{(rnn_{(i)})}$ is computed for every utterance located in the window.

Merging and FeedForward Layers

This step is similar for both CTS-CNN and CTS-RNN models, where the output of the textual representation of each utterance is merged with *Topic and Behavior*, and *User Profile* features. Here I describe all the details of these layers for the CTS-CNN model.

To create the final representation of a $w_{(i)}$ in a conversation, I extract $y_{(cnn_{(i)})}$ from all the utterances located in the window in parallel. Then, the window is fed to an LSTM network with 100 hidden states. I deploy an LSTM instead of an Bi-LSTM since in real conversation, there is not a backward signal. Finally, the output of the

last layer going through

$$w_{(i)} = [utt_{(i-m)}; ...; utt_{(i-m+j)}; ...; utt_{(i)}] \tag{4.30}$$

$$rep_{(w_{(i)})} = [y_{(cnn_{(i-m)})}; ...; y_{(cnn_{(i-m+j)})}; ...; y_{(cnn_{(i)})}] \tag{4.31}$$

$$output = LSTM\Big(rep_{(w_{(i)})}\Big) \tag{4.32}$$

Where, in this equation $j < m < i$. The final output is fed to a feed forward layer of size 256 with a dropout rate of 0.5. A softmax function $f(s)$ is applied to generate a probability distribution over C possible topics. The network was trained with an Adam optimizer with a learning rate of 0.001 using the softmax cross-entropy loss function CE. C is the number of classes, t_i is the one-hot representation of the target label, and s_i are the scores inferred by the model for the i-th class:

$$f(s_i) = \frac{e^{s_i}}{\sum_{i=j}^{C} e^{s_j}} \rightarrow CE = -\sum_{i=1}^{C} t_i \log\left(f(output)\right)) \tag{4.33}$$

I summarize the parameters of the (CNN-) and (RNN-) based models in Table 4.23. The parameters were not tuned and were chosen based on our experience and previous literature.

4.5.7 Hybrid Sequential and Collaborative Filtering: CTS-Seq-CF

I now introduce our new model, CTS-Seq-CF, which augments the sequence modeling approach described above, with additional signals extracted from other users' experiences using collaborative filtering. The proposed models, CTS-CNN-CF and CTS-RNN-CF, incorporate the probability of acceptance of each topic based on similar users' preferences (I describe the collaborative filtering methods used in Section

Parameters	Values
Pooling type	Max-pooling
L2-regularization	0.001
Word embedding length	300
Momentum	0.997
Epsilon	1e-5
Learning rate	0.001
Dropout	0.5
Batch Size	64
Number of layers (CNN)	3
Number of filters(CNN)	128
Filter sizes (CNN)	1,2,3
Hidden state (RNN)	100
Feed forward layer (RNN)	256

Table 4.23: Detailed configuration parameters of the CTS-CNN and CTS-RNN implementations.

4.5.10.) as features into the CTS-CNN and CTS-RNN models, respectively. One of the resulting hybrid models, CTS-RNN-CF, is illustrated in Figure 4.9. CTS-CNN-CF follows the same pattern as CTS-RNN-CF, with the semantic utterance representation being generated using a CNN model. They aggregate contextual evidence from the preceding states by considering a window of size m for each turn. Then, all the system state information for the turns in that window is extracted, which includes all the features in Table 5.1, as well as suggested topic distribution predicted by the collaborative filtering method described in Section 4.5.10. Finally, all the utterance vector embeddings within the window are concatenated with them to form the window vector embedding. Here, I walk through the details for CTS-RNN-CF.

$y_{(rnn_{(i)})}$ is first concatenated with $fv_{(i)}$ to obtain the enriched representation $rep_{(rnn_{(i)}+fv_{(i)})}$ of an utterance. Then, I concatenate them with the CF features, which were extracted by collaborative filtering module to generate the final utterance representation $rep_{utt_{(i)}}$.

$$rep_{(cnn_{(i)}+fv_{(i)})} = [y_{(cnn_{(i)})}; fv_{(i)}] \tag{4.34}$$

$$rep_{(utt_{(i)})} = [rep_{(cnn_{(i)}+fv_{(i)})}; CF_{(i)}] \tag{4.35}$$

To create the final representation of a w_i in a conversation, I extract $rp_{(utt_{(i)})}$ from all the utterances located in the window in parallel. Finally, all the outputs are concatenated together to form the final vector.

$$w_{(i)} = [utt_{(i-m)}; ...; utt_{(i-m+j)}; ...; utt_{(i)}] \tag{4.36}$$

$$rep_{(w_{(i)})} = [rep_{(utt_{(i-m)})}; ...; rep_{(utt_{(i-m+j)})}; ...; rep_{(utt_{(i)})}] \tag{4.37}$$

$$output = LSTM\Big(rep_{(w_{(i)})}\Big) \tag{4.38}$$

Then, $rep_{(w_{(i)})}$ is fed to an LSTM network with 100 hidden states, later the output of the last layer going through a feed-forward layer followed by a softmax layer as described in Section 4.5.6.

4.5.8 Experimental Setup

I now describe the baselines, data, metrics, and experimental procedures used to evaluate our proposed conversational topic suggestion models.

4.5.9 Baseline 1: Popularity Method

The Popularity method is a heuristic method, which suggests the next conversation topic based on overall frequency (popularity) in previous conversation data and previous user ratings, and the approximate time of day. The order of suggestion is *Movies*, followed by *Music*, followed by *Video Games* or *Travel* or *Animals* depending on the time of day to accommodate the expected differences in user demographics. This

heuristic popularity baseline was deployed during the Alexa Prize competition [1].

4.5.10 Baseline 2: Collaborative Filtering (CF)

I adapt the classical approach of CF, originally introduced for item recommendation to the conversational setting using the K-Nearest Neighbors (KNN) model. Each user is represented by the following features:

- User and time features: F_{13}, F_{14}, F_{15} from Table 5.1.
- Suggestion acceptance and rejection rate: The fraction of topic suggestions accepted by the user and the fraction rejected by the user.
- Topical features: F_1 - F_8 from Table 5.1.

For each conversation turn, the feature vector described above is calculated based on the conversation up to this turn. For example, if a user has accepted a suggestion to talk about *Movies* and rejected a suggestion for *Music*, the accept and reject rates would be 0.5. The topical feature vector would contain $\mathbf{1}$ for *Movies* and $-\mathbf{1}$ for *Music*, and then top k users with most similar conversation histories would be retrieved.

More formally,

$$\mathbf{U}_a = [F_1(a) : F_8(a), F_{13}(a) : F_{15}(a), r^{accept}(a), r^{reject}(a)] \tag{4.39}$$

$$sim(\mathbf{U}_a, \mathbf{U}_b) = \frac{\mathbf{U}_a \cdot \mathbf{U}_b}{||\mathbf{U}_a|| \times ||\mathbf{U}_b||} \tag{4.40}$$

$$pred(\mathbf{U}_a, T) = \frac{\sum_{\mathbf{U}_b \in N} sim(\mathbf{U}_a, \mathbf{U}_b) \times s_{(\mathbf{U}_b, T)}}{\sum_{\mathbf{U}_b \in N} sim(\mathbf{U}_a, \mathbf{U}_b)} \tag{4.41}$$

where $\mathbf{U}_a$ is the user who I are calculating the topic scores for, $\mathbf{U}_b$ is one of the neighbors from set N, which is the set of 33 nearest neighbors of $\mathbf{U}_a$, $r^{accept}(a)$ is the suggestion acceptance rate of user $\mathbf{U}_a$, $r^{reject}(a)$ is the suggestion rejection rate of

user $\mathbf{U}_a$, $s_{(\mathbf{U}_b,T)}$ indicates the score of topic T for user $\mathbf{U}_b$, and $pred(\mathbf{U}_a, T)$ represents the predicted score of a topic T for the active user.

For final classification, the predicted topic scores based on 33 nearest neighbors' preferences are fed to a feed-forward layer followed by a softmax layer, as described in Section 4.5.6.

4.5.11 Baseline 3: Contextual Collaborative Filtering: Contextual-CF

Contextual-CF utilizes the collaborative filtering signals extracted from the preceding utterances. Then, a fully connected neural network followed by a softmax is applied to combine the features and provide the final prediction result. To this end, I applied the CF model described in Section 4.5.10 to extract the suggested topics for all the utterances located in a window of size m. To represent the CF features, I considered a one-hot-vector, where the length of the one-hot-vector is equal to the number of available topics that are supported by the conversational agent. The value corresponding to the topic selected by the CF model is assigned as **1**. Then, the one-hot-vectors are concatenated together to create the final vector for the corresponding window. As a result, a vector of size $[window\_size * len(one - hot - vector)]$ is generated. Eq. 4.43 represents the feature vector of $w_{(i)}$.

$$w_{(i)} = [utt_{(i-m)}; ...; utt_{(i-m+j)}; ...; utt_{(i)}] \tag{4.42}$$

$$CCF_{(w_{(i)})} = [CF_{(utt_{(i-m)})}; ...; CF_{(utt_{(i-m+j)})}; ...; CF_{(utt_{(i)})}] \tag{4.43}$$

Where $CCF_{(w_{(i)})}$ indicates the contextual CF features extracted from i-th window and $CF_{(utt_{(i)})}$ represents the CF features extracted from the i-th utterance. For final classification, $CCF_{(w_{(i)})}$ is fed to a feed forward layer followed by a softmax layer as described in Section 4.5.6.

4.5.12 Methods Compared

For convenience, I summarize the methods compared in the next section for reporting the experimental results.

Popularity:A heuristic method, described in Section 4.5.9, using topic frequency in previous conversations.

CF: The collaborative filtering approach, described in Section 4.5.10, using the conversation state (accepted/rejected topic suggestions) as the user profile.

Contextual-CF: The contextual collaborative filtering approach, described in Section 4.5.11, incorporating CF signals from preceding utterances into CF features from the current utterance using a fully connected neural network.

CTS-CRF: The CRF implementation of the CTS approach, described in Section 4.5.5, using only the conversational context (model-based recommendation).

CTS-CNN: The CNN implementation of the CTS approach, presented in Section 4.5.6, using only the conversational context features (model-based recommendation).

CTS-RNN: The RNN implementation of the CTS approach, presented in Section 4.5.6, using only the conversational context features (model-based recommendation).

CTS-CRF-CF: The hybrid model-based and collaborative-filtering based approach, enhancing the CTS-CRF model with collaborative filtering features (Section 4.5.5).

CTS-CNN-CF: The hybrid model-based and collaborative-filtering based approach, enhancing the CTS-CNN model with collaborative filtering features (Section 4.5.6).

CTS-RNN-CF: The hybrid model-based and collaborative-filtering based approach, enhancing the CTS-RNN model with collaborative filtering features (Section 4.5.6).

Movie	20.1%	Music	14.4%	News	18.4%
Pets_Animal	10%	Sci_Tech	6%	Sports	6%
Travel	9.1%	Games	6%	Celebrities	2.5%
Literature	1.5%	Food_Drinks	1.5%	Other	1.5%
Weather	1.5%	Fashion	1%	Fitness	1%
Entertainment and Cars	1%				

Table 4.24: Topics distribution in Alexa dataset.

4.5.13 Dataset: Amazon Alexa Prize 2018

The conversation data were collected by participating in Amazon Alexa Prize 2018 competition [64]. The conversation dataset consisted of 14,707 open-ended conversations longer than four turns (because the first 2-3 turns usually consisted of the required introduction and exchanges of greetings). These conversations were collected from August 1, 2018, to August 15, 2018. The first ten days of conversations were used for training and the rest for testing. The relative topic popularity is shown in Table 4.24. The conversations have an average length of 11.5 turns, where 91% of the conversations contain at least one suggestion, and 60% have at least two explicit topic suggestions.

4.5.14 Evaluation Metrics

To evaluate our approach, I computed the topic suggestion models on off-line data for each of the methods compared. Following the established recommender system research, I used the following metrics:

- **Micro-averaged Accuracy**: The accuracy is averaged across each topic suggestion individually, thus prioritizing more popular topics and potentially longer conversations.
- **Macro-averaged Accuracy**: The accuracy is averaged across each topic class, equally weighing both popular and "tail" topics.

Suggested Topic / Method	Popularity	CF	C-CF	CTS-CRF	CTS-CNN	CTS-RNN	CTS-CRF-CF	CTS-CNN-CF	CTS-RNN-CF
Movies	0.594	0.804	0.827	0.909	0.785	0.794	**0.910**	0.906	0.909
Music	0.533	0.468	0.807	0.802	0.724	0.741	**0.828**	0.800	0.813
Travel	0.445	0.863	0.853	0.720	0.801	0.824	0.833	0.875	**0.902**
Animals	0.425	0.276	0.482	0.780	0.603	0.621	**0.812**	0.702	0.681
News	0.414	0.164	0.466	0.741	0.518	0.555	**0.742**	0.543	0.584
Sports	0.232	0.000	0.316	0.621	0.523	0.544	**0.663**	0.645	0.608
Entertainment and Cars	0.307	0.752	**0.949**	0.651	0.831	0.856	0.855	0.881	0.928
Games	0.321	0.010	0.405	0.748	0.572	0.605	**0.751**	0.689	0.659
Micro-Averaged Accuracy	0.450	0.519	0.640	0.793(+23%)	0.669(+5%)	0.693(+8%)	**0.819(+27%)**	0.754(+18%)	0.765(+20%)
Macro-Averaged Accuracy	0.408	0.482	0.639	0.746 (+16%)	0.668(+4%)	0.692(+8%)	**0.799(+25%)**	0.755(+18%)	0.760(+19%)

Table 4.25: Accuracy of the topic suggestion methods compared: Popularity, CF, and Contextual-CF (C-CF) methods, vs. CTS-CRF and CTS-CNN (model-based), vs. CTS-CRF-CF and CTS-CNN-CF (hybrid models). All the results are reported for a window of size five for contextual models. All the improvements are reported based on the strongest baseline, Contextual-CF, where they are statistically significant using a one-tailed Student's t-test with p-value < 0.05.

4.5.15 Ground Truth Labels

To create the ground-truth labels, two different scenarios have been followed for training and test data.

For *training*, if a topic t was suggested in turn i, and the talks about topic t in turn $i+1$, the label of T_accept is assigned to turn i. If the user rejects the suggestion, or asked for something else, the label was T_reject. Otherwise the label is a $follow-up$ if a user continues to engage with the same topical component, or *chat* if the utterance is classified as non-informational or *phatic*.

At *test time*, the ground truth labels were assigned as follows: if at turn i, a user rejects the suggested topic T and subsequently, in turn $(i+n)$, requests topic T, then the label for turn i is modified from T_reject to T_accept, because it ultimately matched the user interests. Only the turns with T_accept labels were used as ground truth labels, because users accepted those suggestions at some point during the conversation. Other turns, without a true (accepted) topic, were not used for evaluation. The same ground truth labels were used for all the baseline and the proposed methods.

4.5.16 Training CTS-CRF Model

To train both Seq-CTS-CRF and CTS-CRF-CF models, a maximum likelihood algorithm is applied, where the parameters are optimized using Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) method. For both methods, the context of length five turns is considered. In addition, elastic net ($L1 + L2$) regularization is used to avoid overfitting. Finally, a grid search is deployed to find the optimal values for $L1$ and $L2$, where the values of 0.03 and 0.01 are assigned to $L1$ and $L2$, respectively.

To implement the CF method, I trained the KNN model on the training set with a K value of 33, and the cosine similarity is used as a measure of similarity.

4.5.17 Results And Discussion

I first report the main results of evaluating our proposed topic suggestion models against the popularity-based, CF, and Contextual-CF baselines. I then analyze the recommendation performance for different conversation settings and discuss some limitations of the reported experiments.

4.5.18 Main Results

The main results on most popular classes are reported in Table 4.25. The proposed CTS models outperform the strongest baseline Contextual-CF method, where CTS-CRF, CTS-CNN, CTS-RNN, CTS-CRF-CF, CTS-CNN-CF, and CTS-RNN-CF outperform the Contextual-CF method by 23%, 4%, 8%, 27%, 18%, and 20% respectively.

The results show that CRF significantly outperforms CNN and RNN models, which is surprising for a sequence tagging problem. RNN-based models typically outperform CRF-based methods in similar tasks like entity tagging [29]. I conjecture that CRF outperforms RNNs on this task due to two main reasons: first, the available dataset is relatively small compared to standard entity recognition datasets such as DBpedia [96] and OntoNotes 5.0 [139] with more than 1200K and 1600K samples, respectively. Second, random transitions (e.g., due to dialogue breakdowns) in conversations are more frequent compared to conventional, coherent text. Users usually do not follow a standard conversation with the bot, and may randomly jump between topics. Therefore, even more data are needed to properly model the sequences. However, in contrast to deep RNNs, CRF models need significantly fewer data to be trained.

In general, the collaborative filtering approach appears to perform worse than the other models, including the Popularity-based heuristic baseline (which was manually tuned to optimize the experience of the majority of users). However, incorporating contextual information into the prediction process with CF improves accuracy by

Dropout	Num Filters	Hidden States	Batch Size	Accuracy
0.5	128	100	64	0.765
0.25	128	100	64	0.762(-0.0%)
0.5	512	100	64	0.774(+1.2%)
0.5	128	300	64	0.770(+0.7%)
0.5	128	100	16	0.764(-0.0%)

Table 4.26: Macro-averaged accuracy for CTS-CNN-CF with different parameter settings.

