Deep Learning for Spoken Dialogue Systems: Application to Nutrition

by

Mandy Barrett Korpusik

B.S., Franklin W. Olin College of Engineering (2013)
S.M., Massachusetts Institute of Technology (2015)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2019

Author . . . .
Department of Electrical Engineering and Computer Science
May 23, 2019

Certified by . . . .
James R. Glass
Senior Research Scientist
Thesis Supervisor

Accepted by . . . .
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Deep Learning for Spoken Dialogue Systems: Application to Nutrition

by

Mandy Barrett Korpusik

Submitted to the Department of Electrical Engineering and Computer Science
on May 23, 2019, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Personal digital assistants such as Siri, Cortana, and Alexa must translate a user's natural language query into a semantic representation that the back-end can then use to retrieve information from relevant data sources. For example, answering a user's question about the number of calories in a food requires querying a database with nutrition facts for various foods. In this thesis, we demonstrate deep learning techniques for performing a semantic mapping from raw, unstructured, human natural language *directly* to a structured, relational database, without any intermediate pre-processing steps or string matching heuristics. Specifically, we show that a novel, *weakly supervised* convolutional neural architecture learns a shared latent space, where vector representations of natural language queries lie close to embeddings of database entries that have semantically similar meanings.

The first instantiation of this technology is in the nutrition domain, with the goal of reducing the burden on individuals monitoring their food intake to support healthy eating or manage their weight. To train the models, we collected 31,712 written and 2,962 spoken meal descriptions that were *weakly annotated* with only information about which database foods were described in the meal, but not explicitly where they were mentioned. Our best deep learning models achieve 95.8% average semantic tagging F1 score on a held-out test set of spoken meal descriptions, and 97.1% top-5 food database recall in a fully deployed iOS application. We also observed a significant correlation between data logged by our system and that recorded during a 24-hour dietary recall conducted by expert nutritionists in a pilot study with 14 participants. Finally, we show that our approach generalizes beyond nutrition and database mapping to other tasks such as dialogue state tracking.

Thesis Supervisor: James R. Glass
Title: Senior Research Scientist

Acknowledgments

As I sit here in my office on the fourth floor of the funky-looking Stata Center, finally done with faculty job interviews and now wrapping up my thesis, I have the opportunity to reflect back on what a wonderful experience my Ph.D. has been. I feel extremely lucky to have joined the Spoken Language Systems group in the EECS department at MIT, with Dr. Jim Glass as my advisor, because I have always felt welcomed and supported in this friendly, tight-knit community with a kind, flexible advisor who let me work on projects that interested me, and in a department that is exceptionally supportive of its graduate students. This was an incredible journey, with the ups and downs that always come with research, and life in general, but looking back, I am sure I will remember this time in graduate school at MIT as some of the best years of my life!

First, I want to thank my family for their love and support throughout my whole life, but especially during graduate school and in preparing for my future career as a professor. I never could have done all this without them. My parents are always there for me, no matter what, and I am so blessed to have them in my life. My mom in particular is my #1 mentor—she is the best role model I have. I'm looking forward to finally moving back to CA after graduating to be closer to home after 10 years in MA :) I also want to thank my sister Angie for her love and advice (there's so much to learn from the wisdom of younger sisters, and she has a much better fashion sense than I do!). I'm so proud of her for following her passion to study organic chemistry at the University of Florida for her Ph.D., and I wish her the best of luck finishing graduate school!

I also want to thank my Uncle Peter and Aunt Patty for taking me under their wing after I moved across the country. They have become my family away from home, and I will miss them so much when I move back to CA. I really appreciate them inviting me to countless family get-togethers, always checking in with me, driving me to and from

Danvers, and helping me whenever I was sick or needed a ride to or from the hospital. I'm grateful to my extended family as well—my cousins Melissa, Rob, and Adam; Uncle John and Aunt Gail; and Aunt Cathy all welcomed me as part of the family when I moved to the East Coast. Finally, I want to thank my relatives in the South for their advice and support, especially Grandma and Gordon who let us stay at their place during Christmas, Aunt Sharon whose advice was shared with me through my mom, and Aunt Tricia who connected me to distant relatives and gave tips whenever I was traveling.

Next, I want to thank my dear friends. My friend group at MIT (nicknamed MDMRAJ for Mengfei, Danielle, Mandy, Ramya, Amy, and Jennifer) has been the best set of friends I could ever have imagined—these women are absolutely amazing, and are one of the best parts of graduate school!! We celebrate our success and accomplishments, paper acceptances, weddings and bridal showers together; have met up over lunch in Stata and ladies' night dozens of times; and are always there for each other in times of sadness, illness, or stress. I want to thank Mengfei for five wonderful years of roomming together (I wish her the best of luck with her upcoming wedding and move back to Singapore!), Ramya for always being the sweetest, most selfless friend possible, Danielle for her calming presence and wise advice, and Amy and Jennifer who are my favorite workout buddies and planned awesome trips. I also want to thank my zumba buddies Natasha and Annabelle who are also there for me, whether to congratulate me on a job offer, or give support during a breakup. I want to thank my new roommate Li for being so friendly and open from day one, joining me at coffee hour and brunch, and chatting with me every night. I also want to thank my Olin friends Lillian, Sarah (both of them!), and Breauna for doing such a great job of keeping in touch, and Tanner for always brightening my day and helping me design Coco. Thanks to Guha for joining me for a tennis rally every now and then, Cherry for exploring entrepreneurship ideas with me, Leah and Izzy for joining us at Olin lunch, Karthik and Farnaz for advice on the faculty job search, and Pamela for still keeping in

touch after high school despite living all over the world. Last but certainly not least, I'd like to thank my best friend Megan from high school for inviting me to be a maid of honor at her wedding and for staying close all these years apart!

I am so grateful for the many mentors I have had the opportunity to work with over the years. They helped me get to where I am today and are a big reason I wanted to become a professor, so I could give back to others the same kind of mentoring that meant so much to me! Of course, I want to thank my advisor Jim for training me and supporting me in my research, and my committee members Victor Zue and Pete Szolovits for their helpful comments and feedback on my thesis and during the faculty job search (and a special thank-you to Victor for accepting me into MIT and supporting me as my graduate counselor and by writing my recommendation letters). I also want to thank Kathy Huber and Julia Hirschberg, who always were looking out for me and genuinely had my best interest in mind. Kathy helped me not only with entrepreneurship, but also in dealing with tricky situations and preparing for my future career. Julia first taught me natural language processing, helped me get into graduate school and get a faculty job, and is my inspiration for becoming a professor. Thanks to President Miller for his willingness to meet with me and support me in the job search (and even writing me tailored recommendation letters!), Francine Chen and Cynthia Skier for their mentoring and recommendation letters, Patrick Winston for helping me with my job talk and research statement, Julie Shah and Polina Golland for offering their time to help me with the faculty job search, Mark Chang and Lynn Stein from Olin for their advice, Jinane Abounadi for her support of entrepreneurship through the Sandbox program, Dr. Howe for mentoring me in high school and for his continued support at every subsequent stage of my career, and Jessica Wu who barely even knew me for being the best mentor one could ask for during the academic job search.

Lastly, I want to thank SLS, the Spoken Language Systems group, who have been my friends and colleagues throughout my time at MIT. Marcia is not only our administrative assistant—she is one of the wisest, kindest, funniest, most helpful people I have ever had the privilege to know. She is such a good listener, and has given me some of the best life advice out of anyone I have met. I want to thank my officemates Wei-Ning, François, Hao, Jackie, Ekapol, and Yu for all the fun times we have shared in this office, and for teaching me about speech recognition. I can only hope to be half as brilliant as you guys someday, and you inspire me every day to become better and work harder! I also want to thank my labmates Stephen, Di-Chia, Yonatan, Tuka, Mitra, Maryam, Jen, Ann, and Xue for keeping work fun and light-hearted, and for their advice and support through the Ph.D. I'm still working on my Duolingo with the hopes that someday I will be fluent enough in Mandarin to eavesdrop on my Taiwanese labmates :)

While it's bittersweet for this chapter of my life to come to an end in June, I am also very excited to move onto the next phase of my career as an Assistant Professor of Computer Science at Loyola Marymount University in Los Angeles—bring on that sun!!!

This research was sponsored by a grant from Quanta Computing, Inc., and by the Department of Defense (DoD) through the National Defense Science & Engineering Graduate Fellowship (NDSEG) Program.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Today's AI-powered personal digital assistants, such as Siri,[1] Cortana,[2] and Alexa,[3] convert a user's spoken natural language query into a semantic representation that is used to extract relevant information from a structured database. These dialogue systems typically begin with standard natural language processing (NLP) tasks, including intent detection (i.e., determining the user's goal), followed by semantic tagging of the natural language utterance to determine precisely what the user is requesting. For example, the user's intent may be to book a flight, and the relevant semantic tag values are Boston for the departure city, and Toronto for the arrival city. Subsequently, the semantic frame representation of the user's query is converted into a SQL command for querying a structured database, which should contain all the available flights to and from a given city on a particular date.

The problem with these conventional approaches is that they often rely heavily on manual feature engineering and a set of heuristics for mapping from user queries to database entries. There is a fundamental mismatch between how people describe objects, and how

[1] https://www.apple.com/siri
[2] https://www.microsoft.com/en-us/cortana
[3] https://developer.amazon.com/alexa

the corresponding entities are represented in a structured database. For example, when a person describes the food they have eaten, they might say they had a slice of "toast," rather than a piece of bread. However, the matching entries in a food database are various types of "bread, toasted," which is not an exact word match, and to complicate matters further, there are numerous other potential database matches, including French toast, toasted cereal, and toasted nuts or seeds. A similar problem occurs for "oatmeal," which also matches oatmeal bread and oatmeal cookies, as well as "milk," which is mentioned in database entries for yogurt and other dairy products. Historically, researchers have dealt with this mismatch by regularizing text through tokenizing, stemming, and other heuristics.

To avoid this pipeline of text normalization and word matching database lookup, we have instead taken the approach of feeding raw input to neural networks, and allowing the models to learn how to handle the text mismatch internally. This thesis examines the use of deep learning models for the components used in spoken dialogue systems. In particular, we have explored neural models for learning the semantic representations of natural language input, as well as for mapping from a user's query to a structured database. Instead of parsing a user query into a semantic fame and translating it into an SQL query, we let the neural model learn how to transform natural language input and database entries into points in a shared vector space, where semantically similar entities lie nearby.

1.1 Dialogue Systems

With the rise of conversational agents such as Siri and Cortana, dialogue systems that interact with people through spoken language are becoming more and more popular. With the abundance of data and more powerful computation available to us today, we can apply increasingly powerful models to machine learning problems. In particular, neural networks have been shown to outperform prior state-of-the-art statistical models in computer vision and speech recognition, and can learn to handle raw input without requiring any manual feature engineering. Thus, we apply neural methods to spoken dialogue systems, allowing the models to handle raw natural language internally, with minimal pre-processing. While the concept of learning semantic representations of natural language is not new, and has been explored extensively in dialogue systems and in many other NLP applications, our approach to *directly* map from a user query to a matching database entry without running an intermediate SQL query is, to the best of our knowledge, a novel technique.

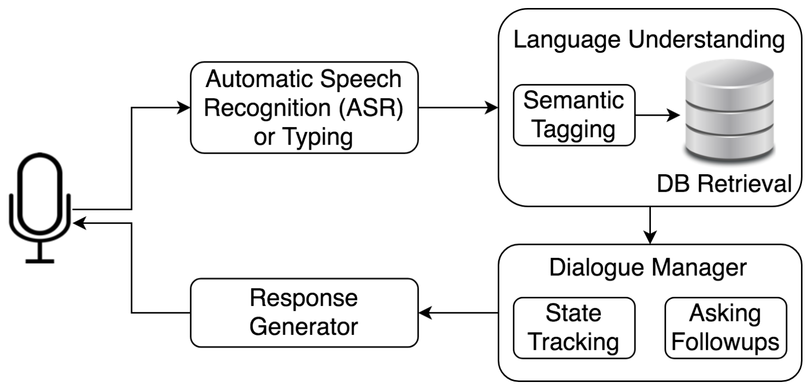


Figure 1-1: A standard pipeline for a spoken dialogue system, where the input spoken user query is passed to an automatic speech recognizer to generate its text, and the generated text or the input textual user query is sent to the language understanding component for pre-processing and semantic tagging. Then, the information is passed to the dialogue manager to update the state of the dialogue and predict the next action the system should take to generate a desired response.

The standard pipeline of steps in a spoken dialogue system (SDS) is shown in Fig. 1-1, where the user query (either written or spoken) is fed through an automatic speech recognizer if it is spoken, and the text of the query is sent to the language understanding component. In this component, the semantic tagging step involves isolating *slots* and corresponding slot *values* (e.g., in a flight booking system, the `departure city` slot may have the value `Boston`). The database retrieval then looks up the relevant information (e.g., flights) from a knowledge base. This information is passed to the dialogue manager, which updates the state of the dialogue (i.e., the user's goals, which are usually represented as a set of slots and matching slot values) and determines the next action the system should take, such as asking a followup question about the user's preferred departure time. Throughout this thesis, we will follow the steps in this pipeline, starting with language understanding, then moving onto database mapping, and finally finishing up with dialogue management by asking followup questions and tracking dialogue state.

We can classify dialogue systems into categories across several dimensions, which allows us to clearly define the problem to be solved, and facilitates comparing systems:

- **Rule-based vs. Statistical:** Early work on dialogue systems used rule-based approaches, whereas more recent work has used statistical models and reinforcement learning methods. In the 1960s, Weizenbaum at MIT developed the first computer therapist called ELIZA, which responded based entirely on hand-crafted rules.[4] As another example, in 2000, Wayne Ward and colleagues built the University of Colorado (CU) communicator automated travel agent system, which relied on parsing for natural language understanding, and a rule-based "event-driven" dialogue manager which mapped from the current dialogue context to the next action using a hierarchy of semantic frames (Pellom et al., 2000). Seneff at MIT built several systems, including the multilingual Voyager system for answering queries about objects in geographical regions (e.g., restaurants, hotels, libraries) in 1995 (Glass et al., 1995), Pegasus in 1994 for travel planning (Zue et al., 1994), the telephone-based Jupiter system for weather information in 1999 (Glass et al., 1999), and a more recent dialogue manager in 2010 that combined rules with a simple statistical model for flight reservations (Xu and Seneff, 2010). Then the field transitioned to statistical machine learning models (Lemon and Pietquin, 2012) and rein-

[4]https://en.wikipedia.org/wiki/ELIZA

forcement learning frameworks for dialogue systems, such as Levin et al. (1998), as well as Young et al. (2013) at the University of Cambridge, who demonstrated the success of POMDP-based methods (i.e., Partially Observable Markov Decision Processes) for SDS. Most recently, neural methods have been explored for specific system components, as well as end-to-end neural networks for the overall dialogue system pipeline (Wen et al., 2016; Li et al., 2017b). In our work, we continue the trend with deep learning models.

- **Task-oriented vs. Chatbot:** Typically, dialogue systems are either meant to accomplish a specific task, such as booking a flight (e.g., the well-known Air Travel Information System (ATIS) (Hemphill et al., 1990)) or reserving a restaurant (such as the University of Cambridge restaurant corpus for dialogue state tracking challenges[5]), or to entertain the user with fun chatting. The task-oriented systems usually involve multiple turns, while the chatbots are often only single-turn interactions, and rely on an encoder-decoder structure for generating a response with a recurrent neural network (RNN) decoder, given the input user query fed to the encoder network (Bahdanau et al., 2014). Xing et al. (2017) guide the decoder's generation with LDA topic information, Qiu et al. (2017) use the encoder-decoder to re-rank retrieved responses from an information retrieval mechanism, and Li et al. (2017a) use adversarial learning to generate responses that are indistinguishable from responses written by humans. In our case, we work in specific domains (e.g., nutrition) that have clearly defined tasks (e.g., food logging).

- **Single-turn vs. Multi-turn:** If a dialogue requires slot filling for accomplishing some task (e.g., determining departure and arrival cities for booking a new flight), the conversation will last for multiple rounds (or turns), going back and forth be-

[5] http://camdial.org/ mh521/dstc/

tween the user and the system. On the other hand, a dialogue may consist of only a single turn, which is typically the case in information retrieval and question answering (e.g., Google search or asking Siri a question). Information retrieval systems have a task very similar to our database mapping task—they rank the similarity of Web documents to a user query (typically with weighted term-based ranking algorithms such as BM25F (Craswell et al., 2005), efficient q-gram character matching algorithms for string similarity search (Hu et al., 2015), or key-value memory networks that leverage key-value pairs stored as memories from a knowledge base of facts (Miller et al., 2016; Eric and Manning, 2017)), whereas we rank possible United States Department of Agriculture (USDA) food matches to a user's meal description. Our database food mapping task is reminiscent of single-turn dialogues, but with followup clarification questions to narrow down the search space as needed.

- **Structured vs. Unstructured:** Finally, the data used for training the system, and thus the types of interactions the system will be able to handle at test time, are usually either structured (e.g., slot filling in the flight and restaurant booking domains) or unstructured (e.g., Ubuntu or Twitter data). In our work, we allow the users to speak completely naturally when describing their meal, which allows our system to handle unstructured natural language user queries.

In summary, we propose novel *neural* techniques to accomplish the database mapping piece of *task-oriented* spoken dialogue systems with *unstructured* user queries. We emphasize that in our work, a key differentiator is how the training data for database mapping was *weakly annotated*—that is, the model was not told explicitly *where* the foods were mentioned in the meal description, but only *which* foods were mentioned in the meal. The complexity of the problem can be defined as the number of slots to be filled for semantic tagging (i.e., five in the nutrition domain, 17 for restaurants, and 127 for flight booking), the number of database matches that need to be ranked (i.e., 220,077 total entries in our food database), and the average length of a user utterance and vocabulary size (i.e., 6.36 words on average per meal, 88 words maximum in a user query, and 14,788 words total in the meal descriptions we collected on the crowdsourcing platform Amazon Mechanical Turk (AMT), where workers, also known as Turkers, are paid to complete tasks for data collection and annotation). Thus, given the scope of the problem, we believe our techniques will apply to the following applications outside of nutrition:

- Retrieving the best answer (or most similar previously asked question, and its corresponding answer) to a question on a forum (e.g., SemEval[6] or StackOverflow posts).
- Mapping patients' notes and other information in electronic health records (EHR) to a medical database of relevant facts (e.g., drug or disease codes in ICD-9[7]).
- Recommending restaurants (or movies) given a user's description of what they want to eat (or watch), and a database of movie summaries[8] (or restaurant reviews[9]).
- Determining which action to take, given a user query (e.g., "make the woman's shirt blue" in Photoshop, "turn on the Wimbledon" for watching TV, "dim the lights" for smart light bulbs, or "play X song" on the radio).
- Mapping a user query to a robot or self-driving car command.

1.2 Thesis Contributions

The contributions of this thesis are four-fold:

[6]http://alt.qcri.org/semeval2019/index.php?id=tasks
[7]https://www.cdc.gov/nchs/icd/icd9cm.htm
[8]https://www.imdb.com/
[9]https://www.yelp.com/dataset

1. **Weakly supervised neural models for learning semantic vector representations:** We demonstrate that convolutional neural network (CNN) architectures are a viable approach for semantic tagging and, more importantly, learning semantically meaningful embeddings of natural language, all without requiring any manual feature engineering or pre-processing. The network was trained on weakly annotated data, without any pre-trained word embeddings or semantic tags, and was not told where the food items were located.

2. **Direct mapping from a user's natural language query to a structured database:** We leverage our learned CNN embeddings to map *directly* from a user query to matching database entries, bypassing the pipeline of string matching heuristics typically used in dialogue systems for API calls to a database. There are no constraints on how the user queries the system—they are encouraged to speak completely naturally, which the model learns to handle automatically.

3. **Deep reinforcement learning for asking followup clarification questions:** We establish that an RL agent achieves a balance between recall and ease-of-use, asking questions that result in higher food recall than a rule-based baseline, while asking fewer questions per dialogue than an entropy-based solution. The RL agent was rated by humans as significantly more natural and less frustrating than the baselines.

4. **Fully functional system prototype for diet tracking on iOS:** We applied our technology to the real-world problem of diet tracking, simplifying the food logging process through speech. We built and deployed a complete food logging application for iOS, which was downloaded by almost 1,000 users within a month of the launch date, and allows us to collect data in the wild from real users to retrain our models.

To assess the deep learning methods we have developed for semantic tagging and mapping from natural language input to structured databases, we have applied this technology in the nutrition domain, which we motivate in the next section. However, our technology is general enough that it can be applied to any domain in which a spoken query must be mapped to a database in order to respond appropriately to the user.

1.3 Chapter Overview

The remainder of this thesis is organized as follows:

- Chapter 2 provides background on artificial neural network models.
- Chapter 3 motivates our focus on the nutrition domain for spoken food logging, and describes the corpora we have collected and annotated for training our models.
- Chapter 4 presents our deep learning approach to semantic tagging of natural language meal descriptions, showing that these perform similar to or even better than prior state-of-the-art statistical models requiring hand-crafted feature engineering.
- Chapter 5 illustrates our novel convolutional neural network architecture for learning a shared semantic embedding space of natural language for mapping directly from user-described foods to structured database food matches.
- Chapter 6 describes our deep reinforcement learning agent for asking followup clarification questions to narrow down the food database search space.
- Chapter 7 demonstrates that our neural architecture generalizes to other tasks and domains, specifically dialogue state tracking and system response selection.
- Chapter 8 summarizes our findings, discusses our ongoing development and evaluation of a full system prototype, and presents several directions for future work.

- Appendix A shows several full interactions with the system, illustrating limiting cases where it does not properly understand the user.
- Finally, we include a glossary of acronyms with their definitions.

Chapter 2

Background: Artificial Neural Networks

Neural network models have become increasingly popular not only in the NLP community, but in the fields of speech recognition and computer vision as well. Deep learning is advantageous because neural networks have the ability to learn their own features, without requiring hand-crafted feature engineering. They are also quite powerful, often outperforming linear classifiers on complex tasks, since they are composed of multiple stacked layers, each with a nonlinear function. In particular, dialogue research has shifted dramatically away from statistical and rule-based approaches to newer neural models such as the encoder-decoder network. Prior state-of-the-art methods for semantic tagging utilized conditional random field (CRF) models (Sutton et al., 2012), or more recently, RNN models (Hochreiter and Schmidhuber, 1997). While we investigate all these approaches, the method we have chosen for our system is a CNN (LeCun et al., 1995), which is commonly used in computer vision tasks, requires fewer parameters and is faster to train than RNNs, and enables interpretation by inspecting the learned convolutional filters for patterns. We will demonstrate the success of CNNs on semantic tagging and database mapping.

2.1 Feed-Forward (FF) Neural Networks

An artificial neural network (ANN) is called "artificial" because it only approximate the neurons in the human brain. ANNs are composed of layers of connected nodes, where each input is multiplied by weights (which are like the dendrites of neurons in the brain) that are learned through a standard machine learning (ML) training process. However, the key difference between neural networks and prior statistical models is that neural networks apply non-linear activation functions (similar to the decision the brain neuron makes to fire, based on whether the electrical state exceeds some threshold), and can have multiple layers stacked on top of each other (which is why it is referred to as "deep" learning). This makes neural networks very powerful, but also require powerful computation, large training data, and the right training algorithm (i.e., backpropagation), which is why they only recently became successful, despite having been developed decades ago.

The most basic type of neural network is the feed-forward neural network (FF), or multi-layer perceptron (MLP), in which each node in the previous layer is connected to every node in the subsequent layer, as shown in the example in Fig. 2-1. During training, labeled examples are fed to the model, which predicts its output, and then updates the weights for each node by propagating gradients of the loss function, with respect to the ground truth label, backwards all the way through the entire network, using the chain rule to multiply incoming gradients by outgoing gradients. In the diagram, the values of the hidden layer h are computed by the following equation

$$h = f(W_h x + b_h) \tag{2.1}$$

where input x is multiplied by the edge weights W_h and summed with a bias term b_h, and then a non-linear activation function f is computed. This function might be the popular rectified linear unit (ReLU) that is 0 for values of x below 0 and linear for all positive values, sigmoid for binary outputs (i.e., with an output range of [0, 1]), or hyperbolic tangent (tanh) with output in the range [-1, 1]. The output o is computed as the softmax of the hidden layer's nodes h multiplied by another set of weights W_o, as in the equation

$$o = \mathrm{softmax}(W_o h + b_o) \tag{2.2}$$

where the softmax outputs probabilities of each possible output label, as follows:

$$\mathrm{softmax}(y_i) = \frac{\exp(y_i)}{\sum_j \exp(y_j)} \tag{2.3}$$

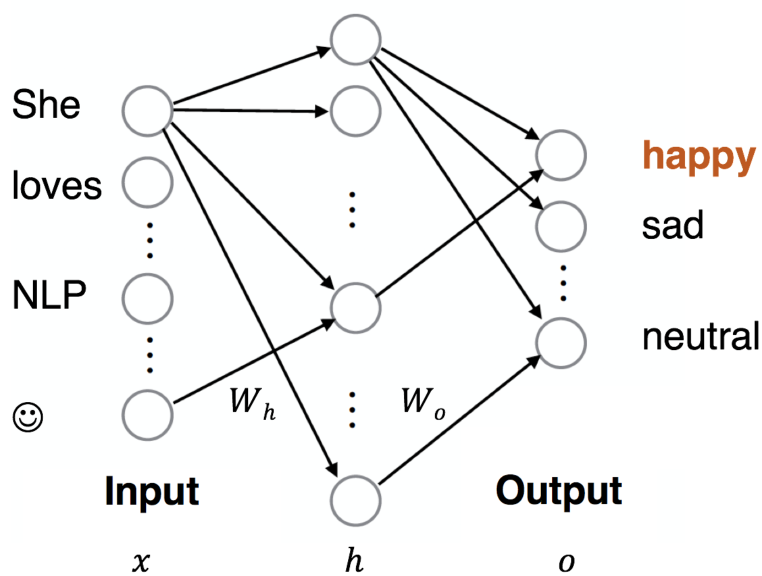


Figure 2-1: A simple example of a feed-forward neural network, in which each node in the input layer (i.e., the sentence "She loves NLP :)" is connected to every node in the hidden layer, which then pass their output to every node in the final output layer that predicts a probability for each possible sentiment label (i.e., happy, sad, or neutral).

2.2 Recurrent Neural Networks (RNN)

In FF networks, information flows in one direction only, from input to output, and each input goes through the network independently from the other inputs. However, in some cases, it is important for the network to "remember" previous timesteps. For example, in machine translation, the model needs to know which word was previously generated before predicting which word comes next. Recurrent neural networks enable this kind of memory by incorporating a feedback loop, where the output from the hidden layer at the previous timestep is fed into the current timesteps hidden layer, along with the input. The vanilla RNN, along with its long short-term memory (LSTM) and gated recurrent unit (GRU) variants that address the vanishing/exploding gradients problem (Hochreiter and Schmidhuber, 1997; Gers et al., 2003), have become popular in speech recognition and natural language processing, including semantic tagging (Mesnil et al., 2013a; Yao et al., 2013). A simple RNN is depicted in Fig. 2-2, where the edges connecting the previous timesteps to the subsequent timesteps are clearly illustrated, resulting in the following equation for the hidden state, where another set of weights U_h are learned for multiplying to the previous hidden state h_{t-1}:

$$h_t = f(W_h x_t + U_h h_{t-1} + b_h) \tag{2.4}$$

The LSTM uses a gating mechanism for explicitly remembering and forgetting specific information, in order to address the vanishing gradient problem, where if the history is too long and the gradient is less than one, the gradients become smaller and smaller as they are multiplied together and passed further down the network to earlier timesteps. The LSTM memory cell shown in Fig. 2-3 has a forget gate f_t which outputs a sigmoid σ

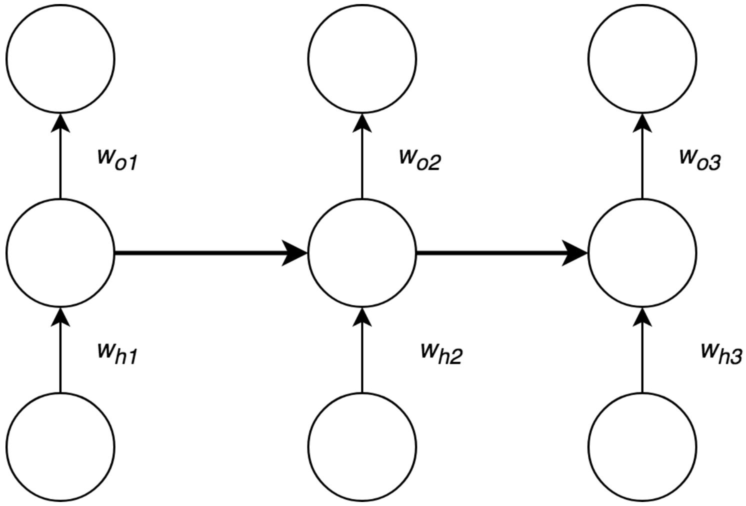


Figure 2-2: A simple illustration of an RNN, where there are recurrent connections (shown in bold) between hidden layers at each subsequent timestep to remember the past.

value between 0 and 1 indicating how much to remember from the previous cell state c_{t-1}:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.5}$$

The input gate i_t decides how much of the candidate memory cell $\widetilde{c_t}$ to use in computing the new memory cell c_t, following these equations:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.6}$$

$$\widetilde{c_t} = \tanh([h_{t-1}, x_t] + b_c) \tag{2.7}$$

$$c_t = f_t \times c_{t-1} + i_t \times \widetilde{c_t} \tag{2.8}$$

Finally, the output gate o_t determines how much of the new memory cell c_t is passed

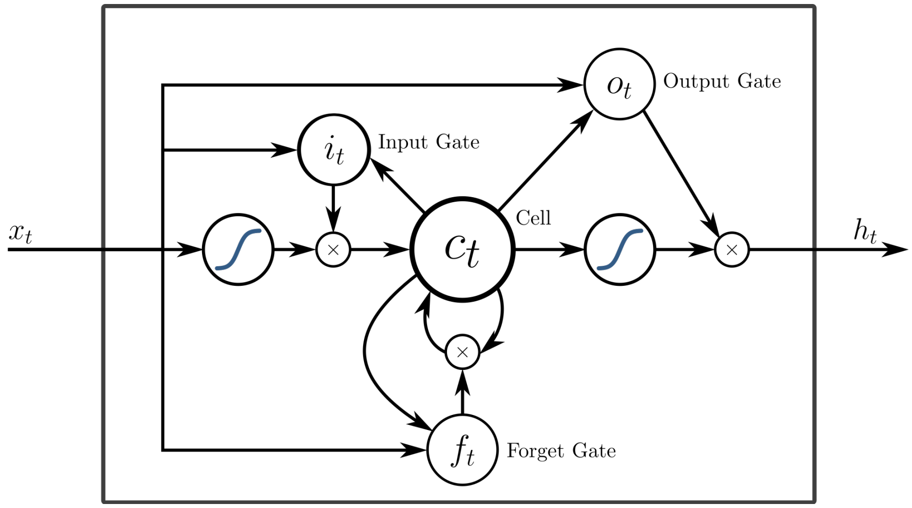


Figure 2-3: The LSTM memory cell, with input, forget, and output gates to control which information is retained from one timestep to the next (Hochreiter and Schmidhuber, 1997).

through to the output of the hidden layer h_t:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.9}$$

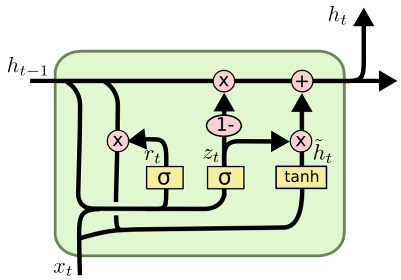
$$h_t = o_t \times \tanh c_t \tag{2.10}$$

The popular simplified variant, called the GRU, is displayed in Fig. 2-4,[1] with only one update gate z_t that are the input and forget gates combined.

2.3 Convolutional Neural Networks (CNN)

CNNs, which were originally developed for computer vision, have also been successfully applied to speech and NLP tasks such as sentence matching (Yin et al., 2016b) and ma-

[1]http://colah.github.io/posts/2015-08-Understanding-LSTMs/



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 2-4: The gated recurrent unit (GRU), which merges the input and forget gates into a single "update" gate z_t, simplifying the LSTM that had three gates.

chine comprehension (Yin et al., 2016a). Recent work has shown significant performance improvement over previous state-of-the art text classification techniques using very deep character-level CNNs (Conneau et al., 2017). Whereas for images the CNN learns filter maps that apply 2D convolutions over regions of images, CNNs can also learn filters that apply 1D convolutions to sequences of words in a sentence. We show a depiction of the convolution process,[2] where a learned kernel, or filter (in gray) is slid across the input (in blue), to generate the output (in green). The empty spaces indicate padding. Typically, each convolution layer is followed by a pooling layer, and many layers can be stacked so that the higher layers see input from increasingly wider regions of the input and, thus, may have a higher-level understanding of the input. For example, in computer vision, lower-level layers may pick up on edges, whereas higher layers may identify objects.

2.4 Attention-based Transformer Network

Recent work has demonstrated that recurrence may not be necessary for state-of-the-art performance on NLP tasks, and that an attention mechanism is all that is required (Vaswani et al., 2017). The Transformer network, shown in Fig. 2-6, is an encoder-decoder architec-

[2] https://github.com/vdumoulin/conv_arithmetic

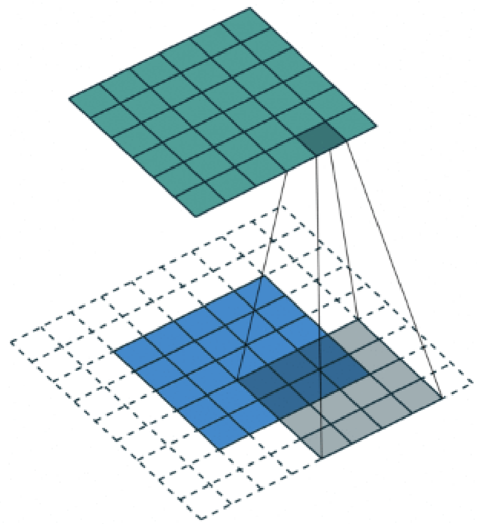


Figure 2-5: An illustration of the convolution process with a 2D filter, or kernel (in gray), that is slid across the input (in blue), and the resulting output (in green), with padding.

ture with N stacked layers, each containing multiple self-attention heads, followed by a feed-forward layer. The self-attention mechanism allows each layer to focus (or "attend") to the positions from the previous layer that are most relevant, resulting in a weighted sum of the inputs from the previous layer, where the weights are the attention values. In addition, there is the standard attention mechanism found in sequence-to-sequence (seq2seq) models between the encoder's output and the decoder, which learns where to pay the most attention over all positions in the input while generating its output.