23%. Contextual-CF produces the best results on *Entertainment and Cars*, while it is among the worst results on the other topics like *Games* and *Animals*. I conjecture that this is because *Entertainment and Cars* is a tail topic that few users chose to engage with, and CF is designed to work well for users with rare preferences.

Similar to Contextual-CF, CTS-CNN, CTS-RNN, and CTS-CRF can effectively capture each specific conversation context, for dramatically more accurate recommendations. In contrast, they are reliable, where they provide high accuracy in all the classes. Interestingly, a hybrid of CTS (model-based) and CF model resulted in a more effective model for the topic suggestion, where CTS-CRF-CF and CTS-CNN-CF boost performance by 4% and 9% respectively compared to the CTS-CRF and CTS-CNN models.

Method	Context Size	Without Features	Topical Features	User Profile Features	All Features
CTS-RNN	1	0.563	0.612(+8.7%)	0.601(+6.7%)	0.665(+18.1%)
CTS-RNN	3	0.584(+3.7%)	0.638(+13.3)	0.621(10.3%)	0.685(+21.6%)
CTS-RNN	5	0.613(+8.8%)	0.674(+19.7%)	0.656(16.5%)	0.693(+23.0%)
CTS-RNN-CF	5	**0.701(+24.5%)**	**0.736(+30.7%)**	**0.717(+27.3%)**	**0.765(+35.8%)**

Table 4.27: Ablation study on different features of the CTS-based models, where Context Size is measured in conversation turns. All the improvements are reported based on CTS-RNN with no dialogue manager state information feature and no context. The Macro-averaged accuracy is reported, and all of the improvements are statistically significant with p-value < 0.05.

4.5.19 Feature Ablation on CTS

CTS-based methods are complex models consisting of different steps built based on deep learning algorithms like CNN and RNN. I performed a comprehensive feature ablation analysis to evaluate the effect of each feature group on the overall performance of the system. Table 5.9 reports the results. Using all the CF, topical, and user profile features in combination, is the most effective approach for CTS-based models. Moreover, the results indicate that the impact of *Topic and Behavior* is higher than *User Profile* information. I conjecture that the *Topic and Behavior* features contain contextual information from previous utterances. Also, as conversations progress, the values of these features are updated for each user. In contrast, *User Profile* information contains static and global information about users, which remain largely unchanged during the conversation, thus having a lower impact.

Parameter Tuning.

To evaluate how parameter tuning contributes to the final results, I performed several experiments with different parameter settings. Table 4.26 shows macro-averaged accuracy of the CTS-CNN method with different parameters.

4.5.20 Discussion

I now discuss the strengths and potential limitations of the proposed CTS models on different topics at different stages in the conversation. Finally, I provide the limitations that I encountered during our experiments.

User Topic Acceptance Rate.

Some topics are more popular and interesting for users, such as *Movie* and *Music*. The Popularity baseline described in Section 4.5.9 is designed based on these metrics. Figure 5.2 shows the topic acceptance rate for the most popular topics in Alexa

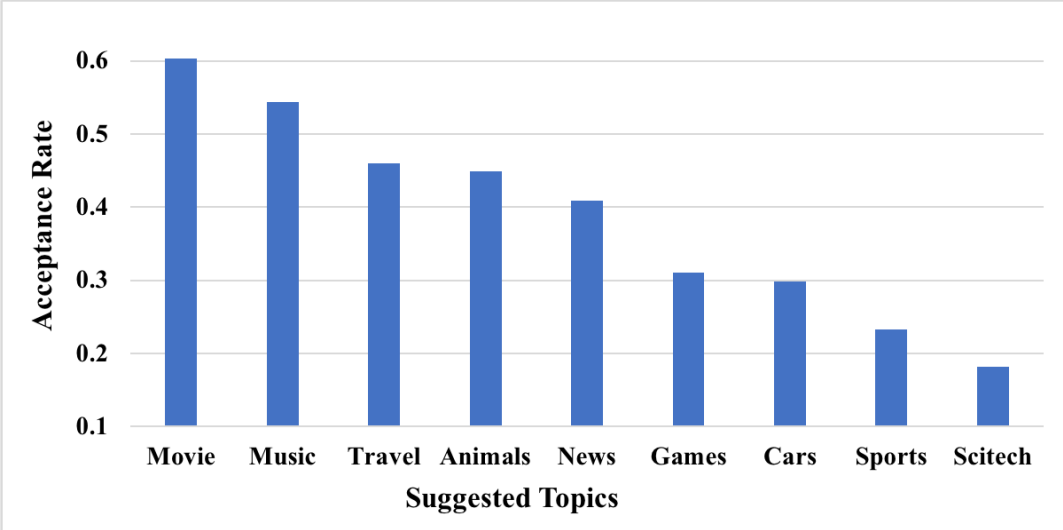

Figure 4.10: Topic acceptance rates in Alexa dataset [5].

dataset. The results indicate that *Movie* is the most popular topic among users with over 60% acceptance rate, and *Scitech* is the least favorite topic with an acceptance rate less than 20%.

Analyzing CF contribution to RNN- based models.

RNN-based methods are known for finely capturing the contextual information within a sequence. The results in Table 4.25 represents that using CF features contributed to the CTS-RNN by extracting relevant knowledge from the dataset that is hidden to CTS-based models. In our specific case, I can elaborate on two reasons, 1) CF features are generated using all the conversation context, while the LSTM model generally considers the history window of size m, and 2) CF features utilize the user-level information like the similarity between user behaviors in accepting or rejecting topics whereas RNN does not consider the user-level information.

Performance for Different Conversation Stages.

As the conversation progresses, the next topic suggestion becomes increasingly challenging, as it is challenging to keep people engaged for long conversations. A proper

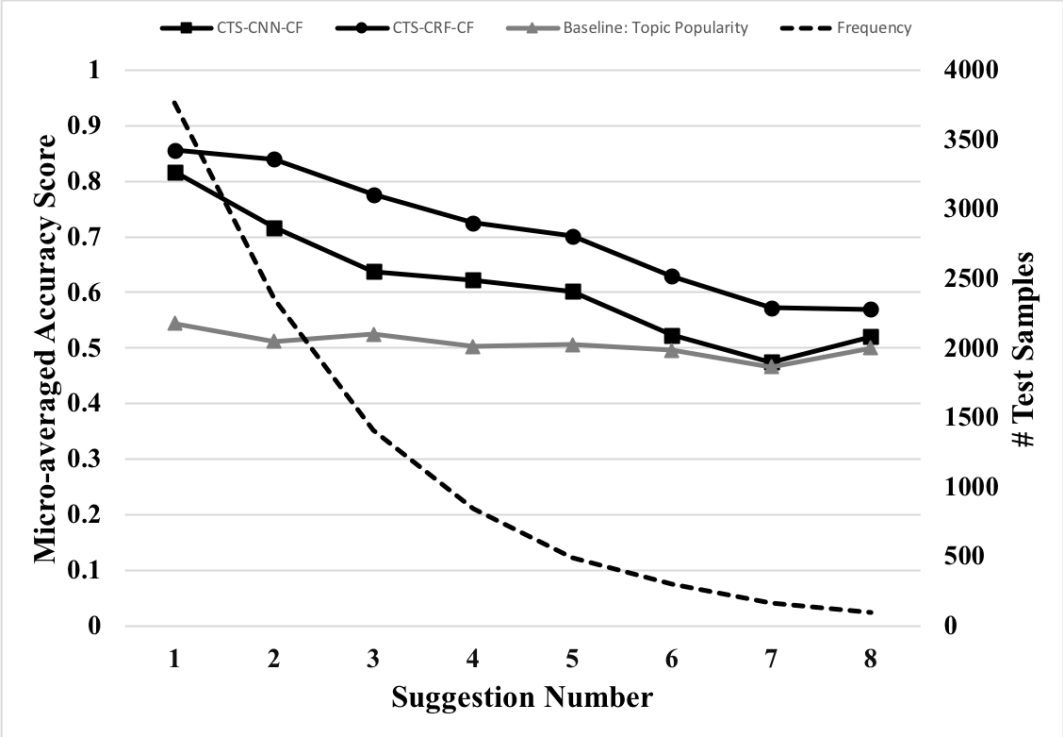

Figure 4.11: Micro-averaged accuracy for CTS-CRF-CF, CTS-CNN-CF, and baseline Popularity model vs. the number of topic suggestions in the conversation [5].

topic suggestion model could encourage a user to engage more with the conversational agent, which has been shown to be associated with an increase in user satisfaction [30, 129]. Figure 4.11 reports Micro-averaged accuracy for CTS-CRF-CF, CTS-CNN-CF, CTS-RNN-CF, and the baseline Popularity model for a varying number of suggested topics per conversation. Surprisingly, the average accuracy of the suggested topic *drops*, as the number of suggestions in a conversation increases. I conjecture that this effect is due to a design decision where a direct topic suggestion was only invoked if a user was not engaged with the current topic or a domain-specific component has returned conversation's control back to the main dialogue manager. These situations indicate that the user may already be not sufficiently engaged in continuing a conversation with the conversational agent past the suggestion point. Also, the rejection of the proposed topic may not be (solely) due to the recommendation algorithm but as a result of user fatigue, or other factors. At the same time,

fewer people continue talking to the conversational agent for the increased number of suggestions. The vast majority of people only interacting with the first one or two suggested topics. Thus, the accuracy of the first handful of suggestions is critical for user experience, as an incorrect first suggestion may cause the user to end the conversation immediately.

4.6 Summary

In this chapter, I described my research on user intent inference in conversational Agents. I explained the architectures of Emersonbot and Irisbot, two open-domain conversational agents that our team at Emory developed for the Amazon Alexa prize 2017 and 2018. Then, I elaborated on my specific contributions on different aspects of user intent inference in conversational agents. I discussed my research on an entity-aware topic classifier named ConCET, a context-aware dialogue act classifier called CDAC, a smart topic suggestion, and an online and offline satisfaction prediction module. All research in this section significantly improved the performance of the current models and was published in well-known conferences and provide insights for future development in advancing conversational AI.

In the next chapter, I present my research on user intent inference in e-commerce search. I explain my contributions in this field and place them in context.

Chapter 5

User Intent inference in E-commerce Search

This chapter explains my research in user intent inference for e-commerce search engines. This chapter focuses on three different models that leverage three different external sources of information, including joint learning, product taxonomies, and unlabeled domain-specific corpora (e.g., catalog). The research was conducted during my collaboration with The Home Depot[1] from Summer 2019 to Spring 2021. In section 5.1, I adopt my paper [4] that was published in SIGIR 2020 under the title "JointMap: Joint Query Intent Understanding For Modeling Intent Hierarchies in E-commerce Search." The paper describes the effectiveness of joint learning of two height-level intent tasks in large e-commerce search engines. It follows by section 5.2 that I discuss the proposed model to leverage product category and taxonomy information for user intent inference in e-commerce [6], which was published in SIGIR 2021-eCom workshop titled "DeepCAT: Deep Category Representation for Query Understanding in E-commerce Search." In section 5.7, I explain my research on pseudo-relevance feedback for query categorization [7], which was published in SIGIR

[1] *https://www.homedepot.com/*

2021 titled "APRF-Net: Attentive Pseudo-Relevance Feedback Network for Query Categorization." In this paper, I investigate product corpus information to improve product categorization for e-commerce search. The insights from this chapter can enable more intelligent user intent inference in an e-commerce search engine.

5.1 JointMap: Joint Query Intent Understanding For Modeling Intent Hierarchies in E-commerce Search

An accurate understanding of a user's query intent can help improve the performance of downstream tasks such as query scoping and ranking. In the e-commerce domain, recent work in query understanding focuses on query to product-category mapping. But, a small yet significant percentage of queries (in our website 1.5% or 33M queries in 2019) have non-commercial intent associated with them. These intents are usually associated with non-commercial information seeking needs such as discounts, store hours, installation guides, and so forth. In this dissertation, I introduce Joint Query Intent Understanding (JointMap), a deep learning model to simultaneously learn two different high-level user intent tasks: 1) identifying a query's commercial vs. non-commercial intent, and 2) associating a set of relevant product categories in taxonomy to a product query. JointMap model works by leveraging the transfer bias that exists between these two related tasks through a joint-learning process. As curating a labeled data set for these tasks can be expensive and time-consuming, I propose a distant supervision approach in conjunction with an active learning model to generate high-quality training data sets. To demonstrate the effectiveness of JointMap, I use search queries collected from a large commercial website. Our results show that JointMap significantly improves both "commercial vs. non-commercial" intent

Search Queries	**intent**	**Product Categories**
where is my shipped order	*non-commercial*	-
how to install my tiles	*non-commercial*	-
cost to rent a carpet cleaner	*non-commercial*	-
18 volt ryobi	*commercial*	[*tools, electrical, lighting*]
24 in. classic Samsung refrigerator	*commercial*	[*appliance, electrical*]

Table 5.1: Dataset sample queries and their associated labels.

prediction and product category mapping by 2.3% and 10% on average over deep learning methods. Our findings suggest a promising direction to model the intent hierarchies in an e-commerce search engine.

5.1.1 Model Overview

In this section, I present the network architecture of JointMap, as shown in Figure 5.1. JointMap utilizes both word and category embeddings in which both representations are jointly trained to achieve an efficient semantic representation for a query. The proposed model consists of two deep learning layers: the first layer for the understanding of the user's commercial intent and the second layer for the prediction of relevant product categories in the taxonomy. As a result, the proposed model contains three embedding layers: a word embedding layer and two category embeddings layers, i.e., commercial vs. non-commercial and product-categories. Both category embedding types are concatenated, to compute the final product category representations. Then, a Compatibility Matrix (CM) is generated by computing the cosine similarity between the label and word representations. CM represents the relative spatial information among consecutive words (phrases) with their associated product category and commercial vs. non-commercial labels. Finally, CM is passed through a Multi-head self-attention layer to calculate attention scores. The word vectors simultaneously go through two Highway layers, and the output of each Highway is multiplied by their corresponding attention scores to generate the final query repre-

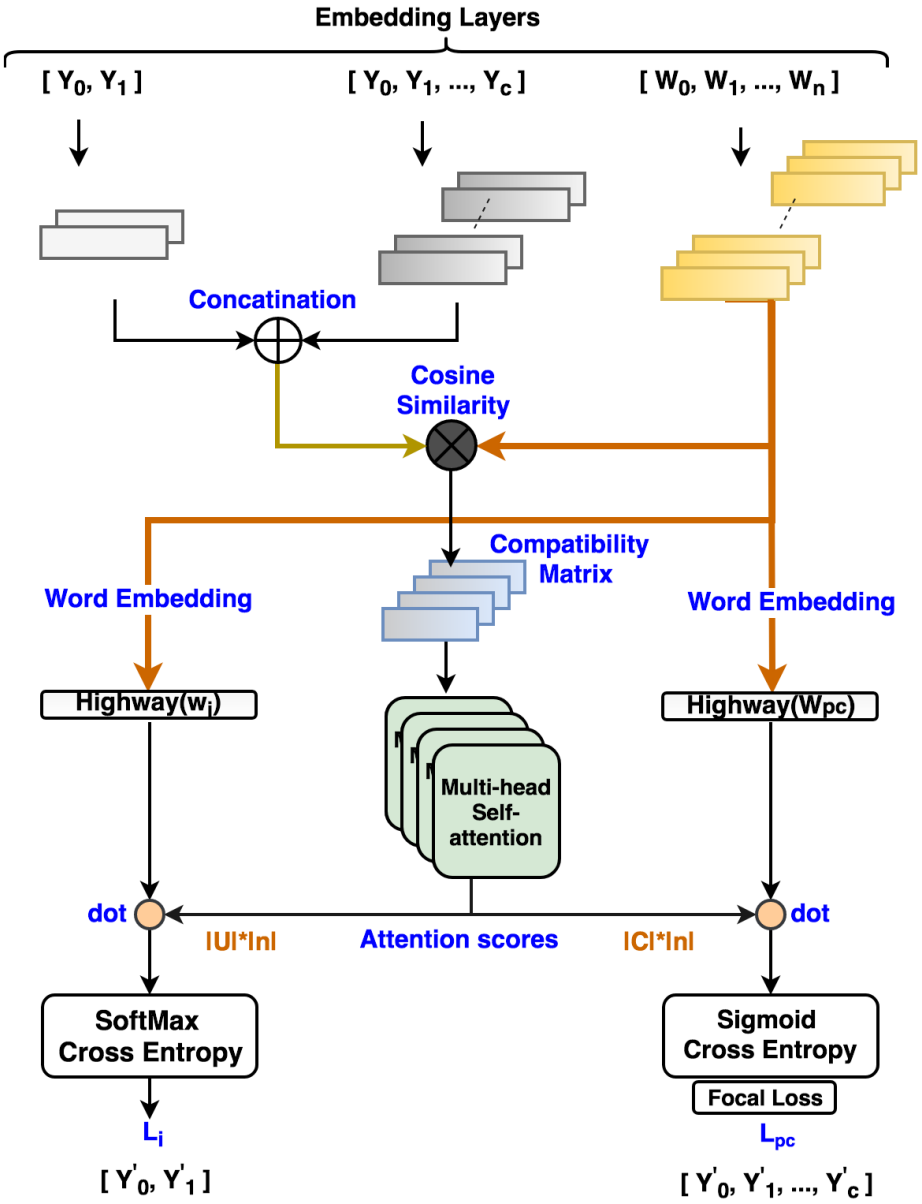

Figure 5.1: JointMap network architecture [4].

sentation. Finally, the loss value of $\mathcal{L}_{pc}$ is computed using sigmoid cross-entropy for the product category mapping. Also, the loss value $\mathcal{L}_i$ is calculated using Softmax cross-entropy for determining the query's commercial intent.