2.5 Word Embeddings

For NLP tasks, a key component of neural networks is the first layer, or the word embedding layer, that maps from input words to vectors (or embeddings) of numbers. These embeddings, which are essentially a lookup table from words in a vocabulary to vec-

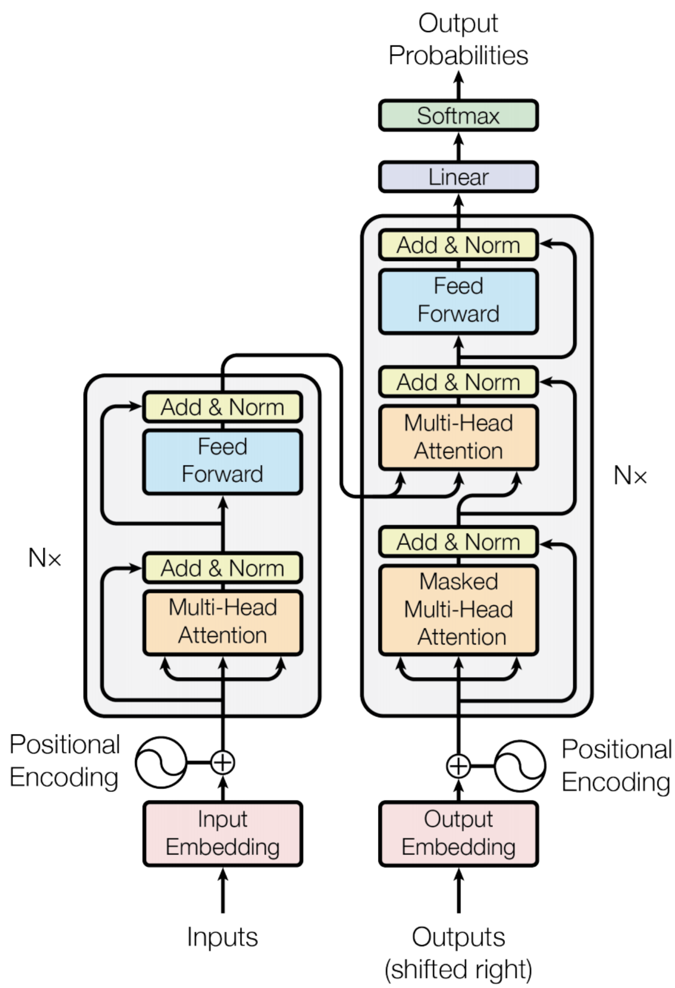


Figure 2-6: The Transformer network (Vaswani et al., 2017), with N stacked layers for both the encoder and decoder, composed of multiple self-attention heads and FF layers.

tors of numbers, are often learned through training a neural network to perform language modeling on a large dataset (e.g., Wikipedia or Google News), and can then be used as pre-trained vectors to initialize the word embedding layer of other NLP tasks, or they can be learned from scratch on the task of interest. There are many approaches to learning pre-trained word embeddings. One popular approach is known as word2vec (Mikolov et al., 2013b), where the objective was to predict the surrounding words given the vector of the center word w_t, as shown Fig. 2-7.

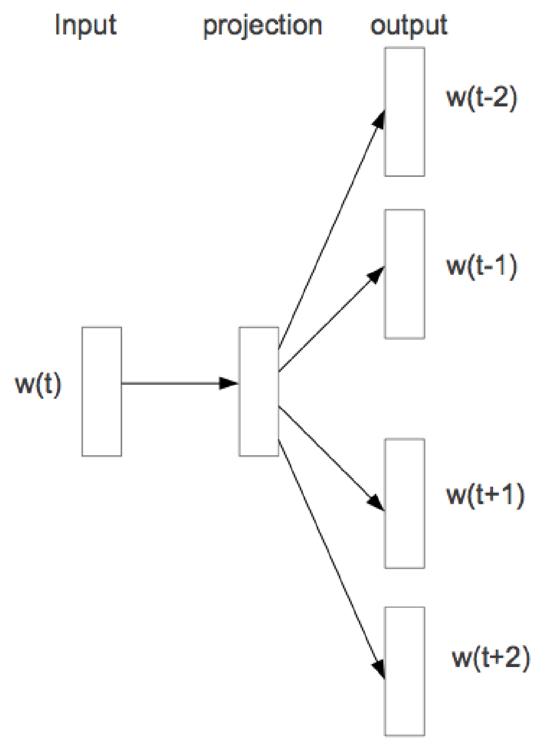


Figure 2-7: The word2vec skip-gram architecture, where the objective is to learn word vectors that are good at predicting nearby words (Mikolov et al., 2013b).

The notion of learning vectors that are good at predicting the context of a word is known as the "distributional hypothesis," where words in similar contexts tend to have the same meaning. Learned word embeddings tend to have interesting geometric properties such that indicate relationships among word meanings. For example, in the plot in Fig. 2-8, the vectors for countries all lie on the left-hand side of vector space, while their capital cities lie in a similar location on the right-hand side. This is all learned simply from the training data, without explicitly telling the model anything about these concepts or their relationships. This allows us to do simple arithmetic with word vectors, such as `vec(man)` $-$ `vec(king)` $+$ `vec(woman)` to get the embedding of `queen`.

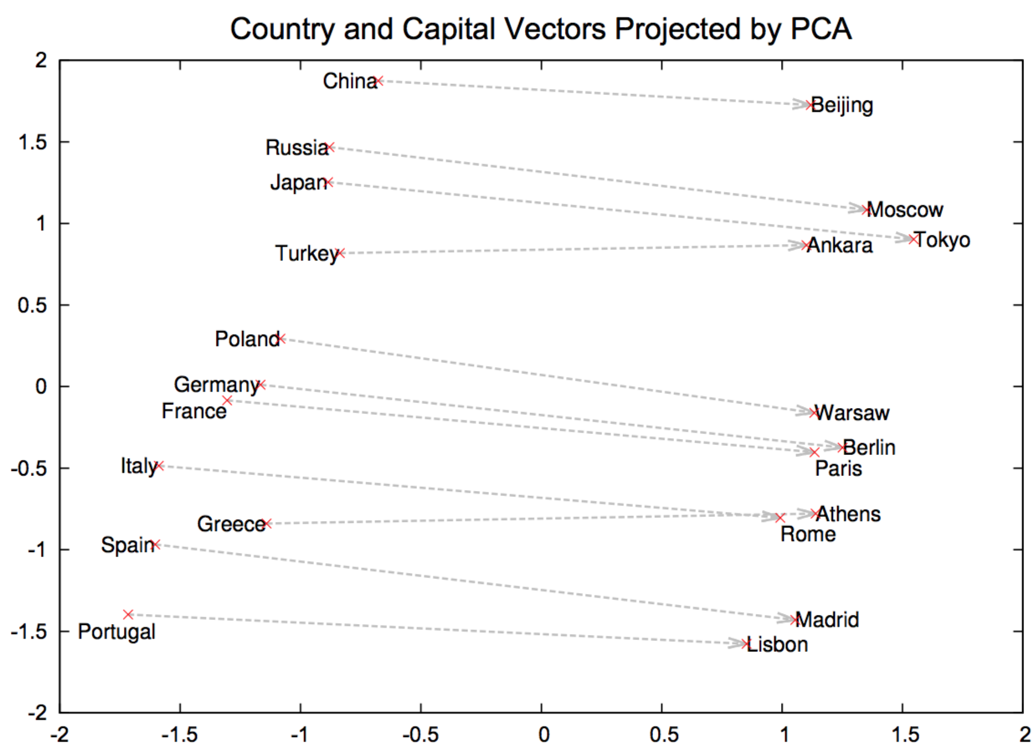


Figure 2-8: A plot of country and capital city words' vectors learned by word2vec, and reduced to two dimensions with PCA.

2.6 Summary

In conclusion, there are many different architecture options for neural networks, but the choice depends on the task at hand. While we explore all of these deep learning architectures in our work, we largely choose to focus on CNNs due to their comparable empirical performance on the tasks and domains we are interested in, with the added benefit of fewer parameters and faster training time. Finally, it is important to keep in mind that while deep learning is a popular method now in ML, this may not always be the technique of choice, and there are limitations. For example, it is still difficult for neural networks to explain *why* they predicted a certain answer, whereas prior statistical models were more interpretable. Hence, we will show throughout this thesis that by inspecting the learned convolutional filters of our CNNs, we do gain some interpretability.

Chapter 3

Background: Nutrition Database and Corpora

3.1 Motivation for Diet Tracking

Today, many Americans are trying to eat more healthy, nutritious diets. This is for a variety of reasons. Some are worried about reducing specific nutrients' intake due to disease; for example, diabetics must limit their carbohydrates and watch their protein. In addition, the rising obesity rate in the United States has led many to worry about losing weight. Adult obesity increased from 13% to 32% between the 1960s and 2004 (Wang and Beydoun, 2007), and over one-third of American adults today are obese (Ogden et al., 2014). This led to an estimated medical cost of $147 billion in 2008 (Finkelstein et al., 2009). Some are following popular diets such as vegan, gluten-free, and Paleo. Still others are simply interested in staying healthy and fit or maintaining their current weight. For all these reasons, nutrition has become a popular area of research in fields such as computer vision and NLP, and many smartphone apps are related to health and nutrition.

However, despite the plethora of diet tracking programs already in existence, the primary challenge with all of them is they are time-consuming and difficult to use. People often give up after only a few days, which makes it hard to sustain long-term weight loss goals. The popular diet tracking app MyFitnessPal, which has over 100 million users, found that users who manage to come within 5% of their weight loss goals use the app for six consecutive days on average, whereas the other users only continue for three consecutive days.[1] Thus, a group of nutritionists at Tufts University approached us five years ago and requested that we collaborate on a project to build a system that uses speech as the input modality for quick and easy diet tracking.

We believe nutrition is an importation application that is still relatively unexplored from an NLP research perspective, and we propose applying cutting-edge conversational agent technology to this domain. We accomplished our goal of merging these two bodies of work (dialogue and nutrition) into a novel system that has the potential to fundamentally change the way people interact with diet tracking applications, enabling them to more easily and efficiently record their dietary intake in order to become more fit and healthy.

3.2 Nutrition Corpora

The first step for any ML task is to gather the data for training the models. Here we describe our two primary datasets: structured food databases, and natural language meal descriptions. In addition, we discuss our spoken corpus, as well as quantity data.

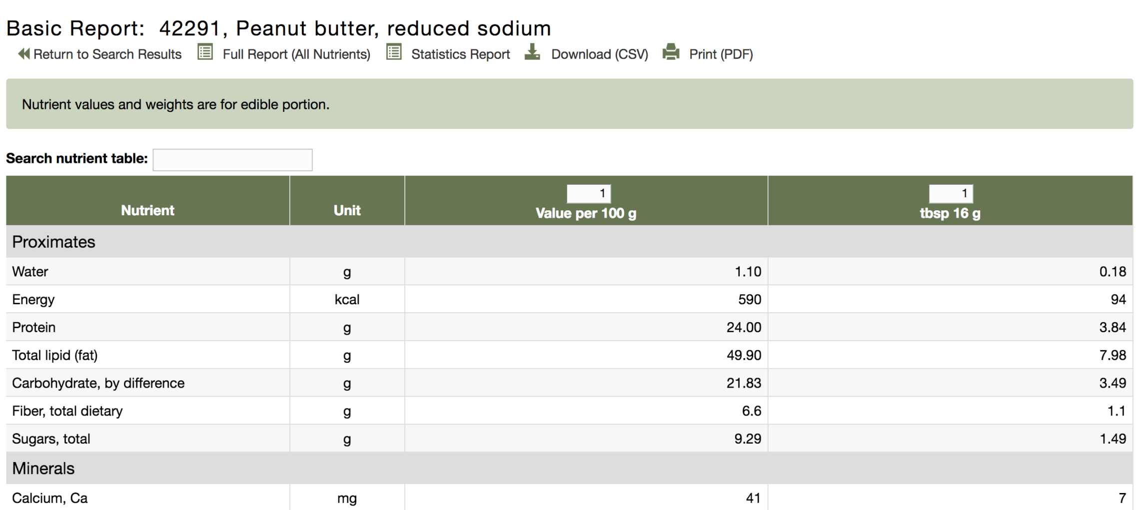
[1] https://www.wsj.com/articles/new-reasons-why-you-should-keep-a-food-journal-1463419285

3.2.1 Structured Food Databases

Ultimately, a diet tracking system responds to the user with nutrition facts for foods they ate. Thus, we need data with nutrient information for various foods. There are two databases with generic and branded food items that we used, and have combined into our back-end Postgresql database: USDA Standard Reference (SR) and USDA Branded foods. The data statistics are shown in Table 3.1, and a screenshot of the online USDA database information for a particular food item is shown in Figure 3-1.

Database	Number of Foods
USDA SR	8,911
USDA Branded	211,166

Table 3.1: Structured food database statistics, where number of "foods" refers to the total number of food entities in each database, either USDA Standard Reference or Branded food products. See Figure 3-1 for an example food entry, with all its nutrition facts.


Basic Report: 42291, Peanut butter, reduced sodium

Return to Search Results | Full Report (All Nutrients) | Statistics Report | Download (CSV) | Print (PDF)

Nutrient values and weights are for edible portion.

Search nutrient table:

Nutrient	Unit	1 Value per 100 g	1 tbsp 16 g
Proximates			
Water	g	1.10	0.18
Energy	kcal	590	94
Protein	g	24.00	3.84
Total lipid (fat)	g	49.90	7.98
Carbohydrate, by difference	g	21.83	3.49
Fiber, total dietary	g	6.6	1.1
Sugars, total	g	9.29	1.49
Minerals			
Calcium, Ca	mg	41	7

Figure 3-1: An illustration of the data contained in the USDA database for each food entry.

3.2.2 Natural Language Meal Descriptions

Since our primary tasks involve semantic mapping of natural language meal descriptions and mapping from natural language input to structured databases, we collected meal descriptions on the AMT crowdsourcing platform and annotated them for semantic tags (i.e., food, quantity, other) and matching food database entries.

In order to generate reasonable meal description tasks, we partitioned the over 5k foods in the USDA database into specific meals such as breakfast, dinner, etc. (see Table 3.2). A given task was randomly assigned a subset of 9-12 food items from different categories. To reduce biasing the language used by Turkers (i.e., an AMT worker), we included images of the food items along with the less natural USDA titles (see Fig. 3-2). Turkers were asked to select at least three of the foods, and generate a meal description using these items (see Fig. 3-3 for the actual AMT interface). This enabled workers to select foods that would typically be eaten together, producing more natural meal descriptions and quantities.

Meal	**# Foods**	**# Diaries**	**# Words per Diary**
Breakfast	1167	4010	18.8
Dinner	2570	3850	21.6
Salad	232	4040	19.1
Sandwiches	375	4000	20.1
Smoothies	384	3850	20.1
Pasta/Rice	1270	4000	20.6
Snacks	1342	4077	19.1
Fast Foods/Meals	669	3886	19.1
All Foods	5124	31712	19.8

Table 3.2: Meal description statistics, organized by category.

Instructions

Using the images below, create a meal as if you had eaten some of the foods depicted:

- You've eaten **at least THREE** of the items in the images
- **Check the boxes** of the items you use
- Database names are provided, however, **describe the foods how you would normally say them** (i.e. Cereals, Oats -> Oatmeal)
- You should try to construct a **SMOOTHIE**, however if another meal would make more sense make that
- **Specify** appropriate **quantities** (i.e, a cup, a bowl, two tablespoons, one)
- **Do not just list** the food items
- **Do not add** additional items to the meal
- If the same image appears more than once, just select one. It doesn't matter which
- You can assume raw food or mixes have been prepared

x
Oranges, Raw, All Commercial Varieties

Corn, Sweet, Yellow, Cooked, Boiled, Drained, Without Salt

x
Strawberries, Raw

Melons, Casaba, Raw

Mushrooms, White, Raw

x
Kiwifruit, Green, Raw

Sweet Potato, Cooked, Baked In Skin, Flesh, Without Salt

Carrots, Cooked, Boiled, Drained, Without Salt

x
Bananas, Raw

GOOD MEAL DESCRIPTIONS:

- Today I ate a smoothie made with a handful of grapes, a can of peaches, an ounce of frozen strawberries, and half a cup of low fat fruit yogurt.
- For breakfast I made a smoothie with 8 ounces of low fat fruit yogurt, an ounce of frozen strawberries, a dozen grapes, and half a can of peaches.

BAD MEAL DESCRIPTIONS:

- I made a smoothie with a dozen frozen strawberries, a frozen banana, an ounce of fruit yogurt and a tablespoon of honey ——— ***[Adds banana and honey, lacks grapes, and peaches]***
- Grapes, Canned Peaches, Fruit Yogurt, and Strawberries Frozen.——— ***[Just lists the food items, specifies no quantites]***

Figure 3-2: The instructions and example meal diaries shown to workers in our AMT data collection task.

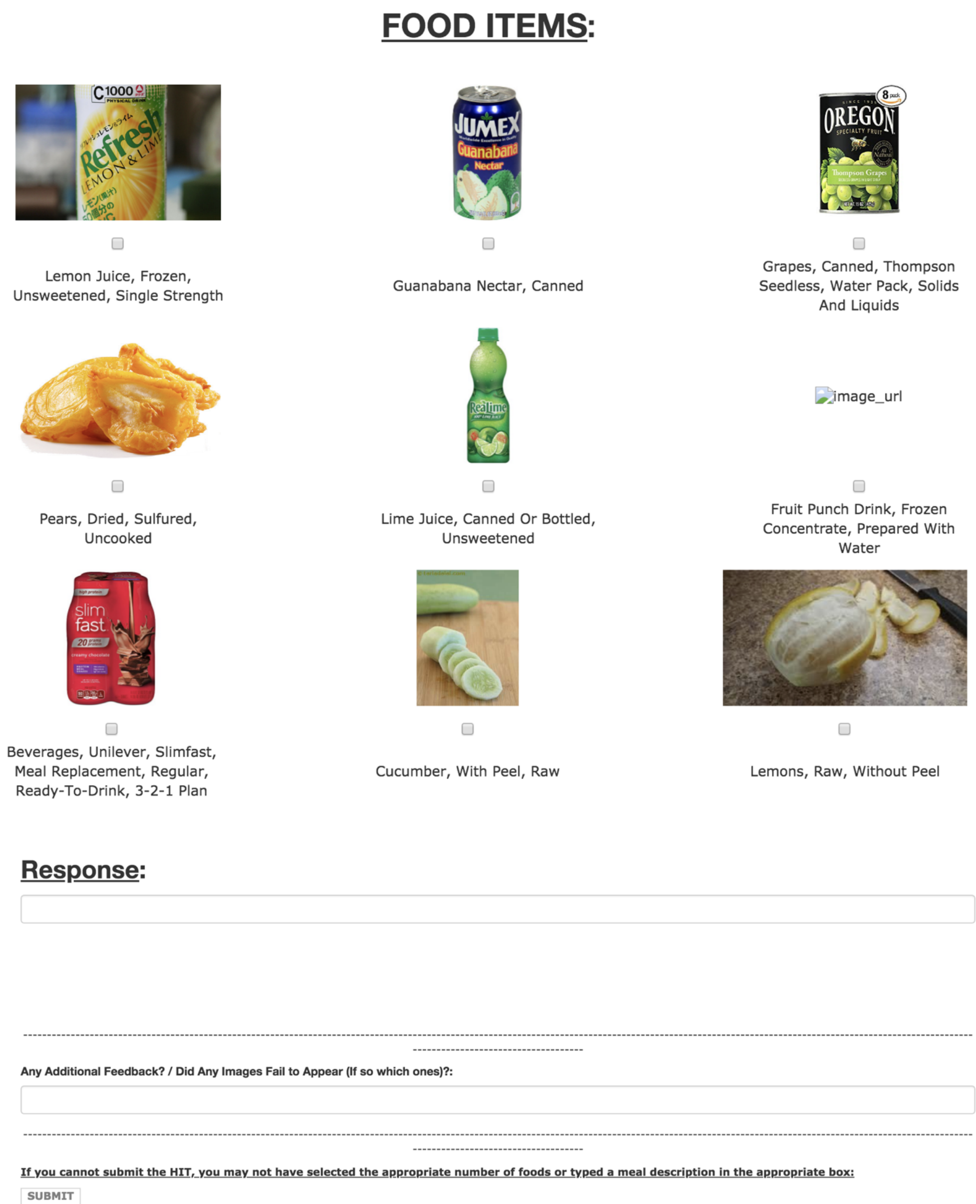


Figure 3-3: The AMT interface, where Turkers had to check off at least three foods, and write a "response" (i.e., meal description containing those food items).

Semantic Tagging Annotation

The diaries were then tokenized and used as input for the second phase, where we asked Turkers to label individual food items within the diaries. The third and final phase combined the meal descriptions with their food labels and prompted Turkers to label the brand, quantity, and description properties associated with a particular food item (Fig. 3-4) (Korpusik et al., 2014; Korpusik, 2015).

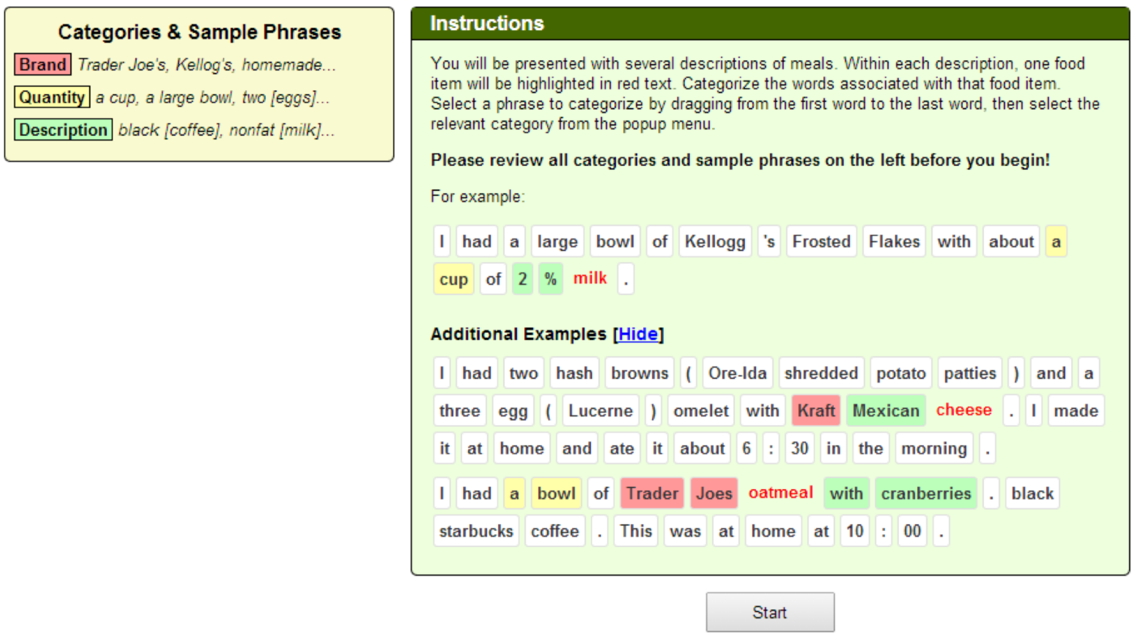


Figure 3-4: After Turkers labeled the foods, the final AMT task asked them to label properties (e.g., quantities in yellow) of a given food highlighted in red.

We labeled semantic tags for a total of 22,000 meal descriptions including breakfast, lunch, dinner, and snacks on AMT, which we used to train our models. The frequency of each tag is shown in Table 3.3. We measured the reliability of the data annotations by calculating the inter-annotator agreement among Turkers. Specifically, we calculated Fleiss' kappa scores for the two labeling tasks: 0.77 for food labeling, and 0.41 for property labeling. The score for the food labeling task indicates substantial agreement. The score for property labeling is lower, but still indicates moderate agreement (Viera et al., 2005).

Label	**Frequency**
Food	76,399
Brand	13,826
Quantity	38,668
Description	46,898
Other	89,729

Table 3.3: Statistics for tokens assigned one of five possible labels out of the 22,000 collected meal descriptions.

We also incorporated algorithms for improving Turker labeling performance. In order to determine whether the food and property labels selected by the Turkers were reasonable, we automatically detected which tokens were foods or properties in each meal description and required Turkers to label these tokens upon submitting a property labeling task. If a token was missing, the submission error message would require the Turker to return to the task to complete the labeling more accurately, but would not reveal which were missing.

To automatically generate hidden food and property labels, we used a trie matching algorithm (Knuth et al., 1977) trained on the USDA food lexicon. A trie is an n-ary tree data structure where each node is a character, and a path from the root to a leaf represents a token. We built a variant of the standard trie where each node contains a token that is part of a USDA food entry, and a path from the root to a leaf represents an entire food phrase. For example, a node might contain the token "orange," and its child node might contain the token "juice" (see Fig. 3-5). Then, the matching algorithm would find every matching entry from the USDA trie that is present in a meal description. Since USDA food entries often contain only the singular form of a food token, we incorporated plural handling into the trie matching, using the Evo Inflector libary's implementation of Conway's English pluralization algorithm (Conway, 1998).

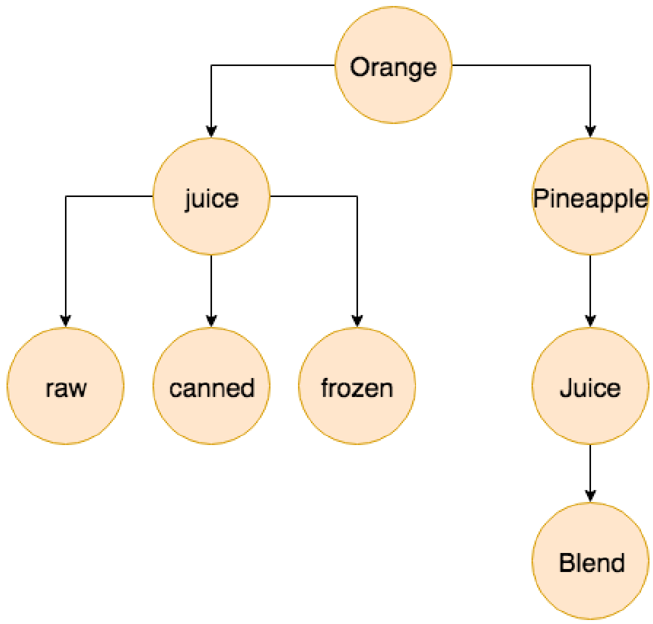


Figure 3-5: An example trie for four USDA foods that all start with “Orange”: *Orange juice raw, Orange juice canned, Orange juice frozen,* and *Orange Pineapple Juice Blend.*

3.2.3 Speech Corpus and Recognizer

The meal descriptions we have described thus far are all written, rather than spoken, whereas at test time the system must be able to handle spoken user input. To address this limitation, we collected a corpus of spoken meal descriptions, and created a nutrition speech recognizer (Korpusik et al., 2016a). We collected the speech data via AMT (Saylor, 2015), where we asked Turkers to record 10 meal descriptions. The diaries were selected from previously collected written meal descriptions, and spelling and grammar errors were manually corrected. The Turkers' recording, converted to text via a recognizer embedded in the AMT task, was required to contain at least 60% of the words in the transcript they were reading in order to submit the task.

We split the resulting 2,962 utterances (from 37 speakers totaling 2.74 hours) into 80% training, 10% development, and 10% test sets, and removed punctuation and capitalization from the text data for training the language model. Using Kaldi (Povey et al., 2011), we trained a fully-connected, feedforward, deep neural network (DNN) acoustic model and a trigram language model on 40,000 written sentences (this is a larger set than the original 10,000 meal logs (Korpusik et al., 2016a) because each meal was split into individual sentences). The DNN's input features were Mel-frequency cepstral coefficient (MFCC) (Vergin et al., 1999) that are the standard for speech recognition. The network was used in conjunction with a hidden Markov model (HMM) (Juang and Rabiner, 1991) recognizer that had 265 tied states; therefore it had 265 outputs. The DNN had 6 hidden layers, each with a sigmoid nonlinearity, followed by a softmax. The decoder had a word error rate (WER) of 7.98% on the test set. We then annotated the semantic tags and food-property associations of the recognizer's output on AMT, as described in subsection 3.2.2 for subsequent understanding evaluation.

3.2.4 Quantity Data

Since we thus far have only collected gold standard labels for *food* database matches, we then also collected over 99k food *and quantity* descriptions. To collect quantity descriptions, we revised the AMT task such that workers were told to select one quantity option from among all the database quantity units available for a given food item. They were instructed to describe this quantity naturally (e.g., *two cups of*), and in a separate textbox, to describe the food item (e.g., *chopped kale*). To reduce biasing the language used by workers, we included images of the foods along with the less natural USDA titles.

For our evaluation on speech data, we collected 9,600 spoken meal descriptions on AMT (1,200 for each of the eight meal categories), using the Google speech recognizer API. The data was collected the same way as the text data, but with speech instead of text, and as a single description for each combined food and quantity.

Meal	**# Quantities**	**# Foods**	**# Diaries**
Breakfast	616	1,477	33,317
Dinner	613	2,556	23,094
Salad	173	232	2,446
Sandwich	234	372	4,474
Smoothies	214	382	5,789
Pasta/Rice	366	1,262	12,715
Snacks	725	1,334	12,041
Fast Food	271	661	5,474
All Foods	1,562	5,156	99,350

Table 3.4: AMT quantity data statistics, organized by meal.

Now that we have motivated the diet tracking problem and described the corpora, we are ready to move onto the three core chapters of this thesis: language understanding of written and spoken meal descriptions, food database mapping, and asking followup clarification questions in order to quickly narrow down the food database search space.

Chapter 4

Spoken Language Understanding

We begin the journey of building a spoken food logging system with spoken language understanding (SLU)—the first step in the pipeline after recognizing the speech of the user utterance, as in Fig. 1-1. Specifically, SLU entails identifying relevant slots and their associated values, also known as semantic tagging or slot filling. For example, in Fig. 4-1, "chocolate" is labeled as `B-Description` (for beginning of the Description semantic tag), and "chip granola" is assigned the semantic tag `I-Food` (for inside the Description). Given these tags, the system then decides which action to take next and how to respond.

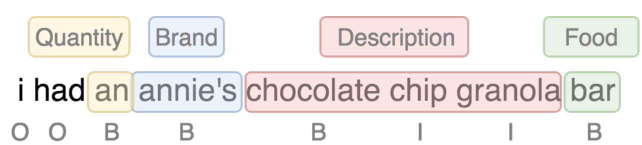


Figure 4-1: Semantic tagging on a written meal description, where "chocolate," "chip," and "granola" are all a Description tag for the food "bar," and where "B" is the beginning of a new semantic tag, "I" is inside a semantic tag, and "O" is other.

We investigate a suite of neural network models for tagging foods, brands, quantities, and descriptions in the natural language meal descriptions (described in Section 3.2.2).

As we noted earlier, these models have the benefit of not requiring hand-crafted feature engineering. However, they are sensitive to hyperparameters; thus, in the last section of this chapter, we show that neural models generalize to other domains without fine-tuning.

This chapter is organized as follows. First, we compare a suite of neural networks (including recurrent, convolutional, and the recently proposed Bidirectional Encoder Representations from Transformers (BERT) model) on written and spoken meal descriptions, demonstrating that the best CRF performs best on written meals, but that the neural models (in particular, a single pre-trained BERT model with fine-tuning of a token classification layer on top) are more robust to speech recognition errors in the spoken test data. Error analysis shows that the models are less confident when making errors, enabling the system to follow up with the user when uncertain. Finally, at the end of this chapter, we also establish that the neural models easily generalize to new domains, since we port the models from the nutrition domain to the ATIS benchmark and a restaurant corpus, without any task-specific fine-tuning.

4.1 Related Work

The SLU literature largely focuses on the ATIS corpus (Hemphill et al., 1990; Dahl et al., 1994), which is composed of spoken queries about flight information. For example, understanding the query "I want to fly to Boston from New York next week" involves identifying the goal as airfare and slot values of Boston, New York, next, and week for the slots departure city, arrival city, relative departure time, and departure time period, respectively.

Research on ATIS has moved from early work involving handmade template matching requiring expensive grammars to more data-driven methods. He and Young (2003) showed improved performance using the expectation maximization algorithm for a generative hidden vector state (HSV) model over hand-crafted semantic grammar rules. Wang et al. (2006) used a discriminative CRF rather than generative models to reduce slot error rate by over 20%. Raymond and Riccardi (2007) similarly demonstrated that the addition of a-priori long-term dependency features in CRF models led to better performance than the generative finite-state transducer, and Meza-Ruiz et al. (2008) also showed that global dependency features in discriminative models outperform the generative HSV model. This motivates our use of the CRF with its sequential processing. Heintze et al. (2010) demonstrated a performance improvement as incrementally longer utterance prefixes are seen by a classifier, and Tur et al. (2011) used dependency parsing to simplify natural utterances into more concise, keyword-style queries that are easier for classifiers to process. Tur et al. (2010) study analyzing the state-of-the-art on ATIS revealed common error patterns that were still unresolved, including long-distance dependencies in slot filling.

More recently, neural networks such as bidirectional RNNs (Mesnil et al., 2013b, 2015) and LSTMs (Yao et al., 2014a), and CNNs (Xu and Sarikaya, 2013), have been shown to outperform CRFs, which motivates our use of neural networks on our tasks. In addition, there has been work on jointly training RNNs for slot filling and intent and domain detection (Hakkani-Tür et al., 2016; Lee et al., 2018; Liu and Lane, 2016; Ma et al., 2017; Goo et al., 2018), as well as end-to-end neural networks for mapping directly from speech to semantic tags (Haghani et al., 2018), and memory networks to remember dialogue history (Chen et al., 2016).

A similar trend has been observed for named entity recognition (NER) as for SLU, where early work applied hand-crafted rules (Nadeau and Sekine, 2007), but transitioned to machine learning methods over the course of the 1990s and 2000s, including supervised learning such as hidden Markov models, support vector machines, and CRFs (Lin and Wu, 2009; Passos et al., 2014), as well as unsupervised clustering and semi-supervised learning from distributional semantics, given a set of seed entities. Recently, CNNs (Collobert et al., 2011) and LSTM-CRFs (Huang et al., 2015; Lample et al., 2016) were explored.

Within the past year, several papers have come out that learn *contextual* representations of sentences, where the entire sentence is used to generate embeddings. Embeddings from Language Models (ELMo) (Peters et al., 2018) uses a linear combination of vectors extracted from intermediate layer representations of a bidirectional LSTM trained on a large text corpus as a language model (LM); in this feature-based approach, the ELMo vector of the full input sentence is concatenated with the standard context-independent token representations and passed through a task-dependent model for final prediction. This showed performance improvement over state-of-the-art on six NLP tasks, including question answering, textual entailment, and sentiment analysis. On the other hand, the OpenAI Generative Pre-trained Transformer (GPT) (Radford et al., 2018) is a fine-tuning approach, where they first pre-train a multi-layer Transformer (Vaswani et al., 2017) as a LM on a large text corpus, and then conduct supervised fine-tuning on the specific task of interest, with a linear and softmax layer on top of the pre-trained Transformer (see Section 2.4). Finally, Google's BERT (Devlin et al., 2018) is a fine-tuning approach similar to GPR, but with the key difference that instead of combining separately trained forward and backward Transformers, they instead use a *masked* LM for pre-training, where they randomly mask out input tokens and predict only those tokens. They demonstrate state-of-the-art performance on 11 NLP tasks, including the CoNLL 2003 NER tagging task.

For the restaurant and nutrition tasks, prior state-of-the-art involved complicated CRFs with carefully hand-crafted features, such as semantic dependency features from query dependency parses (Liu et al., 2013b), and word vector and distributional prototype similarity features (Korpusik et al., 2016a). The motivation of this work is to *avoid manual feature engineering* with neural networks that *automatically* learn features. We demonstrate that our models are easily ported from one domain to another without fine-tuning.

4.2 Models

Although CRFs are a powerful discriminative classifier for sequential tagging problems, they require manual feature engineering. A popular alternative which does not require any feature engineering is a neural network. We investigated a collection of deep learning models for semantic tagging: RNNs, CNNs, the recently proposed attention-based BERT, CRF models trained on logits from the hidden layer of RNNs, and a FF baseline. We also ensembled various models, and initialized with pre-trained embeddings.

4.2.1 RNN

In our PyTorch (Ketkar, 2017) implementation of the RNN, we built a bidirectional GRU on top of a word embedding layer, with a linear layer on top for the final prediction. We used embeddings of dimension 128, hidden layers of size 512, batches with 50 samples each, and trained with the Adam optimizer on cross-entropy loss for 1,000 steps of randomly selected batches. The maximum length was set based on the longest sample in each batch. The FF baseline replaced the recurrent layer with a linear layer of size 512. We implemented the RNN-CRF as in (Yao et al., 2014b) by extracting logits from the trained RNN's hidden layer and feeding these as features to the CRF. Thus, the RNN-CRF does not require any manual feature engineering, unlike the CRFs discussed in Section 4.1.

4.2.2 CNN

A CNN window of 5 tokens can be interpreted as an n-gram of 5 tokens, which directly provides context similar to the features used in a CRF. We implemented variants of the CNN model in Keras (Chollet, 2015). Each model was composed of a word embedding layer initialized uniformly with 150 dimensions, followed by three stacked 1D convolutional layers, with kernel windows spanning lengths of five, five, and three tokens, respectively and 64 filters each. Finally, a fully-connected layer with a softmax activation predicts the semantic tag for each token. We used the Adam optimizer (Kingma and Ba, 2014), binary cross-entropy loss, and early stopping to prevent overfitting, where training continued for up to 15 epochs unless there was no performance gain on the validation set.

We experimented with pre-trained word embeddings from publicly available 200-dimension Glove (Pennington et al., 2014) (trained on Wikipedia and Gigaword), 300-dimension word2vec (Mikolov et al., 2013b) (trained on Google News), and 300-dimension FastText (Bojanowski et al., 2017) (trained on Wikipedia News) standard word embeddings. After initialization, the word embeddings were updated during training.

4.2.3 BERT

Finally, we used the base pre-trained BERT in PyTorch with a fine-tuned token classification layer on top and hyperparameters tuned on 10% validation data (i.e., batch of 32, uncased tokenizer, 3×10^{-5} learning rate, and four epochs). Since BERT uses word pieces, but the data is pre-tokenized, we use only the first sub-token's predicted label during evaluation. As shown in Fig. 4-2, the tokens (or word pieces) for each meal description are fed into the pre-trained base BERT model (i.e., with only 12 multi-head attention layers, instead of 24 in large), and a single classification layer is fine-tuned on top that uses a softmax to predict the semantic tag (or named entity in the original paper) for each token.

4.3 Experiments

We evaluated our deep learning models on a held-out test set and compared our performance to a state-of-the-art CRF model with hand-crafted features. Our metrics are preci-

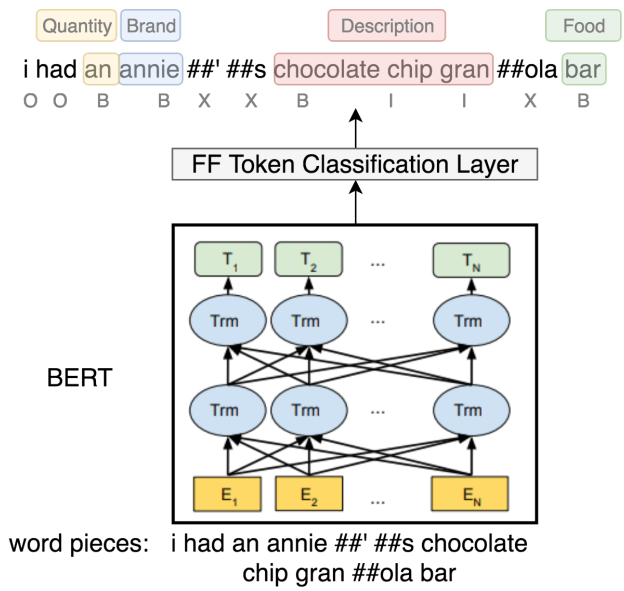


Figure 4-2: An illustration of how BERT is used for named entity recognition, or semantic tagging in our case (Devlin et al., 2018).

sion (i.e., the percentage of correct semantic tags out of all the predicted semantic tags), recall (i.e., the percentage of correct semantic tags out of all the true semantic tags), and F_1 score (i.e., the harmonic mean of precision and recall). We show results for both written and spoken meal descriptions in Tables 4.1 and 4.2, respectively.

One interesting result is that on spoken data, the neural networks actually outperform the best CRF, demonstrating their ability to perform well despite automatic speech recognition (ASR) errors. The poor performance on semantic tagging of brands for both corpora is likely due to the small number of brand tokens (i.e., only 3.4% of the test set's tokens are brands), as well as the difficulty distinguishing between brands and descriptions. Errors could be due to misrecognized brands; for example, "don julio tortillas" was incorrectly recognized as "on whole wheat tortillas," which is a description instead of a brand.

The second takeaway is that BERT outperforms the other neural models on both written and spoken meal descriptions, despite being pre-trained out of the nutrition domain

Model	Food	Brand	Num	Descrip	Avg
CRF (unigram)	92.3	78.5	93.9	86.6	92.4
CRF (+ bigram)	94.1	80.3	95.1	88.9	93.7
Best CRF (Korpusik and Glass, 2017)	94.6	85.7	**95.1**	90.3	**94.4**
FF baseline	85.1	69.0	91.0	74.4	85.3
1 RNN	94.3	77.2	94.2	87.1	92.1
RNN-CRF	94.4	76.8	93.1	86.9	91.7
4 RNNs	**95.1**	80.5	94.5	88.4	92.9
1 CNN	91.9	79.5	95.1	87.1	92.4
CNN + Glove	94.4	84.1	94.7	89.5	93.9
CNN + wd2vc	93.6	83.6	91.0	88.0	92.1
CNN + FastText	93.7	82.9	94.3	88.8	93.2
4 Glove CNNs	94.4	84.4	91.7	89.0	92.7
4 RNN + 4 CNN	76.9	78.3	94.5	89.1	85.3
BERT	94.6	**87.0**	94.7	**90.4**	94.2

Table 4.1: Per-label F1 scores on written meals (Korpusik and Glass, 2017). The CRF performs best, but it requires hand-crafted features, whereas the neural models are competitive without feature engineering. Although BERT is not the best overall, it does particularly well on brands and descriptions, which is hard, even for AMT workers, to distinguish.

Model	Food	Brand	Num	Descrip	Avg
Best CRF (Korpusik and Glass, 2017)	93.3	79.0	96.6	87.7	94.2
FF baseline	87.9	61.7	93.7	78.3	90.1
1 RNN	94.0	81.5	95.9	89.1	94.9
RNN-CRF	94.1	80.6	95.3	87.9	94.5
4 RNNs	94.4	80.9	97.2	89.8	95.3
1 CNN	93.9	78.2	97.5	89.1	95.1
CNN + Glove	94.9	80.9	97.5	91.1	95.5
CNN + wd2vc	95.4	80.6	97.2	90.9	95.7
CNN + FastText	94.8	74.8	97.6	90.4	95.0
4 Glove CNNs	95.1	82.2	97.0	91.4	95.6
4 RNN + 4 CNN	**95.5**	77.4	97.8	**91.7**	95.6
BERT	94.4	**85.3**	**97.8**	91.1	**95.8**

Table 4.2: Per-label F1 scores on *spoken* meal data. All the neural networks outperform the CRF, demonstrating that they are more *robust to speech recognition errors*.

and only adding a task-specific softmax layer on top for fine-tuning. This illustrates how learning a strong representation of language that can be shared among tasks and domains is more important than the previous approach of pre-training only the word embedding layer, and then building a task-specific model on top.

We hypothesize that the models will be less confident in their predictions when making mistakes. This has important applications for real-world tasks, in which the model could learn from human feedback when it is not confident in its prediction. For example, in a diet tracking application, if the model is uncertain about the tag for "oatmeal" in the food description "oatmeal cookie," it might ask, "Was the oatmeal a *description* of cookie, or a separate *food*?" Thus, the model could learn online from users without asking an overwhelming number of questions, only for clarification on those for which it is least certain. In addition, we could use the confidence of the model to discover errors in the test set labels if, for example, the model is very confident when it makes a mistake. Here we show that the models are indeed less confident when making errors, and that high confidence can be used to discover errors in the test data.

We see in Fig. 4-3 that as we increase the confidence threshold[1] from 0.5 to 0.999, the weighted average F-score on the written meal test set increases. In particular, during evaluation, we omit the examples where the model is uncertain, since the predicted tag's probability is below the specified confidence threshold. The performance improvement from eliminating examples where the model is less certain indicates that the model is more confident when its predicted tag is correct, and less confident when it makes errors.

Error analysis reveals that the model tends to have less certainty in its predictions when it is mistaken, and that high-confidence predictions may identify errors in the data annotation. We see in Fig. 4-4 that the high-confidence prediction for "syrup" as a `Food`

[1]Note that we define "confidence" as the probability which the model assigns to the top-predicted label, and is unrelated to speech recognition confidence.

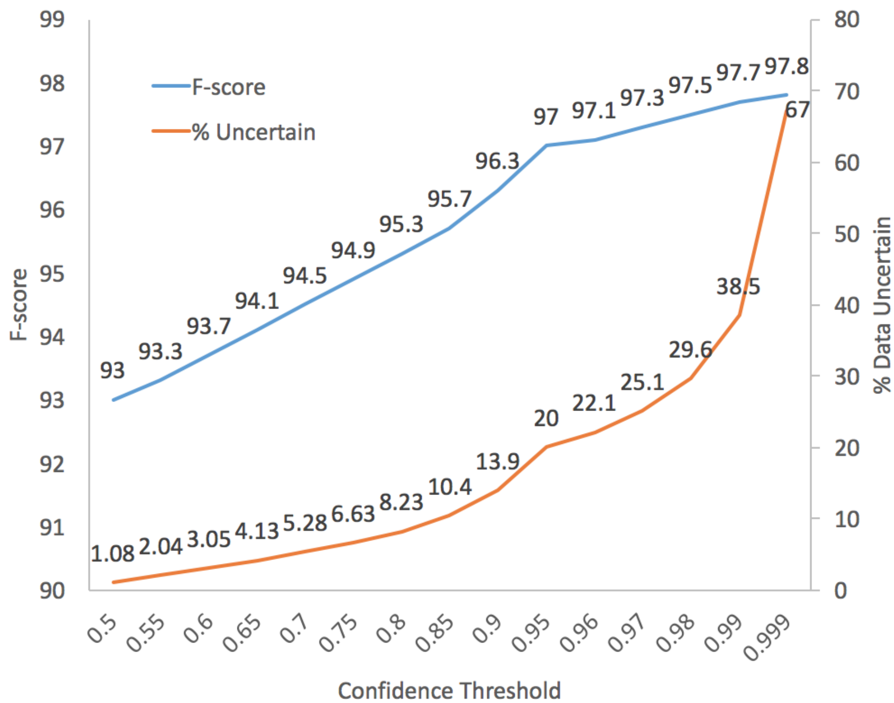


Figure 4-3: The F-score as a function of the confidence threshold, on the written meal description task, where tags that fall below the threshold are omitted from the recompute of the F-score. Performance improves as more data is omitted, since the model is only evaluated on tags where it is more certain (i.e., the predicted tag's probability is above the confidence threshold). While the percent of data omitted increases quite a bit as the confidence threshold rises from 0.9 to 0.999, the F-score gain is incremental in this region.

(i.e., $p = 0.999$) is actually correct, whereas the gold standard tag for `Brand` from AMT is a mistake. We also note that the predicted probabilities for the mistakes are lower, illustrating the model's uncertainty (i.e., $p = 0.69$ and $p = 0.73$ for incorrectly tagged tokens "medium sized" and "butter," respectively).

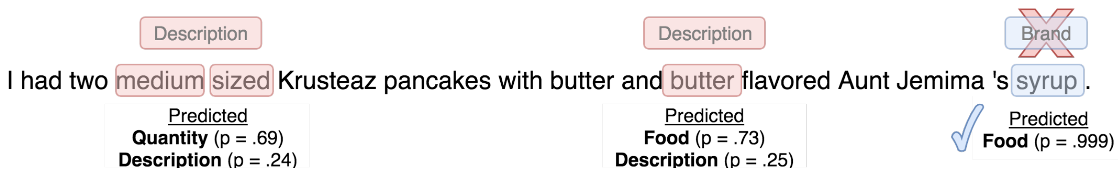


Figure 4-4: Incorrectly predicted semantic tags for a sample meal description, where "syrup" is actually tagged correctly by the model. Thus, this data annotation error can be discovered by identifying incorrect predictions that have high confidence (e.g., $p = 0.999$).

We see a common type of error made by the model in Fig. 4-5, where multiple adjacent food items cause the model to incorrectly predict the first food token as a `Description` rather than a `Food`. In addition, Fig. 4-6 illustrates the difficulty of distinguishing between brands, foods, and descriptions, especially in the presence of out-of-vocabulary words (i.e., tokens unseen during training are `<UNK>`).

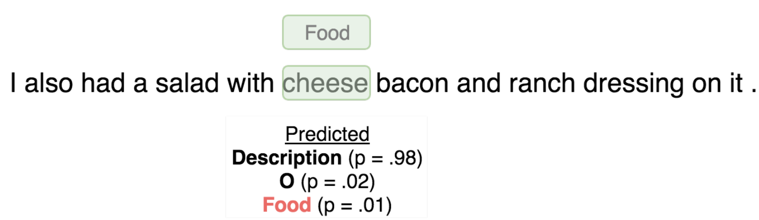


Figure 4-5: The model incorrectly predicts that "cheese" is a description of "bacon" instead of a separate food item.

In addition to error analysis, we also analyzed what specifically the neural network models were learning, since a common criticism is that these networks are "black boxes" and hard for humans to interpret. To do this, we select the tokens that have the highest activation for each of the 64 learned filters in the top layer of the trained three-layer CNN. We

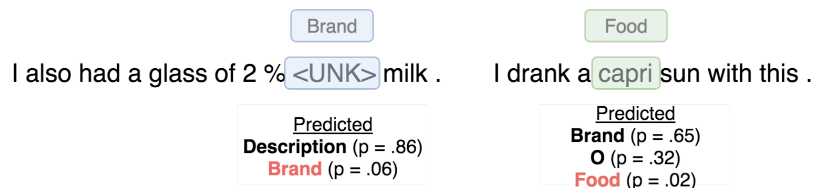


Figure 4-6: The model confuses brands with descriptions and foods.

observe that individual filters seem to pick out *semantically similar* words. In Table 4.3, filter 34 identifies quantities, filters 5 and 44 pick out brands, and filter 20 focuses on food.

Filter	Top-10 Highest Activation Tokens
34	8, 16, 14, 2, had, a, 6, an, ., 12
5	's, of, Natural, Welch, Pantry, brand, Debbie, Ida, Jemima, Arizona
44	Coke, coke, Water, Mountain, Kraft, cheese, Eggs, Dr. Miracle, Mt.
20	chile, tuna, egg, crab, chicken, cottage, butter, oranges, cauliflower, coconut

Table 4.3: Top-10 tokens with high activation for meal CNN filters.

Finally, we discuss the tradeoff between accuracy and model speed/size. When deploying a real-world system for interacting with users, efficiency is critical. While BERT performs best, it has 109.5 million parameters and requires 1.1GB, whereas the CNN is only 3.1 million parameters (and 12MB), as shown in Table 4.4.

Model	# Params	Size	Training Time
biGRU	3,457,159	14MB	31.8s
CNN	3,120,775	12MB	18.4s
BERT	109,486,854	1.1GB	6920s

Table 4.4: A comparison of the training speed and size of the three deep learning models we investigated, where training speed is based on one epoch on an NVIDIA GeForce GTX 1080 GPU. We used eight GPUs in parallel to train BERT, so we multiplied its training time of 865 seconds by eight to get the total time.

4.4 Domain Transferability

To conclude this chapter, we show that the neural models we have explored on the nutrition domain thus far also generalize to two other spoken language understanding domains: flight booking in the well-known ATIS, as well as restaurant booking. ATIS[2], which has been used since the 1990s, and the restaurant task[3] are both publicly available. See Table 4.5 for a summary of the data statistics. As on written meal descriptions, BERT performs best on both ATIS and restaurants.

Dataset	# Train Samples	# Test Samples	# Tags
Written Meals	35,130	3,412	5
Spoken Meals	35,130	476	5
ATIS	4,978	893	127
Restaurants	7,659	1,520	17

Table 4.5: The data statistics for each corpus. The spoken meal description data uses the same training set as the written.

4.4.1 ATIS Corpus

The ATIS task involves dialogues between users and automated spoken dialogue systems for booking flights. The goal is to label each token in a user utterance with the correct semantic tags, in the standard B-I-O format (e.g., `B-fromloc.city_name` refers to the beginning of the departure city's name, and `O` is `Other`). We show an example user utterance with its corresponding gold standard semantic tags in Fig. 4-7. In our work, we start from the same dataset as prior work (He and Young, 2003): the training set consists of 4,978 utterances selected from the Class A (context independent) training data in the

[2] https://github.com/yvchen/JointSLU/tree/master/data
[3] https://groups.csail.mit.edu/sls/downloads/restaurant/

ATIS-2 and ATIS-3 corpora, and the ATIS test set contains both the ATIS-3 NOV93 and DEC94 datasets.

Figure 4-7: Semantic tagging on a user utterance in the ATIS corpus, where BOS and EOS refer to the beginning and end of the sentence, respectively.

As in Section 4.3, we show the experimental results in Table 4.6, and the CNN's learned filters in Table 4.7. We see a similar trend as in the nutrition domain, where pre-training the CNN with word embeddings improves performance, and BERT is best overall. We see a pattern in the CNN's learned filters, where filter 11 identifies numbers, 12 focuses on flight attributes such as airline and number of stops, filter 22 finds cities, and 23 isolates tokens relating to days and time frames.

4.4.2 Restaurant Corpus

The restaurant corpus was collected on AMT, where workers were hired to write queries about restaurants, given a set of keywords (i.e., the semantic tags, as shown in Fig. 4-8) (Liu et al., 2013a). See Table 4.8 for more examples. They used a semi-Markov CRF with hand-crafted features, where they augmented baseline n-gram features with linguistic knowledge such as syntactic structural features, and semantic dependency parse features, from hierarchical parse trees (see Fig. 4-9 for an illustration).

Once again, as shown in Table 4.9, the neural models outperform the prior state-of-the-art CRF with hand-crafted features, and BERT is best. In Table 4.10, we see that learned CNN filter 6 identifies cuisines, filter 32 focuses on location, and filter 35 finds ratings.

Model	Precision	Recall	F-score
FST (Raymond and Riccardi, 2007)	91.6	91.9	91.7
CRF (Tur et al., 2011)	–	–	95.0
R-CRF (Mesnil et al., 2015)	–	–	96.5
bLSTM (Hakkani-Tür et al., 2016)	–	–	95.5
FF baseline	85.8	86.7	85.7
Transformer encoder (Vaswani et al., 2017)	96.6	96.6	96.4
1 RNN	97.9	97.9	97.7
RNN-CRF	97.8	97.9	97.7
4 RNNs	98.0	98.1	97.8
1 CNN	96.4	97.0	96.5
CNN + Glove	97.5	97.6	97.3
CNN + word2vec	97.1	97.4	97.1
CNN + FastText	96.8	97.1	96.7
4 Glove CNNs	97.5	97.7	97.4
4 RNNs + 4 Glove CNNs	97.8	97.9	97.6
BERT	**98.1**	**98.3**	**98.1**

Table 4.6: Precision, recall, and F-scores on ATIS, where the neural methods (other than the FF baseline) outperform the statistical FST and CRF from prior work. Again, ensembles of neural networks, and pre-training help, but a single fine-tuned BERT model outperforms them all.

Filter	Top-10 Highest Activation Tokens
11	12, 5, 230, 4, 7, to, 6, round, 10, 8
12	round, nonstop, us, american, northwest, delta, united, twa, daily, continental
22	phoenix, houston, pittsburgh, denver, detroit, milwaukee, cincinnati, chicago, charlotte, toronto
23	on, thursday, leave, between, from, friday, tuesday, in, leaving, a

Table 4.7: Top-10 tokens with high activation for ATIS CNN filters.

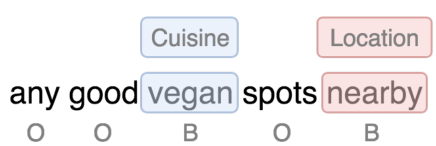


Figure 4-8: Semantic tags for a sample query in the restaurant domain.

Query
a four star restaurant with a bar
any asian cuisine around
any bbq places open before 5 nearby
any place along the road has a good beer selection that also serves ribs
any reasonably priced indian restaurants in the theater district

Table 4.8: Example restaurant queries collected on AMT, illustrating various tags in the restaurant domain (e.g., `Rating` for "four star," `Hours` for "open before 5," and `Dish` for "beer" and "ribs."

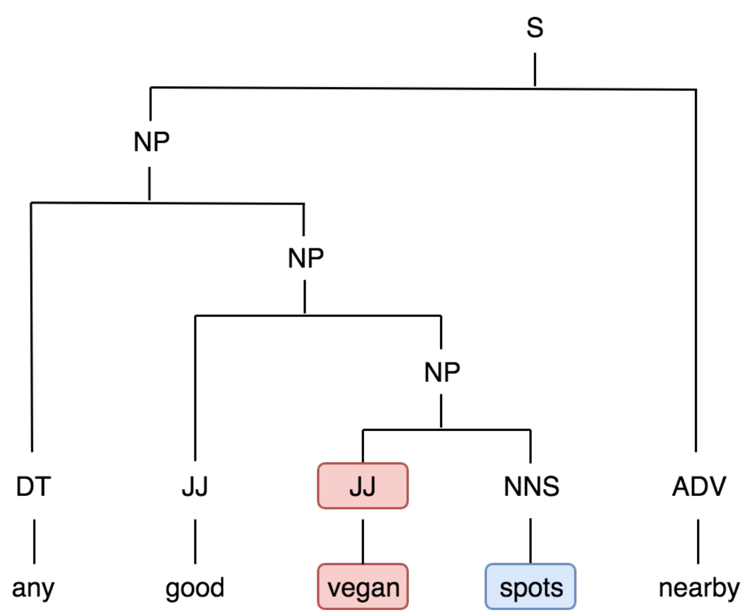


Figure 4-9: Example parse tree on a restaurant query, where the dependency relation between the adjective "vegan" and its parent plural noun "spots" is highlighted.

Model	Precision	Recall	F-score
Best CRF (Liu et al., 2013b)	85.3	83.9	84.6
FF baseline	82.3	80.7	81.0
1 RNN	87.1	87.4	87.2
RNN-CRF	87.1	87.4	87.2
4 RNNs	89.1	89.3	89.1
1 CNN	88.3	88.2	88.1
CNN + Glove	88.8	88.8	88.7
CNN + word2vec	88.9	88.7	88.6
CNN + FastText	88.9	88.9	88.8
4 Glove CNNs	89.7	89.7	89.7
4 RNNs + 4 Glove CNNs	89.9	90.0	89.8
BERT	**91.6**	**91.6**	**91.6**

Table 4.9: Precision, recall, and F-scores on the restaurant test set (Liu et al., 2013b). The gain from Glove and word2vec is not from using larger dimensions (200 and 300, respectively). Without pre-trained embeddings, using larger dimensions *decreases* the F-score (from 88.1 to 87.5 and 87.1, respectively).

Filter	Top-10 Highest Activation Tokens
6	brazilian, authentic, asia, italian, tex, mexican, sandwich, mex, mediterranean, spanish
32	nearest, closest, does, time, the, local, nearby, downtown, italian, at
56	till, until, after, open, at, past, before, a, is, every
35	4, highest, 3, 5, star, 1, starving, three, get, five

Table 4.10: Top-10 tokens with highest activations for four learned CNN filters on the restaurant dataset.

4.5 Summary and Future Work

In this chapter, we have demonstrated that neural networks (and BERT particular) outperform prior state-of-the-art CRFs that required manual feature engineering, on three spoken language understanding tasks: ATIS, restaurants, and spoken meal descriptions (with the exception of written meal descriptions, where a hand-crafted CRF still slightly outperforms BERT). In addition, pre-training the CNN with word vectors consistently boosts performance. Our analysis of the trained CNN shows that its filters are learning semantically meaningful categories related to the semantic tags, as well as predicting tags with lower confidence when making mistakes. In the future, we aim to incorporate a feedback mechanism into our food logging prototype such that the model will ask for clarification when it is uncertain about the semantic tags, thus learning online from users.

Chapter 5

Database Retrieval

In the previous chapter, we addressed the first step for language understanding in spoken dialogue systems (i.e., semantic tagging, or spoken language understanding). We now turn our attention to the second piece—database retrieval (as shown in Fig. 1-1). Many speech understanding applications involve mapping the concepts present in a natural language expression to their corresponding database entries. In recent efforts to create an ability for users to log their food intake by speech, we are faced with this situation, where we need to determine which particular foods have been described, and, ultimately, find their associated database entry in a nutritional database. For example, if someone says, "For breakfast I had a bowl of Kellogg's corn flakes," we would like to find the matching food entries in the USDA nutrient database (i.e., "Cereals ready-to-eat, KELLOGG, KELLOGG'S Corn Flakes"), so we can log the appropriate nutrition information.

As illustrated in Figure 5-1, we first attempted what one might consider the most obvious solution to this database mapping problem (Korpusik et al., 2014, 2016a), where we fed meal descriptions through a pipeline involving 1) semantic tagging of each token (as in Chapter 4), 2) performing food segmentation with simple linear classifiers to associate

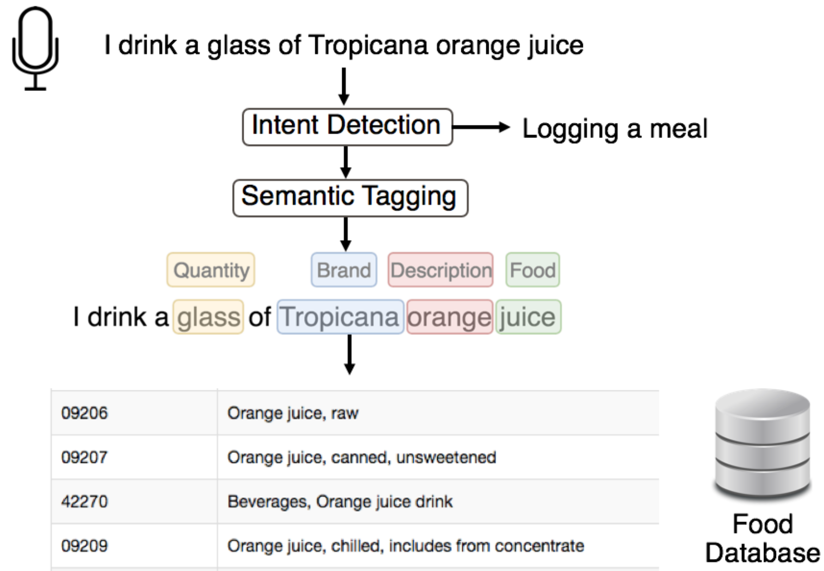


Figure 5-1: The previous system's sequence of steps: semantic tagging, segmenting a meal into food entities, and USDA database lookup using heuristics.

foods with their properties (e.g., brands, quantities, and descriptions), and 3) retrieving matching USDA food entries from a database. This approach had several undesirable properties. First, it was vulnerable to intermediate errors. Second, it relied on heuristics for the database lookup (e.g., stemming) to account for mismatches in language usage between the meal description and the database entry. Unfortunately, there are many instances where the common word is not present in the database entry at all (e.g., there is no exact match if the user says "toast," only various types of "bread, toasted" as well as "french toast" for additional confusion), so additional heuristics are required to bridge this mismatch (Naphtal, 2015). To address these issues, we developed a new model that can *directly map* between the natural language description to the underlying database entries.

The remainder of the chapter is as follows. In Section 5.2, we describe the novel CNN-based architecture for learning semantic embeddings of natural language, which we then use at test time to rank matching database entries. We describe both the overall word-based architecture in Section 5.2.1, as well as character-based embeddings in Section 5.2.2 for handling misspellings. After learning the embeddings, the next step is ranking the database matches using these learned embeddings. We describe three approaches: a finite-state transducer in Section 5.3.1 that requires no semantic tagging pre-processing, and a re-ranking algorithm in Section 5.3.2 that substantially outperforms our previous best configuration. Finally, we discuss multi-task learning in Section 5.4 for simultaneous food *and quantity* mapping in a jointly trained neural network, where we hypothesize that information about quantities is useful for predicting foods, and vice versa (e.g., cheese and toast come in slices, whereas milk and juice are usually in cups).

5.1 Related Work

5.1.1 Learning Embeddings

While in our work we learn embeddings for USDA food entries through CNNs, recent work (Adi et al., 2016) has analyzed the relative strengths of various other sentence embeddings, including averaging word vectors learned with the continuous-bag-of-words method (Mikolov et al., 2013a), LSTM auto-encoders (Li et al., 2015), and skip-thought vectors based on GRU (Kiros et al., 2015). Our approach differs from these in that we use a CNN rather than recurrent networks, and we learn the vectors through a domain-specific binary verification task for predicting whether a USDA food entry matches a meal description.

Similar work in learning joint embeddings for two different modalities or languages explored a margin-based contrastive loss. For ranking annotations given an image, prior work directly incorporated the rank into the model's loss function, along with a hinge loss between true and false annotation samples (Weston et al., 2011); similarly, a margin-based loss was used to learn a joint multimodal space between images and captions for caption generation (Karpathy and Fei-Fei, 2015; Harwath et al., 2016), and sentence/document embeddings were learned through a multilingual parallel corpus with a noise-contrastive hinge loss ensuring non-aligned sentences were a certain margin apart (Hermann and Blunsom, 2014). Other work predicted the most relevant document given a query through the cosine similarity of jointly learned bag-of-words embeddings (Huang et al., 2013).

5.1.2 CNNs for Natural Language Processing

Many researchers are now exploring CNNs for NLP. For example, in question answering, recent work has shown improvements using deep CNN models for text classification (Conneau et al., 2017; Zhang et al., 2015; Xiao and Cho, 2017) following the success of deep CNNs for computer vision. Whereas these architectures take in a simple input text example and predict a classification label, our task takes in two input sentences and predicts whether they match. In work more similar to ours, parallel CNNs predict the similarity of two input sentences. While we process each input separately, others first compute a word similarity matrix between the two sentences, and use the matrix as input to one CNN (Pang et al., 2016; Wang et al., 2016b; Hu et al., 2014).

attention-based convolutional neural network (ABCNN) has also been proposed for sentence matching. The ABCNN (Yin et al., 2016b) combines two approaches: applying attention weights to the input representations before convolution, as well as after convolution, but before pooling. Our method is similar, but we compute dot products (our version of the attention scheme) with the max-pooled high-level representation of the USDA vector. Hierarchical ABCNN applies cosine similarity attention between CNN representations of a query and each sentence in a document for machine comprehension (Yin et al., 2016a).

Thus, the attention comes after pooling across the input, whereas we compute dot products between each meal token and the learned USDA vector.

5.1.3 Multitask Learning

multitask learning (MTL) has been applied successfully to many NLP tasks. Collobert and Weston (2008)'s early exploration of multitask learning involved jointly training a single CNN like ours on several tasks: part-of-speech tagging, chunking, NER, semantic role labeling (SRL), semantic relation prediction, and language modeling. They focus specifically on SRL, while we care about both our tasks equally. Liu et al. (2015) built a multitask DNN that combined two different tasks of multiple-domain query classification and information retrieval for web search ranking. Similar to our work, they embedded an input query into a lower-level shared semantic representation used for the two different tasks at the top layer; however, they use a DNN while we employ a CNN.

Other work in MTL for NLP demonstrated an improvement in sentence compression by incorporating two auxiliary tasks, combinatory categorical grammar (CCG) tagging and gaze prediction, based on the intuition that longer reading time correlates with text difficulty (Klerke et al., 2016); they showed that the cascaded architecture, where auxiliary tasks are predicted at an inner layer, outperforms the model where auxiliary tasks are predicted at the top layer. Luong et al. (2016) investigated MTL for neural machine translation with the seq2seq model, with the surprising result that parsing (i.e., sharing the encoder) and image caption generation (i.e., sharing the decoder) both improve translation, despite the much smaller datasets.

Multi-task learning has also been applied to other fields, including ASR and computer vision. Toshniwal et al. (2017) explored end-to-end speech recognition on the conversational Switchboard corpus, demonstrating gains in character-based ASR by adding supervision at lower layers in a deep LSTM network with two lower-level tasks. In computer vision, Misra et al. (2016) proposed a novel cross-stitch unit that combines CNNs for two tasks by automatically learning an optimal combination of shared and task-specific representations. Wang et al. (2016a) constructed a shared sub-network with higher-level sub-networks for two image representations, in order to achieve high accuracy from cross-image representations while maintaining the efficiency of single-image representations.

5.2 Learning Semantic Embeddings

The natural language understanding component of a diet tracking system requires mapping a spoken or written meal description to the corresponding USDA food database matches. We employed two steps to achieve a system that directly selects the best USDA matches for a given meal description: 1) we constructed a CNN model that learns vector representations (either word-based or character-based) for USDA food items through a binary verification task (i.e., whether or not a USDA item is mentioned in a meal description), and 2) we re-ranked the food database entries based on a combination of dot product and word-by-word cosine similarity scores (Korpusik et al., 2016a). We also discuss an earlier approach to mapping that used a finite state transducer (FST) to map directly to the USDA database matches without semantic tagging as pre-processing, but this does not perform quite as well as a the re-ranking algorithm.

5.2.1 CNN Training Details

We learn the semantic embeddings of food descriptions by training the CNN architecture shown in Figure 5-2 on a simple binary verification task with two inputs—the user's meal description and a food database entry—where the model's goal is to determine whether the

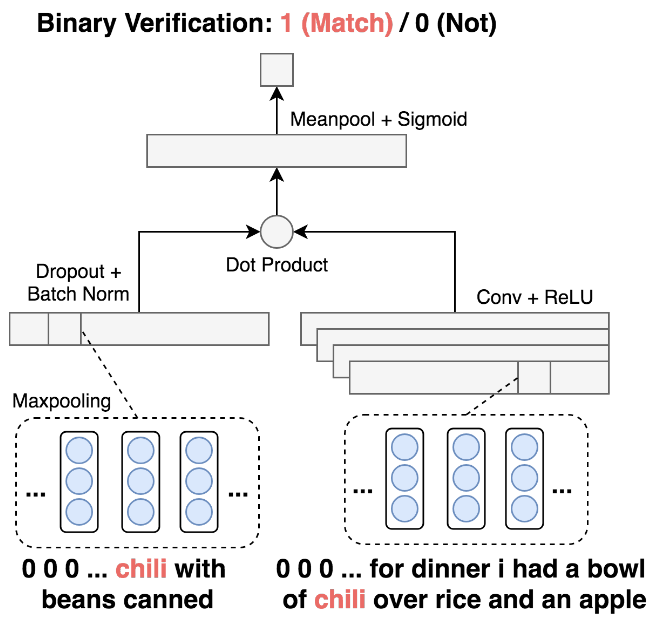


Figure 5-2: Architecture of our CNN model for predicting whether a USDA food entry is mentioned in a meal, which simultaneously learns semantic embeddings.

database food matches the meal description or not.[1] Our model is composed of a shared 64-dimension embedding layer, followed by one convolution layer above the embedded meal description, and max-pooling over the embedded USDA food name. The text is tokenized using spaCy.[2] The meal CNN computes a 1D convolution of 64 filters spanning a window of three tokens with a ReLU activation. During training, both the USDA input's max-pooling and the CNN's convolution over the meal description are followed by dropout (Srivastava et al., 2014) of probability 0.1,[3] and batch normalization (Ioffe and Szegedy, 2015) to maintain a mean near zero and a standard deviation close to one. A dot

[1]Note that while we did not exhaustively try every possible model architecture, and thus cannot guarantee this is the optimal structure, we did iterate through many versions of the model before empirically selecting this particular design (e.g., we initially fed both inputs through a convolutional layer, and tuned hyperparameters such as embedding size and number of filters). Our aim is to demonstrate that this model learns meaningful embeddings, not that this exact architecture gives the best performance.