In the next section I explain the details of the proposed model.

5.1.2 Joint-Learning of High-Level Intent Tasks

I now introduce JointMap, a joint-learning model for high-level user intent prediction.

Suppose there is a search query dataset $D = \{Q, C, U\}$, where Q is a set of search queries, U represents user commercial vs. non-commercial intent, and C is the candidate product category set. Each query consists of a sequence of words $q = [w_1; w_2; ... ; w_n]$ of size $n = 10$, and represents as $\mathbf{W}^{|W| \times V}$. Also, C and U are mapped to the embedding spaces $\mathbf{C}^{|C| \times V}$ and $\mathbf{U}^{|U| \times V}$, respectively. Then, the matrices $\mathbf{C}$ and $\mathbf{U}$ are concatenated to illustrate the whole label space. The word and label embeddings are initialized with Word2Vec and random embeddings of size $|V| = 300$, respectively. Cosine similarity between $\mathbf{L}$ and $\mathbf{W}$ is computed for each query q to extract the relative spatial information among consecutive words with their associated labels, where $\otimes$ indicates the cosine similarity function.

$$\mathbf{H} = (\mathbf{C} + \mathbf{U})^{(|C|+|U|) \times V} \otimes \left(\mathbf{W}^{n \times V}\right)^T \tag{5.1}$$

To extract the contribution of the words concerning their category, a multi-head self-attention mechanism with n different heads is implemented on H. Multi-head self-attention contains a parallel of linear projections of a single scaled dot-product function. Eq. 5.10 shows a single head of the self-attention mechanism.

$$\mathbf{G} = SoftMax(\frac{\mathbf{H}K^T}{\sqrt{d_k}})\, V \tag{5.2}$$

where K is the key matrix, V is the value matrix, and d_k is the dimension of

the keys. Also, each projection is responsible for extracting the attention between word-label in a query and computes using weighted sum of the values. Next, $\mathbf{G}$ is split into two matrices of size $\hat{\mathbf{G}} = (|C| \times n)$ and $\widehat{\mathbf{G}} = (|U| \times n)$. For both tasks, the word embedding vectors $\mathbf{W}$ are fed into a highway encoder layer, which has shown its effectiveness in improving network capacity by allowing unimpeded information flow in the network [154]. The output is multiplied by their corresponding attention scores of $\hat{\mathbf{G}}$.

$$\alpha_{\mathbf{1}} = Highway_1(\mathbf{W}), \alpha_{\mathbf{2}} = Highway_2(\mathbf{W}) \tag{5.3}$$

$$\alpha_{\mathbf{i}} = sigmoid(r_w) \rightarrow r_w = relu(word2vec(w)) \tag{5.4}$$

$$\mathbf{W_1} = \sum_{i=1}^{n} \hat{\mathbf{G}}_i \times \alpha_1, \mathbf{W_2} = \sum_{i=1}^{n} \widehat{\mathbf{G}}_i \times \alpha_2 \tag{5.5}$$

Then, resulted $\mathbf{W_1}$ and $\mathbf{W_2}$ have the size of $(n \times V)$. They go through a fully connected layer to generate the semantic representations of both tasks. For product category mapping, a sigmoid cross-entropy loss function $\mathcal{L}_{pc}$ is used since in sigmoid, the loss computed for every output s_i is not affected by other component values. Also, a binary softmax cross-entropy loss $\mathcal{L}_i$ is applied to train the user commercial vs. non-commercial intent.

$$\mathcal{L}_{pc} = -\sum_{c=1}^{|C|} t_c \log\left(Sigmoid(s_c)\right) \tag{5.6}$$

$$\mathcal{L}_i = -t_1 \log(SoftMax(s_1)) - (1 - t_1) \log(1 - SoftMax(s_1)) \tag{5.7}$$

where s_c represents the prediction distribution and t_c indicates the target labels. To address the class imbalance problem, particularly in the product category dataset, I update the loss values based on focal loss proposed in [85]. The focal loss was initially proposed for object detection and removing the effect of extreme foreground-

background class imbalance in the images.

$$\mathcal{L}_{focal_{pc}} = \sum_{c=1}^{|C|} \alpha_c \big(Sigmoid(s_c) - t_c\big)^{\gamma} \log\big(Sigmoid(s_c)\big) \tag{5.8}$$

where t is the target vector, c is the class index, and $(f(s) - t)^{\gamma}$ is a factor to decrease the influence of well-classified samples.

JointMap overall loss:

The final loss function is computed using a weighted loss over commercial vs. non-commercial, product category mapping intents.

$$\mathcal{L}_{total} = \beta_1 \mathcal{L}_{focal_{pc}} + \beta_2 \mathcal{L}_i \tag{5.9}$$

5.1.3 Dataset Overview

In this section, I describe the dataset collected from search logs of a large e-commerce search engine in July 2019, and provide details the algorithms used for generating user-intent datasets.

I propose an algorithm to simultaneously generate both datasets, which consists of three steps: 1) generating the commercial vs. non-commercial queries, 2) over-sampling of the non-commercial queries to balance the dataset, and 3) creating the product category dataset based on the commercial queries. Algorithm. 2 represents the steps for generating commercial vs. non-commercial samples. In this method, first I need to generate a small-size dataset that covers all expected types of non-commercial queries. This can be different for different e-commerce websites based on their resources for intent handling. For example, if they design a specific webpage for installation guides, this type of queries need to be included in the initial dataset.

Then, I over-sample the non-commercial queries as described in [24] to make the

Result: *Commercial Vs. Non-commercial Dataset*
D_init = A small-size Dataset by human supervision ;
Test = Hold-out test set;
while *Accuracy ¡ threshold* **do**
  D = Expand(D_init) using KNN;
  Confidence Scores = SVM(**D**);
  TS = Find(tricky samples) using confidence scores;
  D = Re-label(TS) using human supervision;
  D_init = **D**;
  Accuracy = Compute_Accuracy(Test);
end

Algorithm 2: Commercial vs. non-commercial dataset.

dataset balanced (only 1.5% of the queries have a non-commercial intent). Similar to [156], I utilize user behavior data like click rate, to generate the category labels associated with each commercial query. Algorithm. 3 describes different steps to create the product mapping dataset.

Result: *Product Category Dataset*
Product_category = {};
for *each query in Q* **do**
  pid_list = Extract(pid that user clicks)
  for *pid in pid_list* **do**
    category_list= Find(category(pid) in taxonomy)
  end
  for *category in category_list* **do**
    if *if click_rate* $\geq r$ **then**
      product_category(query).add(category)
    end
  end
end

Algorithm 3: Product category dataset generator.

Finally, a dataset of size 195K with 32 product categories such as *tools, appliance, outdoors, etc.* extracted from the search logs.

5.1.4 Experimental Setup

In this section, I describe the parameter setting, metrics, baseline models, and experimental procedures used to evaluate JointMap.

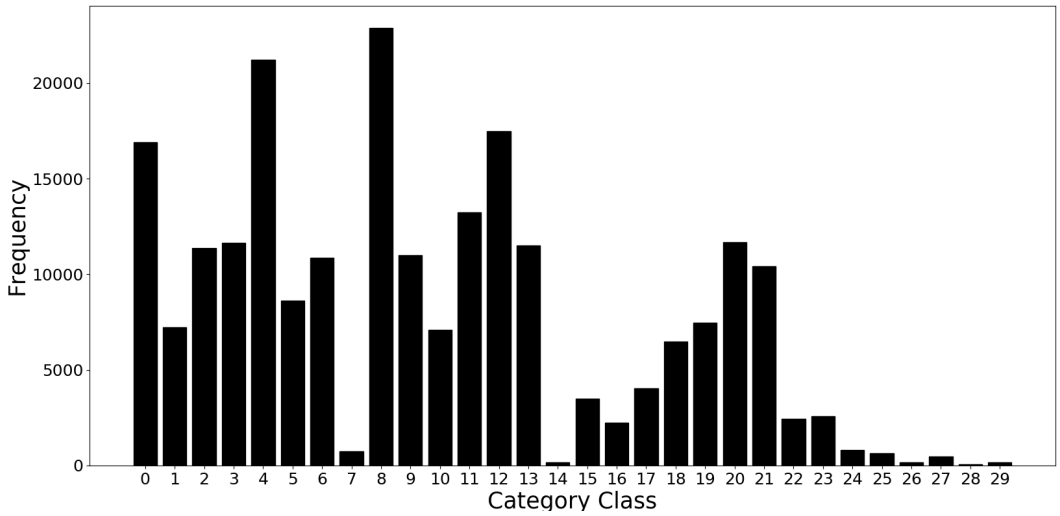

Figure 5.2: Product category distribution [4].

Parameter Setting:

I used Adam optimizer with a learning rate of $\eta = 0.001$ and a mini-batch of size 64 for training. The dropout rate of 0.5 is applied at the fully-connected and ReLU layers to prevent the model from overfitting.

Evaluation Metrics:

To evalate JointMap, both Micro- and Macro- averaged F1-score for both tasks are reported.

Methods Compared:

I summarize the **multi-label** classification methods compared in the experimental results.

- **Tf*idf + SVM:** One-Vs-Rest SVM with a linear kernel.
- **VDCNN:** Very Deep Convolutional Neural Network [34].
- **FastText:** Text classification method developed by Facebook [16].

- **LEAM:** Word-label representation model[130].
- **XML-CNN:** Extreme multi-label text classification [86].
- **JointMap:** The proposed Joint Query Intent Understanding model.

Dataset Experimental Design:

I use an SVM model with n-gram tf*idf as features to perform distant supervision method due to multiple reasons: 1) SVM is fast and scalable, 2) the features and results are interpretable for supervisors, 3) SVM has proved its effectiveness on text data, 4) SVM provides confidence scores to detect the tricky samples. Moreover, two different human annotators were asked to label 540 samples manually. The (Matching, Kappa) scores of (0.98, 0.96) are computed, which is a "significant agreement." The category distribution is shown in Figure 5.2.

5.1.5 Main Results and Ablation Analysis

To evaluate the models described in Section 5.7.6, 70% of the dataset is used for training, 10% for validation, and 20% for test. Table 5.7 summarizes the performance of the models. The results are reported for both commercial vs. non-commercial classification and product category mapping. All the improvements are statistically significant using a one-tailed Student's t-test with a p-value ¡ 0.05.

For the user commercial intent mapping task, the results indicate that the Macro-averaged F1 improves 4.5%,3.8%,2.8%,1.0%, and 1.8% compared to tf*idf, VDCNN, FastText, LEAM, and XML-CNN models respectively. In product category mapping task, the improvements are more significant. There is improvement of 28.4%, 22.1%, 4.2%, 6.3%, and 7.2% over tf*idf, VDCNN, FastText, LEAM, and XML-CNN models, respectively. As a results, JointMap improves macro-averaged F1 scores over deep learning baselines by 2.3% on commercial vs. non-commercial intents, and a 10%

Method	Dataset			
	Commercial vs. Non-commercial		Product Category Mapping	
	Macro-F1	Micro-F1	Macro-F1	Micro-F1
tf*idf+SVM	90.71	90.26	48.75	76.84
VDCNN [34]	91.28	91.34	51.41	79.34
FastText [16]	92.18	92.15	60.06	79.69
XML-CNN [130]	93.11	93.01	58.40	81.61
LEAM [130]	93.96	93.66	58.90	81.31
JointMap	94.80 **(+1.1%)**	94.63 **(+1.0%)**	62.60 **(+6.3%)**	83.01 **(2.1%)**

Table 5.2: Macro- and Micro- averaged F1 for different models. The improvements reported against LEAM.

improvement over product category mapping.

In reference to user commercial intent prediction, a 2.3% improvement is considerable since it is in the context of a large e-commerce search engine that receives billions of search queries per year. For product category mapping, the F1-averaged macro experiences a higher jump when compared to the F1-averaged micro (6.3% vs. 2.1%). This improvement indicates the positive impact of inductive bias between these two tasks, which not only boosts the performance of majority classes, but it also contributes to minority classes. For instance, the Macro-average F1 for 8-button minority classes shows in Figure. 5.2 for XML-CNN and LEAM are 21.76% and 18.33%, respectively, while this number jumps to 31.28% for JointMap.

Parameter Tuning

To evaluate the impact of hyper-parameter tuning in JointMap, I implemented a grid search approach on β_1 and β_2 in Eq. 5.9. I observed that using a smaller β for each task in Eq. 5.9 causes a slower convergence for that specific task. However, the final results is not significantly different. In our experiments, a simple average works as good as a fine-tuned hyper-parameter model. For focal loss hyper-parameter tuning, I repeat the experiments with different γ values of 1, 1.2, 1.5, and 2. I observe that the best results achieve using the $\gamma = 1.5$, where the original dissertation suggested

using $\gamma = 2$ for computer vision application.

Focal Loss Impact on JointMap

Using focal loss deteriorates the overall micro- and macro- averaged F1-scores by 0.6%, 1.5%, respectively. However, the macro-average F1 on 8-button minority classes without focal loss is 31.28%, while with presence of focal loss is 33.81%. This shows a relevant improvement of 8.1%. Furthermore, I observed that in absence of focal loss, the performance of at least two of the minority classes is 0%, therefore making the use of focal loss necessary.

5.2 Label Representation for Product Category Mapping in E-commerce Search

In this dissertation, I introduce a data-driven approach named DeepCAT for query understanding. Our model consists of a pipeline of deep learning models that utilize both word-category and category-category interactions. In summary, our contributions are: (1) proposing a novel deep learning model for joint word-category representation, and (2) introducing a new loss function to incorporate pairwise category information into the query understanding process.

5.3 DeepCAT: Model and Implementation

In this section, I present our DeepCAT model. First, I provide a high-level overview of the model architecture and then describe the model implementation's details. Then , I describe *query representation* and *word-category representation*, and *category-category representation* models, followed by our new loss function to incorporate category co-occurrences.

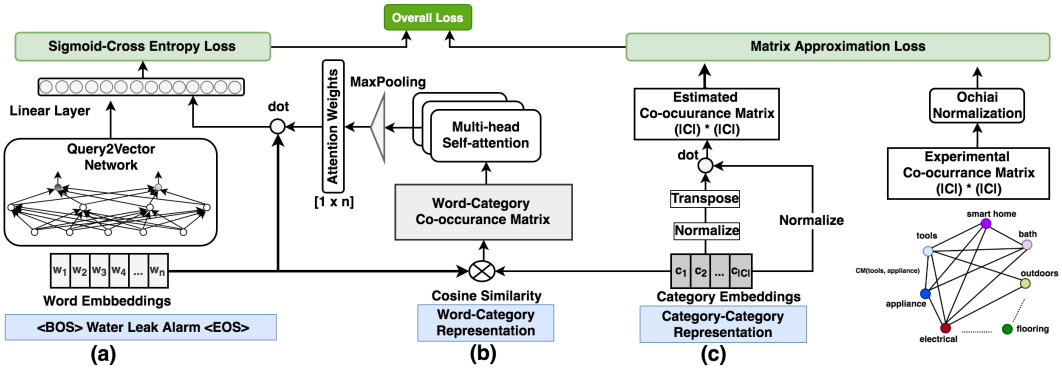

Figure 5.3: DeepCAT architecture, (a) query representation (b) word-category representation (c) category-category representation [6].

5.3.1 Model Overview

The DeepCAT network architecture is illustrated in Figure 5.3. DeepCAT consists of three main components: (a) query representation, (b) joint-word-category representation, and (c) category-category representation. Any deep network could be used to develop the query representation (Query2Vector Network). I deploy a CNN-based model, which consists of convolutional layers followed by highway layers [154], to add more non-linearity to the model and improve the model capacity by allowing information flow in the network. Recurrent [27] or transformer [128] neural models could also be used as an alternative for Query2Vector network. However, due to their high latency time during inference compared to feed-forward neural models, I decided to choose the convolutional neural network-based models for query representation.

I leverage the word-category co-occurrence concepts for joint-word-category representation, which computes using a cosine similarity between query words and their associated categories. Then, a multi-head self-attention deploys to generate the contribution of each word to each specific product category. These attention scores utilize to modify the word's contribution in the query modeling. Finally, category and query representations are concatenated to create the final query representation. A sigmoid cross-entropy is deployed to compute the loss values for this multi-label problem. For category-category representation, first, I extract the experimental category co-

occurrence matrix *CM* from the training set. Next, it normalized using the *Cosine* normalization method. In each training step, the *CM* is estimated using the category representations, and the loss values are propagated through the network using matrix approximation [81].

5.3.2 Query Representation (Query2Vector Network)

Suppose there is a search query dataset $D = \{Q, C\}$, where Q is a set of search queries and C is candidate product categories. Each query consists of a sequence of words $q = [w_1; w_2; ... ; w_n]$ of size $n = 10$, and is represented as $q_w^{|n| \times V}$. Also, C is mapped to embedding spaces of $\mathbf{C}^{|C| \times V}$. The word and category embeddings are initialized with Word2Vec and random embedding of size $|V| = 100$, respectively. For the query representation, any complex deep learning model could be used. Our implementation of Query2Network uses a 3-layer CNN model, where it receives the word embeddings and produces the query representation. $cnn(q_w)$, goes through a *highway* layer [154]. A highway layer combines a ReLU function for a non-linear projection, followed by a sigmoid function for smoothing the projection of each convolutional layer, $highway(q_w) = relu\Big(sigmoid\big(cnn\big)\Big)$.