[2]https://spacy.io

[3]Performance was better with 0.1 dropout than 0.2 or no dropout.

product is performed between the max-pooled 64-dimension USDA vector and each 64-dimension CNN output of the meal description. Mean-pooling[4] across these dot products yields a single scalar, followed by a sigmoid for final prediction.[5] This design is motivated by our goal to compare the similarity of specific meal tokens to each USDA food.

To prepare the data for training, we padded each text input to 100 tokens,[6] and limited the vocabulary to the most frequent 3,000 words, setting the rest to UNK (i.e., unknown). We trained the model to predict each (USDA food, meal) input pair as a match or not (1 or 0, respectively) with a threshold of 0.5 on the output. The model was optimized with Adam (Kingma and Ba, 2014) on binary cross-entropy loss, norm clipping at 0.1, a learning rate of 0.001, early stopping after the loss stops decreasing for the second time on the validation data (i.e., 20% of the data), and mini-batches of 16 samples. We removed capitalization and commas from USDA foods.

5.2.2 Character-based Models

In this section, we address the problem of what happens when a user misspells a food or brand in their meal description, or when there is a new food that the system has not seen before in the training data. For example, if the user misspells "cheerios" as "cherios," ideally the system would be able to correctly interpret this unknown word as the cereal "cheerios." To handle cases like these, we built a character-based CNN that learns embeddings for each character in a token, rather than only at the word level. Thus, with a character model, out-of-vocabulary (OOV) words are represented as character sequences and can

[4]The inverse (mean-pooling before dot product) hurt performance.

[5]Note that our approach would work for newly added database entries, since we can feed the new database food's name into the pre-trained CNN to generate a learned embedding. This is the strength of using a binary prediction task, rather than a softmax output, so we do not have to re-train the network every time the database adds a new entry.

[6]We selected 100 as an upper bound since the longest meal description in the data contained 93 words.

be used to predict matching foods with similar characters, while the previous word-based approach would not be able to handle such OOV words. Another approach which we leave for future work is learning word pieces rather than words or characters.

Levenshtein Edit Distance

As a baseline, we implemented the Levenshtein edit distance (Marzal and Vidal, 1993) for determining the distance between two sequences of characters. The minimum distance is computed through the following recurrence relation, where each term corresponds to deletion, insertion, and substitution of a character, respectively:

$$d[i,j] = \min\left(d[i-1,j]+1, d[i,j-1]+1, d[i-1,j-1]+c\right) \tag{5.1}$$

where cost c is 0 if the two characters are equal and 1 if not. During the ranking step, if an OOV word is encountered, Levenshtein edit distance is used to determine which USDA foods are most similar to the tagged food segment containing the OOV.

Character-based Embeddings

The second method we explored for handling OOV words is a character-based convolutional neural network (charCNN) that processes each character in a token to generate a word embedding Korpusik et al. (2017b). As shown in Figure 5-3, each token is padded to 24 characters and fed through the shared network, which consists of a 15-dimension embedding layer over each character, followed by a convolution of 64 filters over windows of three characters with hyperbolic tangent activation,[7] max-pooling to reduce the 24 embeddings to one embedding, and a highway network on top (Srivastava et al., 2015)

[7]We also tried rectified linear unit, but this did not perform as well.

to allow information flow along highways connecting layers.[8]

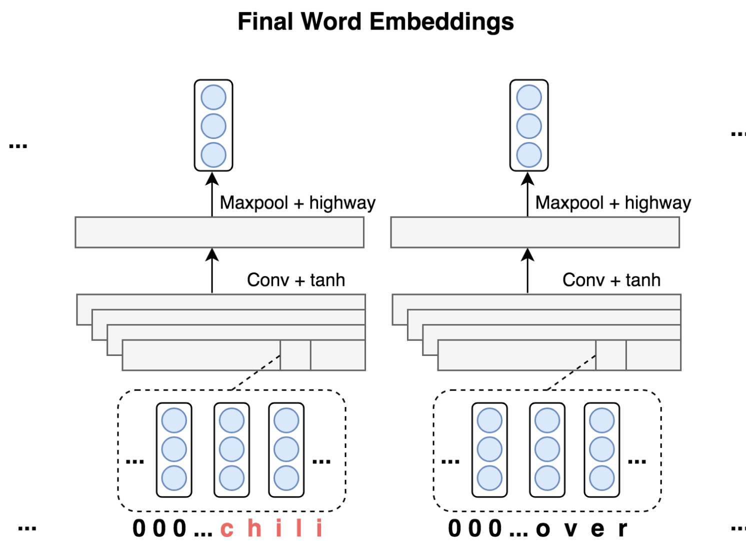


Figure 5-3: Architecture of the character-based CNN model.

Since the character-based models predict USDA matches primarily based on similar character sequences, rather than semantic similarity (e.g., "cheerios" would have a high similarity to "cheese"), we modified the USDA food database names during the ranking step for the character-based models. Specifically, we shortened the USDA food names to more concise descriptions that only included those tokens that had an α weight above a threshold of 0.25 (where α_i is the probability that token i was used to describe that particular USDA food item in the training data). Thus, the new USDA names only included tokens that were used most often in the training data to describe those USDA foods, which are more likely to match the characters of that food in a user's meal description. For example, the USDA food "Cereals ready-to-eat, GENERAL MILLS, CHEERIOS" would become simply "cheerios."

To evaluate how well the standard word-based model compares to the character-based

[8]Without the highway layer, the performance is not as good.

models on noisy data, we artificially induced typos in the test set. The original test data was clean so the performance was the same with the standard word-based model as well as when we augmented it with character-based models. We measure the ranking performance using mean average precision (MAP). The MAP score, which is a commonly used metric in information retrieval, measures the average precision as:

$$\frac{\sum_{k=1}^{n} prec(k)rel(k)}{\text{correct \# foods in the meal}}, \tag{5.2}$$

where k is the rank of a USDA food entry, $prec(k)$ is the precision of the food entry at rank k (i.e., how many correct foods have been identified so far, divided by the current rank k), and $rel(k)$ is 1 if the USDA food at rank k is actually in the meal description, and 0 otherwise. In Table 5.1, the last two rows apply the word-based model if there are no OOVs, use character-based models for food segments consisting of one OOV token, and combine the rankings generated by both models for food segments containing OOV and in-vocabulary words by taking the maximum similarity value per USDA food. We constructed a test set where 10% of the characters in each meal description were randomly deleted, substituted, or inserted. The error type, the index of the modification, and the new character for substitutions and insertions were all randomly selected. We see that the charCNN has the highest performance on the noisy test data.

Model	MAP	Recall
Baseline Levenshtein	6.27	15.5
Char CNN	8.65	17.1
Word CNN	12.0	26.3
Word CNN + Levenshtein	13.4	29.1
Word CNN + Char CNN	**13.8**	**30.0**

Table 5.1: Comparison of character-based approaches to the basic word-based CNN model on the all-food data with 10% typos.

5.2.3 Analysis of Learned Semantic Embeddings

To better understand the behavior of our models, we qualitatively analyze the top predicted USDA hits at test time, as well as the learned embeddings. In Table 5.2, we observe reasonable top-3 USDA foods predicted for each tagged food segment by the word-based model augmented with charCNN for the meal "I had a bowl of cherios with a banana and a glass of 1% milk." Note that since "cheerios" is misspelled as "cherios," the word model would predict matches for UNK as it is an OOV word; however, the charCNN correctly handles the error. In addition, we demonstrate that the charCNN is able to learn meaningful word embeddings. In Table 5.3, we see that the top-5 neighbors of the USDA food item are intuitive, where we computed the nearest neighbors by minimizing the Euclidean distance between a given food's embedding and all the other USDA foods.

Food	**Top-1**	**Top-2**	**Top-3**
cherios	cheerios	oat cluster cheerios	frosted cheerios
banana	banana	banana pudding	banana pepper
1% milk	1% milk	dry whole milk	milk low sodium

Table 5.2: Top-3 predicted USDA hits for tagged foods in the meal "I had a bowl of **cherios** with a **banana** and a glass of **1% milk**." with the word-based model combined with the charCNN and re-ranking.

Top-5 USDA Foods
beef new zealand imported kidney raw
beef new zealand imported inside raw
beef new zealand imported sweetbread raw
beef new zealand imported heart raw
beef new zealand imported manufacturing beef raw

Table 5.3: Top-5 nearest USDA neighbors, based on Euclidean distance for embeddings learned by the charCNN, to the food "beef new zealand imported tongue raw."

We also show through qualitative analysis that the word-based CNN model is indeed learning meaningful vector representations of the USDA food entries, which is why it successfully maps from meal descriptions to USDA foods. If we look at the nearest neighbor to three USDA foods (see Table 5.4) using Euclidean distance, we observe that the neighbors are in fact semantically similar. In addition, we can look at a t-SNE plot of the learned vectors for each USDA food item, where a point is a single database food entry, and the color is the food category. In Fig. 5-4, we see that there are clusters corresponding to food categories (e.g., breakfast cereals are all yellow, baked products are green, and dairy and egg food items are red), which indicates that the learned embeddings for semantically related foods lie close together in vector space, as expected.

Table 5.4: Nearest neighbor foods to three learned USDA food vectors.

USDA Food	**Nearest USDA Food**
Rice white short-grain...	...Mexican Spanish Rice
Fast Foods Chicken Tenders	Chicken Broiler or Fryers...
Beans Baked Canned...	Beans Black Mature...

Looking at the model's predicted dot products between USDA foods and each token in a meal description, we observe spikes at tokens corresponding to that USDA food entry. We visualize the spike profile of the dot products at each token for the top USDA hits in Fig. 5-5. Despite the spelling mismatch, the USDA foods "McDonald's Big Mac" and "Fast Foods, Cheeseburger" spike at "big mac," "Catsup" peaks at "ketchup," and "Bananas, Raw" matches "peeled banana."

5.3 Mapping Approaches

While the NN model learns to predict whether or not a USDA food is mentioned in a meal, it does not directly retrieve the matching USDA foods from a meal. To do this, as a first approach, we trained a finite state transducer (FST) to take as input the model's learned

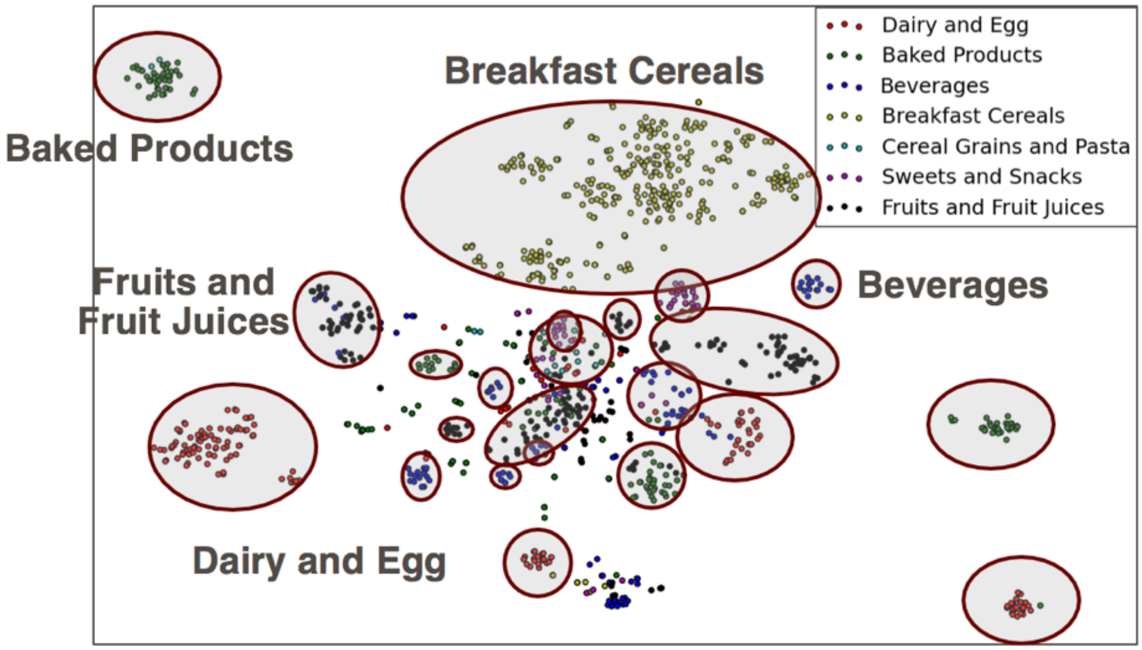


Figure 5-4: A t-SNE plot of the learned embeddings for each USDA food database entry, where the color corresponds to the food category, and foods in the same category tend to cluster together. The center consists of several different clusters. The black points are fruits and fruit juices, and the dark blue are beverages. Since these are both drinks, they overlap. We hypothesize that the red cluster for dairy and egg that lies near the center is milk, since the center clusters are primarily related to beverages. In addition, just below the breakfast cereals, we note there are several teal points for cereal grains and pasta, which again has a reasonable location near cereals. The two small clusters of purple points, one which overlaps with breakfast cereals, and the other inside the cereal grains and pasta cluster, represent sweets and snacks. These may lie close to cereals since some sweets and snacks (e.g., Rice Krispie treats, Chex Mix) contain cereal.

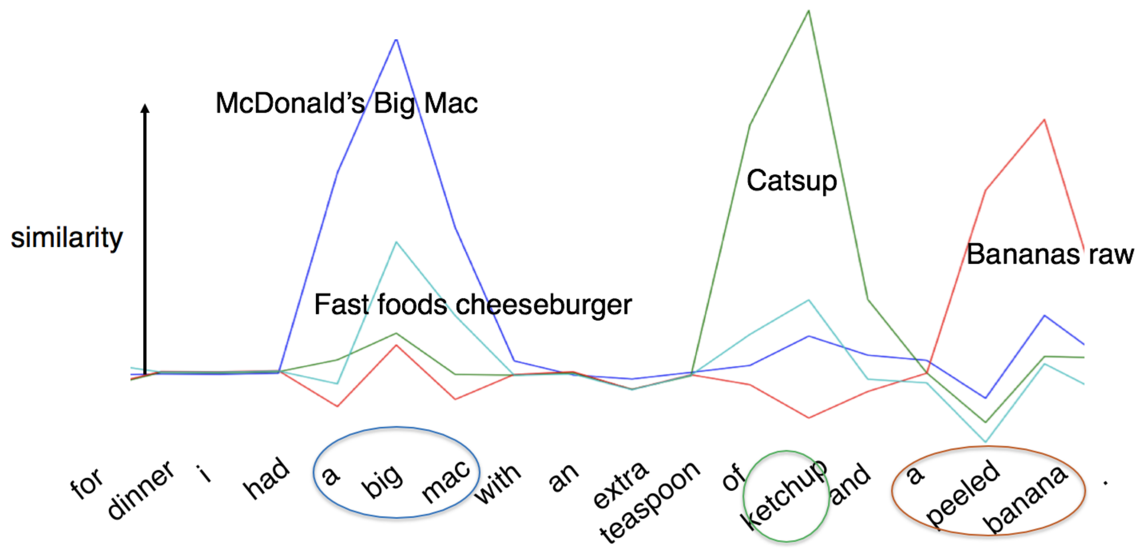


Figure 5-5: Dot products between top USDA hits and meal tokens for the meal description *"for dinner I had a big mac with an extra teaspoon of ketchup and a peeled banana."*

dot products (between each token in a full meal description and all possible USDA foods), and *automatically* output the most likely sequence of USDA foods for the *whole meal*. Afterward, we leveraged semantic tagging as a pre-processing step to identify individual food segments, and found that it performed better at ranking food database matches.

5.3.1 Finite State Transducer

We constructed a generic FST, composed it with a meal-specific FST, and determined the top-5 sequences of USDA foods in the meal with n-best Viterbi decoding Korpusik et al. (2017a). The model FST, shown in Figure 5-6, enables transitions between `Other` states (i.e., no USDA food is discussed at that token) and three interleaved USDA food states, where there are transitions for each possible USDA entry. Each state has a self-loop, which allows multiple consecutive tokens to have the same USDA food. This model forces the output to have a sequence of exactly three USDA food items, with `Other` in between.

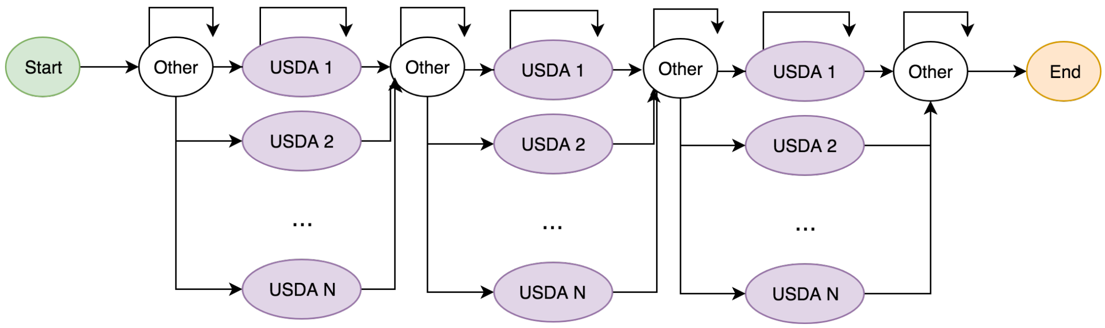


Figure 5-6: A generic FST with a sequence of three USDA foods, with `Other` in between.

The meal FST consists of start and end states, as well as one state for every token in a given meal description (e.g., *slice of toast* in Fig. 5-7), where the transitions to each state are all the possible USDA foods, and the weights are the negative dot products between each token and all possible USDA food entries. This FST forces the output to have the same length as the input meal description, and will select for each token the USDA food that has the highest similarity score between its learned embedding and the token embedding. Composing the generic FST with the meal FST yields alignments (similar to aligning speech to text in speech recognition)—for each token, `Other` or a USDA item with a large dot product is aligned with it.

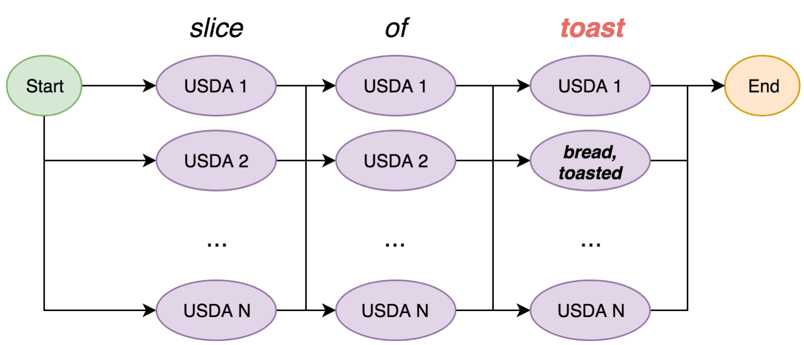


Figure 5-7: The meal-specific FST, which has one state per token, and the weights for the transitions are proportional to the similarity score between a USDA food's learned embedding and the learned token embedding.

We used these FST-predicted alignments to generate additional positive and negative training examples to improve performance. For each input (USDA food, meal) pair, we also generated an example of a shortened food segment and its corresponding USDA entry, where the segment is all the tokens that were aligned with that USDA food in the alignment. Thus, we fine-tuned the network so that it knows which tokens in particular are associated with which USDA entries.

One issue with using the alignments directly for predicting the most likely USDA matches is that the n-best lists only differ in position of USDA foods, rather than generating alternative USDA entries (e.g., in one alignment, the USDA entry for "banana" might be assigned to the first three tokens, whereas in the second alignment, "banana" might span the first four tokens). To force the FST n-best decoder to generate different USDA food choices, we thus composed the model and diary FSTs with a third high-level FST that outputs nothing for `Other` states and USDA self-loops. For example, the alignment "Other ID18290 ID18290 Other ID7953 ID7953 ID7953 Other ID20137" maps to "ID18290 ID7953 ID20137," and the n-best list would contain different choices of USDA foods. From the positions of the USDA foods in this final sequence of IDs, we can determine which food alternatives group together and their ranking.

Binary Verification and Ranking Experiments

For the following experiments, we focused on a subset of 101 USDA foods (i.e., 16,589 total meal descriptions) for training and evaluation. First, we show in Table 5.5 performance of the neural network model for binary verification (i.e., predicting whether a USDA food item is mentioned in a meal description) and ranking (i.e., how highly it ranks the correct USDA foods in a meal description). We see in Table 5.5 that the neural models with the base LSTM and CNN perform similarly on binary verification, but the LSTM is worse at the ranking task. Incorporating aligned token segments specific to each USDA food as additional training examples improves performance in both binary verification and ranking.

Model	Train Acc.	Test Acc.	MAP
baseline word matcher	59.4	59.0	0.067
baseline classifier	90.9	89.8	0.077
base LSTM	95.1	92.9	0.020
base CNN	94.2	92.7	0.853
base CNN + alignments	**96.4**	**95.7**	**0.892**

Table 5.5: Performance on binary verification and ranking. The base LSTM uses LSTMs instead of CNNs to process the USDA and meal description inputs before computing dot products. The baselines are a lexical matcher (i.e., at least two shared words is a match) and logistic regression classifier trained on n-grams and word matches.

In-the-loop Database Lookup Experiments

We ultimately care how well the system will perform at test time with actual users, so we also compared the performance of the existing system set up with tagging followed by food-property association and rule-based database lookup to the performance using the new neural architecture followed by FST n-best prediction. We evaluated the system on two metrics: the percent of correct USDA entries recalled by the system, and the number of alternative foods shown to the user on average. It is preferable to have fewer alternatives to avoid overwhelming the user with options, while still having high recall. As shown in Table 5.6, the CRF tagger with many manually defined features performs similarly to the CNN tagger with no feature engineering, while the neural architecture with FST n-best predictions has much higher recall than either tagger, from 79% up to 92.6%.

Model	USDA Recall	Avg. # Food Options
CRF tagger	78.9%	0.12
CNN tagger	78.5%	**0.09**
FST alignments	**92.6%**	0.16

Table 5.6: System evaluation with tagging followed by USDA lookup vs. neural-FST setup with direct USDA predictions on a held-out test set of 5k meals with only 101 USDA foods. The third column (average number of food options) is the average number of lower probability food alternatives retrieved by the system per food entity.

5.3.2 Semantic Tagging and Re-ranking

Since the FST approach must learn which USDA foods match a given meal description without any information about where the foods occur in the sentence, we investigated whether providing the system with such information via semantic tags would boost performance. At test time, rather than feeding the entire meal description into the CNN, we instead first perform semantic tagging (see Chapter 4) as a pre-processing step to identify individual food segments (Korpusik et al., 2014, 2016a; Korpusik and Glass, 2017), and subsequently rank database foods via dot products with their learned embeddings. A CNN tagger labels tokens as `Begin-Food`, `Inside-Food`, `Quantity`, and `Other`. Then, we feed a food segment into the pre-trained embedding layer in the model described in Section 5.3 to generate vectors for each token. Finally, we average the vectors for tokens in each tagged food segment (i.e., consecutive tokens labeled `Begin-Food` and `Inside-Food`), and compute the dot products between these food segments and each previously computed and stored USDA food vector.[9] The dot products are used to rank the USDA foods in two steps: a fast-pass ranking, followed by fine-grained re-ranking that weights important words more heavily Korpusik et al. (2017b). For example, simple

[9]Our approach with CNN-learned embeddings significantly outperforms re-ranking with skipgram embeddings (Mikolov et al., 2013c). For comparison, on breakfast descriptions, our model achieves 64.8% top-5 recall, whereas re-ranking with skipgrams only yields 3.0% top-5 recall.

ranking would yield generic milk as the top hit for 2% milk, whereas re-ranking focuses on the property 2% and correctly identifies 2% milk as the top match.

- **Ranking:** initial ranking of USDA foods using dot products between USDA vectors and food segment vectors.
- **Re-ranking:** fine-grained word-by-word *cos* similarity[10] ranking of the top-30 hits with weighted distance D (Korpusik and Glass, 2018a).

$$D = \sum_i \alpha_i \max_j (w_i \cdot w_j) + \frac{1}{N} \sum_j \beta_j \max_i (w_i \cdot w_j) \tag{5.3}$$

where N refers to the length of the tagged food segment. The left-hand term finds the most similar meal description token w_j to each USDA token w_i, weighted by the probability α_i that token was used to describe the USDA food item in the training data. In the same way, the right-hand term finds the most similar USDA token w_i to each meal token w_j, weighted by the probability β_j that token w_j was used to describe that USDA food item in the training data (see Fig. 5-8).[11]

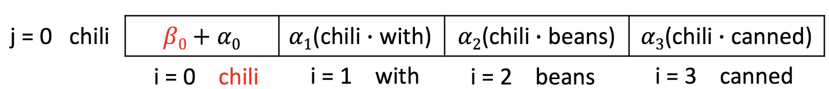


Figure 5-8: A re-ranking example for the food "chili" and matching USDA item "chili with beans canned." There is only one β_0 term in the right-hand summation of equation 1, since there is only a single token "chili" from the meal description.

[10]Note that for the initial ranking, we use dot product similarity scores, but for word-by-word similarity re-ranking, we use cosine distance. These distances were selected empirically, where on the dataset of all foods, these metrics yielded higher food recall than using dot products for both ranking steps, or cosine distance for both, and Euclidean distance was the worst (see Table 5.9).

[11]Although the sum appears biased toward longer USDA foods, the right-hand term is over each token in the food segment, which is fixed, and the left-hand term is normalized by the α weights. Dividing D by the number of tokens in the USDA food item hurt performance (39.8% recall on breakfast data versus 64.8% with the best model).

Ranking Experiments

We see in Table 5.7 that the new re-ranking algorithm outperforms the FST decoding (see Section 5.3.1) on a 101 foods case study, boosting top-5 food recall from 91.9% to 96.2%.

Model	MAP	Top-5 Recall
CNN + FST decode	86.9	91.9
CNN + re-ranking	**92.7**	**96.2**

Table 5.7: FST decoding compared to the new ranking method on 101 foods data. These results are high because we collected 16k meals for an initial case study with only 101 foods, whereas we only used 4k for the other meals with up to 2570 foods.

We also observe an improvement between just ranking and two-pass re-ranking when we apply the re-ranking algorithm to the larger all-food dataset, which is what the system will be using at test time with real users. In Table 5.8, the top row applies only the first-pass ranking to predict the top-5 USDA matches. With the full two-pass re-ranking algorithm in the second row, top-5 recall goes up to 64%.

Model	DB	MAP	Top-5 Recall
CNN + ranking	USDA	20.6	58.1
CNN + **re-ranking**	USDA	**31.3**	**64.0**

Table 5.8: Evaluation of ranking once vs. re-ranking on the all-food dataset.

We evaluated the CNN with re-ranking on all eight meal categories, in addition to the all-food dataset, where each model is trained/tested only on its meal category. Note in Table 5.9 there is a correlation between the number of foods in the meal category and the difficulty of the task, since dinner has more foods than any other meal (2,570 foods total) and has the lowest scores, whereas salads have the fewest foods (only 232) and achieve a recall of 92.5%. These experiments demonstrate that for most meal categories it is best to use cosine similarity as the distance metric for both ranking steps, and pre-training with

FastText; however, for the all-food category, which we use at test time, it is best to use a combination of dot products for initial ranking, followed by cosine similarity word-by-word re-ranking, and initialization with word2vec.

Meal	Cos	Euclidean	Dot	Dot + Cos	Glove	word2vec	FastText
Breakfast	65.9	9.64	65.1	64.8	60.8	67.3	**67.5**
Dinner	47.1	5.47	44.5	45.1	43.0	46.9	**48.8**
Salad	92.3	65.6	92.4	92.5	87.5	91.2	**92.7**
Sandwiches	**76.7**	32.5	76.5	76.5	73.3	76.5	76.5
Smoothies	74.9	22.9	75.6	75.1	69.6	**78.9**	76.4
Pasta/Rice	**52.5**	9.38	51.6	51.9	48.3	51.0	51.8
Snacks	68.1	15.5	67.9	67.3	62.3	67.7	**68.2**
Fast Foods	70.2	25.1	70.5	70.3	67.1	71.0	**71.5**
All Foods	63.4	28.5	17.7	64.0	62.8	**65.1**	65.0

Table 5.9: Top-5 USDA food recall per meal on the held-out test data, for various distance metrics, as well as with and without pre-trained word vectors (where we use the combination of dot products for ranking, and cosine distance for word-by-word re-ranking).

5.4 Quantity Mapping with Multitask Learning

In the previous sections, we investigated the problem of mapping natural language meal descriptions to their corresponding food database entries. But this was limited to food matching, whereas we also need to address the remaining challenge of mapping user-described quantities to matching database quantity entries. This is a difficult problem because user descriptions are often very different from database entries. For example, a user might say "a bowl" or "a handful," but these do not easily map to database quantities, such as cups or grams. In a scenario where the user says, "**a spoonful** of *peanut butter*," the system should determine that the database food match is *Peanut butter, smooth style, with salt* with the corresponding quantity **1 tbsp**.

In this section, we tackle the quantity mapping problem by developing a new CNN architecture that is trained with a softmax layer on top to directly predict the most likely database quantities, whereas our prior food mapping network in Section 5.2.1 used a binary classification task to learn embeddings for each database food entry, which were then ranked via cosine similarities at test time. In addition, we explore multitask learning to jointly predict both the matching food and quantity database entries given a single input meal description. We show that by leveraging the close relationship between quantities and foods, we can use predicted quantity matches to improve food ranking performance.

5.4.1 Models

We implemented two variants of the CNN architecture for mapping natural language quantity descriptions to the USDA database: the first is reminiscent of the food mapping approach described in Section 5.2.1, learning USDA quantity embeddings via binary classification with a sigmoid output, and the other is a different approach that directly predicts the most likely database matches via a softmax layer on top. In this section, we first describe two baseline methods for ranking database quantities (using the longest common substring, and number of exact token matches). We then detail our two sigmoid and softmax CNN models. Finally, we explain the new multitask training mechanism. The data we used for these experiments are described in Section 3.2.4.

Baselines

A simple lexical approach for ranking the most likely database quantity entries, given a user's meal description, is to use the number of tokens[12] that are an exact match between the two. Those database quantities with the maximum number of tokens in common would

[12]We used the Spacy toolkit (https://spacy.io/) for tokenization.

be ranked most highly. Our second baseline uses the length of the longest common substring (LCS) between the user's meal description and each database quantity, where we implement a string matching algorithm that stores the number of matching characters seen so far in a dynamic programming table. For the food mapping task, we compare against our prior state-of-the-art CNN with re-ranking (see Section 5.3.1).

Sigmoid CNN

The sigmoid model (see Fig. 5-9) is the same as that used in Section 5.2.1 for mapping natural language meal descriptions to their associated food database matches, except we pad the input quantity descriptions to 20 tokens instead of 100.[13] The input 50-dimension embedding layer is followed by a 1D convolution with 64 filters spanning windows of three tokens, with a ReLU activation and dropout of probability 0.2. This network is trained for a binary verification task, where each input pair consists of a user-described quantity and a USDA quantity that either matches the user's description or not.[14] Through learning to complete this binary verification task, the network learns semantic embedding representations of each USDA database quantity, which are then used at test time to rank all the possible database quantity options based on the cosine similarity score with the user-described quantity embedding (which is generated by feeding the input meal description through the meal portion of the network, consisting of an embedding layer followed by a convolution and max-pooling). The model is trained with the Adam optimizer (Kingma and Ba, 2014) on binary cross-entropy loss.

[13]The padding results in dot products with each of the 20 input tokens.

[14]For each positive match we collected, we sampled a random negative quantity from among those quantities not described by the user.

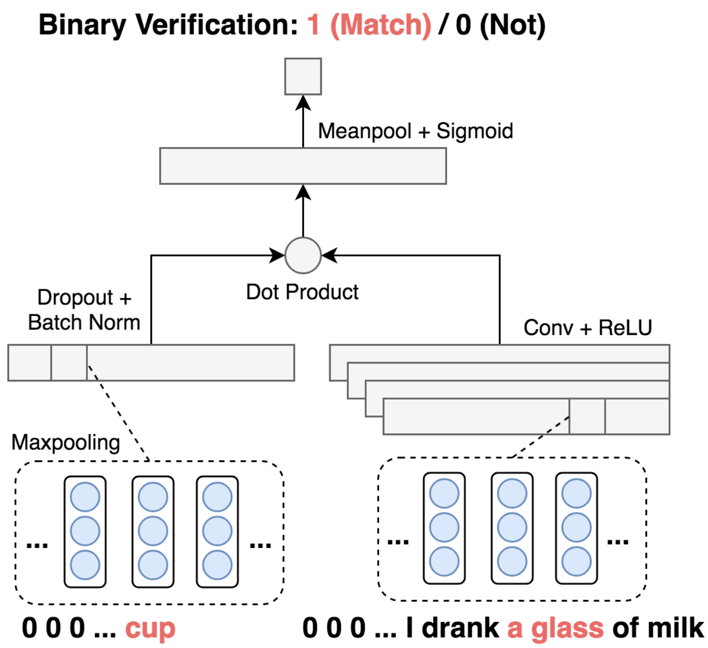


Figure 5-9: The sigmoid CNN for predicting whether or not an input USDA quantity name and user-described quantity match or not.

Softmax CNN

The softmax CNN (see Fig. 5-10) is a new architecture that we implemented to directly rank all the USDA database quantities within the network itself, rather than requiring the multi-step process of generating embeddings with the network and subsequently ranking all the USDA quantities with cosine similarity scores. Rather than feeding the network only a single USDA quantity option, we input all possible USDA quantities along with the user's meal description. The USDA quantities are embedded and used in dot product computation with the convolved meal description the same way as in the sigmoid network; however, this model performs the computation for *every* USDA quantity, with a final feed-forward layer on top to output a probability distribution over all quantities via a softmax.

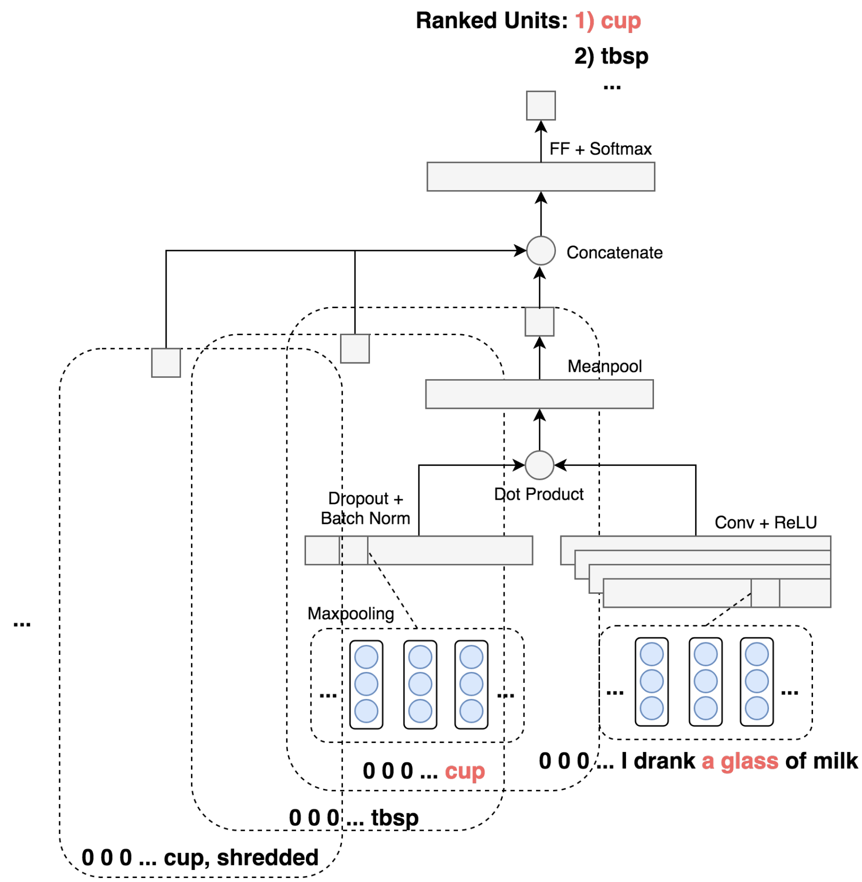


Figure 5-10: The softmax CNN for directly ranking all the USDA quantity options for a given user's input meal description.

Simple Softmax CNN

The simple softmax CNN (see Fig. 5-11) is another new neural architecture that feeds only the input meal description into the embedding and convolution layers before the final feed-forward layer with a softmax output over all possible food or quantity options.

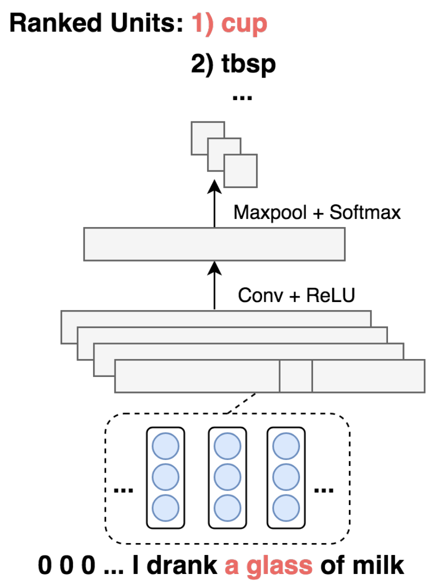


Figure 5-11: The simple softmax CNN for directly ranking all the USDA quantity options for a given user's input meal description.

Multitask CNN

The new multitask model is structured the same way as the sigmoid and softmax CNNs for quantity mapping, but has an additional output layer for predicting the USDA food match. Thus, the majority of the network is shared between the two tasks, and the loss is the combination of the quantity prediction and food prediction losses.

5.4.2 Experiments

Here, we demonstrate that the new softmax model outperforms the previous best binary verification model (see Section 5.2.1) for food mapping. We also show that jointly training the CNN to predict both USDA food and quantity matches yields higher quantity recall for most meal categories. We note that since the quantity predictor has high performance, we can leverage the predicted quantities to rerank the USDA food options to favor those that have the highest ranked quantities as available options in the database, which consistently boosts performance. We evaluate on both written and spoken held-out test sets[15] using top-5 recall, which indicates the percentage of test cases in which the correct food or quantity option appears in the top-5 hits.

Sigmoid vs. Softmax

We see in Table 5.10 that all the CNN models significantly outperform the longest common substring (LCS) and number of word matches (WM) baselines on the quantity mapping task; thus, simply counting word or character matches is not sufficient, and a more sophisticated model is required. The softmax is superior to the sigmoid, for both food and quantity matching (trained separately). For all subsequent experiments, we use the simple softmax as the default softmax since it performs best; we conjecture this is due to the relatively simple task, for which the complex softmax is too powerful and overfits the training data. In Table 5.11, we evaluate the new CNN models on food mapping and discover that the simple softmax model improves on our prior state-of-the-art CNN with word-by-word similarity re-ranking algorithm (Section 5.3.1), yielding an 83.3% gain on all foods.

Spoken Data

Because our production system will enable both text and speech input, here we investigate whether the models trained on text data still perform well on speech data. As shown in Table 5.12, when evaluated on 9,600 spoken meal descriptions (1,200 per meal category), the softmax quantity and food mapping models still perform quite well.

[15] We divide the 99,350 text samples into 80% train/10% dev/10% test.

Meal	LCS	WM	Sigm.	Soft.	Simple Soft.
Breakfast	13.5	7.87	71.1	94.8	**96.9**
Dinner	10.9	10.4	82.1	94.8	**98.2**
Salad	25.1	36.9	75.5	82.7	**97.4**
Sandwich	19.2	30.1	77.7	92.0	**97.2**
Smoothies	18.5	37.1	75.3	92.6	**98.7**
Pasta/Rice	11.7	12.6	84.0	95.5	**98.1**
Snacks	15.8	12.3	63.7	93.4	**96.9**
Fast Food	16.5	13.7	72.2	93.8	**98.7**
All Foods	13.9	13.3	70.0	96.5	**97.3**

Table 5.10: Top-5 quantity recall per meal category for LCS and word match (WM) baselines, the sigmoid, softmax, and simple softmax.

Meal	Baseline CNN	Sigmoid	Simple Soft.
Breakfast	47.3	34.6	**95.8**
Dinner	38.5	25.4	**91.6**
Salad	75.9	40.4	**98.4**
Sandwich	70.8	46.4	**97.9**
Smoothies	69.5	53.7	**97.2**
Pasta/Rice	39.2	29.9	**89.5**
Snacks	60.2	41.4	**96.9**
Fast Food	53.6	38.7	**98.2**
All Foods	49.7	46.4	**91.1**

Table 5.11: Top-5 food recall per meal category for the new sigmoid and softmax CNNs, compared to the baseline CNN reranker (Korpusik et al., 2017b).

Meal	Q LCS	Q WM	Q Best	F CNN	F Best
Breakfast	15.2	8.36	92.0	54.5	80.9
Dinner	14.5	14.6	92.8	45.7	67.3
Salad	27.2	40.9	89.5	82.6	90.9
Sandwich	24.8	29.3	89.2	82.4	88.8
Smoothies	26.0	37.8	90.8	73.6	88.6
Pasta/Rice	14.5	14.0	89.4	43.7	60.2
Snacks	19.5	16.8	90.3	64.4	84.1
Fast Food	19.0	13.3	84.3	58.2	84.0
All Foods	20.1	21.9	91.6	62.6	80.1

Table 5.12: Top-5 quantity (Q) and food (F) recall per meal category for the best simple softmax and baseline models on *spoken* data.

Quantity Input Only

Since it would seem that the interaction between foods and quantities helps the models learn to predict relevant foods and quantities, we ran an experiment to see whether the quantity mapping performance would suffer if the input to the network was only the quantity description, without the associated food's description. For example, with the user-described input meal diary "I had a cup of cheese," the model might tend to prefer database units that relate to cheese, such as "cup, shredded" rather than the generic "cup" or "cup, diced." However, in this experiment, the input would simply be "cup." To convert the full meal to a quantity segment, we ran our CNN tagger from Chapter 4 that labels food and quantity tokens, and extracted only the tagged quantity tokens. As expected, quantity mapping performance is much worse without the full meal input. On Breakfast, the top-5 quantity recall is only 55.8 for sigmoid and 70.0 for softmax (leading to 21.5% and 26.2% drop in performance for the sigmoid and softmax models, respectively, with only quantities as input); the top-5 quantity recall scores for the other meals are similarly all below 70 for sigmoid models and below 80 for softmax.

Multitask Learning

Finally, we investigated MTL to determine whether a single model that jointly predicts food and quantity labels would perform better than either model individually. MTL with the simple softmax model improves quantity mapping for most meal categories (see Table 5.13); however, the food mapping task is more challenging, as there are far more food options than quantities, so MTL does not benefit this task. This indicates that MTL can improve the task with fewer labels, but not the harder task (Bingel and Søgaard, 2017).

Since we also want to boost food mapping by leveraging the quantity mapping task, as an alternative approach to training a joint multitask model, we used the best quantity softmax trained on all data (since if we only used training data, then it could not accurately predict quantities seen only in the test data) to rerank the predicted foods. This boosts the top-1 food mapping performance on test data for all meals except Fast Food (see Table 5.13). First, we predict the top-5 USDA quantities. Then, we rerank the predicted USDA foods that have at least one of the top-5 predicted quantities as a unit option above those that do not. This gain indicates that we can leverage a higher-performing task to improve a weaker, closely related task.

Meal	**Q Soft.**	**MTL Q**	**F Soft.**	**Reranked F**
Breakfast	**89.6**	88.7	80.4	**81.5**
Dinner	89.6	**89.7**	71.8	**72.9**
Salad	**89.0**	88.2	83.7	**84.1**
Sandwich	**89.6**	88.4	82.7	**83.6**
Smoothies	89.3	**89.9**	82.0	**82.0**
Pasta/Rice	90.0	**90.1**	66.5	**67.4**
Snacks	85.9	**86.5**	82.4	**83.0**
Fast Food	92.5	**93.4**	**88.8**	88.3
All Foods	**86.6**	86.3	70.3	**71.1**

Table 5.13: Top-1 recall for MTL Quantity and reranked Foods.

Nearest Neighbors

When users interact with our live nutrition system, we must ensure the rankings generated by our food and quantity mappers at test time are reasonable. To qualitatively evaluate the performance of our CNN model, we observe that its predictions make sense intuitively. For example, in the test meal description "*I had a **cup** of milk and a **tablespoon** of honey*," with the softmax model trained on Breakfast data, the quantity ranking for milk is {cup, fl oz, quart} and {tbsp, cup, packet (0.5 oz)} for honey, which matches commonsense.[16]

By inspecting the nearest neighbors of the learned USDA quantity embeddings (see Table 7.6), we see that the Pasta softmax model (i.e., the complex softmax CNN trained on the Pasta meal category[17]) is learning meaningful representations of quantities, where those of a similar unit are close to each other in vector space. We can also determine what the 64 CNN filters over the embedded quantities learned by inspecting which tokens cause the filters to fire with the highest activations. This analysis shows that filter 46 tends to identify meat-related tokens (i.e., tenderloin, beef, loin, strip, steak, pork, wagyu, roast, dried, and strips are the top-10 tokens in order of descending filter response), while filter 53 picks out numbers (i.e., three, one, a, two, eight, five, four, six, twelve, and seven).

Quantity	**Neighbor 1**	**Neighbor 2**	**Neighbor 3**
cup	cup whole	cup slices	cup shredded
oz	oz whole	oz boneless	oz serving 2.7 oz
serving 1/2 cup	serving 1 cup	cup slices	cup whole

Table 5.14: Top-3 neighbors to three USDA quantities, based on Euclidean distance of learned embeddings from a Pasta softmax model.

[16]A pre-trained semantic tagger (Korpusik and Glass, 2017) identifies each food/quantity segment.
[17]In the deployed system, we would use the full Allfood softmax model.

5.5 Summary and Future Work

The contributions of this chapter are three-fold: 1) in Section 5.2, we learned semantic vector representations of natural language with a novel CNN model trained on *weakly annotated data*, and designed a new re-ranking algorithm in Section 5.3.2, for mapping directly from natural language meal descriptions to matching USDA food database entries using these learned semantic embeddings, 2) in Section 5.2.2, we demonstrated that character-based models can handle misspellings and other out-of-vocabulary words at test time, and 3) in Section 5.4, we expanded our initial work on mapping natural language meal descriptions to their corresponding USDA *food* database entries to address the remaining challenge of mapping meal descriptions to their associated *quantity* database hits, using multitask learning. In future work, we will investigate contextual understanding to determine whether the user has refined their meal description, and we may explore speech-to-speech networks and input lattices to account for speech recognition errors. In the next chapter, we will address the challenge of directly mapping to a large database of possible food matches, given only the initial user's meal description, by asking followup clarification questions to narrow down the search space.

Chapter 6

Followup Clarification Questions

A dialogue system will often ask followup clarification questions when interacting with a user if the agent is unsure how to respond. In this chapter, we explore deep reinforcement learning (RL) for asking followup questions when a user records a meal description, and the system needs to narrow down the options for which foods the person has eaten. We build off of Chapter 5, in which we trained novel CNN models to bypass the standard feature engineering used in dialogue systems to handle the text mismatch between natural language user queries and structured database entries, demonstrating that our model learns semantically meaningful embedding representations of natural language. In the nutrition domain, the followup clarification questions consist of possible attributes for each food that was consumed; for example, if the user drinks a cup of milk, the system should ask about the percent milkfat. We investigate an RL agent to dynamically follow up with the user, which we compare to rule-based and entropy-based methods. On a held-out test set, when all followup questions are answered correctly, deep RL boosts top-5 food recall, from 54.9% without followup, to 89.0%. We also demonstrate that a hybrid RL model achieves the best perceived naturalness ratings in a human evaluation.

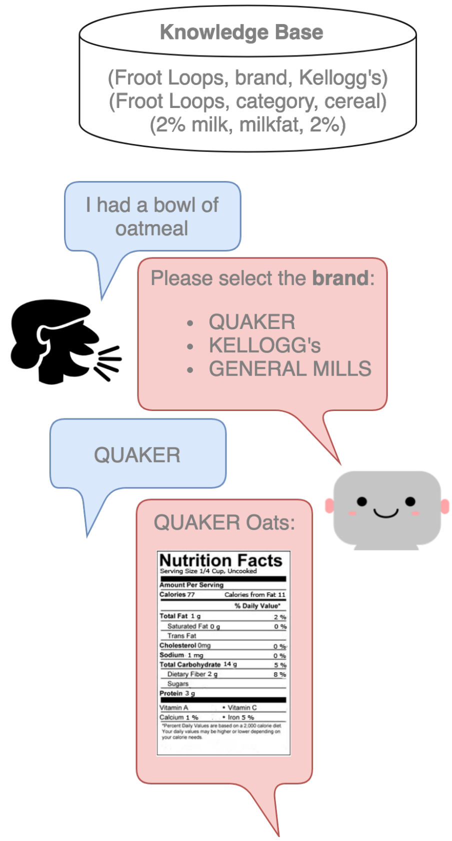


Figure 6-1: In this sample dialogue, the AI agent asks a followup clarification question about the brand of oatmeal.

In this chapter, we continue the difficult task of mapping a natural language meal description to a set of matching foods found in the USDA food database. The huge search space[1] presents a challenge for search algorithms that aim to find the correct USDA food—it is almost impossible to pick the right food from a single user input. Thus, the system needs to ask followup clarification questions to narrow down the search space (see Fig. 6-1 for an example dialogue). However, the system should not ask too many questions of the user, since that may annoy the user. The followup questions must also be intuitive

[1]There are 5,124 food entries in the Standard Reference subset that we use and 215,557 branded food products.

for humans (e.g., the system should not ask about percent milkfat for vegetables), so the system needs to learn which attributes are reasonable to ask, for which foods. Hence, we must balance food recall with efficiency and ease of use—suitable for applying the reinforcement learning (RL) framework.[2]

We use supervised learning to train the initial food ranker (Korpusik et al., 2017a); however, when asking followup questions, it is unclear which order is best at each turn of the dialogue. We do not know ahead of time whether the questions asked will yield the optimal ranking until the dialogue is complete, and at the same time, we wish to finalize the ranking as quickly as possible to avoid annoying the user with excessive questioning. Thus, we investigate deep RL that relies on a reward function for determining the best order of food attributes to ask in order to narrow down the top-500 USDA hits as quickly as possible, while keeping the correct match ranked highest.

In our setup, we train a deep Q-network (DQN) to predict Q-values (i.e., the expected sum of future rewards) for actions (i.e., food attributes, such as the brand). In our experiments with a logistic Q-function, we verify that the DQN learns better value functions than linear approximators, as shown previously (He et al., 2016; Narasimhan et al., 2015). The agent selects the question with the highest Q-value and asks the user to select the correct option for that attribute. The current ranked list of USDA hits gets updated accordingly, the new state is fed to the Q-network to determine the next best action to take, and so on, until the dialogue ends when there are no more questions left or fewer than five USDA hits remaining in the top-n. The full system is depicted in Fig. 6-2.

We compare the RL agent to a rule-based model and an entropy-based solution. In addition, in some tasks, expert knowledge can help guide machine learning models (e.g., intuitively, it makes sense to ask how much milkfat is in milk, but not caffeine for veg-

[2]Our RL reward function favors short dialogues (ease of use) and followups that lead to the correct food (high recall).

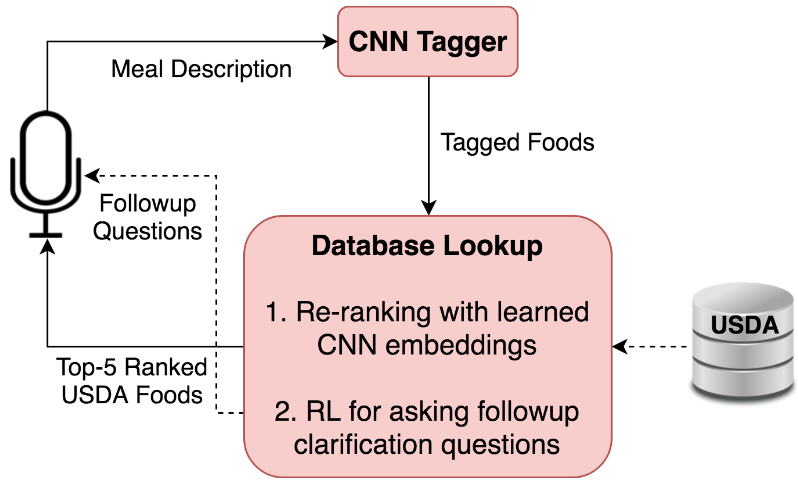


Figure 6-2: The full system framework, where the user's meal is first tagged, then passed to the database lookup component, which consists of re-ranking based on learned CNN embeddings, followed by RL for asking followup clarification questions to narrow down the top-n ranked USDA hits to five.

etables). With this motivation in mind, we explore a new hybrid model for combining expert-engineered solutions with deep RL. We discover a tradeoff between asking fewer questions and achieving high food recall. Evaluated on humans, the rule-based model has fewest turns and lowest recall, while entropy has the most turns and highest food recall. Hybrid RL achieves a balance between the two, with 4.15 turns on average, 89.4% top-5 recall, and significantly better frustration and naturalness ratings.

6.1 Related Work

Although RL has been used for many years in a wide array of fields, including dialogue systems and robot planning, only recently has deep RL begun to gain popularity among NLP researchers. Mnih et al. (2013)'s work on playing Atari games led the shift from previous state-of-the-art Markov decision processes (Young et al., 2010) to the current use of

Q-networks to learn which actions to take to maximize reward and achieve a high score. While Mnih et al. (2013) used convolutional neural networks with video input for playing games, and Narasimhan et al. (2015) used deep RL for playing text-based games, the same strategy is also applicable to dialogue systems. Li et al. (2016b) showed that deep RL models enabled chatbots to generate more diverse, informative, and coherent responses than standard encoder-decoder models. Other work leveraged RL to construct a personalized dialogue system for a coffee-ordering task, where action policies were the sum of general and personal Q-functions (Mo et al., 2016). Li et al. (2016a) used RL in the movie domain to learn when to ask a teacher questions, and showed that the learner improved at answering factual questions about a movie when it clarified the question or asked for help or additional knowledge. Similar to our hybrid model that balances a rule-based approach with RL, Henderson et al. (2005) used a hybrid supervised and RL technique trained on the COMMUNICATOR corpus for flight booking, although they found that a supervised approach outperformed the hybrid.

Deep reinforcement learning has also been successfully applied to task-oriented dialogue, which is similar to our diet tracking application. Zhao and Eskenazi (2016) used a deep Q-network to jointly track the dialogue state with an LSTM network and predicted the optimal policy with a multilayer perceptron to estimate the Q-values of possible actions. In their case, the goal was to guess the correct famous person out of 100 people in Freebase by playing a 20 questions game and asking the user simulator questions relating to attributes such as birthplace, profession, and gender. While their questions are binary Yes/No questions, ours have many options to choose from. Another similar work responded to users with a matching movie entity from a knowledge base after asking for various attributes, such as actors or the release year (Dhingra et al., 2016). Their system first learned from a rule-based agent and then switched to RL. They used a GRU network to track dialogue state and another GRU with a fully-connected layer and softmax on top to model policies. The main difference between our work and Dhingra et al. (2016)'s is they modeled uncertainty over database slots with a posterior distribution over knowledge base entities. Williams and Zweig (2016) focused on initiating phone calls to contacts in an address book, and discovered that RL after pre-training with supervised learning accelerated the learning rate. Peng et al. (2017) used a hierarchical deep RL model to conduct dialogues with multiple tasks interleaved from different domains.

Finally, Li et al. (2017b) focused on optimizing all components in a task-oriented dialogue system simultaneously, in a single end-to-end neural network trained with RL for booking movie tickets. In our case, the tagger and database ranker are separately trained components, and the RL policy is learned solely for asking followup clarification questions, without affecting the other steps in the pipeline. In future work, it would be interesting to explore jointly learning tagging, database mapping, and asking followup questions all in one model. In addition, since we do not have any dialogue data, we cannot train a user simulator and dialogue manager as is typically done. Instead of allowing open-ended responses from the user, the system provides a sample of possible options from which the user selects one. Therefore, the user simulator does not need to generate responses, and can be assumed to select the correct option each time (or we could introduce random noise if we wanted to simulate users occasionally answering questions incorrectly). Finally, the tagger in Li et al. (2017b)'s work is an LSTM that jointly predicts intent and slots. In our work, the tagger is a CNN that only predicts slots, since there is only one intent (i.e., logging food). In summary, our RL agent for asking followup clarification questions is easily ported from one dialogue system and domain to another, where all of the components are already trained, rather than requiring the entire system to be trained from scratch, and our approach works even when no training dialogues are available.

6.2 Models

The CNN re-ranking approach described in Section 5.3.1 only yields 54.9% top-5 test set recall (i.e., how many times the correct USDA food appears in the top-5 ranking), but we would ideally achieve 90% top-5 recall in a system deployed to real users. Thus, in this chapter, we demonstrate that asking followup clarification questions about food attributes boosts top-5 recall to 89.0%. We investigated two hybrid reinforcement learning (RL) models: one using the rule-based method, and another based on entropy. The hybrid approach asks the first question according to the rule-based or entropy-based method's ordering, and afterward selects actions using the RL strategy. This method enables us to combine an expert-engineered solution (either starting with the most intuitive high-level question about category defined by hand-crafted rules, or computed using entropy), while also applying a deep Q-network to learn which attributes to ask next.

6.2.1 Rule-based Followup

In this baseline approach, the dialogue agent asks about each food attribute in a pre-defined order: category, name, brand, type, milkfat, fat, sweetness, addons, salt, packing style, preparation method, and caffeine (see Table 6.1). Any attributes for which the remaining hits all have the same value are skipped, since asking the value of these attributes would not narrow down the USDA hits any further.

6.2.2 Entropy-based Followup

In a 20-questions-style task, such as that implemented by *Zhao & Eskenazi* to guess famous people in Freebase (Zhao and Eskenazi, 2016), an analytic solution based on an entropy measurement may be used to determine the optimal question to ask at each dia-

Attribute	Examples
Category	apples, cereal, pork, snacks
Name	apples, oatmeal, ham, popcorn
Brand	QUAKER, DOMINO'S
Type	instant, wheat, cheddar
Milkfat	nonfat, 1%, 2%, whole
Fat	low fat, reduced fat
Sweetness	sweetened, unsweetened
Addon	with beef, with added vitamin A
Salt	salted, unsalted, low sodium
Packing	regular pack, water pack
Preparation	raw, cooked, microwaved, boiled
Caffeine	with caffeine, no caffeine

Table 6.1: Possibles actions (i.e., food attributes) available to the model, with example food items for each.

logue turn. In this scenario, Yes/No questions are asked (e.g., "Is it a man?") in an order chosen to minimize the number of questions required. Thus, the goal is to select the action that maximizes information gain (i.e., the reduction in entropy after asking a question), where entropy is defined as:

$$H(X) = \sum_{x \in X} P(x) \log_2 \frac{1}{P(x)} \tag{6.1}$$

However, in our work, instead of asking simple binary Yes/No questions, we are asking more complex questions that have multiple options as choices for the user to select from.[3] Instead of asking, "Was the brand Kellogg's?" we are asking, "Which brand was it?" This is similar to the entropy-based questions asked in Gorry et al. (1973) with multiple symptom options used to diagnose acute renal failure. We select the food attribute with the maximum entropy at each turn. For a given attribute, we define X as the set of possible

[3] If we were to ask Yes/No questions, we would have to choose from 4,880 possible (attribute, value) pairs.

values among the current top-n ranked foods (including null), and compute $H(X)$ via Eq. 6.1. See Fig. 6-3 for an example of how entropy would select percent milkfat as the next action, since each yogurt option has a different milkfat, whereas all the yogurts are from the same dairy category, which would yield no new information if asked.

Food	Category Attribute	Milkfat Attribute
01116 Yogurt, plain, whole milk	Dairy	whole
01117 Yogurt, plain, low fat	Dairy	2%
01118 Yogurt, plain, skim milk	Dairy	nonfat

Figure 6-3: A simple example of followup with entropy, where the question that narrows down the search space most (i.e., reduces entropy most) is the percent milkfat food attribute, whereas asking about category yields no new information.

6.2.3 RL Followup

Since the rule-based approach always asks questions in the same order, we investigated whether a machine learning approach could figure out the best order of questions to ask in order to boost food recall further over the deterministic ordering. We do not know the optimal order until the end of the dialogue is reached and we can check whether the matching USDA hit was in the top-5 results, so we explored RL for this task because it uses delayed rewards computed at the end of each dialogue to update the model. As in a typical RL setup, the agent performs actions given the current **state** of the environment, and these **actions** result in **rewards**, which the agent learns from in order to choose future actions that maximize reward.

State The state s consists of the food segment's tokens (e.g., "bacon"); the USDA food IDs remaining in the narrowed-down, ranked list of matches; and the list of remaining actions, where each index is a possible food attribute, and the value at that index is 1 if the action has not been asked yet, or 0 if it has already been asked. For example, the binary action mask would be [0, 1, ... ,1] after one turn where the system asked about category (assuming the first action refers to the category attribute).

Action At each step, the RL agent must determine which action a to take by selecting one of the food attributes (see Table 4.5) to ask a followup question about. Given state s_t, an action a_t is selected either randomly with probability ϵ, which decays from 1.0 to 0.01 at a rate of 0.995 in each minibatch, or as the argmax of the Q-network.

Reward r is defined as in (Dhingra et al., 2016):

$$r = \begin{cases} -0.1 \times \textit{turn} & \text{if dialogue not done} \\ 2(1 - (\textit{rank} - 1)/5.0) & \text{else if rank} \leq 5 \\ -1 & \text{otherwise} \end{cases} \tag{6.2}$$

where *turn* refers to the index of the followup question that is being asked, and *rank* is the final ranking of the correct food item (i.e., 1 is best).

The RL agent uses a two-layer feed-forward neural network (see Fig. 6-4) to estimate the Q-value (i.e., the expected sum of discounted future rewards). The dialogue is considered done when there are no more attributes remaining, or there are fewer than five USDA hits left to narrow down. Every 500 steps, the network gets updated based on a randomly sampled minibatch of 16 previously stored turns (i.e., using experience replay (Zhao and Eskenazi, 2016)).

The Q-network predicts Q-values for each action given the current input state s with a softmax layer. We define our policy as selecting the next action a either randomly with probability ϵ (i.e., exploration) or via the argmax of the predicted Q-values with probability $1 - \epsilon$ (i.e., exploitation). The loss for the Q-network, given chosen action a, is as follows,

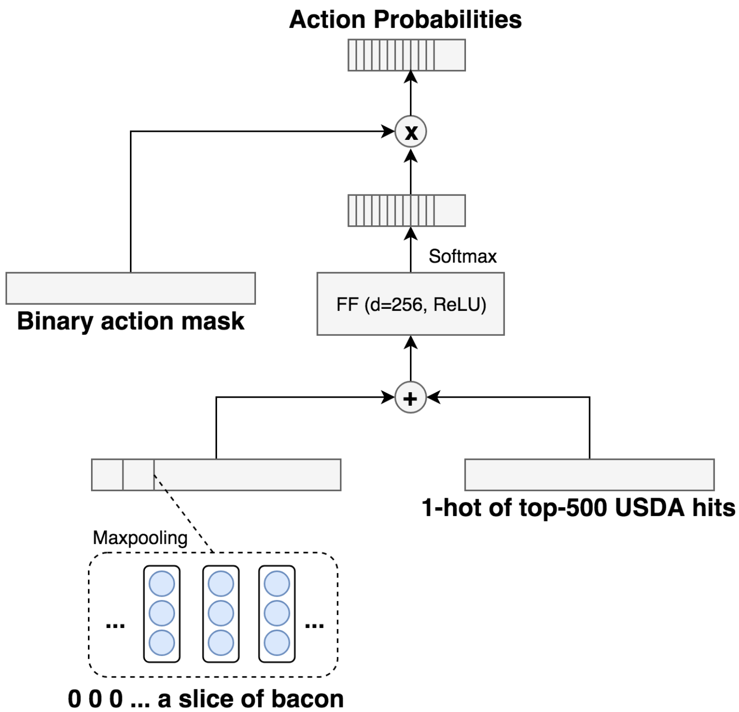


Figure 6-4: The RL Q-network architecture, composed of a simple feed-forward (FF) layer followed by a softmax, which takes as input the tagged food segment (embedded and max-pooled), concatenated with a 1-hot vector of the top-500 ranked hits. The softmax output is multiplied by a binary action mask with zeros masking out all the previously selected actions and ones for the remaining actions. The first step is ranking all possible USDA food database matches, or hits, and selecting the top-n ($n = 500$). Each of the USDA foods is assigned an index in the 1-hot vector where the number of dimensions is the number of unique USDA foods, and the vector is initialized with zero. The foods that remain in the top-n ranking (starting with 500, and narrowed down after each turn) are assigned a value of one in the 1-hot vector.

where discount factor $\gamma = 0.9$:

$$L = \frac{1}{2}(r + \gamma \max_{a'} Q(s', a') - Q(s, a))^2 \tag{6.3}$$

As in Algorithm 1, during training, we first initialize the experience replay memory and Q-network. Then, for each training sample, we iterate through the cycle shown in Fig. 6-5 until the dialogue ends. Each dialogue begins with the user's food description (e.g., "a slice of bacon"), which is converted to the start state s_1. Given the current state s_t, an action a_t is selected either randomly with probability ϵ, which decays from 1.0 to 0.01 at a rate of 0.995 during each minibatch update, or as the argmax of the Q-network.

Using selected action a_t, the system follows up by asking about the chosen food attribute: "Please select the *category* for bacon: meat, oils..." and the user selects the correct attribute value (e.g., meat). The reward r_t is computed and new state s_{t+1} determined by narrowing down the remaining top-n USDA hits based on the user's chosen attribute value. The experience $(s_t, a_t, s_{t+1}, r_t, done)$ is saved to the replay memory, where $done$ is a boolean variable indicating whether the dialogue has ended. The loop repeats, feeding the new state s_{t+1} into the Q-network again to generate Q-values for each action, until the dialogue ends and $done$ is true. The dialogue ends when there are five (or fewer) foods remaining, and the system returns the ranked list of USDA hits (e.g., "The results for bacon are: 10862–Pork, cured, bacon, pan-fried; 10998–Canadian bacon, cooked, pan-fried").

We use Adam (Kingma and Ba, 2014) to optimize the Q-network, ReLU activation and a hidden dimension size of 256 in the feed-forward layer, and 50-dimension embeddings. We process the user's raw input by tokenizing with the Spacy toolkit. Each token is converted to a vocabulary index, padded with zeroes to a standardized length (i.e., the longest food segment), and fed through an embedding layer mapping each token index to a 50-dimensional vector. Maxpooling selects the maximum value across all tokens for

Algorithm 1 RL Training Algorithm

Initialize experience replay memory $\mathcal{D}$
Initialize DQN parameters θ
for $i = 1, N$ **do**
    Initialize start state s_1 by ranking top-50 hits for meal
    while $\neg done$ **do**
        if $random() < \epsilon$ **then**
            Execute random action a_t
        else
            Execute action $a_t = \text{argmax} Q(s_t, a)$
        Observe next state s_{t+1} and reward r_t
        Determine whether dialogue is $done$
        Store memory $(s_t, a_t, r_t, s_{t+1}, done)$ in $\mathcal{D}$
        Sample random mini batch of memories $(s_j, a_j, r_j, s_{j+1}, done_j)$ from $\mathcal{D}$
        $y_j = \begin{cases} r_j & \text{if } done_j \\ r_j + \gamma \max_{a'} Q(s_{j+1}, a'; \theta_t) & \text{else} \end{cases}$
        Perform gradient descent step on the loss $\mathcal{L}(\theta) = \frac{1}{2}(y_j - Q(s_j, a); \theta)^2$

each dimension, resulting in a single 50-dimensional vector representation of the user's input. The 1-hot vector of top-500 hits is a binary list of all possible USDA hits in the food database, where each index corresponds to a food item, and the value at that index is 1 if that food is still in the ranked list, or 0 if not.

6.3 Experiments

For all our experiments, we used the written food descriptions and corresponding USDA food database matches collected on AMT, as described in Section 3.2.4, with 78,980 samples over the full USDA database of 5,156 foods, where Turkers wrote natural language descriptions about USDA foods (Korpusik and Glass, 2018a). To build the food knowledge base, we parsed each USDA entry and determined the values for the 12 manually defined food attributes (see Table 6.1) with a set of heuristics and string matching. We

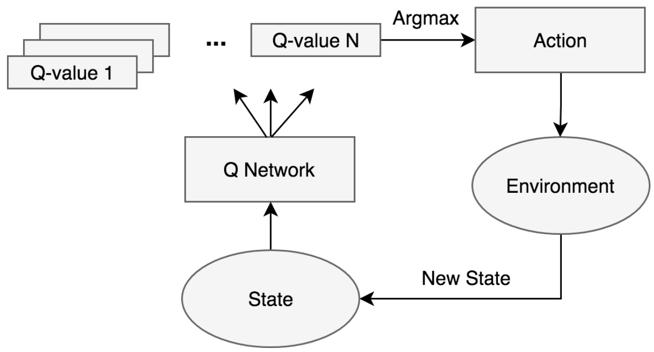


Figure 6-5: The RL flow, where the current state is fed through the Q network to generate Q-values for each possible action. The action with the highest Q-value (or a random action) is selected, which is used to determine the reward and update the new state, continuing the cycle until the dialogue ends.

computed top-1 and top-5 USDA food recall (i.e., percentage of instances in which the correct USDA match was ranked first or in top-5) on 10% held-out test data.

6.3.1 Results from a User Simulator

We see in Table 7.16 the performance of each of our expert-engineered and RL models for asking followups, along with the baseline re-ranking method taken from Chapter 5. We also compare against a logistic regression baseline, which performs significantly worse than deep RL, illustrating why we need a deep Q-network with more than one layer. These results rely on a user simulator that is assumed to always choose the correct attribute value for the gold standard food item. The ground truth data consist of user food logs (e.g., "a slice of bacon") and matching USDA foods (e.g., 10862–Pork, cured, bacon, pan-fried). At each turn, the system asks the user to select an attribute value (e.g., "Please select preparation style for bacon: fried, baked..."), and the simulator selects the value for the correct USDA food (e.g., `preparation style=fried`).[4]

[4]In our work, if an attribute does not apply to a particular USDA food, it is assigned the `null` value, which is one solution for handling attributes that do not apply to all answer candidates.