5.3.3 Word-Category Representation

To train category representation, first, in each training step, I form a word-category co-occurrence matrix. The index (i, j) of this matrix indicates the co-occurrence of word i and associated category j of the query. To estimate this matrix during the training, I need a dot-product between word representations of query $(n \times V)$ with the category representations $(|C| \times V)$. The output is of size $(n \times |C|)$, where $n, |C|$, and $|V|$ indicate the query length, number of categories, and embedding size, respectively. After estimating the word-category co-occurrence matrix, I need to extract each word's contribution in the query to all product categories. I deploy a

self-attention mechanism with $n = 10$ different heads to compute the scores. I use ten heads since I consider each query at most includes ten words. Finally, an attention matrix of size $(n \times |C|)$ creates $A_{wc} = Self\_Attention\big(l2\_norm(q_w) \odot l2\_norm(C)\big)$, where the value at (i, j) represents the contribution of word i to category j. The output goes through a max-pooling layer to form the attention weights. The attention weights multiples to the word vectors to generate the weighted word embeddings $R_{wc} = q_w \odot A_{wc}$. A multi-head self-attention mechanism applies to q_w. Multi-head self-attention contains several linear projections of a single scaled dot-product function that are parallelly implemented $head_i = SoftMax\Big(\frac{q_w K^T}{\sqrt{d_k}}\Big)\ V$. Where $\odot$ indicates a dot-product. Finally, R_{q_w} and R_{wc} go through a linear layer to form R, the final joint word-category representation.

5.3.4 Category-Category representation

A co-occurrence matrix creates on training data to model the category-category interactions. In this matrix each element (i, j) represents the co-occurrence frequency between label-pair of (c_i, c_j) in the training set. Finally, category-category co-occurrence matrix has the size of $|C| \times |C|$. Then, the final matrix is calculated by applying a matrix normalization. I deployed *Cosine* normalization to normalize the CM, where the values on the main diagonal are one. Moreover, the experimental category-category CM is computed using category co-occurrences in the training set. To estimate the normalized matrix, *Cosine* similarity is used between category representations.

5.3.5 Computation of Loss Function

Joint Word-Category Loss

A sigmoid cross-entropy loss function $\mathcal{L}_{pc}$ uses for final product category classification. Sigmoid cross-entropy applies since, in sigmoid, the loss computed for every output s_i

is not affected by other component values. $\mathcal{L}_{pc} = -\sum_{c=1}^{|C|} t_c \log\left(Sigmoid(s_c)\right)$. Where s_c represents the predictions and t_c indicates the targets.

Category-Category Loss

The estimation error is calculated based on a matrix approximation loss [81], $\mathcal{L}_{CM} = \frac{1}{mn}\sum_{i,j\in C} log(1 + exp(C\hat{M}_{ij} \odot CM_{ij}))$.

The Overall Loss

To compute the overall loss, a weighted average of $\mathcal{L}_W$ and $\mathcal{L}_{CM}$ is computed as $\mathcal{L}_{overall} = \lambda_1\mathcal{L}_{CM} + \lambda_2\mathcal{L}_W$.

5.4 Experimental Evaluation

This section describes dataset overview, experimental design, parameter setting, metrics, baseline models, and evaluation.

5.4.1 Dataset Overview

Similar to [156], I utilize customer behavior feedback (e.g., click rate) to obtain the category labels associated with each search query. I collect two weeks of search log to create both training and test sets, where the first week is used to create the training set and the second week for the test set. The training set contains more than 11M search queries. I used 25% of the training data for validation. To generate the test set, I map queries into three different buckets using a simple query frequency. Queries with only one occurrence experimental period are considered as *tail* queries; the ones between 2 and 100 impressions are counted as *torso*, and the rest as *head* queries. Then, to fairly evaluate the models' performance, stratified sampling [11] is used to

Method	Leaf Nodes (Product Categories)									
	P@1	R@1	F1@1	P@3	R@3	F1@3	P@5	R@5	F1@5	MAP@5
TF*IDF BOW	0.783	0.259	0.356	0.617	0.478	0.538	0.514	0.594	0.551	0.623
FastText [15]	0.856	0.2001	0.324	0.634	0.444	0.522	0.504	0.557	0.542	0.666
XML-CNN [86]	0.875	0.314	0.463	0.683	0.549	0.609	0.568	0.666	0.613	0.703
LEAM [130]	0.862	0.302	0.447	0.676	0.531	0.595	0.566	0.651	0.606	0.697
DeepCAT	**0.888***	**0.325***	**0.475***	**0.690**	**0.560***	**0.619***	**0.576**	**0.680***	**0.624***	**0.717***

Table 5.3: Performances on *Product Categories* with about 4200 categories. "*" indicates statistically significant improvements $p < 0.05$.

generate the test set, where I randomly select 2000 different queries from each bucket to create the test set.

5.4.2 DeepCAT Experimental Design

I designed two different experiments to evaluate DeepCAT. In the first experiment, I assess the DeepCAT capability in mapping an input query to the first level in the taxonomy hierarchies, *L1*, with 33 different classes. The *L1* level contains the most abstract product categories (e.g., "appliances", "tools", and "flooring"). This experiment is mainly outlined to estimate the performance of *minority* classes. The *minority* classes include the categories that contain a fairly small number of samples in the training set due to customer click behavior and category overlaps or correlations. The second experiment evaluates DeepCAT on actual product categories in the last layer of taxonomy *Product Categories* with 4115 distinct categories (e.g., "replacement engine parts", "wood adirondack chair", and "window evaporative coolers").

5.4.3 Parameter Setting

I used an Adam optimizer with a learning rate of $\eta = 0.001$, a mini-batch of size 64 for training, and embedding of size 100 for both word and category. The dropout rate of 0.5 is applied at the fully-connected and ReLU layers to prevent the model from overfitting.

5.4.4 Evaluation Metrics

Following the conventions of the search literature to evaluate DeepCAT, I reported the overall Macro- and Micro- averaged *F1*, *P@K, R@K, F1@K* and *MAP@K* on the top-K results. Also, query understanding is a multi-label problem; I reported precision and recall since a practical solution must cover broader possible correct categories while simultaneously keeping precision as high as possible [156].

5.4.5 Methods Compared

I summarize the **multi-label** classification methods compared in the experimental results.

- **TF-IDF + SVM:** One-Vs-Rest SVM with a linear kernel.
- **FastText:** Text classification method by Facebook [15].
- **XML-CNN:** Extreme multi-label text classification [86].
- **LEAM:** Word-label representation model [130].
- **DeepCAT:** The proposed word-label representation.

5.5 Results and Discussion

Table. 5.4 and 5.3 summarizes the performance of different baselines on curated datasets described in section 5.4.1. The results show that DeepCAT significantly improves Macro- and Micro-average *F1*, and *MAP@3* by (3.6%, 1.5%, and 1.2%) over LEAM, as the best model among deep networks, on *L1* level. As a results, an average improvements of (6%, 2.8%, and 4%) on Macro- and Micro-averaged *F1*, and *MAP@3* over deep learning baselines. For *product categories*, DeepCAT outperforms LEAM by (6.2%, 4%, 3%, and 3%) on *F1@1, F1@3, F1@5,* and *MAP@5*, respectively.

Method	First Layer (L1)		
	Macro-F1	Micro-F1	MAP@3
TF*IDF BOW	0.466	0.669	0.669
FastText [15]	0.496	0.686	0.653
XML-CNN [86]	0.511	0.706	0.694
LEAM [130]	0.521	0.709	0.701
DeepCAT	**0.540***	**0.720***	**0.710***

Table 5.4: Performances on *L1* with 33 categories. "*" indicates statistically significant improvements $p < 0.05$.

5.5.1 Results on Minority Classes

Table. 5.4 indicates that Macro-averaged *F1* improves by 2% over Micro-averaged *F1*, which shows a higher impact on the minority classes. This impact is more noticeable on 8-button minority classes, where the Macro-averaged *F1* for the for XML-CNN and LEAM are 0.41.01%, 42.90%. At the same time, this number jumps to 47.16% for DeepCAT, which shows more than 12% and 10% relative improvements, respectively.

5.5.2 Results on Traffic Buckets

Table. 5.8 shows the performance of the models described in section. 5.7.6 across three main buckets of *tail, torso,* and *head*.

Method	FastText	LEAM	XML-CNN	DeepCAT
Head	0.508	0.563	0.560	**0.565 (+0.0%)**
Torso	0.584	0.646	0.648	**0.682 (+5.3%)**
Tail	0.381	0.337	0.373	**0.401 (+7.1%)**

Table 5.5: *F1@3* results on *head, torso,* and *tail* buckets.

The results show that DeepCAT significantly outperforms the other models on both *tail* and *torso* buckets, while it reaches competitive results to XML-CNN and LEAM on "head" bucket. According to higher traffic on both *tail* and *torso* queries, the overall performance of DeepCAT is significantly higher compared to the other models. The *F1@3* is lower on *head* compared to *torso* queries due to a significantly

higher number of correct (relevant) categories, which causes a higher *P@3* and a significantly lower *R@3*.

5.6 Ablation Analysis

DeepCAT is a complex model that consists of several components. I performed a comprehensive ablation study to evaluate each component's impact on the overall performance of DeepCAT. Table. 5.9 reports the contribution of each component on performance. The results illustrate that utilizing the category representation describe in section. 5.3.4 provides a (5.1%, 3.2%) improvement on Macro- and Micro-averaged *F1*, respectively. Moreover, using $\mathcal{L}_{CM}$ improves Macro-averaged *F1* by (2.8%, 1.3%), respectively.

Method	**Macro-F1**	**Micro-F1**
Word Rep.	0.500	0.689
Joint Word-Category Rep.	0.526 (+5.0%)	0.711 (+3.1%)
Joint Word-Category Rep. + $\mathcal{L}_{CM}$	**0.540 (+2.9%)**	**0.720 (+1.3%)**

Table 5.6: Ablation analysis results.

5.7 APRF-Net: Attentive Pseudo-Relevance Feedback Network for Query Categorization

Query categorization is an essential part of query intent understanding in e-commerce search. A common query categorization task is to select the relevant fine-grained product categories in a product taxonomy. For frequent queries, rich customer behavior (e.g., click-through data) can be used to infer the relevant product categories. However, for more rare queries, which cover a large volume of search traffic, relying solely on customer behavior may not suffice due to the lack of this signal. To improve categorization of rare queries, I adapt the Pseudo-Relevance Feedback (PRF)

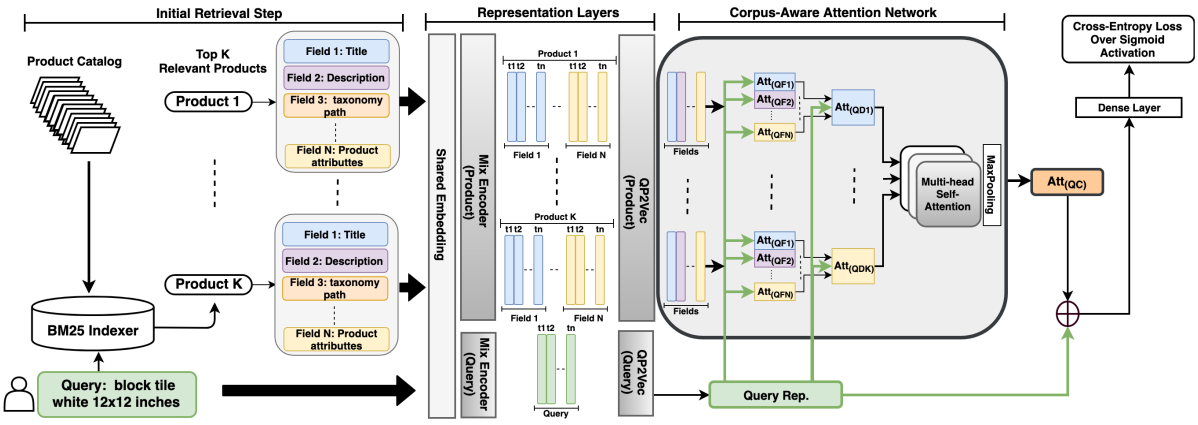

Figure 5.4: Architecture of the proposed attentive pseudo-relevance feedback network (APRF-Net) [7].

approach to utilize the latent knowledge embedded in semantically or lexically similar product documents to enrich the representation of the more rare queries. To this end, I propose a novel deep neural model named **A**ttentive **P**seudo **R**elevance **F**eedback **Net**work (APRF-Net) to enhance the representation of rare queries for query categorization. To demonstrate the effectiveness of our approach, I collect search queries from a large commercial search engine, and compare APRF-Net to deep learning models for text classification. Our results show that the APRF-Net significantly improves query categorization by 5.9% on $F1@1$ score over the baselines, which increases to 8.2% improvement for the rare (tail) queries. The findings of this dissertation can be leveraged for further improvements in search query representation and understanding.

5.7.1 Model Overview

APRF-Net aims to map a query to one or multiple product categories by incorporating its top-ranked retrieved products' content. As illustrated in Figure 5.4, APRF-Net is comprised of three components:

1. **Initial Retrieval Step:** This component is devised to return top-k relevant product documents for an issued query.

2. **Representation Layers:** These layers jointly learn sophisticated query and product field context vectors (Figure. 5.5).

3. **Corpus-Aware Attention Network:** A hierarchical attention network that models the hierarchical structure of the top retrieved product documents (field, document, and corpus).

5.7.2 Initial Retrieval Step

APRF-Net uses BM25 model to retrieve the top-k relevant product documents (corpus) $C_k = \{d_1, ..., d_k\}$ for an initial query q. Each product document is broken down into its corresponding fields (title, color,...). Finally, The query and its corresponding documents' fields are fed into the representation layers (subsection 5.7.3).

5.7.3 Representation Layers

The objective of these layers is to generate context vectors for a query and its retrieved product documents' fields. Representation Layers consist of the following layers: (i) Embedding (ii) Mix Encoder (iii) Query-product-to-vector (QP2Vec). All the layers above are shared between queries and products. A shared embedding could bridge the vocabulary gap between queries and documents. Also, shared encoders enable the network to transfer the captured knowledge across both product documents and queries.

Mix Encoder Layer.

Tail queries generally contain many infrequent terms due to different reasons, such as typos. To make the model less susceptible to out-of-vocabulary issues and spelling errors, I enrich word-level representations with character-aware embeddings introduced by [154].

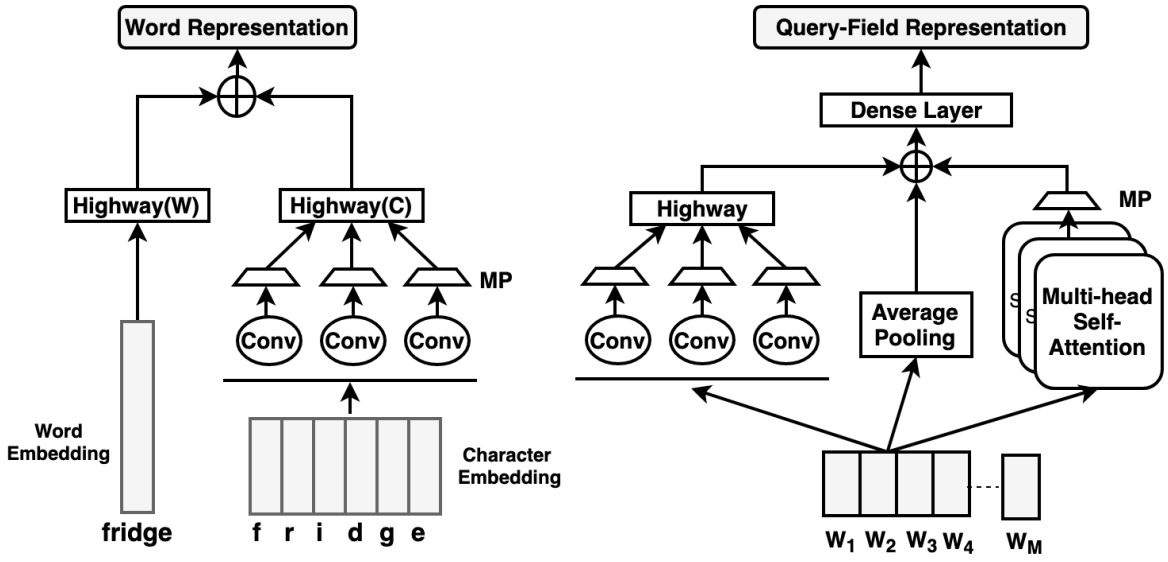

Figure 5.5: (a) Mix Encoder layer (b) QP2V layer [7].

Query-Product-to-Vector Layer (QP2Vec):

QP2Vec composes enhanced word representations from mix encoder to generate a query or products' field-level representations (Figure 5.5.b). The network takes word embeddings and passes them into three parallel neural layers. (i) A three-layer CNN model followed by a max-pooling layer, (ii) an average pooling layer, and (iii) a multi-head self-attention network. Multi-head self-attention contains several linear projections of a single scaled dot-product function that are parallelly implemented. Eq. 5.10 shows a single head in self-attention.

$$head_t = softmax\Big(\frac{E_i K^T}{\sqrt{d_k}}\Big)\ V, t = 0, ..., h \tag{5.10}$$

Where K is the key matrix, V is the value matrix, and d_k is the dimension of the keys, E_i is word embedding for $i-th$ token generated by the mix encoder. Finally, heads' outputs are concatenated and multiplied to a linear transformation, and passed to a max-pooling layer to generate the final representation (Eq. 5.11).

$$SelfAttention(E) = MaxPooling([head_1; ...; head_h].W) \tag{5.11}$$

Where W is the weights of a linear function. All the three outputs from previous steps are concatenated for final representations of query or product's field (Eq. 5.12).

$$QP = [Avg(E); SelfAttention(E); CNN(E)], QP \in R^F \tag{5.12}$$

Where F is the dimension of the query or fields. The QP vectors are sent to the corpus-aware attention network (section. 5.7.4).

5.7.4 Corpus-Aware Attention Network

It is a hierarchical attention network which enriches query representations by incorporating informative contents from top retrieved documents. Corpus-aware attention network contains three levels of abstractions: (i) **query-field attention layer:** I apply a separate query-field attention operation, Att_{QF_i}, across all document fields $i \in \{1, .., N\}$ and top-K retrieved documents. As a result of this operation, $(K \times N)$ vectors of size $(1 \times F)$ are produced, where N is the number of document fields and K is the number of top retrieved documents. To form document attention representation $Att_{D_k} \in R^{N \times F}$, I stack document's query-field attentions: $Att_{QF[k \times N:(k+1) \times N]}$ where $k \in \{0, .., K-1\}$, (ii) **query-document attention layer:** Att_{D_k} matrices are fed into another attention layer to generate query-document attention matrix Att_{QD} (Eq. 5.13).

To implement the attention layers for both (i) and (ii), I used Loung-style attention [93].

$$Att_{QD} = [Q \odot Att_{D_0}; ...; Q \odot Att_{D_K}], Att_{QD} \in R^{K \times D} \tag{5.13}$$

Where D is the dimension of query-document attention.

Method	**P@1**	**R@1**	**F1@1**	**P@2**	**R@2**	**F1@2**	**P@3**	**R@3**	**F1@3**	**MAP@3**
CNN+SCE	0.540	0.486	0.511	0.327	0.574	0.416	0.245	0.633	0.353	0.371
BiLSTM+SCE	0.546	0.419	0.448	0.317	0.554	0.403	0.241	0.619	0.347	0.367
FastText	0.571	0.450	0.503	0.366	0.576	0.448	0.271	0.639	0.380	0.403
XML-CNN	0.564	0.507	0.539	0.339	0.593	0.431	0.252	0.650	0.363	0.385
LEAM	0.578	0.518	<u>0.546</u>	0.344	0.601	<u>0.438</u>	0.253	0.652	<u>0.365</u>	<u>0.391</u>
APRF-Net, top-K=1	0.599*	0.540*	0.567*	0.353*	0.622*	0.450*	0.260*	0.676*	0.376*	0.404*
APRF-Net, top-K=3	**0.610***	**0.548***	**0.578***(+5.9%)	**0.361***	**0.633***	**0.460***(+5.0%)	**0.265***	**0.6879***	**0.383***(+4.9%)	**0.412***(+5.4%)

Table 5.7: Performances of query categorization on TT test set. "*" indicates statistically significant improvements $p < 0.05$.(X%): shows relative improvement between bolded and underlined scores. P and R stand for precision and recall.

(iii) **query-corpus attention layer:** the query-document attentions are passed into a self-attention with K heads followed by a max-pooling to generate query-corpus attention Att_{QC} (Eq. 5.14).

$$Att_{QC} = Maxpooling(selfattention(Att_{QD})), Att_{QC} \in R^{1\times C} \tag{5.14}$$

Where $\odot$ and ; indicate the attention layer and concatenation, respectively. Att_{QC} is the final PRF signal from top-K returned documents to expand the query representations. To do so, I concatenate Att_{QC} to the original query representation. In other words, I expand the query model using the document model in the latent space. Finally, the output is fed into a dense layer to fuse query and document representations before the final category prediction.

5.7.5 Dataset Overview

This section describes our dataset creation process from e-commerce search logs. I use six months of search data to create train, validation, and test sets. Similar to [156], I utilize click-through data to approximate ground truth labels (product categories) by considering the following steps: for a query, (i) leverage query-product clicks to find candidate categories, (ii) aggregate the number of clicks across these categories, and (iii) assign only categories with more than 5% of total clicks. The most fine-grained product categories located in the leaf nodes of taxonomy paths are considered as final classes (e.g., "$Bath > BathroomFaucets > BathroomSinkFaucets > SingleHoleBathroomFaucets'' - >'' SingleHoleBathroomFaucets''$). For each query, I also collect the top-ranked product documents. The table below summarizes fields included in a product document. I randomly sample 60% of the data to create the training, 10% for validation, and the remaining 30% for test sets. I collect 60M search queries (2.5M unique queries) that contain customer signal information, including

4,665 unique product categories.

Fields	Short Description
Title	Product Title
Description	Product descriptions
Taxonomy path	Product taxonomy path from root to leaf
Color/material	Color and material information
Numerical	Numeric values (height, width, etc.)
Brand	Brand information (brand name)

I segment the test set into two types: (i) **Traffic Type (TT):** I utilize the query's frequency information from one year (the refined queries are used to compute the query frequency) with a certain threshold to map each test query to a traffic bucket (head, torso, and tail). The distributions of head, torso, and tail queries in the Traffic Type test set are 8%, 62%, and 30%, respectively. (ii) **Query Type (QT):** queries could contain one or several attributes. I randomly sample 10k queries from the initial test set to assign them to these five attributes: brand name, product name, ID (model number, universal product code, and serial number), numerical (dimension, units, etc.), and color/material. Note that the aforementioned query types are not mutually exclusive, i.e., a query could be mapped to multiple attributes. I leverage some rule-based algorithms to create the initial query type; later, the ones with low confidence scores are relabeled by human annotators.

5.7.6 Experimental setup

This section describes the parameter settings, metrics, baselines, results, and experimental procedures used to evaluate our model.

Parameter Settings

An Adam optimizer was used with a learning rate of $\eta = 0.001$, a dropout rate of 0.5, a mini-batch of size 64, and embedding sizes of 300 and 16 for word and characters, respectively. For convolutional layers, I employed 128 filters with kernel sizes of 1, 2, and 3. I utilized Word2Vec to initialize the word embeddings, and character embeddings were initialized randomly. I employed Sigmoid Cross-Entropy (SCE) for the loss function as it shows superior results compared to the rank loss for XMTC problems [106]. All models were implemented by Tensorflow 2.1 with a single NVIDIA P100 GPU.

Evaluation Metrics

Following the conventions of the literature for XMTC problems and specifically in product categorization [156], I reported Precision@K ($P@K$), Recall@K ($R@K$), $F1@K$, and Mean Average Precision@K ($MAP@K$) on the top-K results.

Methods Compared

I employed the following deep learning models as baselines to evaluate the effectiveness of APRF-Net.

- **CNN+SCE:** 3-layers convolutional neural network.
- **BiLSTM+SCE:** Bidirectional recurrent neural network.
- **FastText:** Bag of tricks for efficient text classification[15].
- **XML-CNN:** Deep learning for XMTC [86].
- **LEAM:** Joint label embedding attentive model [131].
- **APRF-Net:** The proposed model.

5.7.7 Empirical Results and Discussion

Table. 5.7 summarizes the performances of models in section. 5.7.6. As shown in Table 5.7, APRF-Net achieved 5.9% relative improvement on $F1@1$ compared to LEAM, the best performing baseline @1. As K increases, $P@K$ scores drop since each query on average has 1.26 relevant categories (head : 3.3, torso : 1.2, and tail : 1.0).

Results on Traffic and Query Types.

Table 5.8 shows the performance of APRF-Net compared to LEAM on different buckets. APRF-Net advantages are promising, stable, and stronger consistently across all metrics and buckets. I can conclude that APRF-Net has a more significant impact on rare queries across all metrics from our experimental results. For instance, in terms of $F@1$ score, it achieves (3.8%, 5.2%, and 8.2%) relative improvements on (head, torso, and tail), compared to LEAM, respectively. This result proves our earlier claim that our proposed model can make the tail queries less sparse by transferring knowledge from semantically similar documents across queries, specifically, from frequent to rare queries.

Results on the QT test set are also promising. In particular, the improvement is significant for queries with ID attributes (with 38% relative improvement on $F1@1$ over LEAM). This low performance is caused by the lack of natural language content in ID queries, which can be alleviated by including PRF signals.

Results on Minority Classes.

Due to the intrinsic imbalanced nature of e-commerce query categorization, our training data exhibits a power-law distribution. I used the method introduced by [57] to select the minority classes, then evaluate APRF-Net on the picked minority categories. As a result, 3,130 out of 4,665 categories are considered as minority classes

Test	Bucket	Method	F1@1	F1@2	F1@3	MAP@3
TT	head	LEAM	0.444	0.428	0.412	0.567
	torso	LEAM	0.541	0.426	0.351	0.384
	tail	LEAM	0.539	0.418	0.339	0.362
TT	head	APRF-Net,3	0.461	0.436	0.414	0.576
	torso	APRF-Net,3	0.569	0.445	0.367	0.402
	tail	APRF-Net,3	0.583	0.449	0.365	0.391
Test	**Bucket**	**Method**	**F1@1**	**F1@2**	**F1@3**	**MAP@3**
QT	brand	LEAM	0.605	0.475	0.391	0.437
	product	LEAM	0.604	0.477	0.395	0.442
	num	LEAM	0.663	0.503	0.409	0.456
	ID	LEAM	0.174	0.140	0.114	0.122
	c/m	LEAM	0.730	0.548	0.439	0.498
QT	brand	APRF-Net,3	0.641	0.500	0.410	0.460
	product	APRF-Net,3	0.626	0.495	0.409	0.457
	num	APRF-Net,3	0.701	0.535	0.427	0.482
	ID	APRF-Net,3	0.241	0.191	0.162	0.441
	c/m	APRF-Net,3	0.744	0.546	0.441	0.503

Table 5.8: Performances across buckets. c/m:color/material

in our dataset. The LEAM achieved (0.293%, 0.257%, 0.226%, and 0.222%), and APRF-Net reached (0.350%, 0.295%, 0.257%, and 0.256%) on ($F1@1$, $F1@2$,$F1@3$, and $MAP@3$), respectively. The results show a 19.5% $F1@1$ relative improvement over LEAM.

5.7.8 Ablation Analysis

APRF-Net is a complex model that consists of several components: shared embedding, mix encoder, QP2Vec, and corpus-aware attention network. I performed a comprehensive ablation study to evaluate each component's impact on the overall performance (Table. 5.9). The results illustrate that by utilizing PRF signals, APRF-Net achieved relative boosts by (8.0%, 5.4%, 3.9%, and 8.9%) compared to QP2Vec on ($F1@1, F1@2, F1@3$, and $MAP@3$), respectively.

To analyze the impact of the number of retrieved documents on APRF-Net model,

Method	F1@1	F1@2	F1@3	MAP@3
QP2Vec	0.525	0.427	0.362	0.371
QP2Vec+MixEncoder	0.536(+2.1%)	0.431(+0.1%)	0.366(+0.1%)	0.383(+2.2%)
APRF-Net,1 w/o SE	0.553(+5.3%)	0.441(+3.2%)	0.369 (+1.9%)	0.396(+6.7%)
APRF-Net,1	0.567(+8.0%)	0.450(+5.4%)	0.376 (+3.9%)	0.404(+8.9%)
APRF-Net,3	**0.578(+10.1%)**	**0.460(+7.7%)**	**0.383 (+5.8%)**	**0.412(+11.1%)**

Table 5.9: Impact of APRF-Net components: QP2Vec, added by MixEncoder, corpus-aware attention network, and shared embedding. SE stands for the shared embedding.

I ran experiments with different top-K (document) values. As shown in Figure. 5.6, there is a sharp increase from 1 to 3, and it plateaus out afterward. This is due to two reasons: (i) after 3, the relevancy of documents drops, and (ii) the model has already captured the informative contents. Thus, I chose 3 the optimal value for our model.

Comparing Corpus-Aware Attention Network with RM3.

To further investigate the robustness of the corpus-aware attention network for expanding queries in a latent space, I compared it with standard PRF models (e.g., RM3 [77]) that expand queries at the term-level. To do so, first, I expanded the queries using the top 10 relevant documents to [query $<$ *expand* $>$ new terms] utilizing RM3[2], then, a multi-label classifier such as QP2Vec was used for final classification. QP2Vec+RM3 obtained (0.548, 0.442, 0.371, and 0.394) on ($F1@1, F1@2, F1@3$, and $MAP@3$), which showed 4.4% relative improvement on $F1@1$ compared to QP2Vec. Although RM3 provided improvements on all metrics compared to QP2Vec, I experienced a more significant improvement when I expanded the queries in a latent space using APRF-Net (Table. 5.9).

[2]I used Lucene available at `https://github.com/castorini/pyserini`

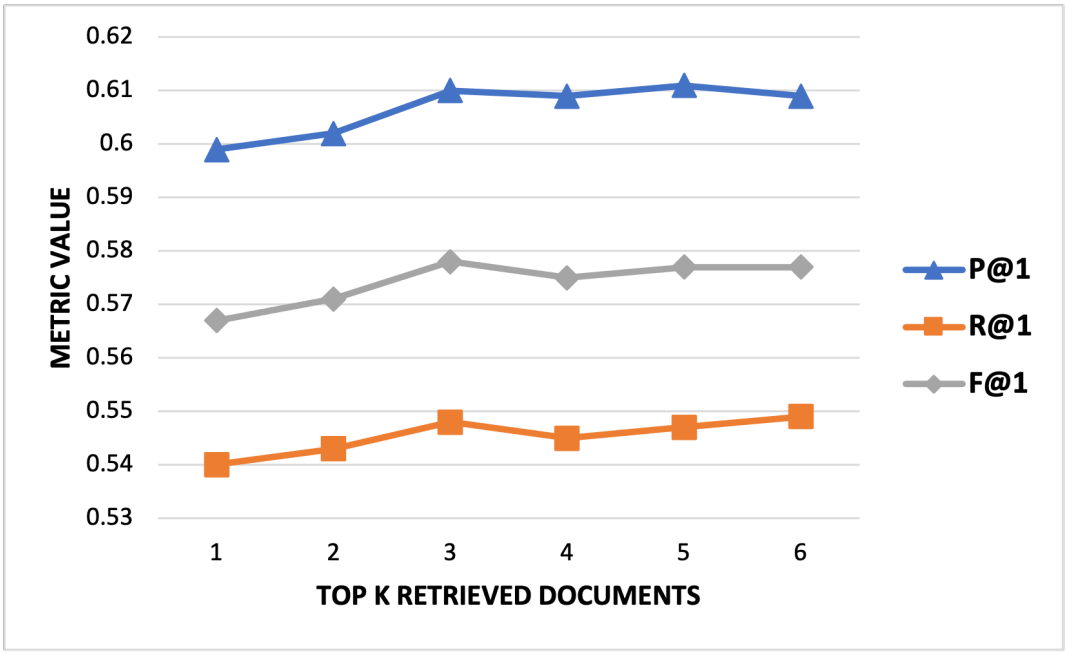

Figure 5.6: Impact of top-K documents on APRF-Net [7].

5.8 Summary

In this chapter, I described my research on user intent inference in an e-commerce search engine. I primarily worked on the real data collected from a large e-commerce search engine like The Home Depot. I started the chapter by describing my research on joint learning of high-level user intents. Then, it was followed by my research on category representation to improve the quality of query categorization. Our findings related to user intent inference in e-commerce search suggest a promising direction to improve search quality in the e-commerce platforms.

In the next chapter, I discuss the results of the proposed approaches by providing their strengths and potential limitations.

Chapter 6

Discussion

In this chapter, I start by discussing the strengths of the proposed approaches and then presenting some potential limitations. The chapter's content related to conversational agents is based on references [3, 2, 5], published in CIKM'2019, SIGIR'2019, and CHIIR'2020. Also, the content related to e-commerce search is based on references [4, 6, 7], published in SIGIR'2020, SIGIR e-com 2021, and SIGIR'2021.

My dissertation introduces several models to utilize knowledge from different resources and actual utterance or query text to improve the performance of the intent inference. Moreover, as the results showed, incorporating an external source of knowledge can improve user intent inference for imbalanced e-commerce datasets and for queries that are not well represented in the training set, a.k.a *tail* and *torso* queries. Also, this dissertation investigates the overfitting problem that might happen on the available external data, a case study on the entity knowledge base. In the next section, I discuss the effectiveness of the proposed knowledge-aware models on addressing imbalanced extreme multi-label classification, queries that are not well represented in the training set, and avoiding overfitting on the frequent meta-data information. Incorporating knowledge from other resources is effective, as I have shown throughout this dissertation, while sometimes it might have some restrictions that must be

investigated carefully.

6.1 Strengths of Knowledge-aware Based Models: Addressing RQ1, RQ2, and RQ3

This section discusses the research questions and how the proposed models addressed them throughout the dissertation.

6.1.1 Addressing RQ1

Incorporating knowledge from external resources can improve the overall performance of the user intent inference, as we discussed through chapter 4 and 5 on different metrics and over deep learning baselines. The proposed models in this dissertation show that integrating entity knowledge-bases, conversation context, and user profile information effectively improves user intent inference for conversational agents. The results indicated that the ConCET model significantly improves topic classification accuracy by 15%, the CDAC model can increase the dialogue act accuracy by 8%, and CTS base models can enhance user engagement performance by 16% for conversational agents.