All the followup approaches significantly outperform the baseline, boosting top-1 recall from 27.3% to 68.0% with entropy. The rule-based and entropy-based solutions are at opposite ends of the spectrum: using rules results in shorter dialogues, but lower recall, whereas the entropy solution has the highest recall, but longer dialogues. The RL agents find a balance between short dialogues and high recall. This tradeoff is similar to that in (Varges et al., 2010), where longer dialogues achieved higher precision with followup. This is because the rule-based method asks questions that have many possible attribute values as options, so when one of these options is chosen, the dialogue is already close to completion; however, since we limit the options shown to 10 to avoid overwhelming the user when the system is used by humans, the correct USDA food may be omitted, lowering food recall. The hybrid RL model strikes a balance between the rule-based method with fewer turns, and the entropy solution with high food recall.

Model	Turns	Top-1 Rec.	Top-5 Rec.
Re-ranking **	N/A	27.3	54.9
Entropy-based	6.00	**68.0**	**89.5**
Hybrid entropy	4.96	65.1	89.2
Rule-based **	**3.03**	59.4	89.1
Hybrid rules	4.15	66.4	89.4
Logistic RL **	3.60	64.6	88.4
Deep RL **	4.21	66.6	89.0

Table 6.2: Food recall with followups, compared to a re-ranking CNN baseline without followup (Chapter 5). Statistical significance (t-test based on final rank of the correct food) is shown, for **hybrid rules-RL** compared to other methods (** for $p < .00001$).

6.3.2 Results from an Online Study with Humans

While the user simulator answers every question correctly, this is likely not true for humans interacting with the live system. Thus, we cannot assume the user will perfectly answer every question. In addition to maximizing recall, we want to minimize frustration. Therefore, we conducted studies on AMT to evaluate the various systems on both metrics, recall and ease-of-use, to confirm our hypothesis that the questions asked by the RL models are indeed the most intuitive and natural to people, enabling them to answer questions more accurately and resulting in higher recall scores.

We incorporated our followup clarification dialogue system into a server and built an AMT task in which workers interacted with the system to narrow down the top-500 food options to the correct match, given the spoken food description and correct USDA food. We showed the possible attribute values at each turn, and displayed the top-10 food names per option (see Fig. 6-6). We paid $\$0.10$ per task, and evaluated 1000 test samples for three models: rules, entropy, and hybrid rules-RL. We asked Turkers to rate the interaction for perceived accuracy, naturalness, and ease-of-use on a scale of 1 to 3, where 3 is the best.

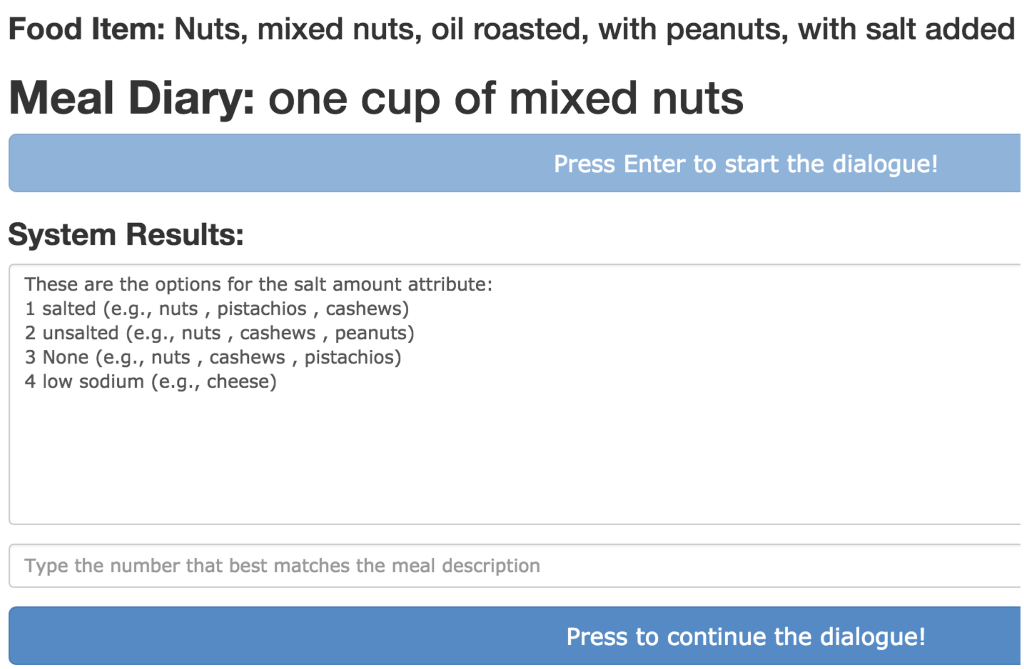


Figure 6-6: The AMT task for online evaluation of the dialogue system.

In Fig. 6-7, we see the same pattern for food recall on humans as with the user simulator: the entropy-based solution has the highest recall scores, with hybrid RL in between, and rule-based the worst. We see in Fig. 6-8 that the rule-based model has the shortest dialogues, again, and entropy-based longest. Interestingly, in Fig. 6-9, note that the hybrid rules-RL model has significantly better naturalness and ease-of-use scores than rules and entropy, respectively. Despite entropy's high recall, Turkers rated it as least accurate.

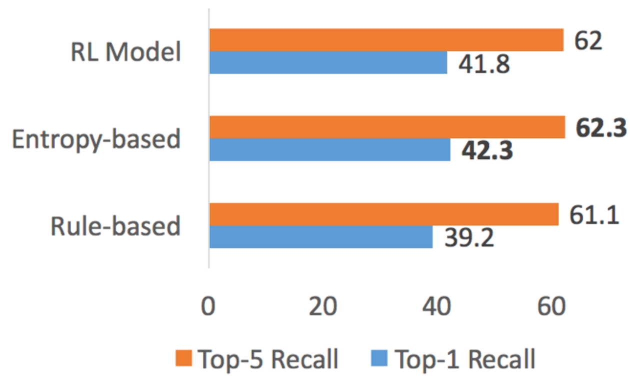


Figure 6-7: Top-1 and top-5 food recall per model, evaluated on AMT.

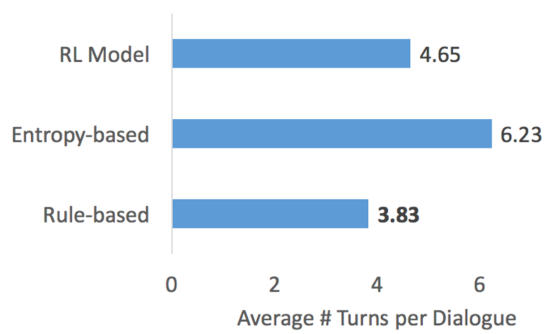


Figure 6-8: Average turns per model, evaluated on AMT.

An analysis of the order of attributes asked by the RL model indicates a meaningful relationship with the target USDA food item described by the user. For example, in Table 6.3, we see that the model asks about brands for branded food products; fat for meat,

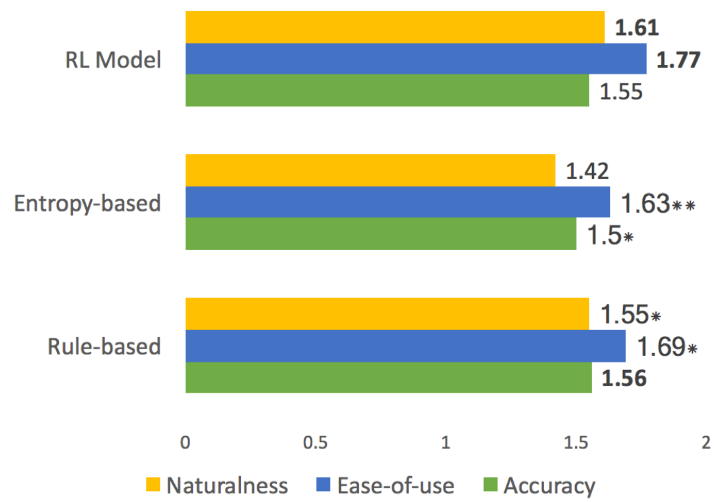


Figure 6-9: Average perceived accuracy, naturalness, and ease-of-use scores (higher is better) in the AMT study (* for $p < 0.05$, as compared to **hybrid rules-RL**).

baked goods, and dairy; salt for vegetables and cheese; and addons for composite foods.

6.3.3 Sample Dialogues

We show sample interactions with the hybrid rules-RL model in Table 6.4 for the meal "I had eggs with bacon and a glass of milk," where we assume the user has eaten fresh scrambled eggs, regular bacon (i.e., as opposed to meatless or low sodium), and a glass of 1% milk. We observe that, in general, the questions asked by the hybrid RL model seem fairly intuitive. For example, it asks about the preparation style of the eggs and bacon, as well as the milkfat and addons for milk (i.e., with or without added vitamins). We show sample interactions with the entropy-based and rule-based models in Tables 6.5 and 6.6, respectively, for the same meal. Note the contrast between the short interactions with the rule-based model, versus the lengthy conversations with entropy.

Attribute Followup	Example Foods
Brand	ON THE BORDER, taco ...
Brand	McDONALD'S Vanilla ...
Brand	Cereals, KASHI ...
Fat	Cookies ... reduced fat
Fat	Beef, ground, 93% lean meat ...
Fat	Cheese, ... low fat
Packing style ...	Fruit-flavored drink ...
Packing style ...	Plums, raw
Packing style ...	Tomato sauce, canned ...
Salt	Lentils ... without salt
Salt	Pickles ... reduced sodium
Salt	Cheese, parmesan, low sodium
... Addons	... with cheese and sausage
... Addons	... with added nutrients
... Addons	... with vitamin E added

Table 6.3: Example target USDA foods for five learned followup patterns asked with RL.

6.4 Summary and Future Work

In this chapter, we demonstrated the success of deep RL for asking followup clarification questions in the novel nutrition domain. We compared RL to rule-based and entropy-based solutions for asking followup clarification questions, demonstrating a tradeoff between shorter dialogues and higher food recall. Asking followup questions significantly boosts recall, from 27.3% without any clarifications, to 66.4% top-1 recall with hybrid rules-RL followup. We also demonstrated that the hybrid RL model achieves the highest perceived naturalness and frustration scores when evaluated on humans.

In future work, we will add an option for users to speak or type a response to followup questions and will automatically narrow down the options as the user speaks or types. We will make use of the attributes specified in the user's initial meal diary to avoid re-asking that attribute. Finally, we will investigate RL for online learning of user preferences.

Table 6.4: Sample dialogue with the **hybrid RL** approach, which selects the **correct** top-1 hit for all eaten foods.

I had eggs with bacon and a glass of milk.
System: Please select the **category** for: **eggs** • egg • sweets • meals
User: egg
System: Please select the **sweetness**: • None • sweetened
User: None
System: Please select the **preparation style**: • raw • dried • None • frozen • poached • fried • scrambled • hard-boiled
User: scrambled
Results for: **eggs** • 01132: Egg, whole, cooked, scrambled ✓
System: Please select the **category** for: **milk** • milk • snacks • sweets
User: milk
System: Please select the **preparation style**: • None • canned • dried
User: None
System: Please select the **addon**: • without added vitamins • with lauric acid oil • with added vitamins
User: with added vitamins
System: Please select the **milkfat**: • nonfat • None • whole • 2% • 1%
User: 1%
Results for: **milk** • 01082: Milk...1% milkfat, with added vitamins ✓ • 01104: Milk, chocolate..., with added vitamins

System: Please select **category** for: **bacon** • meat • oils • vegetables
User: meat
System: Please select **preparation style**: • None • cooked • fried • baked • heated
User: fried
Results for: **bacon** • 10862: Pork, cured, bacon, ...pan-fried ✓ • 43378: Pork...bacon...reduced sodium • 10998: Canadian bacon, cooked, pan-fried

Table 6.5: Sample dialogue using the **entropy** solution, which asks many questions, some of which are odd, given the food (e.g., milkfat for eggs). If the user mistakenly selects salted instead of `None`, only raw eggs or egg noodles are returned.

I had eggs with bacon and a glass of milk.

System: Please select the **milkfat** for: **eggs** • None • whole
User: None
System: Please select the **fat**: • None • nonfat
User: None
System: Please select the **sweetness**: • None • sweetened
User: None
System: Please select the **salt**: • None • salted • unsalted
User: salted
Results for: **eggs** • 01160: Egg, yolk, raw, frozen, salted... • 01255: Egg, whole, raw, frozen, salted... • 20310: Noodles, egg, cooked...with added salt • 20510: Noodles, egg...unenriched, with added salt

System: Please select the **fat** for: **bacon** • None • rendered fat
User: None
System: Please select the **sweetness**: • None • sweetened
User: None
System: Please select the **salt**: • None • low sodium
User: None
System: Please select the **preparation style**: • fried • baked • heated • cooked • frozen
User: fried
Results for: **bacon** • 10862: Pork, cured, bacon, ...pan-fried ✓ • 10998: Canadian bacon, cooked, pan-fried

System: Please select the **salt** for: **milk** • None • low sodium
User: None
System: Please select the **sweetness**: • None • sweetened • low calorie sweetener • unsweetened
User: None
System: Please select the **preparation style**: • None • canned • raw • frozen • dried
User: None
System: Please select the **fat**: • None • nonfat
User: None
System: Please select the **brand**: • None • MARS • HERSHEY'S • ROLO • CYTOSPORT • SYMPHONY
User: None
System: Please select the **milkfat**: • whole • None • nonfat • 1% • 2%
User: 1%
Results for: **milk** • 01175: Milk, fluid, 1% fat, without added vitamins • 01082: Milk...1% milkfat, with added vitamins ✓ • 01104: Milk, chocolate...lowfat, with added vitamins • 01088: Milk, buttermilk, fluid, cultured, lowfat

Table 6.6: Sample dialogue with the **rule-based** approach, which asks fewer questions, but yields incorrect results for eggs and bacon due to ambiguous names (e.g., egg vs. eggs, and bacon vs. pork).

I had eggs with bacon and a glass of milk.

System: Please select the **category** for: **eggs** • egg • sweets • meats
User: egg
System: Please select the **name**: • egg • egg yolk • egg white • egg substitute • eggs
User: eggs
Results for: **eggs** • 43285: Eggs, scrambled, frozen mixture

System: Please select the **category** for: **milk** • milk • snacks • sweets
User: milk
System: Please select the **name**: • milk • ready-to-drink reduced fat beverage • chocolate milk • almond milk
User: milk
System: Please select the **type**: • whole • filled • dry • imitation • None • nonfat • human • fluid • reduced fat • lowfat • goat
User: lowfat
Results for: **milk** • 01082: Milk...1% milkfat, with added vitamins ✓

System: Please select the **category** for: **bacon** • meat • oils • vegetables
User: meat
System: Please select the **name**: • bacon • bacon bits • bacon and beef sticks • pork • turkey bacon • canadian bacon
User: bacon
Results for: **bacon** • 16104: Bacon, meatless

Chapter 7

Dialogue State Tracking

In the previous chapters, we focused on semantic tagging and database mapping (i.e., the language understanding component) in a spoken food logging system. In this chapter, however, we turn to a different piece of the dialogue system pipeline—dialogue state tracking and response selection, which are both part of the dialogue manager (see Fig. 1-1), and we move into other domains beyond nutrition, such as making restaurant reservations.

Spoken dialogue systems have gained popularity recently with the rise of personal intelligent assistants such as Siri and Cortana. These systems can be divided into two categories: chatbots that entertain the user through fun conversation, and task-oriented dialogue systems that accomplish a goal for the user, such as making a restaurant reservation or booking a flight. The standard approach for task-oriented dialogue typically follows a pipeline of steps, starting with intent detection (i.e., determining the user's goal), followed by SLU of the user utterance to determine precisely what the user is requesting. For example, the user's intent may be to book a restaurant, for which the relevant semantic tag values would be `Chinese` for the `food` slot, and `centre` for the `area` slot. Subsequently, the user's goal is updated based on the output of the SLU component, the next system action is selected via the predicted user goal, and finally the system responds according to the chosen action (see Fig. 7-1 for the standard pipeline).

In such a framework, however, there are several drawbacks. First, the SLU component is an intermediate step between the user query and updating the dialogue state (i.e., revising the current prediction for the user's goal) (Young et al., 2010; Wang and Lemon, 2013; Williams, 2014), which may cause errors to accumulate further down the pipeline. Recent approaches avoid this by eliminating the intermediate SLU and directly tracking the state of the dialogue, given the user utterance (Zilka and Jurcicek, 2015; Mrkšić et al., 2015). In addition, prior approaches often relied on hand-crafted features, such as semantic dictionaries that map words to synonyms of each other (e.g., `area` may map to `part of town`), or even entirely rule-based dialogue systems. Current approaches explore end-to-end neural models for dialogue management instead (Mrkšić et al., 2016; Zhong et al., 2018). In this chapter, we show through three dialogue state tracking challenges how 1) our convolutional architecture generalizes to new tasks and domains, and 2) we can leverage deep learning to avoid the typical pipeline of steps required for dialogue management.

7.1 Related Work

Traditionally, spoken dialogue systems relied on separately trained components for SLU and dialogue state tracking. The SLU component would identify slot-value pairs from the speech recogition output, which would be passed to the state tracking module to update the belief state (Thomson and Young, 2010; Wang and Lemon, 2013). However, this pipeline of steps would accumulate errors, as the SLU component often would not have the necessary context to accurately predict the slot values. Thus, belief tracking research shifted to jointly predicting slot-value pairs and updating the dialogue state (Henderson et al., 2014c; Sun et al., 2014).

Typically, these jointly trained SLU and dialogue state updating models rely on a delexicalization strategy, which translates various instantiations of slot and value mentions in the user utterance into generic labels; this approach requires hand-crafted semantic dictionaries in order to perform the mapping from specific wordings to generic slot-value labels. Prior work by Henderson et al. (2014b) fed delexicalized user utterances into a recurrent neural network, which output a distribution over slot values. However, delexicalizing the input requires a manually defined semantic dictionary that maps from slot-value pairs to all possible text forms, or synonyms (see Table 7.1 for examples).

Slot-Value	**Synonyms**
Food=Cheap	[affordable, budget, low-cost, low-priced, ...]
Area=Centre	[center, downtown, central, city centre, ...]
Rating=High	[best, high-rated, highly rated, top-rated, ...]

Table 7.1: Example rephrasings for three slot-value pairs in a semantic dictionary for restaurant booking.

To avoid this reliance on hand-crafted semantic dictionaries, Mrkšić et al. (2016) recently demonstrated the ability of their Neural Belief Trackers (NBT) to match the performance of delexicalization-based models, without requiring any hand-crafted semantic dictionaries, as well as the ability to significantly outperform such models when the semantic resources are not available. However, the NBT still requires pre-trained word vectors tailored to retain semantic relationships. While our work is similar to theirs in that we both leverage CNNs for dialogue state tracking, our work, on the other hand, does not rely on pre-trained word vectors, and directly predicts matching slot values instead of doing binary classification for each slot-value pair; in contrast, the NBT is trained to learn representations of user utterances and slot-value pairs that are used for binary classification (i.e., whether or not a given slot-value pair is mentioned in the user utterance).

In addition, Zhong et al. (2018) state-of-the-art work has explored deep learning methods for dialogue state tracking, but with recurrent (instead of convolutional) self-attentive encoders, and again considers each slot-value pair one at a time, while we predict the matching slot value from among all options simultaneously. Their self-attentive RNN model encodes user utterances, system actions, and each slot-value pair under consideration, but still relies on pre-trained Glove word vectors (Pennington et al., 2014), and character embeddings, while our model does not require pre-trained embeddings. Rastogi et al. (2017) also feed delexicalized utterances into their multi-domain deep learning model for state tracking.

For the DSTC6 challenge, the top-2 performing methods each used separately trained slot-value trackers to refine the initial ranking of system response candidates generated by an action selector (Ham et al., 2017; Bai et al., 2017). We take a similar approach, but use a CNN instead of an LSTM network to select system actions. We also use a CNN for semantic tagging, rather than the CRF used in Bai et al. (2017) or the LSTM inHam et al. (2017) for slot-value tracking, and we use less feature engineering than these systems. Bai et al. (2017) use a heuristic strategy for the final scoring, with two branches based on the preliminary scoring module (e.g., if `api_call` is selected, then update a candidate response's score based on the relative index of the word that last occurred in the dialogue history, using weights according to the output of a separately trained "Uncertainty CRF").

7.2 2nd Dialogue State Tracking Challenge (DSTC2)

In this section, we examine the WOZ 2.0 written dataset for restaurant booking dialogues (Williams et al., 2013; Henderson et al., 2014a), where the task is to predict the state of the dialogue at each turn. In particular, the dialogue state consists of the user's goal at that turn, which is composed of a set of slots that the user is either requesting (e.g., "What is the phone number?" would indicate the requested slot `phone`) or informing the system (e.g., "I want Turkish food," which maps the informable slot `food` to the value `Turkish`). This requires keeping track of the history of the conversation, as well as the context from the previous system response, as illustrated in Table 7.2.

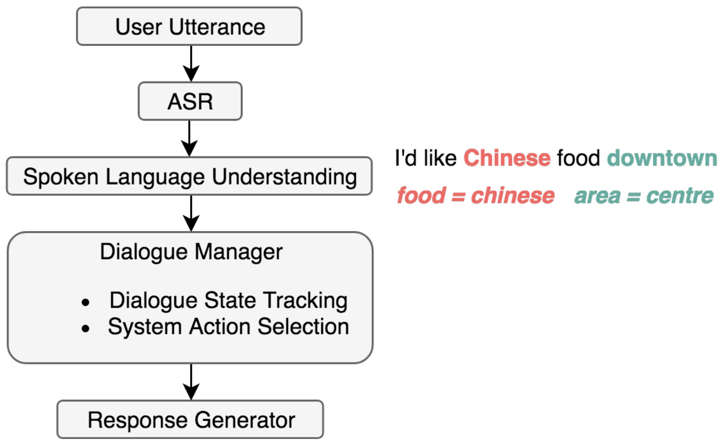


Figure 7-1: The typical flow of a dialogue system, with spoken language understanding followed by dialogue state tracking.

The approach we take in this work is a deep learning model, specifically a CNN architecture for tracking the user's goal at each turn in a restaurant booking dialogue. We show that without relying on any semantic dictionaries or pre-trained word vectors, our model is competitive with state-of-the-art, achieving 86.9% joint goal accuracy and 95.4% requestable slot accuracy.

7.2.1 Models

The goal of our work is to accurately update the current belief state of the dialogue by predicting, at each turn, the correct slot values specified by the user, specifically for a restaurant booking task. There are two types of slots: informable (i.e., the user is providing information about the type of restaurant they want, such as the cuisine), and requestable (i.e., the user is asking for information about the restaurant, such as the telephone num-

User: Is there any place here in the centre that serves corsica food? `food = corsica`; `area = centre`
System: What price range are you looking for? **User:** Any price range will do. `food = corsica`; `area = centre`; `price = dontcare`
System: There are no restaurants available matching your criteria. Would you like to try a different area, price range, or food type? **User:** Are there any restaurants in the centre that serves North American type of food? `food = north_american`; `area = centre`; `price = dontcare`
System: Yes. The gourmet burger kitchen serves north american food and is located in the centre part of the city. Would you like their location? **User:** Can I get the phone number? `food = north_american`; `area = centre`; `price = dontcare`; `requested = phone`
System: The phone number for gourmet burger kitchen is 01223 312598. **User:** Thank you. Good bye. `food = north_american`; `area = centre`; `price = dontcare`

Table 7.2: Example WOZ 2.0 dialogue snippet, with the corresponding slots specified by the user at each turn. Note that midway through the dialogue, the user changes the goal `food` slot value from `corsica` to `north_american`.

ber),[1] as shown in Table 7.3 which enumerates all possible informable and requestable slots, as well as the number of values available for each slot. Since the informable slots are also requestable, the system must differentiate whether the user is providing or requesting information. In addition, there is an imbalance of data, since the `Food` informable slot has many possible values, whereas `Area` and `Pricerange` have fewer than 10, and some slot-value pairs appear more often than others in the training data.[2]

Slot	**Type**	**Num Values**
`Food`	Informable, Requestable	75
`Area`	Informable, Requestable	7
`Pricerange`	Informable, Requestable	4
`Name`	Requestable	N/A
`Address`	Requestable	N/A
`Phone`	Requestable	N/A
`Postcode`	Requestable	N/A
`Signature`	Requestable	N/A

Table 7.3: All possible informable and requestable slots.

As discussed in Section 7.1, prior work has either used hand-crafted features and semantic dictionaries for dialogue state tracking with delexicalization, or neural models relying on pre-trained semantic word vectors, while ours does not.

Below, we describe in detail our novel convolutional neural dialogue state tracker, with two variants: one model with a binary sigmoid output layer indicating the presence of each requestable slot (Fig. 7-3), and a separately trained model for each informable slot (Fig. 7-2) with a softmax output layer to predict the slot value. We discuss two post-processing techniques for boosting performance to illustrate the importance of error analysis for gaining insight into why a system is underperforming and finding a solution based on human

[1] http://camdial.org/ mh521/dstc/downloads/handbook.pdf

[2] Note that requestable-only slots (e.g., address) do not have values that can be specified by the user, since the user is requesting the value.

intuition about the task. Thus, we combine deep learning models with expert knowledge into a hybrid approach in order to overcome the limitations of purely neural methods.

Informable Slot Models

As shown in Fig. 7-2, we separately trained a model for each of the informable slots (i.e., `Food`, `Area`, and `Pricerange`).[3] Each model is composed of an embedding layer (which is not pre-trained, and is learned during training), into which we fed the user utterance concatenated with the previous system response as the input $\boldsymbol{x}$, where $\boldsymbol{x}$ is composed of the sequence of learned word vectors for the input tokens $\boldsymbol{w_1}, \boldsymbol{w_1}, ..., \boldsymbol{w_n}$. The input had two options, chosen via the development set:[4]

1. The user utterance concatenated with the full system response, omitting the system response if the user utterance starts with "no" (i.e., correcting the system), and using only the final question asked by the system.

2. The user utterance concatenated with all the slots requested by the system (i.e., in its dialogue act).

This is followed by a single convolutional layer with maxpooling to get a representation $\boldsymbol{r}$ of the embedded input $\boldsymbol{x}$:

$$\boldsymbol{r} = \texttt{maxpool}(\texttt{ReLU}(\texttt{Conv1D}(\boldsymbol{x}))) \tag{7.1}$$

Finally, a feed-forward layer with a softmax on top is used to directly predict the proba-

[3]We also tried jointly training all the slots, but found that separately training the models boosts performance over a single jointly trained model.

[4]Separately processing the user utterance and system response was worse.

bility of all possible slot values:

$$\boldsymbol{o} = \mathtt{softmax}(W\boldsymbol{r} + b) \tag{7.2}$$

where W is a learned weight matrix and b is a bias term in the final feed-forward layer.

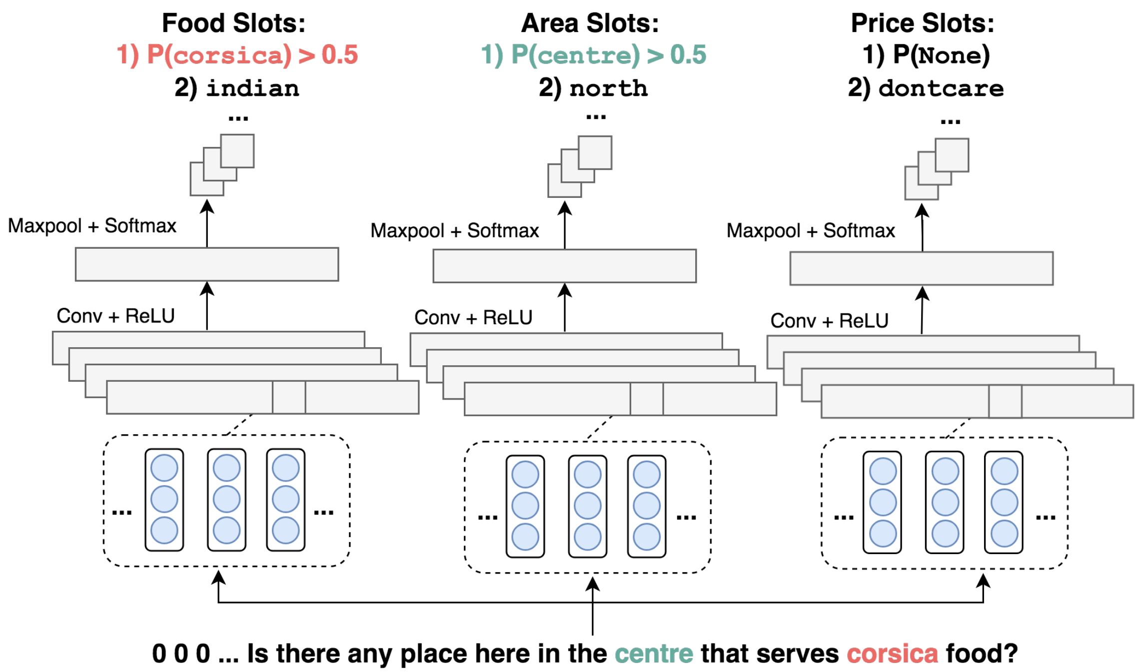


Figure 7-2: The separately trained CNN models for each informable slot type.

Requestable Slot Model

The requestable slot model is also a CNN (shown in Fig. 7-3), but with a separate binary sigmoid output layer for each possible requestable slot (see Table 7.3), instead of one softmax layer on top, as in the informable slot models. The input to this model is the first option we tried for the informable slot models (i.e., the user utterance concatenated with the full system response, omitting the system response if the user says "no," and using only the final question asked by the system).

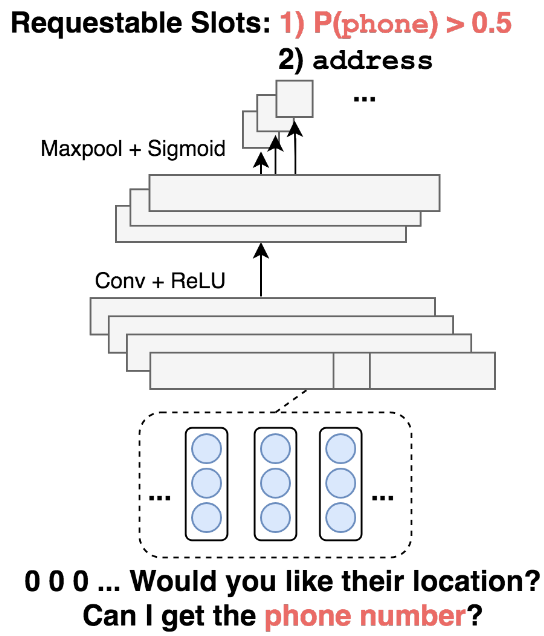


Figure 7-3: The CNN architecture for the requestable slot model.

Hybrid Deep Learning and Knowledge-Based Method

While neural network models are incredibly powerful and have demonstrated success over prior state-of-the-art approaches in many fields, including computer vision, speech recognition, and natural language processing, there are still limitations to using these models that are often referred to as "black boxes." In our work, we simply feed the raw user utterance and system response into the model, which then outputs predicted slot-value pairs, requiring no manual feature engineering, pre-trained word vectors, or semantic dictionaries. However, this can make it difficult to interpret why the model behaves the way it does, and may limit performance since the model does not inherently have common-sense knowledge about the real world, or in this case, the restaurant booking task. Thus, by manually investigating test examples where the system made prediction errors, we are able to boost the model's performance by guiding it in the right direction based on expert knowledge of the task and dataset. Such an approach is illustrative of a hybrid between deep learning without any manual feature engineering, and expert knowledge-based systems, which we use to address the limitations of the purely neural model.

Post-Processing Techniques to Boost Performance

1. Delexicalization of the *input* to the model is common practice on dialogue state tracking tasks. In our work, we take a different approach, and perform string matching of slot values as a post-processing step to correct for any omitted slots (e.g., if some slot values were not seen in training), *after* the model makes its predictions.

2. We also check whether any slots that were requested by the system in a given turn (as specified by the system's dialogue act) were not predicted by that slot's model (i.e., the top value was `None`). If so, we add the next-highest predicted value for that slot to the goal state.

Implementation Details

We pad the input to 51 tokens (i.e., the maximum length of the concatenated user utterances and system responses seen during training). The input 64-dimension embedding layer is followed by a 1D convolution with 64 filters spanning windows of three tokens, with a ReLU activation and dropout of probability 0.2. Each network is trained to predict the matching one-hot label array given the input user utterance; that is, the softmax is trained to predict 1 for each slot-value pair that is specified by the user, and 0 for all others. For the separately trained models, we add a `None` value for each slot type, and assign this value a 1 for each example that does not contain the specified slot type. The model is trained with the Adam optimizer (Kingma and Ba, 2014) on binary cross-entropy loss. The setup is the same for the requestable slot model, with a threshold of 0.5 at test time for each requestable slot.

We tune several threshold hyperparameters on the development set: at the start of a dialogue, we use a threshold of 0.5 for a predicted slot value when adding new slots to the goal state, while a higher threshold of 0.9 is best for adding new slots during the dialogue, and an even higher threshold of 0.99 for updating the value of slots already in the state. In addition, we set a threshold of 0.2 that must be exceeded in order to add slots requested by the system's dialogue act. Finally, the best input for the `Area` slot (and for all slots in the Sim-GEN movie booking task in Section 7.2.2) is the full system response concatenated with the user input, while the best input for the WOZ `Pricerange` and `Food` slots is the user utterance concatenated with the system's requested slots.

7.2.2 Experiments

Datasets

For our experiments, we report results on the WOZ 2.0 dataset,[5] in which Turkers assumed the role of the system or user in dialogues similar to those used in the 2nd Dialogue State Tracking Challenge (DSTC2),[6] so we can compare our performance to that of state-of-the-art approaches on a standard dialogue system benchmark. This task involves restaurant booking, where the user specifies his or her goal as a set of informable and requestable slots, as described in Section 7.2.1. The WOZ data is written, not spoken, requiring semantic understanding rather than robustness to speech recognition errors. Our final model is trained on the full training and development set, with hyperparameters tuned on the

[5] http://mi.eng.cam.ac.uk/ nm480/woz_2.0.zip
[6] http://camdial.org/ mh521/dstc/

development set, and is evaluated on the test utterances. To demonstrate our model's generalization capability, we also evaluate on the Sim-GEN dataset of conversations between an agent and a simulated user for buying movie tickets (Shah et al., 2018).

Metrics

As is commonly used in dialogue state tracking experiments, we report results on two slot tracking metrics:

- **Goals**: the proportion of dialogue turns where all the user's *informable* slots (i.e., search goal constraints) were correctly identified.
- **Requests**: the proportion of dialogue turns where all the user's *requestable* slots were correctly identified.

Results

As seen in Fig. 7-4, our CNN model without semantic dictionaries or pre-trained word vectors, achieved 86.9% goal accuracy and 95.4% requests accuracy on the held-out WOZ 2.0 dataset. In Fig. 7-5, our CNN outperforms the state-of-the-art hierarchical LSTM for jointly tracking dialogue state and predicting system actions (Liu et al., 2018) on the Sim-GEN movies.