Moreover, I examine three external information sources for e-commerce to improve query intent inference: relevant task, product taxonomies, and unlabeled domain-specific corpora (e.g., catalog). To do so, I proposed three different deep learning models: JointMap, DeepCAT, and AFRF-NET that incorporate relevant tasks, product taxonomies, and product catalog information, respectively. The results are promising; I achieved a 6%-8% average boost on the performance of the top-1 retrieved documents.

6.1.2 Addressing RQ2

In this dissertation, I proposed six different external sources of information to be integrated with the textual information from both utterance and query to improve the performance of user intent inference in both conversational agents and e-commerce search. I discuss the different methods that this dissertation proposed for each specific underlying problem for user intent inference.

- **ConCET**
  - **External source:** ConCET leverages an entity knowledge base to improve the conversation topic prediction. It uses an entity-linker to extract the entity-type distributions from the entity knowledge-base.
  - **Process to incorporate the external information:** ConCET utilizes both sparse and dense representation through a feature engineering module and a pipeline of deep neural networks to incorporate this knowledge.
- **CDAC**
  - **External source:** CDAC leverages conversation context and domain adaptation to enhance dialogue act prediction. The model uses preceding utterances and system state information from the dialogue manager.
  - **Process to incorporate the external information:** CDAC utilizes both sparse and dense representation through feature engineering modules and a cascade of deep neural networks to incorporate this information.
- **CTS**
  - **External source:** CTS leverages user profile information to enhance conversational topic suggestion. The model uses collaborative filtering signals, user profile, and behavioral features.

  - **Process to incorporate the external information:** CTS models both sparse and dense representation through a collaborative filtering module and pipeline of deep networks across a conversation window.

- **JointMap**

  - **External source:** JointMap leverages inductive bias from relevant tasks to enhance query intent understanding in e-commerce search engines.
  - **Process to incorporate the external information:** JointMap utilizes a dense representation through a joint word-label network to transfer knowledge among two relevant tasks.

- **DeepCAT**

  - **External source:** DeepCAT utilizes taxonomy information to improve user intent inference in e-commerce search. DeepCAT uses a co-occurrence matrix collected from user co-click behavior through the search process.
  - **Process to incorporate the external information:** DeepCAT leverages dense representation of word-label and label-label interactions to model the user co-click behavior.

- **APRF-NET**

  - **External source:** APRF-NET utilizes product corpus (catalog) information to improve user intent inference in e-commerce search. APRF-NET uses top-k relevant structured product data.
  - **Process to incorporate the external information:** APRF-NET utilizes dense representation of the structured and relevant product documents through a hierarchical attention network and expands query's dense representation using document representation in latent space.

The results indicated that leveraging external sources of information can significantly improve the performance of the base model without using the external knowledge. In the conversational agent setting, incorporating a knowledge base can improve the ConCET base model by 17%, adding context can improve the base DAC model by 10%, and user profile information has a 35% relative improvement over the base CTS model. For e-commerce search setting, incorporating relevant task information can improve the base JointMAP model by 6%, adding taxonomy information can enhance the base DeepCAT model by 3%, and product corpus information has 10% relative improvement over the base APRF-NET.

6.1.3 Handling Data Imbalance: Addressing RQ3

Three different external sources of knowledge, such as joint learning of relevant tasks, product taxonomy (catalog), and product documents, are used to enhance user intent inference. This section examines their impact on the imbalance dataset and how they can handle minority classes. Due to the intrinsic imbalanced nature of e-commerce query categorization, our training data exhibits a power-law distribution. This dissertation uses the method introduced in [57] to select the minority classes, then evaluate three different strategies mentioned above to enrich user intent inference on the picked minority categories. As a result, 3,130 out of 4,665 categories are considered as minority classes in the available dataset.

The proposed model, JointMap, leverages inductive bias from relevant classes through joint and multi-task learning for user intent inference. Deploying JointMap on the curated dataset show that the F1-averaged macro experiences a higher jump when compared to the F1-averaged micro (6.3% vs. 2.1%). This improvement indicates the positive impact of inductive bias between commercial and non-commercial user intent classification, which boosts the performance of majority classes and contributes to minority classes. For instance, the Macro-average F1 for 8-button mi-

nority classes for XML-CNN and LEAM are 21.76% and 18.33%, respectively, while this number jumps to 31.28% for JointMap, which indicates a higher impact on the extreme minority classes.

The results are also promising in the case of product taxonomy and catalog knowledge through the DeepCAT model. The results indicate that Macro-averaged *F1* improves by 2% over Micro-averaged *F1*, which shows a higher impact on the minority classes. This impact is more noticeable on 8-button minority classes, where the Macro-averaged *F1* for the for deep learning baselines such as XML-CNN and LEAM are 0.41.01%, 42.90%. At the same time, this number jumps to 47.16% for DeepCAT, which shows more than 12% and 10% relative improvements, respectively.

Finally, In this dissertation, I introduce APRF-NET, a deep learning model that incorporates product corpus knowledge into the user intent inference process. The impact of APRF-NET on minority classes is also promising. The LEAM model as a strong deep learning model baseline achieved (0.293%, 0.257%, 0.226%, and 0.222%), and APRF-Net reached (0.350%, 0.295%, 0.257%, and 0.256%) on ($F1@1$, $F1@2$,$F1@3$, and $MAP@3$), respectively. The results show a 19.5% $F1@1$ relative improvement over LEAM.

The results show that including product corpus information has the highest impact on the minority classes, followed by joint learning of relevant tasks, and finally, incorporating the product taxonomy structure.

6.1.4 Handling *tail* and *torso* Queries: Addressing RQ3

In Web search, the query intent inference models require a robust performance on queries that do not represent well in the training set (e.g., *tail* and *torso* queries). *Tail* and *torso* queries cover a significant amount of search traffic, and different mechanisms need to be designed to handle them efficiently. E-commerce search as a subcategory of the Web search is not an exception. In this dissertation, I study the impact of

document corpus and product taxonomy information on these queries. To this end, we first partitioned data into different buckets and separately reported results on each partition.

In the case of leveraging product documents into the intent inference process, we evaluate the APRF-Net model with the other state-of-the-art deep learning models for extreme multi-label problems. APRF-Net compared to LEAM on different buckets. APRF-Net advantages are promising, stable, and more robust consistently across all metrics and buckets. I can conclude that APRF-Net has a more significant impact on rare queries across all metrics from our experimental results. For instance, in terms of $F@1$ score, it achieves (3.8%, 5.2%, and 8.2%) relative improvements on *(head, torso, and tail)*, compared to LEAM, respectively. This result proves our earlier claim that our proposed model can make the *tail* queries less sparse by transferring knowledge from semantically similar documents across queries, specifically, from frequent to rare queries.

Also leveraging the product catalog, the results show that DeepCAT significantly outperforms the other models on both *tail* and *torso* buckets by 7.1% and 5.3% on $F@1$. At the same time, it reaches competitive results to XML-CNN and LEAM on *head* bucket. According to higher traffic on both *tail* and *torso* queries, the overall performance of DeepCAT is significantly higher compared to the other models.

The results show that utilizing both product documents and catalog effectively improves the user intent inference in e-commerce search. Moreover, incorporating product documents is more effective on all three traffic buckets, where using a product catalog can be helpful on *tail* and *torso* queries. Moreover, both models have competitive results for *torso* queries, and in the case of *tail* queries, product document information shows a more significant positive impact on the results.

6.1.5 Handling Overfitting: Addressing RQ3

This section discusses a specific case where the knowledge-aware-based models might cause overfitting on the source of transferred knowledge, with particular focus on the proposed entity-aware-based model, ConCET, introduced in this dissertation.

Generally, entity-aware classifiers are prone to overfitting to the majority entity-type. I addressed this difficulty by adding sparse and dense representations of the entity-types, which helps in smoothing the entity representation. In other words, using an additional network and separately training the entities reduced the bias towards entity-types.

Furthermore, there are entity-types like *Movie_Names*, which are notoriously problematic for classification. For example, the utterance "Fabulous how are you echo" can be easily misclassified if the entity-aware model is biased toward certain entity-types. In this example, "Fabulous" could be a *Movie_Name*, and "Echo" could be a *City* located in Oregon. In such cases, the proposed model avoids this error in two different ways. First, because combinations like these appear in all classes, the classifier tends to be less biased to these entities. Second, two different joint deep network layers are used in the proposed model, making the system more robust to entity-type errors. Third, the proposed model enriches the textual representation of an utterance with entity information for topic classification. By simultaneously learning the text and entity-types, the proposed model captures the likelihood of the appearance of a specific entity-type in an utterance text to learn a specific topic label. Moreover, to model semantic (dense) representations of the entity-types, I computed an entity-type sequence as Equation 4.14. The interactions among entity-types, when more than one entity-type appear in utterance, and the order of their appearances in utterance, can, therefore, be inferred. As a result, the proposed model can jointly learn a semantic (dense) representation and the distribution of entity-types with textual information to represent an utterance. Training dense and sparse representation of the entity-type

will give the model a higher generalization and makes it more robust to overfitting on the frequent entity-types.

6.2 Limitations of Using External Sources of Information

This dissertation proposed deploying entity knowledge base, contextual information, and user profile information for user intent inference for conversational agents. Also, joint learning of the relevant tasks, product catalog, and unlabeled domain-specific corpora information for e-commerce search. Although considering these sources of knowledge is effective, it might cause an increase in the response latency or inference time (section 6.2.1). As the model becomes more complex, the latency increases accordingly. Moreover, our results show that the proposed models become even more effective when the external source of knowledge selectively provides information or is better aligned with the target domains (section 6.2.2). Finally, offline and online evaluations are another factor that needs to be conceded for an efficient user data analysis (section 6.2.3).

6.2.1 Higher Latency

Incorporating external resources into the user intent inference process resulted in a more complex model. Deploying a more complex system in production could potentially degrade system performance by introducing higher response latency. This latency is an important issue, as response latency has a dramatic effect on the user experience. Interestingly, the classification latency for the proposed approach is not substantially higher compared to the baseline classifier that operates on an utterance text alone. The main reason is that all four stages of the proposed model can be run in parallel. In addition, while entity linking requires a knowledge base lookup, mod-

ern in-memory knowledge-based storage implementations support candidate entity retrieval and matching in only 10s of milliseconds, which does not introduce perceptible increases to response latency. Finally, the proposed model can be executed in parallel for different conversations, allowing the system higher overall throughput without increasing latency for each user.

6.2.2 Aligning Knowledge Source with the Target Dataset

This dissertation investigates the impact of source knowledge base on the overall performance of the user intent inference. A case study examined how much a sizeable generic entity knowledge base and entity linker (e.g., DBpedia) and a curated customized, small knowledge base and entity linker (e.g., PMI-EL) effectively affect the topic classification process for the conversational agents. Although the proposed model outperforms all of the state-of-the-art baselines with both general and customized entity linkers introduced in section 4.3.2, I observe more significant improvements on Alexa data with the PMI-EL domain-specific linker. I conjecture that PMI-EL is designed to identify the entity-types supported by the conversational agent, which are better aligned with the target domains. Nevertheless, ConCET exhibits significant improvements over the previous state-of-the-art with an off-the-shelf generic entity linker and, when available, can take advantage of the domain-specific entity linking for additional improvements.

6.2.3 Offline vs. Online Evaluation

The experimental evaluations for the proposed user profile information suggest the next most interesting topic used offline analysis, and the results might differ in the online setting. As such, I plan to explore it in a follow-on live user study. However, I do not anticipate that the conclusions would change: I emphasize that our reported results are a *lower bound* on performance since I rely on conversations continuing be-

yond the current turn in order to give "credit" to our proposed suggestions that were not recommended at the appropriate time during the live competition. Another potential limitation is the form of the suggestions themselves. In this study, the system proposed a general topic like *Sports* for some topics. Still, I found that proposing a specific item for the topic, e.g., "News about the Yankees" instead of just *Sports* may be more effective and would be a promising complementary direction to the current work.

6.3 Summary

In this chapter, I presented the strength and limitations of the proposed approaches in this dissertation. I explicitly discussed the impact of an individual source of knowledge on the final intent inference. The results showed a positive impact on imbalanced data, *tail* and *torso*, and potential impact on avoiding overfitting. Finally, I presented three limitations that need to be investigated before using an external source of information, including higher latency, aligning source and target knowledge resources, and offline and online evaluations.

In the next chapter, I present the research conclusion and the potential future research to advance the current models proposed in this dissertation.

Chapter 7

Conclusions

This chapter concludes the dissertation by providing summary of the main results in section 7.1, potential future directions in section 7.2, and a summary of the dissertation in section 7.3. The chapter's content related to conversational agents is based on references [3, 2, 5], published in CIKM'2019, SIGIR'2019, and CHIIR'2020. Also, the content related to e-commerce search is based on references [4, 6, 7], published in SIGIR'2020, SIGIR e-com 2021, and SIGIR'2021.

7.1 Summary of the Results:

In this section, I provide a summary of the results for diffident models presented in this dissertation.

ConCET: Entity-Aware User Intent Inference: I introduced ConCET, a novel and effective entity-aware classifier that fuses textual and semantic entity-oriented information to determine the topic of the utterance. The results of the extensive experimental evaluation on two different datasets indicated that ConCET significantly outperformed all the existing deep learning based utterance classifiers introduced both for generic text and for conversational data. The results showed that both variations

of ConCET outperformed the classifier baselines Fastext, VDCNN, and ADAN on the Alexa dataset by large margins of 13%, 23%, and 10%, respectively in terms of Micro-Averaged F1 score.

CDAC: Context-Aware User Intent Inference: I proposed a contextual dialogue cct classification model, CDAC, which incorporated lexical, syntactic, and semantic information in context. Additionally, I introduced a new group of context features to capture the internal system information. Finally, I demonstrated good use of fine-tuning on a limited set of labeled human-machine conversations to decrease manual annotation requirements and utilize the existing human-human labeled conversation data. As a result, CDAC was able to outperform DA classification baselines: by 8.0% on Switchboard data, and by 9.6% on the Alexa Data, and performed comparably to the latest reported and more complex contextual DA classification model. CDAC was also shown to be general enough to be easily fine-tuned for DA classification in human-machine conversations. I believe CDAC represents a promising advance in general user intent classification for intelligent conversational agents.

CTS: User Profile-Aware Intent Inference: I developed and formalized the problem of conversational topic suggestions for mixed-initiative open-domain conversational agents, specifically designed to deliver relevant and interesting information to the user. I presented and explored three approaches for this problem: 1) a collaborative filtering-based approach, 2) a model-based sequential topic suggestion model (CTS-Seq), implemented using CRF, CNN, and RNN models, and 3) a hybrid model which combined sequence modeling approach with traditional collaborative filtering methods (CTS-CRF-CF and CTS-CNN-CF). Topic sequence models I introduced demonstrated significant improvements over previous methods by incorporating collaborative filtering signals derived from the previous choices of similar users. I showed that contextual, sequence-based recommendation significantly outperforms a heavily

tuned, popularity- and time-based baseline, which does not consider the current conversation context and prior user preferences even if available.

JointMap: Relevant Task-Aware User Intent Inference: I proposed a joint learning model that leverages inductive bias among relevant tasks into the user intent inference process. I introduced JointMap, a deep learning model designed for jointly learning two high-level intent tasks on e-commerce search data. JointMap utilized word and label representations and leveraged focal loss to tackle the class imbalance problem in catalog categories. Compared to the strong deep learning models, the results were promising, with an average raise of 2.3% and 10.9% on Macro-averaged F1 in user commercial vs. non-commercial intent and product category mapping.

DeepCAT: Product Catalog-aware User Intent Inference: I developed a deep learning model, DeepCAT, for query understanding in e-commerce search. DeepCAT contains a new joint word-category representation component in which category representations are learned using word-category co-occurrences. Then, I proposed a novel loss function utilizing category representations to model category-category co-occurrences. The comprehensive experiments showed that using category representation significantly improved the results, particularly on minority classes and *tail* queries. DeepCAT achieved a 10% improvement on *minority* classes and a 7.1% increase on *tail* queries over a strong label embedding model.

The experimental results showed the robust performance of DeepCAT compared to deep learning based models. For *minority* classes, *tail*, and *torso* queries, I observed 10%, 7%, and 5.3% relative improvements, respectively. I also reported the performance on the last layer (leaf nodes) of product taxonomy consisting of 4115 categories. The results showed that DeepCAT achieves (6.2%, 4%, 3%, and 3%) increase on *F1@1, F1@3, F1@5,* and *MAP@5*, respectively. In ablation analysis, I proved that the improvements from all three components of DeepCAT and especially

the joint word-category representation that improved the query representation by 5%, and the loss function could improve it by 2.9%.

APRF-Net: Product Corpus-Aware User Intent Inference: I introduced APRF-Net, which adapts the idea of pseudo-relevance feedback (PRF) for the query categorization task in e-commerce search, especially to improve performance for rare queries. I proposed APRF-Net, a novel corpus-aware attention neural model to incorporate the PRF information, representing a query using three abstraction levels (e.g., fields, documents, and corpus) on top-K retrieved products. The goal was transferring customer signal information from head queries to tail queries by leveraging semantic knowledge shared in the overlapping or similar retrieved product documents. The results demonstrated the APRF-Net significantly improved query categorization by 5.9% relative improvement on $F1@1$ score over the baseline, particularly 8.2% improvement for tail queries.

7.2 Future Research Directions

In this section, I present some potential future research directions. Although there have been many breakthroughs in user intent inference for conversational agents and e-commerce search, many fundamental research challenges remain open and need further improvements. The following research tracks are among the important issues that require to be addressed in the future.

Topic Switching in conversational Agents: Topic switching is one of the critical challenges in designing a conversational agent. The performance of topic switching has a direct impact on the conversation's coherency and final user satisfaction. An abrupt change in conversation topic or not properly switching to a new one frustrates the user and reduces their engagement with the agent. Although topic switching

is essential to design a coherent conversation, it remains an open problem and requires further research to address the issue. Throughout the Amazon Alexa Prizes, we developed different methodologies to handle topic switchings, such as leveraging system-driven requests (do you wish to continue talking about "X"), sentiment-based signals, turn-based info, and combinations of the above. However, all of these could be improved with additional customer information to make personalized decisions. Another method I specifically developed in this dissertation was leveraging an embedded recommender system to reduce the rate of topic switching. Although it is interesting to address the problem by reducing its occurrence rate, it does not directly address it. As a result, an exciting research direction is leveraging conversation context and user profile information such as user demographic information to reduce the topic switching rate and improve the performance when the model needs to handle a topic switching intent.

Complex Agreement/Rejection in Conversational Agents: In daily conversations with an accountant in a grocery store, with a banking agent, or even with loved ones, the conversations involve different ways to express the agreement or disagreement with the other person. Humans can state these agreements and disagreements in a complex way that is straightforward for us to understand. We found this very challenging for a machine (conversational agent). For example, imagine a chatbot suggesting a user start talking about an actor. In the response, the user says, "No, I do not want to talk about this, I like to talk about Cristiano Ronaldo". This situation becomes even more challenging when it also involves a topic switching intent. I found handling this highly difficult, and I suggest implementing primary and secondary topic classification (primary = NO, secondary = Movies in the example above). However, many exceptional cases exist, and this problem remains unsolved. I believe that contextual topic and intent classification appears promising, but more

work is needed to improve the performance on such intents to the human level.

Topic classification for conversational Agents: A potential future work for topic classification in the conversational agent can be improving the ConCET model in a more robust way to handle unexpected ASR errors. ASR errors are common in conversational Agents as the current models are not in the human-level performance. For example, by relying less on the exact and complete entity detection and experimenting with character-based representation models. Improving entity recognition models to incorporate contextual information from the preceding utterances can improve the entity recognition module's quality and directly impact the performance of the entity-aware topic classifier.

Another promising direction is to explore other neural network architectures (e.g., transformer-based models) which can incorporate longer contextual dependencies into the prediction process. Finally, Current user topic classification models considered a predefined list of topic information. In real-world scenarios, the conversation chatbot learns new conversation topics and updates its policies by modifying the class boundaries while not forgetting previous knowledge (catastrophic forgetting phenomena). Current trends in machine learning, such as meta-learning-based models and continual learning methods, are other attractive research tracks. These models gradually learn new topics without experiencing catastrophic forgetting phenomena that future research should explore for topic classification in conversational agents.

Smart topic suggestion for Conversational Agents: A key challenge for having a smooth and coherent conversation is to balance familiarity (proposing topics well-received by customers in the past) and surprising and delighting returning customers by helping them discover new and interesting information. A critical future plan is to build on the initial personalization work I started with IrisBot to design new experiences that combine the best previously explored topics and the novelty and value of

proposing novel information. Another major challenge is combining different domains (hobbies/activities and potentially more) to the search process. It becomes challenging to retrieve and rank the results that are semantically relevant to the context. Traditional keyword-based search on metadata information works well for finding entities containing keywords but performs poorly for finding entities that share similar concepts without matching keywords. Incorporating "concept relatedness" and "contextual relevance" into the search process remains challenging. A promising direction for future work is to explore representation learning and deep learning techniques further, more effectively model the conversation context, and experiment with reinforcement learning-based methods for studying the space of interests for new users during the conversation more rapidly. These approaches would naturally fit into and extend the CTS models proposed in this paper, ultimately enabling more intelligent and proactive conversational agents.

Contextual Intent Modeling for Taxonomy Learning: One possible future research is tuning the proposed JointMap model to accept other user behavior information, particularly incorporating contextual information from previous queries within a session. The idea of using contextual information can even be expanded to leveraging multiple sessions in a search log where all the sessions belong to a user for performing a complex search task. Moreover, the idea of learning the correlations between fine-grained categories for label representation is practical; however, it still loses the essential information in the hierarchical information encoded in product taxonomy. A model can be used to train the dependencies to learn the interaction and correlation between the product categories within a product-category level and learn the structural information during product taxonomy mapping.

Dynamic Intent Modeling for Product taxonomy mapping in E-Commerce Search: Current user intent modelings only consider a predefined tree structure on

the product category hierarchy, which is barely the case for colossal e-commerce product data. This taxonomy structure updates over time in real scenarios while getting outdated as the new product is coming or removed from the taxonomy. As a result, new product categories are formed or removed from the current taxonomy. Having a dynamic model that can be updated with the latest changes in the taxonomy is an attractive direction for future research in user intent inference. Like the conversation topic prediction models, current trends in machine learning, such as meta-learning-based models and continual learning methods that gradually learn new product categories, can provide a robust roadmap for future research. These models can learn new tasks without experiencing catastrophic forgetting phenomena, making them practical for dynamic intent modeling tasks.

7.3 Summary of the Dissertation

My dissertation work focused on knowledge-aware user intent inference for both conversational agents and e-commerce search engines. In my dissertation, I proposed several methods related to knowledge-aware user intent inference for intent understanding for conversational agents leveraging different sources of knowledge, including entity knowledge base, conversation context, and user profile information. As a result, I presented three other models in this dissertation that focus on these three sources of knowledge:

1. **ConCET:** a deep learning model for entity-aware topic classification, where ConCET incorporated entity and entity-type information from an entity-knowledge base into textual representation for topic modeling.

2. **CDAC:** a deep learning model for context-aware classification, where CDAC utilized contextual information and transfer learning to adapt models trained on human-human conversation corpus to infer dialogue acts in open-domain

human-machine dialogues.

3. **CTS:** a deep learning model for user profile-aware topic switching, where CTS leveraged user profile information to design a knowledge-aware topic suggestion component to propose the next most interesting macro-topic to the user.

All the models were evaluated on real data logs collected from Amazon Alexa Prize.

For user intent inference in Web search, with a focus on e-commerce search, my dissertation extended the latest knowledge-aware based methods in Web search intent understanding to the e-commerce domain, including:

1. **JointMap:** a deep learning model for relevant task-aware user intent inference, where it introduced a join-learning model to transfer the inductive bias between relevant query understanding tasks.

2. **DeepCAT:** a deep learning model for taxonomy-aware user intent understanding, where DeepCAT proposed a new label-representation to incorporate knowledge from category interactions into user intent understanding.

3. **APRF-Net:** a deep pseudo-relevance-based model for corpus-aware user intent inference, where APRF-Net developed a pseudo-relevance feedback model to enhance the representation of rare queries for query categorization.

All the models were evaluated on real search logs collected from The Home Depot search engine.

The results showed that incorporating knowledge from external resources can significantly improve the overall performance of the user intent inference on different metrics and over deep learning baselines for both settings (e.g., conversational agents and e-commerce search). The results indicated that the proposed models significantly improve topic classification accuracy by 15% and dialogue act accuracy by

8% for conversational agents. For the e-commerce search setting, the dissertation demonstrated that joint learning, product taxonomies, and product corpus (catalog) could significantly improve intent inference accuracy. The proposed models boosted the performance of the top-1 retrieved documents by 6%-8% on the standard metrics. I also demonstrate that incorporating an external source of knowledge can improve user intent inference for imbalanced e-commerce datasets and for queries that were not well represented in the training set, a.k.a *tail* and *torso* queries. The results showed that the proposed models improved the data imbalance problem on average by 19.5%, and they achieved (5.2% and 8.2%) relative improvements on *tail* and *torso* queries, respectively on standard metrics over strong deep learning baselines.

To sum up, this dissertation explained methods and approaches to enhance the user intent inference in intelligent information systems for two settings: 1) conversational agents and 2) e-commerce search. Throughout the dissertation, I complemented the quantitative results with a detailed system performance analysis, which can be used for further improvement in both fields of conversational agents and the e-commerce search. The proposed methods can also be integrated into the new generation of social bots (e.g., Amazon Alexa and Google Home) and e-commerce search engines (e.g., Amazon and The Home Depot) that daily interact with hundreds of millions of users. Finally, the proposed models in this dissertation that focused on user intent inference, a fundamental building block of information seeking processes, can find various new applications, from healthcare and education to media and entertainment.

Bibliography

[1] Ali Ahmadvand, Ingyu Choi, Justus Schmidt Harshita Sahijwani, Mingyang Sun, Sergey Volokhin, Zihao Wang, and Eugene Agichtein. Emory irisbot: An open-domain conversational bot for personalized information access. In *2nd Proceeding of Alexa Prize*, 2018.

[2] Ali Ahmadvand, Jason Ingyu Choi, and Eugene Agichtein. Contextual dialogue act classification for open-domain conversational agent. In *Proceedings of SIGIR*, pages 1273–1276, Paris, 2019. ACM.

[3] Ali Ahmadvand, Harshita Sahijwani, Jason Ingyu Choi, and Eugene Agichtein. Concet: Entity-aware topic classification for open-domain conversational agents. In *Proceedings of CIKM*, pages 1371–1380. ACM, 2019.

[4] Ali Ahmadvand, Surya Kallumadi, Faizan Javed, and Eugene Agichtein. Jointmap: Joint query intent understanding for modeling intent hierarchies in e-commerce search. In *proceedings of SIGIR*. ACM, 2020.

[5] Ali Ahmadvand, Harshita Sahijwani, and Eugene Agichtein. Would you like to talk about sports now? towards contextual topic suggestion for open-domain conversational agents. In *Proceedings of the 2020 Conference on Human Information Interaction and Retrieval*, pages 83–92, 2020.

[6] Ali Ahmadvand, Surya Kallumadi, Faizan Javed, and Eugene Agichtein. Deep-

cat: Deep category representation for query understanding in e-commerce search. *arXiv preprint arXiv:2104.11760*, 2021.

[7] Ali Ahmadvand, Sayyed M Zahiri, Simon Hughes, Khalifa Al Jadda, Surya Kallumadi, and Eugene Agichtein. Aprf-net: Attentive pseudo-relevance feedback network for query categorization. *arXiv preprint arXiv:2104.11384*, 2021.

[8] Nabiha Asghar, Pascal Poupart, Xin Jiang, and Hang Li. Deep active learning for dialogue generation. In *6th Joint Conference on Lexical and Computational Semantics (SEM)*, 2017.

[9] Azin Ashkan, Charles LA Clarke, Eugene Agichtein, and Qi Guo. Classifying and characterizing query intent. In *European Conference on Information Retrieval*, pages 578–586. Springer, 2009.

[10] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[11] Brian Babcock, Surajit Chaudhuri, and Gautam Das. Dynamic sample selection for approximate query processing. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 539–550, 2003.

[12] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language mode. *Journal of machine learning research.*, (3): 1137–1155, 2003.

[13] Phil Blunsom and Nal Kalchbrenner. Recurrent convolutional neural networks for discourse compositionality. In *Proc. of the Workshop on Continuous Vector Space Models and their Compositionality*, 2013.

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

[15] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *TACL, ACL*, 5:135–146, 2017.

[16] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. pages 135–146, 2017.

[17] Chandrakant Bothe, Cornelius Weber, Sven Magg, and Stefan Wermter. A context-based approach for dialogue act recognition using simple recurrent neural networks. In *arXiv preprint arXiv:1805.06280*, 2018.

[18] Chandrakant Bothe, Cornelius Weber, Sven Magg, and Stefan Wermter. A context-based approach for dialogue act recognition using simple recurrent neural networks. In *arXiv preprint arXiv:1805.06280*, 2018.

[19] Gerlof Bouma. Normalized (pointwise) mutual information in collocation extraction. *Proceedings of GSCL*, pages 31–40, 2009.

[20] Eliot P Brenner, J Zhao, Aliasgar Kutiyanawala, and Z Yan. End-to-end neural ranking for ecommerce product search. *Proceedings of SIGIR eCom*, 18, 2018.

[21] Andrei Broder. A taxonomy of web search. In *ACM Sigir forum*, pages 3–10. ACM, 2002.

[22] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[23] Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, and et al. Universal sentence encoder. In *arXiv preprint arXiv:1803.11175*, 2018.

[24] Francisco Charte, Antonio J Rivera, María J del Jesus, and Francisco Herrera. Dealing with difficult minority labels in imbalanced mutilabel data sets. *Neurocomputing*, 326:39–53, 2019.

[25] Evi Yulianti Mark Sanderson Chen, Ruey-Cheng and W. Bruce Croft. On the benefit of incorporating external features in a neural architecture for answer sentence selection. In *n Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1017–1020. ACM, 2017.

[26] Jilin Chen, Rowan Nairn, Les Nelson, Michael Bernstein, and Ed Chi. Short and tweet: experiments on recommending content from information streams. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1185–1194. ACM, 2010.

[27] Jindong Chen, Yizhou Hu, Jingping Liu, Yanghua Xiao, and Haiyun Jiang. Deep short text classification with knowledge powered attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6252–6259, 2019.

[28] Ruey-Cheng Chen, Evi Yulianti, Mark Sanderson, and W Bruce Croft. On the benefit of incorporating external features in a neural architecture for answer sentence selection. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1017–1020. ACM, 2017.

[29] Jason PC Chiu and Eric Nichols. Named entity recognition with bidirectional lstm-cnns. In *Transactions of the Association for Computational Linguistics*, pages 357–370, 2016.

[30] Jason Ingyu Choi, Ali Ahmadvand, and Eugene Agichtein. Offline and online

satisfaction prediction in open-domain conversational systems. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1281–1290, 2019.

[31] Jason Ingyu Choi, Ali Ahmadvand, and Eugene Agichtein. Offline and online satisfaction prediction in open-domain conversational systems. In *Proceedings of CIKM*, pages 1281–1290. ACM, 2019.

[32] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. Towards conversational recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 815–824. ACM, 2016.

[33] Yu-An Chung, Hung-Yi Lee, and James Glass. Supervised and unsupervised transfer learning for question answering. In *Proc. of NAACL-HLT*, 2017.

[34] Alexis Conneau, Holger Schwenk, Loc Barrault, and Yann Lecun. Very deep convolutional networks for text classification. In *15th Conference of the European Chapter of the Association for Computational Linguistics(EACL)*, 2017.

[35] Marco Cornolti, Paolo Ferragina, Massimiliano Ciaramita, Stefan Rüd, and Hinrich Schütze. A piggyback system for joint entity mention detection and linking in web queries. In *Proceedings of the 25th International Conference on World Wide Web*, pages 567–578, 2016.

[36] W Bruce Croft, Michael Bendersky, Hang Li, and Gu Xu. Query representation and understanding workshop. In *SIGIR Forum*, volume 44, pages 48–53, 2010.

[37] W Bruce Croft, Donald Metzler, and Trevor Strohman. *Search engines: Information retrieval in practice*, volume 520. Addison-Wesley Reading, 2010.

[38] Liang CWu, Diane Hu, Liangjie Hong, and Huan Liu. Turning clicks into purchases: Revenue optimization for product search in e-commerce. In *The 41st International ACM SIGIR Conference on Research Development in Information Retrieval)*, pages 365–374, 2018.

[39] Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*, pages 271–280. ACM, 2007.

[40] Fernando Diaz, Bhaskar Mitra, and Nick Craswell. Query expansion with locally-trained word embeddings. *arXiv preprint arXiv:1605.07891*, 2016.

[41] Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. Wizard of wikipedia: Knowledge-powered conversational agents. *CoRR*, abs/1811.01241, 2018. URL `http://arxiv.org/abs/1811.01241`.

[42] Debora Donato, Pinar Donmez, and Sunil Noronha. Toward a deeper understanding of user intent and query expressiveness. In *ACM SIGIR, query representation and understanding workshop*, 2011.

[43] Sarah E Finch, James D Finch, Ali Ahmadvand, Xiangjue Dong, Ruixiang Qi, Harshita Sahijwani, Sergey Volokhin, Zihan Wang, Zihao Wang, Jinho D Choi, et al. Emora: An inquisitive social chatbot who cares for you. *arXiv preprint arXiv:2009.04617*, 2020.

[44] Kathleen Kara Fitzpatrick, Alison Darcy, and Molly Vierhile. Delivering cognitive behavior therapy to young adults with symptoms of depression and anxiety using a fully automated conversational agent (woebot): a randomized controlled trial. *JMIR mental health*, 4(2), 2017.

[45] FA Fonte, Juan Carlos, Burguillo Rial, and Martín Llamas Nistal. Tq-bot: an aiml-based tutor and evaluator bot. *Journal of Universal Computer Science*, 15(7):1486–1495, 2009.

[46] CJ Hutto Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proc. of ICWSM*, 2014.

[47] Sergio Grau, Emilio Sanchis, Maria Jose Castro, and David Vilar. Dialogue act classification using a bayesian approach. In *Proc. of Conference on Speech and Computers.*, 2004.

[48] Fenfei Guo, Angeliki Metallinou, Chandra Khatri, Anirudh Raju, Anu Venkatesh, and Ashwin Ram. Topic-based evaluation for conversational bots. In *NIPS*, 2018.

[49] Jung-Woo Ha, Hyuna Pyo, and Jeonghee Kim. Large-scale item categorization in e-commerce using multiple recurrent neural networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 107–115. ACM, 2016.