Ablation Study

Here we show the importance of the two post-processing techniques discussed in Section 7.2.1 that combine the deep learning model with expert knowledge based on error analysis. As shown in Table 7.4, the biggest gain in performance on the WOZ dataset is from exact string matching of slot values in the user utterance, and the best overall model

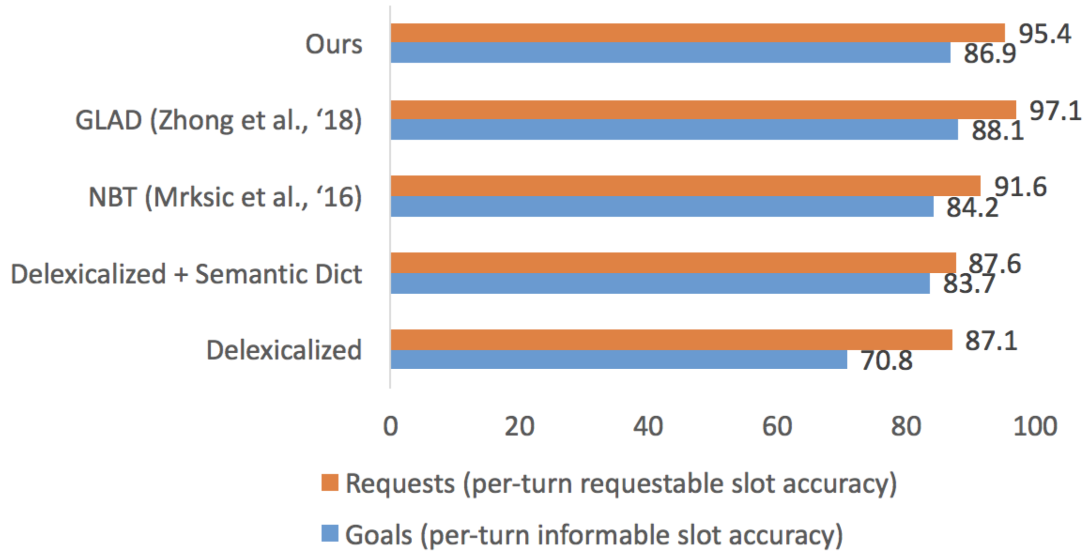


Figure 7-4: Goal and request accuracy of our model compared to a strong delexicalization baseline and two state-of-the-art neural methods: NBT (Neural Belief Tracker) and GLAD (Global-Locally Self-Attentive Dialogue State Tracker).

is achieved by using both techniques. We also note that there is no gain from applying these techniques to the Sim-GEN movies dataset.

Model	WOZ Goals	Sim-GEN Goals
Best	**86.9**	**96.5**
w/o technique 1	76.6	96.5
w/o technique 2	82.6	96.5
w/o technique 1 or 2	72.7	96.5

Table 7.4: Goal development set accuracy of our model, on WOZ and Sim-GEN, without the two post-processing techniques in Section 7.2.1: 1) exact string matching of slot values in the user utterance, and 2) adding the slot value with highest predicted probability for slots requested by the system.

We see examples of errors made by the model on WOZ in Table 7.5, where due to synonyms such as "expensively" and "upscale" for "expensive," the model is unable to recognize out-of-vocabulary words that have similar meaning. In addition, the user cor-

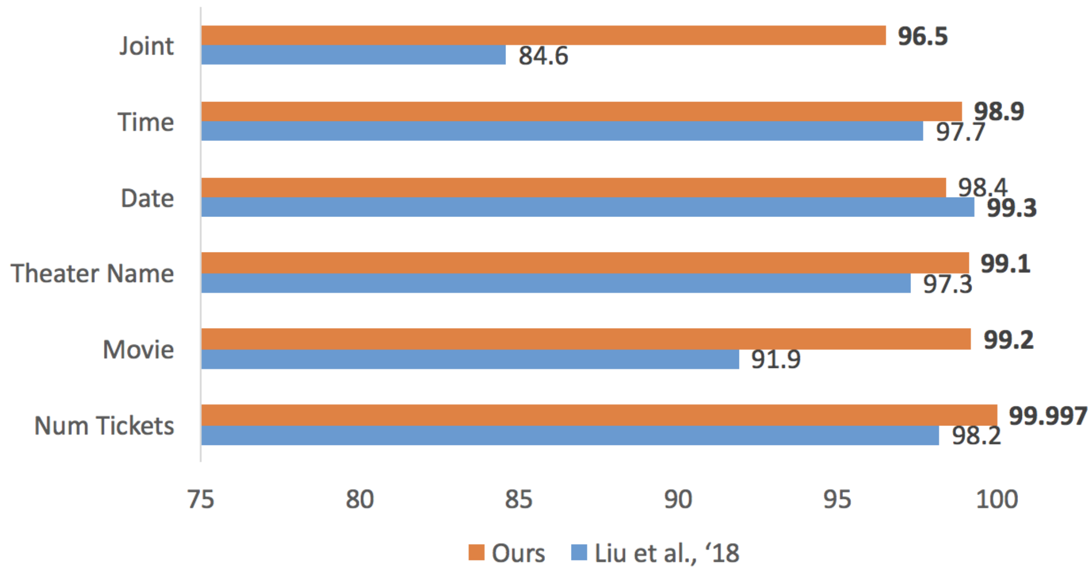


Figure 7-5: Accuracy of our model, as compared to the state-of-the-art (Liu et al., 2018) on slots in the Sim-GEN movies dataset.

rection for the `area` slot in the third example, where the user specifies "anywhere," is challenging since the system asks about the "centre" of town. Finally, understanding that the user wants a "cheap" restaurant if they are "close to broke" requires advanced commonsense reasoning.

Qualitative Analysis

To see whether our model is learning semantically meaningful embeddings after passing the input user utterance through the convolutional layer followed by maxpooling, we used Euclidean distance to identify the top-n nearest neighbor embedded utterances to several utterances selected from the held-out test set (note that we are focusing on the model trained to predict requestable slots for the purposes of this analysis). In Table 7.6, we note that nearest neighbor embeddings are indeed similar in meaning to the query user utterance (e.g., "any" is most similar to the learned vectors for "uh any" and "ah any"), and "chinese

User: Hello, I'm looking for a **nice** restaurant with vegetarian food. **True**: `food = vegetarian` **Pred**: `food = vegetarian`; `price =` **`expensive`**
User: Hi, I want a Tuscan restaurant that's **expensively** priced. **True**: `food = tuscan`; `price =` **`expensive`** **Pred**: `food = vegetarian`; `price =` **`cheap`**
System: No such results found. Would you like me to search for any Mediterranean restaurants in the **centre**? **User:** Is there a Lebanese place **anywhere** around? **True**: `food = lebanese`; `area =` **`dontcare`**; `price = dontcare` **Pred**: `food = lebanese`; `area =` **`centre`**; `price = dontcare`
User: I like Persian but I'm close to **broke**. **True**: `food = persian`; `price =` **`cheap`** **Pred**: `food = persian`
System: I will search for the most nearby English restaurant. **User:** It should be an **upscale** English restaurant. **True**: `food = english`; `price =` **`expensive`** **Pred**: `food = english`

Table 7.5: Examples of incorrect slot-value predictions made by the system due to lexical variation used by Turkers in the WOZ 2.0 dataset, which requires semantic understanding.

food" is most similar to the learned embedding for "um chinese food"), as expected.

To further understand the behavior of our neural network model and illustrate that is interpretable, rather than simply a black box, we also extracted the top-10 tokens that had the highest activations when passed through the learned CNN filters in the requestable slots model. As shown in Table 7.7, some filters appear to be identifying requestable slots (e.g., postcode, post, center), whereas others are focused specifically on finding different types of food (e.g., caribbean, indian, etc.).

User Utterance	Top-3 Nearest Neighbors
phone number	and phone number whats phone number phone number please
any	uh any ah any any range
chinese food	um chinese food what about thai food romanian food

Table 7.6: Top-3 nearest neighbors for three test user utterances, using Euclidean distance on the model's learned embeddings (i.e., after convolving and maxpooling the input).

7.2.3 Summary and Future Work

We have demonstrated that our novel convolutional architecture that directly predicts a user's goal slots during a task-oriented dialogue in the restaurant booking domain, given the user utterance and system response, achieves 86.9% joint goal accuracy and 95.4% requested slots on the WOZ 2.0 test set, without any semantic dictionaries or pre-trained word vectors. In future work, we plan to predict not only the user's goal, but the next system response. We also plan to modify our model so as to handle the noisy ASR test set of DSTC2—this may require tricks such as summing the scores from each ASR hypothesis, applying word dropout, and learning character n-gram embeddings (Zhong et al., 2018). In the next section, we present our novel approach to the 6th Dialogue State Tracking Challenge (DSTC6) track for end-to-end goal-oriented dialogue, in which the goal is to select the best system response from among a list of candidates for restaurant booking.

CNN Filter	Top-10 Tokens
11	caribbean, indian, type, food, bistro, serve, something, thai, singaporean, romanian
13	european, canapes, indian, bistro, japanese, caribbean, world, persian, italian, british
16	postcode, post, center, thank, restaurant, then, i, need, could, uh
19	phone, telephone, does, their, the, is, south, east, i, in
50	code, expensive, type, moderate, serving, kind, any, my, anything, cheap

Table 7.7: Top-10 highest activation tokens for several learned CNN filters, where filters 11 and 13 isolate cuisines, and filters 16, 19, and 50 focus on three types of requestable slots: postcode, phone, and pricerange, respectively.

7.3 6th Dialogue State Tracking Challenge (DSTC6)

In this section, we now turn our attention to the 6th Dialogue State Tracking Challenge (DSTC6) (Boureau et al., 2017) track with a corpus collected by Facebook AI Research for end-to-end goal-oriented dialogue. In this task, the goal is making a restaurant reservation for the user, given all their constraints on the location, cuisine, price range, atmosphere, and party size. This overall task is broken down into five subtasks: 1) issuing API calls, 2) updating API calls, 3) displaying options, 4) providing extra information, and 5) conducting full dialogues (see Fig. 7-6). This requires not only dialogue management, but also querying knowledge bases (KB) of restaurants to respond with the correct information to the user. The motivation for this challenge is to build end-to-end models that have the potential to scale up and generalize to new domains better than existing SDS methods, where the dialogue state is designed to be domain-dependent.

In particular, we focus on the first two tasks in this section, as in (Bai et al., 2017), since tasks three and four can be simply handled by KB lookup. We compare our CNN

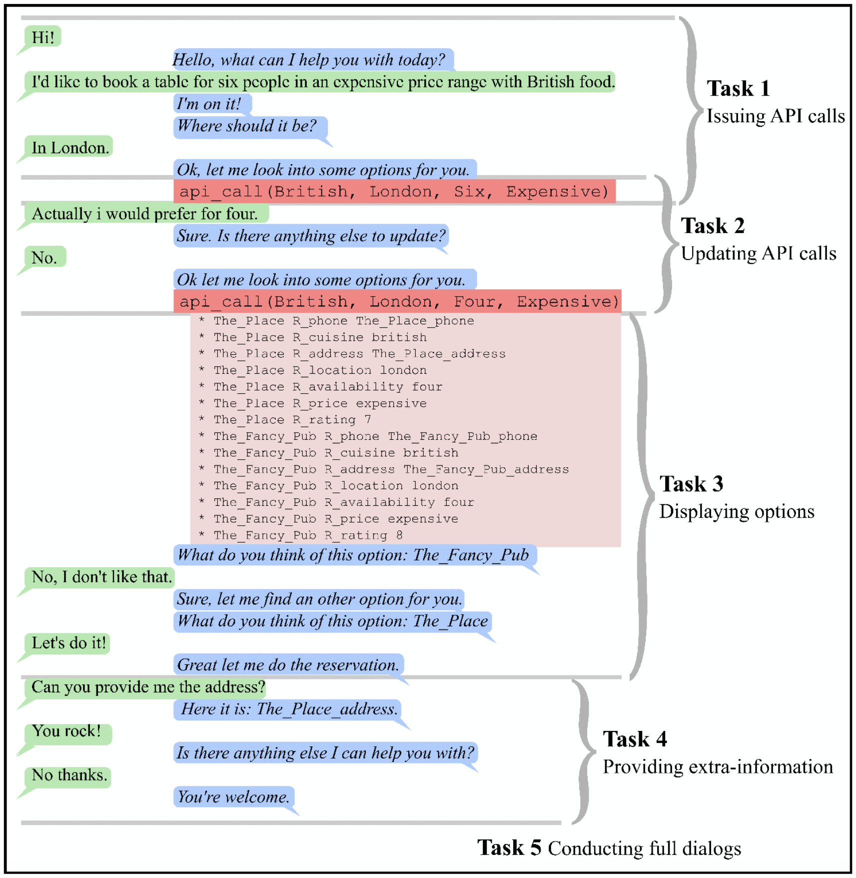


Figure 7-6: An illustration of the five subtasks in the DSTC6 track for end-to-end goal-oriented dialogue (Bordes and Weston, 2016).

approach to the top-2 DSTC6 participants, since they achieve 100% top-1 precision (P@1) on the test set for all subtasks (Korpusik and Glass, 2019b). We demonstrate that our CNN technique is competitive, reaching 100% P@1 on subtasks 1 and 2, without requiring any LSTMs, as were used in both the top submissions. In addition, our approach uses the CNN semantic tagger we developed in Chapter 4, which we establish is *generalizable* to other tasks and domains, such as restaurant booking in DSTC6, without requiring task-specific hyperparameter fine-tuning.

7.3.1 Models

Baselines

We compare against the baselines used by *Bai et al.* (Bai et al., 2017): ranking candidate system responses randomly, according to tf-idf values, with a support vector machine (SVM), with a vanilla LSTM, and with a hierarchical LSTM. We also implement a binary CNN model with a sigmoid output layer (see Fig. 7-13) that predicts whether each candidate is a good system response, given the input user utterance and dialogue history. To get the final ranking, we use the sigmoid output probability for each candidate.

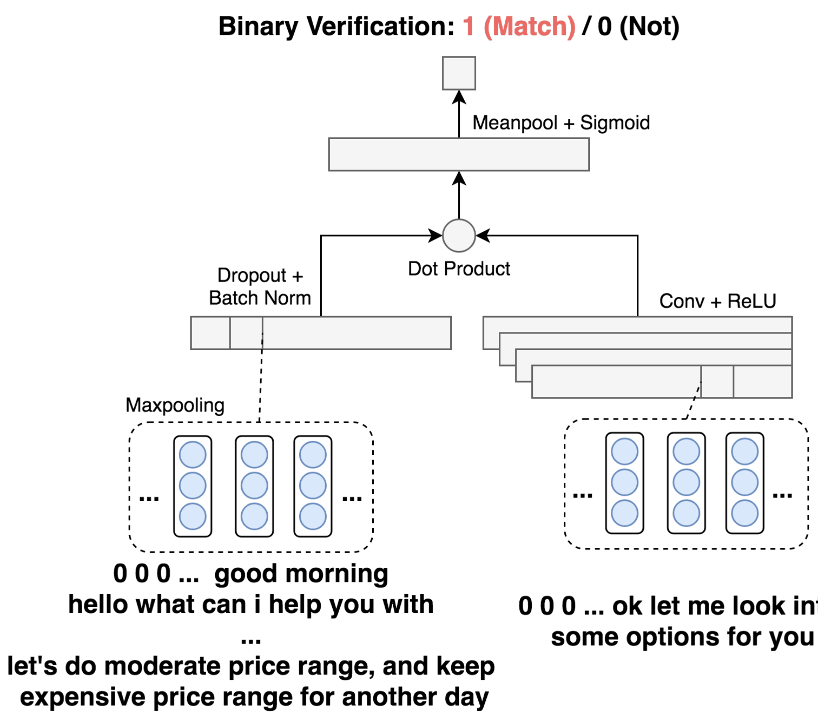


Figure 7-7: The binary CNN baseline predicts whether each candidate system response is the correct answer. There are two inputs to the network: the full dialogue history concatenated with the last user utterance, and a candidate system response on the right. We pad input tokens with zero to the maximum utterance length seen in training, apply 0.1 dropout, and perform batch norm after maxpooling.

CNN Architecture

Our best model pushes us closer to *generating* the next system response, rather than simply selecting the best response from among the list of candidates. Our full system architecture, shown in Fig. 7-8, consists of one CNN for semantic tagging and updating the dialogue state (see Fig. 7-9 for an example utterance with its tags), another CNN for action selection (see Fig. 7-10), and a final response generation step filling in the template with the slot values in the final dialogue state.

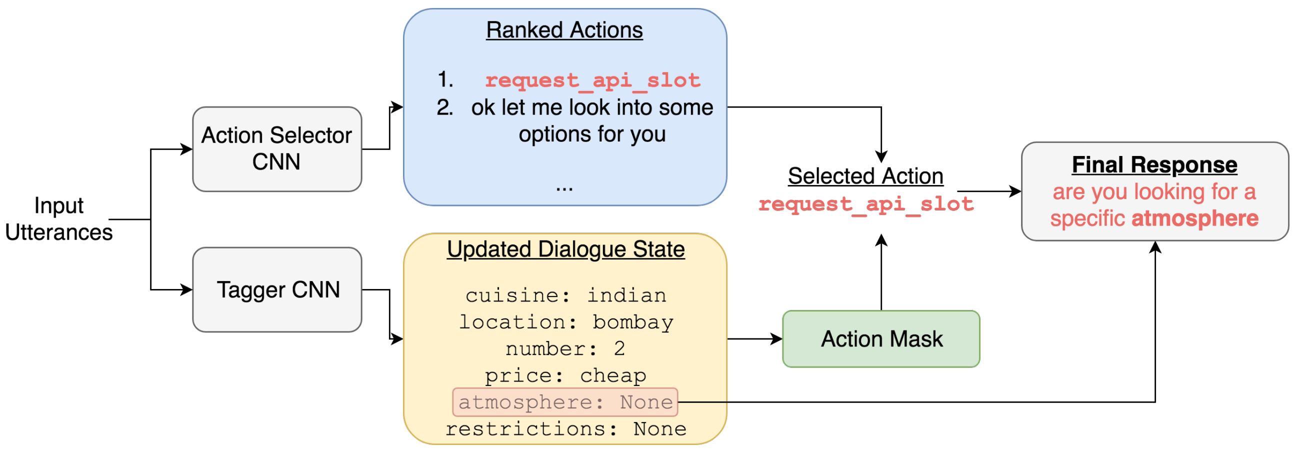


Figure 7-8: The full system architecture, where the action selector CNN ranks the 15 possible system action templates; the tagger CNN updates the dialogue state by tagging each utterance in the dialogue history; and the final response is generated by selecting the top-ranked action, applying an action mask, and populating slots with values. Here, since there are missing slot values, the `api_call` action is masked out, and the next slot missing a value is `atmosphere`, so it is requested by the system as the final response.

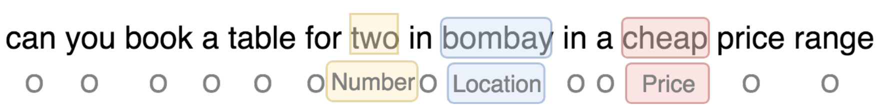


Figure 7-9: A sample user utterance with its color-coded semantic tags (i.e., number in the party, restaurant location, and price range).

Semantic Tagging

As in Chapter 4, where we used CNNs for semantic tagging on meal descriptions for a diet tracking application (Korpusik et al., 2014), here we apply a CNN to semantic tagging of the user utterances in the restaurant booking conversations. We select a CNN here rather than a CRF because we demonstrated it has comparable performance, without requiring any feature engineering. We choose it over the more typical LSTM due to its faster training time.[7] We also illustrate the interpretability aspect of the learned convolutional filters in Table 7.12, by identifying patterns among the tokens that have the highest activation from the filters. As in Chapter 4, the CNN tagger is composed of a word embedding layer followed by three stacked 1D convolutional layers, with kernel windows spanning lengths of five, five, and three tokens, respectively. It learned 150-dimension embeddings without using pre-trained word vectors, used 64 filters per convolutional layer, applied ReLU activation, and trained with the Adam optimizer on cross-entropy loss for up to 15 epochs with early stopping determined by no performance gain on the validation set (20% split). We trained a separate tagger for each of the two subtasks we evaluated in the DSTC6 challenge, since that performed better than jointly training a tagger on both.

To convert the DSTC6 data to training data for semantic tagging, we searched for `api_call` utterances (see Table 7.8) within the list of utterances for each dialogue, since that provided us with the gold standard value for each slot. Each `api_call` has the following order: `cuisine`, `location`, `number`, `price`, `atmosphere`. For example, a possible utterance might be `api_call italian paris four cheap romantic`. We then found exact string matches for each of the slot values among the previous utterances, labeling each utterance with that slot type (e.g., the `cuisine` tag would be assigned to each token `italian`) and the remaining tokens with the other se-

[7]Training the LSTM takes $>$ 5x longer than training the CNN on Task 2.

mantic tag `O`. At test time, we ran each user utterance in the dialogue history for the DSTC6 challenge through the trained CNN tagger, updating the dialogue state each time a new slot was identified, using the tagged token as the value for that slot (e.g., `two` is the value specified for the `number` slot in the example utterance shown in Fig. 7-9).

Action Selection

The second CNN in our system is trained to predict the best candidate system response template from among the 15 possible options in Table 7.8. As shown in Fig. 7-10, this CNN takes only one input (the full dialogue history of utterances concatenated together) and feeds it to a learned word embedding layer of 64 dimensions. The embeddings for each token are fed through a 1D convolutional layer with a window size of three tokens, ReLU activation, and 64 filters. The output of this layer is then maxpooled before passing through a final feed-forward layer with a softmax activation to output a probability for each of the 15 possible candidate system response templates.

Final Response Generation

After the dialogue state is updated by feeding each utterance through the CNN tagger, and the possible system actions are ranked by the CNN action selector, an action mask is applied (as in *Ham et al.* (Ham et al., 2017)), and the action templates are populated with slot values from the dialogue state. The action mask is formed based on the dialogue state—if any slot values are not yet specified, the `api_call` action is masked out, and if all slot values are specified, the `request_api_slot` action is masked out. In the final step, if the `api_call` action is selected, the values are populated using the current dialogue state; likewise, if the `request_api_slot` action is selected, the system response for the next slot that is still missing its value (again according to the current dialogue state) is chosen

Action Template
ok let me look into some options for you
`api_call`
i'm on it
hello what can i help you with today
sure is there anything else to update
you're welcome
what do you think of this option:
great let me do the reservation
sure let me find another option for you
here it is
whenever you're ready
the option was
i am sorry i don't have an answer to that question
is there anything i can help you with
`request_api_slot`

Table 7.8: The 15 possible system action templates. The `api_call` action is populated with the values for each of the slots (i.e., cuisine, location, number, price range, atmosphere) in the current dialogue state, while the `request_api_slot` template is mapped to a response for the next missing slot the system needs to call the API.

(see Table 7.9 for the responses generated for each requested slot).[8]

7.3.2 Experiments

7.3.3 Data

The DSTC6 dialogue data we used is an extension of the bAbI dialogue data in *Bordes and Weston* (Bordes and Weston, 2016), where the dialogues for restaurant reservation are generated through simulation based on a knowledge base (KB) of restaurants. Each restaurant is specified by a cuisine (e.g., French), location (e.g., Tokyo), price range (e.g.,

[8]We use the deterministic order for requesting missing slots.

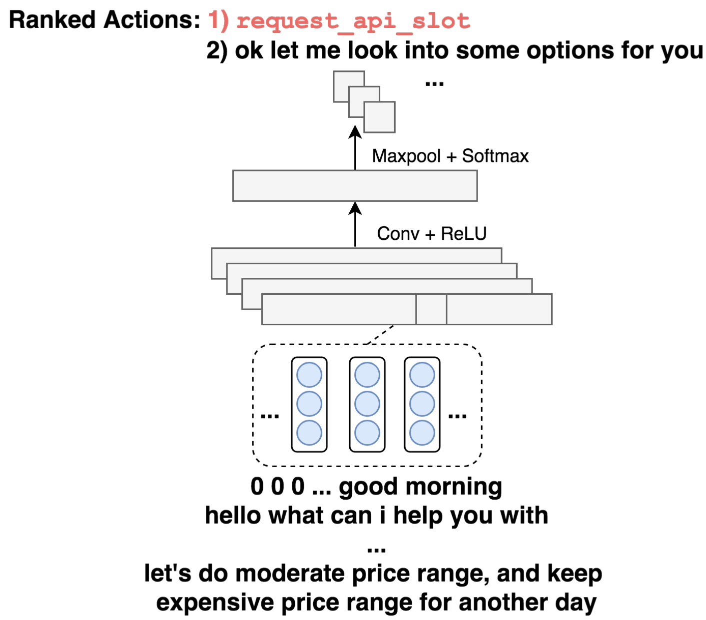


Figure 7-10: The CNN for action selection, which takes as input the full dialogue history of utterances concatenated together, and outputs the probabilities for each possible response.

expensive), atmosphere, and dietary restrictions. Each restaurant has a party size option of 2, 4, 6, or 8, and a phone number and address. There are 10,000 example dialogues in the training set for each subtask, for which we train tagger and action selector CNNs *separately*.

We report precision @ $\{1, 2, 5\}$ as our evaluation metric (i.e., the number of times in the test set that the correct system response candidate is ranked first, in the top-2, and among the top-5 responses, respectively). In our preliminary experiments, we evaluate on the first of four test sets, in which the knowledge base is the same between training and test, and where dietary restrictions are omitted.

Slot	System Response
Cuisine	any preference on a type of cuisine
Location	where should it be
Number	how many people would be in your party
Price	which price range are you looking for
Atmosphere	are you looking for a specific atmosphere

Table 7.9: The system response returned for each of the six possible slots when the `request_api_slot` action template is chosen.

Results

As shown in Table 7.10, our model is competitive with the top participant (Ham et al., 2017) in the DSTC6 challenge, achieving 100% precision on the first test set for subtasks 1 and 2. Our binary CNN model is outperformed by the SVM, vanilla LSTM, and hierarchical LSTM in P@1, though it does better than the SVM in P@2 and outperforms both the SVM and LSTM in P@5. Note that the action mask is a critical piece of our system (due to predicting `api_call` with missing slots, and `request_api_slot` with all slots filled)—without it, P@1 drops from 100% to 91.1% on Task 1 (and remains 100% on Task 2). In Table 7.11, we show the results of the tagger on our automatically generated tagging data for subtask 1. We see examples of tricky user utterances in Fig. 7-11, where the model makes a mistake when there are two possible tokens for the same tag. In Table 7.12, we show the tokens that have the highest activations for the tagger's CNN filters.

7.3.4 Summary and Future Work

In this section, we have demonstrated that the CNN tagger we designed in Chapter 4 for semantic tagging of natural language meal descriptions is general enough to be directly applied to the 6th Dialogue State Tracking Challenge (DSTC6) without requiring task-specific hyperparameter fine-tuning. Our model, which combines the CNN tagger with

	Task 1			Task 2		
Model	P@1	P@2	P@5	P@1	P@2	P@5
Random	10.2	20.4	50.9	0.95	19.5	46.7
TFIDF	21.0	29.9	52.2	36.7	47.4	66.9
SVM	81.3	81.6	83.0	74.5	76.4	78.9
LSTM	84.3	90.6	98.5	77.8	84.0	97.8
Hier. LSTM	88.6	94.1	99.9	81.7	92.6	100
Bai et al.	99.8	100	100	99.7	100	100
Ham et al.	100	100	100	100	100	100
Binary CNN	78.9	88.9	99.7	69.0	79.3	99.6
Our Model	100	100	100	100	100	100

Table 7.10: We report the precision for each of the baseline methods, the top-2 submissions to the DSTC6 challenge (Ham et al., 2017; Bai et al., 2017), our baseline binary CNN, and our final softmax CNN.

a CNN action selector, achieves 100% precision on subtasks 1 and 2 of the end-to-end goal-oriented dialogue track, and is competitive with the top challenge participants.

In future work, we will experiment on the remaining three subtasks (displaying options, providing extra information, and conducting full dialogues), as well as the other three test sets for each subtask. We could also add a feature to our CNN that indicates whether all the slots have been filled or not when predicting the action template, which should allow the network to automatically learn the action mask. Finally, we aim to jointly

Semantic Tag	**Precision**	**Recall**	**F-score**
Cuisine	100	96.9	98.4
Location	100	95.9	97.9
Number	100	100	100
Price	96.9	96.5	96.7
Atmosphere	100	100	100
All	99.8	99.8	99.8

Table 7.11: Precision, recall, and F-scores of our CNN semantic tagger on each of the semantic tags in the automatically generated tagging test set for the first subtask of DSTC6.

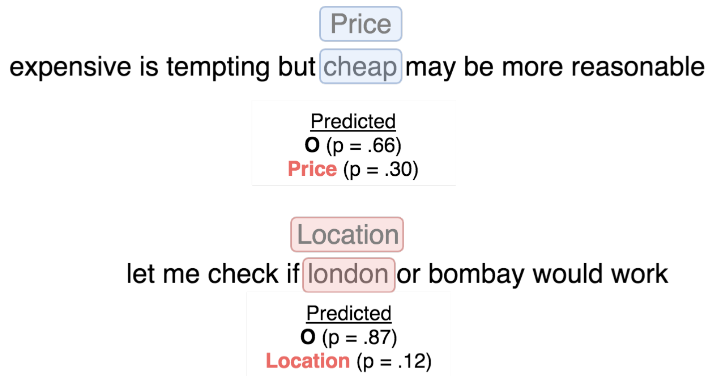


Figure 7-11: Example semantic tagging errors, where the model incorrectly labels both `cheap` and `london` as `O`, rather than correct tags `Price` and `Location`, respectively.

Filter	**Top-3 Highest Activation Tokens**
19	french, spanish, italian
52	two, six, four
63	bombay, london, paris

Table 7.12: Top-3 tokens with high activation for the learned filters in the semantic tagger's third CNN layer—filter 19 picks out cuisines, filter 52 isolates party numbers, and filter 63 identifies locations.

train the tagger and action selector CNNs as one fully end-to-end model. In the next section, we demonstrate that our CNN approach to selecting the next best system response in a spoken dialogue system is also competitive on the recently released 7th Dialogue System Technology Challenge (DSTC7).

7.4 7th Dialogue System Technology Challenge (DSTC7)

In this section, we again focus on the final response generation step of a dialogue system, taking the user's query and dialogue history as the only input, completely bypassing the language understanding and dialogue management components. Therefore, the model is trained fully end-to-end, with only two inputs: the previous two utterances in a dialogue between a student and the advisor, and a candidate utterance for the next system response. Given a set of 100 possible response candidates, the goal is to select the next best response based on the dialogue history (see Table 7.13 for an example partial dialogue). We entered our CNN approach, which does not require any manual feature engineering or semantic dictionaries, into the 2019 DSTC7 challenge, focusing in particular on subtask one of the Advising dataset in track one (Korpusik and Glass, 2019a).[9]

The challenge provided five subtasks: 1) selecting the next response from among 100 candidates, 2) selecting the next response from 120,000 responses, 3) selecting the next response and its paraphrases from among 100 candidates, 4) selecting the next response from among 100 candidates that may not contain the correct response, and 5) selecting the next response from among 100 candidates and incorporating the provided external knowledge. We demonstrate that our convolutional neural encoder placed 11th out of 20 participants on the first subtask, with a *Recall@50* score of 82.4%.

7.4.1 Models

Baseline Dual LSTM Encoder

We compare our model to a strong baseline—the dual LSTM (Hochreiter and Schmidhuber, 1997) encoder (Lowe et al., 2015). The inspiration for this model is the seq2seq (Sutskever et al., 2014) approach often applied to machine translation, where an encoder (usually a recurrent model, such as an LSTM) encodes the input sentence in the source language, and an LSTM decoder is fed the encoded representation to generate the sentence in the target language.

The LSTM encoder works as follows (see Fig. 7-12).[10] Each token in the dialogue his-

[9] https://ibm.github.io/dstc7-noesis/public/index.html
[10] http://www.wildml.com/2016/07/deep-learning-for-chatbots-2-retrieval-based-model-tensorflow/

Advisor: Hello Mingyang! Are you doing well?
Student: Hi advisor. I'm doing alright. I would like some advice on which courses to take next semester.
Student: My interested area is Software Development and Intelligent system.
Advisor: you have three choices namely, EECS481 Software Engineering, EECS492 Introduction to Artificial Intelligence, and EECS 381 Object Oriented and Advanced Programming.
Student: how many difficulty levels do these classes have?
Advisor: EECS381 is not easy
Advisor: any thoughts about that?
Student: What time does the course occur? I like afternoon classes and will find something else if it's scheduled too early.
Advisor: EECS351 is after lunch. The others are before. EECS481 is from nine to ten thirty and EECS492 is from ten thirty to twelve.

Table 7.13: Example DSTC7 dialogue snippet, between a student and the advisor. 100 candidate responses are provided, and the system must choose the correct response, given the conversation history: "*481 is the early morning and is quite similar to EECS381, so you might want to skip it.*"

tory is fed through an embedding layer (initialized with Glove (Pennington et al., 2014)), followed by a recurrent layer, to yield an encoded vector representation of the context, D_e. Likewise, each token in a candidate response is fed through an embedding layer and a recurrent layer, generating the encoded representation C_e. This is done for each of the 100 candidate responses, and the similarity of each candidate response with the dialogue context is computed using a learned similarity matrix M, via the matrix multiplication $D_e M C_e^i$, where i refers to the index of the candidate response. Each of these similarity scores is fed through a final softmax layer to generate probabilities of each candidate response. The whole model is trained with cross-entropy loss.

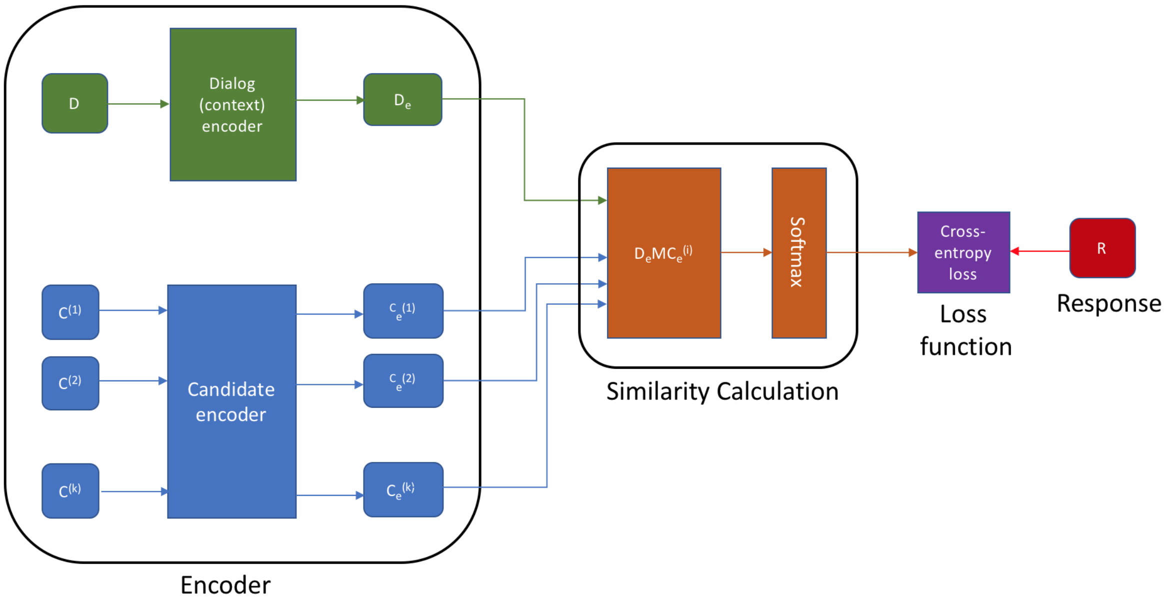


Figure 7-12: The strong baseline dual encoder LSTM network for predicting the next best system response.