[50] Stevan Harnad. The annotation game: On turing (1950) on computing, machinery, and intelligence. In *The Turing test sourcebook: philosophical and methodological issues in the quest for the thinking computer*. Kluwer, 2006.

[51] Mohammed Hasanuzzaman, Sriparna Saha, Gaël Dias, and Stéphane Ferrari. Understanding temporal query intent. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 823–826. ACM, 2015.

[52] Brent Hecht, Jaime Teevan, Meredith Ringel Morris, and Dan Liebling. Searchbuddies: Bringing search engines into the conversation. In *Sixth International AAAI Conference on Weblogs and Social Media*, 2012.

[53] Jonathan L Herlocker, Joseph A Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 1999*, pages 230–237. Association for Computing Machinery, Inc, 1999.

[54] Sebastian Hofstätter, Navid Rekabsaz, Mihai Lupu, Carsten Eickhoff, and Allan Hanbury. Enriching word embeddings for patent retrieval with global context. In *European Conference on Information Retrieval*, pages 810–818. Springer, 2019.

[55] Jian Hu, Gang Wang, Fred Lochovsky, Jian-tao Sun, and Zheng Chen. Understanding user's query intent with wikipedia. In *Proceedings of the 18th international conference on World wide web*, pages 471–480. ACM, 2009.

[56] Jian Hu, Gang Wang, Fred Lochovsky, Jian tao Sun, and Zheng Chen. Understanding user's query intent with wikipedia. In *Proceedings of the 18th international conference on World wide web*, pages 1471–480. ACM, 2009.

[57] Bin Jiang and Xintao Liu. Scaling of geographic space from the perspective of city and field blocks and using volunteered geographic information. *International Journal of Geographical Information Science*, 26(2):215–229, 2012.

[58] Joty, Muhammad Tasnim Mohiuddin, and Dat Tien Nguyen. Coherence modeling of asynchronous conversations: A neural entity grid approach. In *proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 558–568, 2018.

[59] Shafiq Joty, Muhammad Tasnim Mohiuddin, and Dat Tien Nguyen. Coherence modeling of asynchronous conversations: A neural entity grid approach. In

proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, pages 558–568, 2018.

[60] D. Jurafsky. Switchboard swbd-damsl shallow-discourse-function annotation coders manual, draft 13. In *Technical Report*. University of Colorado, 1997.

[61] Nal Kalchbrenner and Phil Blunsom. Recurrent convolutional neural networks for discourse compositionality. In *In Proceedings of the 2013 Workshop on Continuous Vector Space Models and their Compositionality.*, 2013.

[62] Maryam Kamvar and Richard A Miner. In-conversation search, May 12 2015. US Patent 9,031,216.

[63] Chandra Khatri, Rahul Goel, Behnam Hedayatnia, Angeliki Metanillou, Anushree Venkatesh, Raefer Gabriel, and Arindam Mandal. Contextual topic modeling for dialog systems. In *IEEE 2018 Spoken Language Technology (SLT)*. IEEE, 2018.

[64] Chandra Khatri, Behnam Hedayatnia, Anu Venkatesh, Jeff Nunn, Yi Pan, Qing Liu, Han Song, Anna Gottardi, Sanjeev Kwatra, Sanju Pancholi, Ming Cheng, Qinglang Chen, Lauren Stubel, Karthik Gopalakrishnan, Kate Bland, Raefer Gabriel, Arindam Mandal, Dilek Hakkani-Tür, Gene Hwang, Nate Michel, Eric King, and Rohit Prasad. Advancing the state of the art in open domain dialog systems through the alexa prize. *In CoRR*, 2018.

[65] Chandra Khatri, Behnam Hedayatnia, Anu Venkatesh, Jeff Nunn, Yi Pan, Qing Liu, Han Song, Anna Gottardi, Sanjeev Kwatra, Sanju Pancholi, et al. Advancing the state of the art in open domain dialog systems through the alexa prize. *arXiv preprint arXiv:1812.10757*, 2018.

[66] Y. Kim. Convolutional neural networks for sentence classification. In *proceedings*

of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014.

[67] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[68] Young-Bum Kim, Sungjin Lee, and Karl Stratos. Onenet: Joint domain, intent, slot prediction for spoken language understanding. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 547–553. IEEE, 2017.

[69] Youngho Kim, Ahmed Hassan, Ryen W White, and Imed Zitouni. Modeling dwell time to predict click-level satisfaction. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 193–202. ACM, 2014.

[70] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

[71] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[72] Martijn Koster. Aliweb-archie-like indexing in the web. *Computer Networks and ISDN Systems*, 27(2):175–182, 1994.

[73] Ben Krause, Marco Damonte, Mihai Dobre, Daniel Duma, Joachim Fainberg, Federico Fancellu, Emmanuel Kahembwe, Jianpeng Cheng, and Bonnie Webber. Edina: Building an open domain socialbot with self-dialogues. In *1st Proceedings of the Alexa Prize*. Amazon, 2017.

[74] Harshit Kumar, Arvind Agarwal, Riddhiman Dasgupta, and Sachindra Joshi. Dialogue act sequence labeling using hierarchical encoder with crf. In *Proc. of AAAI*, 2018.

[75] Saar Kuzi, Anna Shtok, and Oren Kurland. Query expansion using word embeddings. In *Proceedings of the 25th ACM international on conference on information and knowledge management*, pages 1929–1932. ACM, 2016.

[76] John Lafferty, Andrew McCallum, and Fernando C-N Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[77] Victor Lavrenko and W Bruce Croft. Relevance-based language models. In *ACM SIGIR Forum*, volume 51, pages 260–267. ACM, 2017.

[78] Ji Young Lee and Franck Dernoncourt. Sequential short-text classification with recurrent and convolutional neural networks. In *arXiv preprint arXiv:1603.03827*, 2016.

[79] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.

[80] Canjia Li, Yingfei Sun, Ben He, Le Wang, Kai Hui, Andrew Yates, Le Sun, and Jungang Xu. Nprf: A neural pseudo relevance feedback framework for ad-hoc information retrieval. *arXiv preprint arXiv:1810.12936*, 2018.

[81] Dongsheng Li, Changyu Miao, Stephen Chu, Jason Mallen, Tomomi Yoshioka, and Pankaj Srivastava. Stable matrix approximation for top-n recommendation on implicit feedback data. In *Proceedings of the 51st Hawaii International Conference on System Sciences*, 2018.

[82] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM, 2010.

[83] Raymond Li, Samira Ebrahimi Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. Towards deep conversational recommendations. In *Advances in Neural Information Processing Systems*, pages 9748–9758, 2018.

[84] Xiao Li, Ye-Yi Wang, and Alex Acero. Learning query intent from regularized click graphs. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 339–346. ACM, 2008.

[85] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, pages 2980–2988, 2017.

[86] Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 115–124. ACM, 2017.

[87] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Recurrent neural network for text classification with multi-task learning. In *proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI)*, 2016.

[88] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Adversarial multi-task learning for text classification. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1–10, 2017.

[89] Yang Liu, Kun Han, Zhao Tan, and Yun Lei. Using context information for

dialog act classification in dnn framework. In *Proc. of EMNLP*, pages 2170–2178, 2017.

[90] Yang Liu, Kun Han, Zhao Tan, and Yun Lei. Using context information for dialog act classification in dnn framework. In *In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2170–2178, 2017.

[91] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer, 2011.

[92] Zhongqi Lu, Zhicheng Dou, Jianxun Lian, Xing Xie, and Qiang Yang. Content-based collaborative filtering for news topic recommendation. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[93] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[94] Saurav Manchanda, Mohit Sharma, and George Karypis. Intent term weighting in e-commerce queries. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2345–2348. ACM, 2019.

[95] Prem Melville, Raymod J Mooney, and Ramadass Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *Eighteenth national conference on Artificial intelligence (AAAI)*, pages 187–192. American Association for Artificial Intelligence, 2002.

[96] Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. Db-

pedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th international conference on semantic systems*, pages 1–8. ACM, 2011.

[97] Donald Metzler and W Bruce Croft. A markov random field model for term dependencies. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 472–479. ACM, 2005.

[98] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *arXiv preprint arXiv:1301.3781*, 2013.

[99] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[100] Tom M Mitchell. *Machine learning*. Number 35. Burr Ridge, IL: McGraw Hill 45, 1997.

[101] Andriy Mnih and Ruslan R Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2008.

[102] Ali Montazeralghaem, Hamed Zamani, and James Allan. A reinforcement learning framework for relevance feedback. In *SIGIR*, pages 59–68, 2020.

[103] Andrea Moro, Alessandro Raganato, and Roberto Navigli. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2:231–244, 2014.

[104] Robert R Morris, Kareem Kouddous, Rohan Kshirsagar, and Stephen M Schueller. Towards an artificially empathic conversational agent for mental

health applications: System design and user perceptions. *Journal of medical Internet research*, 20(6), 2018.

[105] Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. How transferable are neural networks in nlp applications? In *In* Proc. *of EMNLP*, 2016.

[106] Jinseok Nam, Jungi Kim, Eneldo Loza Mencía, Iryna Gurevych, and Johannes Fürnkranz. Large-scale multi-label text classification—revisiting neural networks. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 437–452. Springer, 2014.

[107] Hien D. Nguyen and Faicel Chamroukhi. Practical and theoretical aspects of mixture-of-experts modeling: An overview. In *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, page e1246, 2018.

[108] Ioannis Papaioannou and Oliver Lemon. Combining chat and task-based multimodal dialogue for more engaging hri: A scalable method using reinforcement learning. In *Proceedings of the Companion of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, pages 365–366. ACM, 2017.

[109] Silvia Pareti and Tatiana Lando. Dialog intent structure: A hierarchical schema of linked dialog acts. In *Proc. of LREC*, 2018.

[110] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[111] Owen Phelan, Kevin McCarthy, and Barry Smyth. Using twitter to recommend real-time topical news. In *Proceedings of the third ACM conference on Recommender systems*, pages 385–388. ACM, 2009.

[112] Matt Post and Shane Bergsma. Explicit and implicit syntactic features for text classification. In *proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 866–872, 2013.

[113] Filip Radlinski and Nick Craswell. A theoretical framework for conversational search. In *proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval*, pages 117–126. ACM, 2017.

[114] Ashwin Ram, Rohit Prasad, Chandra Khatri, Anu Venkatesh, Raefer Gabriel, Qing Liu, and Jeff Nunn. Conversational ai: The science behind the alexa prize. In *1st Proceedings of Alexa Prize*. Amazon, 2018.

[115] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*.

[116] Rishiraj Saha Roy, Rahul Katare, Niloy Ganguly, Srivatsan Laxman, and Monojit Choudhury. Discovering and understanding word level user intent in web search queries. *Journal of Web Semantics*, 30:22–38, 2015.

[117] Dou Shen, Jian-Tao Sun, Qiang Yang, and Zheng Chen. Building bridges for web query classification. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 131–138. ACM, 2006.

[118] Parikshit Sondhi, Mohit Sharma, Pranam Kolari, and ChengXiang Zhai. A taxonomy of queries for e-commerce search. In *The 41st International ACM SIGIR Conference on Research Development in Information Retrieval*, pages 1245–1248. ACM, 2018.

[119] Yangqiu Song, Haixun Wang, Weizhu Chen, and Shusen Wang. Transfer understanding from head queries to tail queries. In *CIKM*, pages 1299–1308, 2014.

[120] Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. Dialogue act modeling for automatic tagging and recognition of conversational speech. In *Computational linguistics*, pages 339–373, 2000.

[121] Pei-Hao Su, Nikola Mrksic, Inigo Casanueva, and Ivan Vulic. Deep learning for conversational ai. In *proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorial Abstracts*, pages 27–32, 2018.

[122] Yueming Sun and Yi Zhang. Conversational recommender system. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 235–244. ACM, 2018.

[123] Jian Tang, Meng Qu, and Qiaozhu Mei. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, .

[124] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *In Proceedings of the 24th international conference on world wide web*, .

[125] Paul Thomas, Mary Czerwinski, Daniel McDuff, Nick Craswell, and Gloria Mark. Style and alignment in information-seeking conversation. In *proceedings of the 2018 Conference on Human Information InteractionRetrieval*, pages 42–51. ACM, 2018.

[126] Michael Tomasello. *Origins of human communication*. MIT press, 2010.

[127] Andrew Trotman, Surya Kallumadi, and Jon Dagenhardt. Introduction to spe-

cial issue on ecommerce search and recommendation. *Information Retrieval Journal*, pages 1–2, 2020.

[128] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[129] Anu Venkatesh, Chandra Khatri, Ashwin Ram, Fenfei Guo, Raefer Gabriel, Ashish Nagar, Rohit Prasad, Ming Cheng, Behnam Hedayatnia, Angeliki Metallinou, Rahul Goel, Shaohua Yang, and Anirudh Raju. On evaluating and comparing conversational agents. In *NIPS*, 2018.

[130] Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. Joint embedding of words and labels for text classification. In *ACL*, 2018.

[131] Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. Joint embedding of words and labels for text classification. *arXiv preprint arXiv:1805.04174*, 2018.

[132] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Dkn: Deep knowledge-aware network for news recommendation. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1835–1844. International World Wide Web Conferences Steering Committee, 2018.

[133] Jin Wang, Zhongyuan Wang, Dawei Zhang, and Jun Yan. Combining knowledge with deep convolutional neural networks for short text classification. In *IJCAI*, pages 2915–2921, 2017.

[134] Jin Wang, Zhongyuan Wang, Dawei Zhang, and Jun Yan. Combining knowledge with deep convolutional neural networks for short text classification. In Proceedings of IJCAI, 2017.

[135] Jingang Wang, Junfeng Tian, Long Qiu, Sheng Li, Jun Lang, Luo Si, and Man Lan. A multi-task learning approach for improving product title compression with user search log data. In *proceeding of AAAI*, 2018.

[136] Zhongyuan Wang and Haixun Wang. Understanding short texts(tutorial). In ACL, 2016.

[137] Zhongyuan Wang Dawei Zhang Wang, Jin and Jun Yan. Combining knowledge with deep convolutional neural networks for short text classification. In *In Proceedings of IJCAI*, volume 350, 2017.

[138] Zihao Wang, Ali Ahmadvand, Jason Ingyu Choi, Payam Karisani, and Eugene Agichtein. Emersonbot: Information-focused conversational ai emory university at the alexa prize 2017 challenge. In *1st Proceeding of Alexa Prize*, 2017.

[139] Martha Palmer Mitchell Marcus Eduard Hovy Sameer Pradhan Lance Ramshaw Nianwen Xue et al. . Weischedel, Ralph. "ontonotes release 5.0 ldc2013t19.". In *Linguistic Data Consortium*. Philadelphia, PA, 2013.

[140] Joseph Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, 1966.

[141] Tsung-Hsien Wen, Yishu Miao, Phil Blunsom, and Steve Young. Latent intention dialogue models. In *proceedings of ICML*, pages 3732–3741, 2017.

[142] Tsung-Hsien Wen, David Vandyke, Milica Gasic Nikola Mrksic, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. A network-based end-to-end trainable task-oriented dialogue system. In *proceedings of EACL*, pages 438–449, 2017.

[143] Jason D. Williams and Geoffrey Zweig. End-to-end lstm-based dialog control optimized with supervised and reinforcement learning. In *CoRR*, 2016.

[144] Zhang Xiang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *NIPS*, 2015.

[145] Chenyan Xiong, Jamie Callan, and Tie-Yan Liu. Bag-of-entities representation for ranking. pages 181–184. ACM, 2016.

[146] Wayne Xiong, Lingfeng Wu, Fil Alleva, Jasha Droppo, Xuedong Huang, and Andreas Stolcke. The microsoft 2017 conversational speech recognition system. In *In 2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5934–5938. IEEE, 2018.

[147] Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. Learning distributed representations of texts and entities from knowledge base. pages 397–411. MIT Press, 2017.

[148] Rui Yan and Dongyan Zhao. Smarter response with proactive suggestion: A new generative neural conversation paradigm. In *IJCAI*, pages 4525–4531, 2018.

[149] Rui Yan, Dongyan Zhao, et al. Joint learning of response ranking and next utterance suggestion in human-computer conversation system. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 685–694. ACM, 2017.

[150] Xuesong Yang, Yun-Nung Chen, Dilek Z. Hakkani-Tur, Paul Crook, Xiujun Li, Jianfeng Gao, and Li Deng. End-to-end joint learning of natural language understanding and dialogue manager. In *proceedings of ICASSP*, pages 5690–5694, 2017.

[151] Zhi-Xiu Ye and Zhen-Hua Ling. Hybrid semi-markov crf for neural sequence labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, page 235–240. Association for Computational Linguistics, 2018.

[152] Dawei Yin, Yuening Hu, Jiliang Tang, Tim Daly, Mianwei Zhou, Hua Ouyang, Jianhui Chen, Changsung Kang, Hongbo Deng, Chikashi Nobata, et al. Ranking relevance in yahoo search. In *KDD*, 2016.

[153] Hamed Zamani and W Bruce Croft. Relevance-based word embedding. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 505–514. ACM, 2017.

[154] Hongfei Zhang, Xia Song, Chenyan Xiong, Corby Rosset, Paul N Bennett, Nick Craswell, and Saurabh Tiwary. Generic intent representation in web search. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 65–74. ACM, 2019.

[155] Xiang Zhang and Yann LeCun. Text understanding from scratch. In *arXiv preprint arXiv:1502.01710*, 2015.

[156] Jiashu Zhao, Hongshen Chen, and Dawei Yin. A dynamic product-aware learning model for e-commerce query intent understanding. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1843–1852. ACM, 2019.