Convolutional Encoder

Our approach is similar to that of the dual LSTM encoder, but with the differences that: (i) we use a CNN instead of the LSTM, (ii) we only feed the last two utterances into the context encoder rather than the full dialogue history, which would likely require attention over all the previous utterances in order to ensure the most important information from the most recent utterances is not lost among the full dialogue history, as shown in related work on multi-turn dialogue (Wu et al., 2016; Zhou et al., 2018), and (iii) we compare against each candidate response one at a time with a sigmoid layer instead of a softmax over all candidates. Again, our motivation for using convolutional rather than recurrent models is they train faster, and are more easily interpretable by inspecting which tokens for each learned filter have the highest activation.

As shown in Fig. 7-13, the model is composed of two inputs: one input layer for the

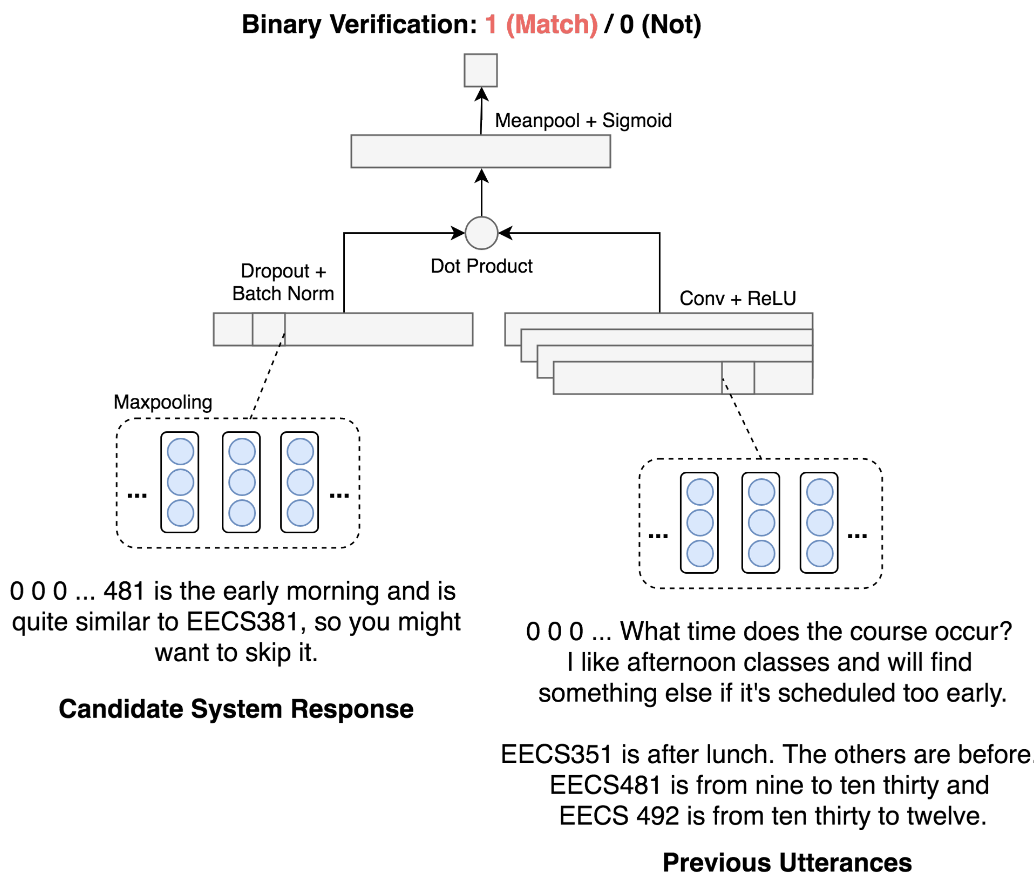


Figure 7-13: The CNN architecture for predicting whether a given candidate system response is the correct match for the previous two utterances in the dialogue between a student and their advisor.

previous two utterances in the conversation, concatenated together, and another input layer for a candidate advisor response. Each of the two inputs is first tokenized using spaCy,[11] lowercased, and padded with zeros to a fixed length of 50 tokens. Each candidate system response is fed through a shared word embedding layer (note that we do not use pre-trained word vectors in the system we submitted to the challenge, although we have since compared to pre-training with Glove (Pennington et al., 2014) and word2vec (Mikolov

[11]https://spacy.io

et al., 2013b) on the validation set), and is max-pooled to generate a single 256-dimension vector representation of the dialogue context. At the same time, the dialogue history is fed through the 64-dimension shared word embedding layer and a 1-dimension convolutional layer of 256 filters spanning a window of three tokens with a ReLU activation. During training, this is followed by a dropout of probability 0.1, and batch normalization (Ioffe and Szegedy, 2015). Following the input encoding step, a dot product is performed with the candidate response vector and each 256-dimension CNN output of the dialogue history. Mean-pooling is then performed across these dot products to produce a single scalar value, which we force to be between zero and one with a final sigmoid layer. We train the model with binary cross-entropy and the Adam optimizer (Kingma and Ba, 2014).

Convolutional Ensemble

In the convolutional encoder described above, the final ranking of the candidate responses is generated by computing the sigmoid output probability for each response. We experimented with ensembling several randomly initialized convolutional models, and found that an ensemble of seven models (six with 256-dimension embeddings and learned CNN filters, and one with 128 dimensions instead), performed best, while the best individual model was the CNN with 256 dimensions. We also found that ensembling by summing the ranked indices predicted by individual models performed better on the development set than averaging the predicted probability scores for each candidate response.

7.4.2 Experiments

In the 7th Dialogue System Technology Challenge (DSTC7) (Yoshino et al., 2018), the goal is to predict the next utterance in the dialogue, given the previous utterances in the dialogue history. There are two corpora—the Ubuntu corpus (Kummerfeld et al., 2018), which is based on chat logs from the Ubuntu channels, and Advising data that simulates a discussion between a student and an academic advisor. We focus on the Advising data, where the purpose of the dialogue is to guide the student to pick the courses that best fit their curriculum, as well as personal preferences about time, difficulty, and area of interest. These conversations were collected by asking students at the University of Michigan to play the role of both the student and the advisor, using provided personas. The statistics of the data for the first subtask are shown in Table 7.14.

	min	max	mean	median
History length	1	41	9.2	8
Utterance length	1	384	10.3	9
Candidate answer length	1	384	12.4	10

Table 7.14: The Advising dataset's statistics for subtask 1.

For our experiments, we focus on the first subtask, in which there are 100 candidate responses for each dialogue snippet, where only one is the correct match, and the others are distractors. We evaluate performance with the *Recall@n* metric, where *R@n* indicates how often the model ranked the correct response among the top-n. The second metric we use for evaluation is the mean reciprocal rank (MRR), which is the average of the reciprocal ranks (i.e., the multiplicative inverse of the rank of the first correct answer) of results for a sample of candidate responses Q:

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i} \tag{7.3}$$

To select the best model, we evaluated individual CNNs, as well as ensembles, on the validation set for Advising subtask 1 (see Table 7.15), and compared against the dual LSTM encoder baseline. In Table 7.16, we show the Recall and MRR scores of our system on the two held-out test sets for the Advising subtask 1. Our ensemble of seven convolutional encoders outperforms the dual LSTM encoder baseline, placing us in 11th place for this subtask, and ranked 13 out of 20 total participants in the first track of the challenge.

Model	R@1	R@10	R@50
Dual LSTM Encoder	6.20	36.0	80.0
Single CNN (meanpool)	3.43	27.2	97.9
Single CNN (maxpool)	10.9	46.3	97.2
2-CNN Ensemble	11.1	48.0	97.2
3-CNN Ensemble	11.8	47.5	97.4
4-CNN Ensemble	12.0	46.7	97.0
5-CNN Ensemble	12.2	46.3	97.0
6-CNN Ensemble	12.4	46.9	97.0
7-CNN Ensemble	12.6	46.9	97.0
8-CNN Ensemble	12.4	46.9	97.0
Single CNN (Glove)	12.2	**50.3**	**98.3**
Single CNN (word2vec)	**14.8**	47.5	97.0
Single CNN (FastText)	11.4	44.1	96.2

Table 7.15: We report the recall for several methods on the validation dataset for Advising subtask 1. Optimizing for R@1, we select the 7-CNN ensemble for the final evaluation (since at the time of submission, we were not using pre-trained word embeddings). With more than 7 CNNs, performance starts dropping. Note that with mean-pooling instead of max-pooling over the candidate response, recall is lower.

Model	**Data**	R@1	R@10	R@50	MRR
CNN Ensemble	Test 1	20.6	54.8	82.4	32.3
CNN Ensemble	Test 2	8.8	32.0	72.8	16.9

Table 7.16: We report recall and mean reciprocal rank (MRR) for our CNN on the two test sets for Advising subtask 1.

7.4.3 Analysis

Since one common critique of neural network models is that they are mysterious "black boxes," here we yet again analyze the learned CNN filters in order to make the model's behavior more interpretable. Specifically, we identify which tokens in the development set yield the highest activation for each of the learned CNN filters, and manually inspect these top-10 highest activation tokens to determine whether there is an intuitive pattern. In Table 7.17, we can see that filter 1 seems to identify greetings (e.g., "hello" and "hi"), filter 11 picks out tokens where the student thanks the advisor and ends the dialogue (e.g., "thanks" and "goodbye"), filter 144 isolates tokens related to course names (e.g., "operating system" and "eecs"), and filters 185 and 209 seem to identify tokens related to personal preferences such as time and workload (e.g., "difficult" and "morning").

CNN Filter	Top-10 Tokens
1	'hello', 'for', 'hi', 'today', '?', 'afternoon', 'full', 'doing', 'one', 'in'
11	'thankful', 'for', 'goodbye', ',', 'thanks', '!', 'thank', '.', 'bye', 'will'
144	'system', 'operating', '482', 'heard', 'really', 'any', 'calc', 'eecs', 'last', 'have'
185	'programming', 'difficult', 'light', 'course', 'in', 'workload', 'take', 'load', 'junior', 'computing'
209	'morning', 'light', 'class', 'in', 'relatively', 'semester', 'prefer', 'a', 'like', 'which'

Table 7.17: Top-10 activated tokens for learned CNN filters.

In addition, we inspect the errors made by our best system (i.e., the ensemble of CNNs) on the validation set to determine 1) whether the mistakes seem reasonable, and 2) to come up with ideas for improving performance. In Table 7.18, we see that the system is confused by out-of-vocabulary words (i.e., <UNK>) that were unseen during training. One approach for handling this better in future work is to use letter trigrams or character-based embeddings, rather than full word embeddings. We also note that in the second example, the student thanks the advisor, and the predicted responses all seem reasonable (e.g., *"you're welcome"*). In the third example mistake, where the system is unable to compare two courses, the limitation is that our system does not use the full conversation history, but only the previous two utterances. In this scenario, it would help the system to know that "they" refers to courses EECS370 and EECS280. The final mistake requires a deeper semantic understanding than our system is capable of currently—the correct response requires commonsense reasoning that by registering for the class, the student can observe firsthand the professor's style, and the predicted responses make sense, but illustrate that the system does not realize "it" refers to courses rather than the professor (e.g., *"it has a high degree of easiness"*).

7.4.4 Summary and Future Work

In this section, we have demonstrated that a CNN encoder outperforms a strong dual LSTM encoder baseline for selecting the correct response from a list of possible candidates. We report a *Recall@50* score of 82.4% on the first held-out test set for the Advising subtask 1 of the first track of the 7th Dialogue System Technology Challenge (DSTC7). In future work, we will explore methods of incorporating more contextual information and leveraging the full dialogue history and provided knowledge base of course information. In order to handle unknown words at test time, we will investigate character embeddings and delexicalization. We will also apply our models to the other subtasks and Ubuntu data.

Student: does eecs370 have <UNK> ? **Correct**: nope , no labs in eecs 370 . **Top-4 Predicted**: 1) no <UNK> . have a good one ! 2) it starts at 3:00 and ends at 4:30pm. 3) i would suggest taking eecs 280 4) how much challenge do you want ?
Student: many thanks , and i hope to never see you again . **Correct**: my wish for you is that you graduate quickly . **Top-4 Predicted**: 1) you ’re welcome . 2) i believe that is a good decision . i wish you luck . 3) what time of day do you need ? 4) do you need anything else ?
Student: do they pair well together ? **Correct**: it depends on your schedule . eecs 370 is a easy class whereas , according to statistics <UNK> from past year students , eecs 281 is not an easy class , so i recommend <UNK> it with an easier class . **Top-4 Predicted**: 1) it depends 2) the students do not like eecs376 . 3) i suggest strongly you take eecs 280 4) you have met all the requirements , so it is possible .
Student: as a professor how is he ? **Correct**: if you want , you can see for yourself by registering for the class . **Top-4 Predicted**: 1) well , that depends 2) 370 has a score of <UNK> for being relatively <UNK> . 3) it has a high degree of easiness , helpfulness , and clarity 4) it ’s not a very heavy load

Table 7.18: Examples of incorrect top-4 response predictions.

Chapter 8

Conclusion

AI has the potential for a positive impact on society through healthcare. For example, by creating a dialogue system that is able to carry on a conversation, listen, and empathize with us, rather than sounding stiff and robotic like many existing systems (illustrating the difficulty of the problem), a conversational agent can act as a therapist to alleviate symptoms of anxiety and depression,[1] or as a companion to elderly patients with dementia who need someone with whom to socialize, and for whom speaking is easier than typing. Bickmore's group at Northeastern has already developed a collection of relational agents that help with tasks such as educating patients on anethesia or hospital discharge instructions, adhering to medication for Schizophrenia patients, promoting healthy eating and exercise for weight loss, and explaining exhibits at science museums.[2]

Dialogue systems research has a long history involving the work of many prominent researchers, from early rule-based systems to statistical models and reinforcement learning, and now the current deep learning paradigm using end-to-end neural architectures. However, despite the substantial body of work devoted to dialogue systems, this is not yet

[1] https://woebot.io/
[2] https://www2.ccs.neu.edu/research/rag/

a solved problem, though ripe for a solution. Companies often still use heavily rule-based systems, and responses are generated either by selecting from a set of pre-defined candidates or filling values in hand-crafted response templates. Such systems tend to sound unnatural, like a machine, and often cannot remember what was previously said or lack basic commonsense knowledge. Despite the recent trend toward end-to-end learning, the standard real-world system is still composed of several pieces: speech recognition to convert the user's speech to text, spoken language understanding to identify slots and values (e.g., `movie_name=Jurassic Park`), dialogue state tracking, dialogue management to select the next system action, and response generation. Furthermore, while one system may work well in a particular domain, such as flight booking, it often does not generalize to other domains, and a whole new system must be constructed for a new domain. Even more importantly, task-oriented dialogue systems cannot chat with users for fun, and chatbots are not yet able to carry on an extended conversation *naturally* with a human.

In this thesis, we designed a novel, weakly supervised NN model and demonstrated that it successfully performs a semantic mapping from human natural language to a structured relational database. In the process, it learns semantically meaningful vector representations, such that entities with similar meanings generate embeddings that lie close together in vector space. We deployed this technology to the nutrition domain, which is motivated by the rising obesity rate in the United States (from 13% among US adults in the 1960s to 39.8% in 2015-2016, leading to an estimated medical cost of $147 billion in 2008[3]). Also, the tedious, time-consuming nature of many existing diet tracking applications causes people to give up after only a few days.[4] We generalized the technology to booking movies, flights, and restaurants (Korpusik and Glass, 2018b)), showing its applicability to any domain in which a query is mapped to a database to respond to the user.

[3]https://www.cdc.gov/obesity/data/adult.html

[4]https://www.wsj.com/articles/new-reasons-why-you-should-keep-a-food-journal-1463419285

One problem with conventional dialogue systems is they often rely heavily on manual feature engineering and a set of heuristics for mapping from user queries to database entries. There is a fundamental mismatch between how people describe objects, and how the matching entities are represented in a structured database. For example, a person may say they had a slice of "toast," rather than a piece of bread. However, the food database may only contain "bread, toasted," which is not an exact word match, as well as incorrect matches, such as French toast and toasted nuts. Historically, researchers dealt with this mismatch through text normalization (e.g., removing punctuation, lowercasing). To avoid this pipeline of text regularization and word matching lookup, we designed a model that is fed raw input and handles the text mismatch internally. Rather than requiring manually defined semantic dictionaries of synonyms, or checking for plurals and misspellings, the model automatically learns to identify segments of text that are semantically similar.

We designed a novel CNN and empirically established that it learns vector representations of natural language queries that lie close to embeddings of database entries with semantically similar meanings (Korpusik et al., 2017a; Korpusik and Glass, 2018a; Korpusik et al., 2017b). We trained a CNN on a binary verification task, which was fed two inputs—a USDA food name, and a natural language meal description. It predicted whether the input USDA food was mentioned in the meal. The network was trained on *weakly annotated data, without any pre-trained word embeddings or semantic tags*, and was not told where the food items were located in the meal. Rather, the network had to *automatically infer* which words in the human-described meal corresponded to each USDA food, since the meals contained at least three foods, as well as non-food words. By successfully solving this task, the network learned meaningful vector representations of USDA food items such that foods from the same category lay nearby in vector space. At test time, we ranked all possible USDA food items by feeding a user's meal log through the CNN to generate a vector, and scored its similarity with each USDA food's embedded vector representation, yielding 64% top-5 food recall and outperforming a rule-based database matching approach, as is commonly used in dialogue systems.

We demonstrated that asking followup clarification questions narrows down the search space of database matches, boosting top-5 food recall to 89%. We established that a *deep reinforcement learning* (RL) agent is more natural for humans to interact with than rule-based or entropy-based solutions when asking followup questions about food attributes, since RL uses a reward function to balance asking few questions with achieving high recall. In future work, we will continue exploring RL to learn dialogue policies online from interacting with users, and apply end-to-end deep learning models to dialogue systems. We will also investigate interpretable *memory* modules with *attention* over previous context.

Finally, to make the system as interactive and helpful as possible, like a personal digital nutritionist, we are investigating nutrition question answering and personalized food recommendation. For example, the system should be able to respond if the user asks a question such as "How many calories are in a cup of milk?" or "How many grams of fat have we eaten today?" and recommend foods based on which nutrients the user is missing and what kind of food they tend to eat (e.g., if the user is low in iron, the system may recommend spinach if they like green vegetables, or steak if they're a meat-eater). Similar to how in our previous work, we fed the history of a user's tweets through an LSTM to estimate a user's purchase behavior (Korpusik et al., 2016b; Sakaki et al., 2016), we could feed previously eaten meals through an LSTM to estimate which foods they are most likely to enjoy, while filtering for healthy options.

8.1 Ongoing Work

We are currently implementing a fully functional system prototype for iOS, with the semantic tagging and database mapping models embedded in a server that the client application calls. Our motivation is three-fold: 1) we enable actual users who want to track their diet to use our spoken food logging system, 2) we gather feedback and evaluate performance on real-world users, and 3) we collect additional data that better matches the test data, since it is organic and taken from people using the application in the wild.

To evaluate the system, we used three different studies: 1) computing the correlation between food logged with the system and gold standard 24-hour dietary recall with expert nutritionists, 2) food and quantity recall from workers interacting with the top-n ranked matches on Amazon Mechanical Turk, and 3) statistics on system usage from users interacting with the live application after launch to the Apple Store.

8.1.1 Tufts Pilot Study

In collaboration with nutritionists at Tufts University, we conducted a pilot study with the objective to evaluate the accuracy of our application for self-monitoring dietary intake using natural spoken language. We recruited 14 participants total, who were instructed to record daily dietary intake using our application for a period of at least five days. Two 24-hour dietary recalls were conducted between day 3 and day 5 as the gold standard for evaluating total energy intake (TEI; measured in kcal). The two-day energy estimates were averaged for each assessment method. Estimates of TEI from our system were compared to estimates obtained from the 24-hour dietary recalls by a paired samples t-test. Pearson's correlation coefficient was used to assess the validity of the system.

Participants were an average of 23.2 ± 2.60 (SD; standard deviation) years of age, 92.3% female, and 14.3% were actively trying to lose weight. The results of the patient satisfaction survey administered at the end of study indicated that, on a scale from 1-5 (where 1 is best, and 5 worst), our system was rated 1.86 ± 0.53 for perceived accuracy of nutrition facts, 1.79 ± 0.70 for personalization, and 1.43 ± 0.65 for appealing interface design. Compared to existing food logging applications, on a scale of 1-3 (where 1 is better, 2 the same, and 3 worse), our application was rated better or the same as existing methods on 8 out of 9 questions, with a score of 1.67 ± 1.16 for difficulty, 1.33 ± 0.58 for personalization, and 1.67 ± 0.58 for how fun it was. 14.3% of participants said they would definitely continue using our system, and 28.6% said they probably would.

On average, participants consumed three meals per day and recorded dietary intake for six days using our application. The average TEI was 1782 ± 773 kcal for all self-recorded days (range: 4 to 10 days). The mean TEI measured by 24-hour dietary recall was 1791 ± 862 kcal. The mean TEI measured by our system was 1818 ± 916 kcal for days corresponding to the 24-hour recall. We observed a significant correlation between the two assessment methods ($r = 0.58$; $P = 0.03$), and there was no significant difference in TEI estimates obtained from our application compared to the 24-hour recall ($P = 0.90$). This indicates the potential validity of our novel approach to capture dietary data.

8.1.2 AMT System-in-the-Loop Evaluation

In addition to the Tufts pilot study, we also created an AMT task to collect more data (as shown in Fig. 8-1), as well as to evaluate the system's performance in terms of food and quantity recall over time. We used the Google recognizer in the browser, which provides us with transcripts in addition to recordings that we can use to train our own in-house speech recognizer. In total, we collected 152,981 spoken food logs. We believe that by combining this speech data with the large number of written meal descriptions we have already collected, we can leverage a powerful language model specific to the nutrition domain to guide our ASR.

In addition, since we ask workers to select the best matching foods and quantities from among the predicted top-n ranked results retrieved by our server, we have an additional

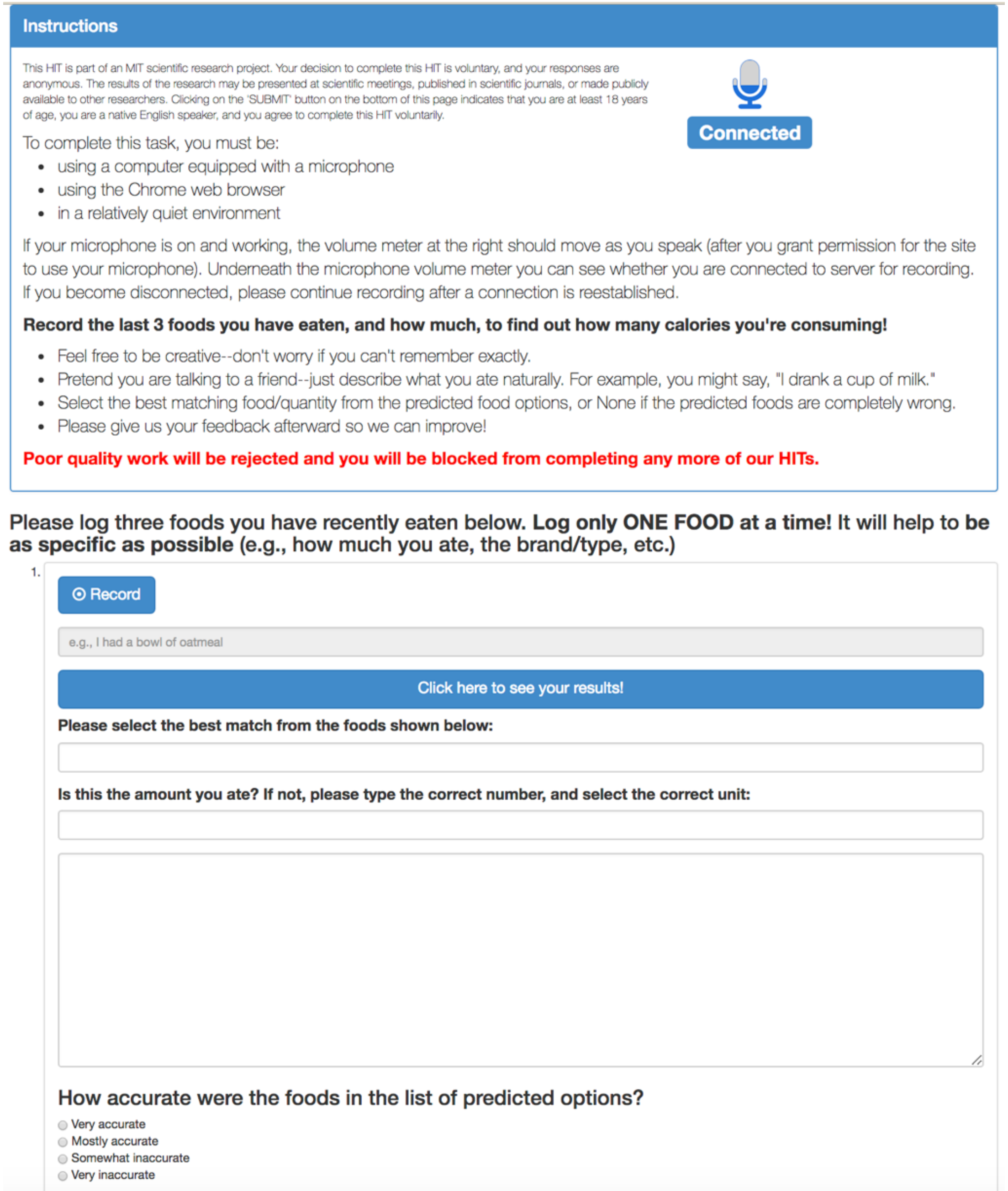


Figure 8-1: The AMT task for collecting more data and evaluating the system. Workers were instructed to record three foods verbally, including quantities, and to select the correct food and quantity matches. We asked workers for perceived accuracy and ease-of-use, feedback, and any questions they wish to ask Coco.

source of data with which to re-train our database mapping model. We launched the AMT tasks in batches of 1,000 over a period of several months, allowing us to track the trend for food and quantity recall over time. As shown in Fig. 8-2, we observe that recall improves over time because the ranked hits are sorted based on how often the food match was eaten, so as more workers interact with the system and select the correct food matches from the ranked results, the system learns from this data and improves over time. We also note that top-15 food recall plateaus at 100% recall, enabling us to reduce our top-n from 30 to only 15 matching foods. We also see that top-5 quantity recall is consistently high, since often individual foods do not have more than 5 unit options to choose from.

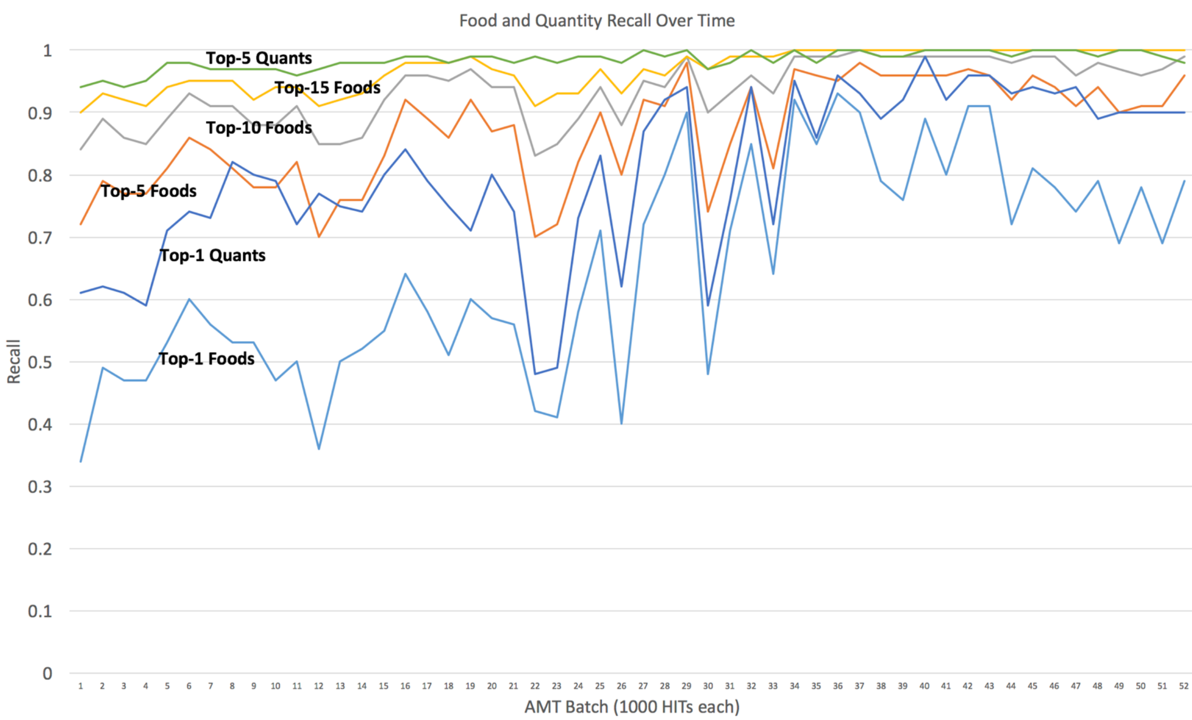


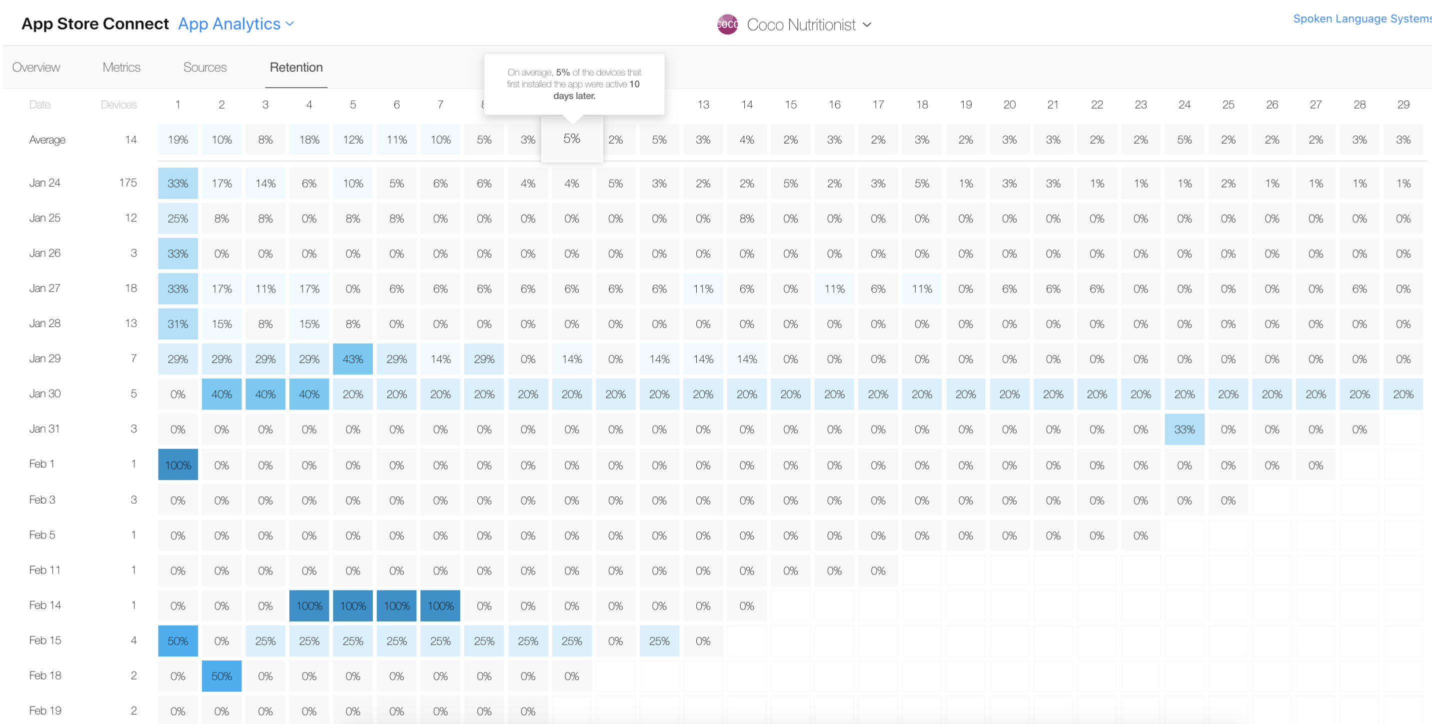
Figure 8-2: Food and quantity recall over time (i.e., per AMT batch of 1,000 HITs).

8.1.3 Apple Store Launch

The final method we used for collecting real-world data and evaluating our system was launching our *Coco Nutritionist* iOS application to the Apple Store. See Appendix A for full sample interactions with the system. As users interact with Coco, they have the option to select a better matching food or quantity than the top-1 predicted match. This gives us more training data for database mapping, whereby we can assume whatever the user selects is the correct match, and that if they leave the prediction alone, then we can assume the model's prediction was correct. If they delete a food, we do not know where the tagger or the food match was incorrect, so we need a mechanism for feedback on semantic tags. We store the audio for the spoken meals, along with the transcripts generated by Siri, for additional ASR training data. Finally, the user has the option to take a picture of their meal, which we can leverage in the future for predicting foods from images.

The Apple Store provides us with analytics on Coco's usage. For example, Fig. 8-3 is the retention plot, showing the percentage of users still using Coco after a certain number of days past the time they first downloaded the app. Note that the top row is the average, which indicates that 19% of users were active one day after download, 5% 10 days later, and 3% after 29 days. In addition, we can measure the number of days that each user logged meals with Coco (i.e., the *streak* length), which is shown in Fig. 8-4. As we can see, although most users only logged meals for one day at a time, there were four users within a month of download who logged their meals for 20 or more days in a row.

Finally, in order to evaluate Coco, we computed the top-1 food and quantity recall, in which we assumed the food selected by the user was the correct match (or close enough to the true answer in their mind). One month after launch to the Apple Store, *Coco Nutritionist* has been downloaded onto 980 iOS devices, with top-1 and top-5 food recall scores of 92.3% and 97.1%, respectively. The quantities and amounts are still using basic



Date	Devices	1	2	3	4	5	6	7	8					13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
Average	14	19%	10%	8%	18%	12%	11%	10%	5%	3%	5%	2%	5%	3%	4%	2%	3%	2%	3%	2%	3%	3%	2%	2%	5%	2%	2%	2%	3%	3%
Jan 24	175	33%	17%	14%	6%	10%	5%	6%	6%	4%	4%	5%	3%	2%	2%	5%	2%	3%	5%	1%	3%	3%	1%	1%	1%	2%	1%	1%	1%	1%
Jan 25	12	25%	8%	8%	0%	8%	8%	0%	0%	0%	0%	0%	0%	0%	8%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
Jan 26	3	33%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
Jan 27	18	33%	17%	11%	17%	0%	6%	6%	6%	6%	6%	6%	6%	11%	6%	0%	11%	6%	11%	0%	6%	6%	6%	0%	0%	0%	0%	0%	6%	0%
Jan 28	13	31%	15%	8%	15%	8%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
Jan 29	7	29%	29%	29%	29%	43%	29%	14%	29%	0%	14%	0%	14%	14%	14%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
Jan 30	5	0%	40%	40%	40%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%	20%
Jan 31	3	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	33%	0%	0%	0%	0%	
Feb 1	1	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%		
Feb 3	3	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%				
Feb 5	1	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%						
Feb 11	1	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%												
Feb 14	1	0%	0%	0%	100%	100%	100%	100%	0%	0%	0%	0%	0%	0%	0%															
Feb 15	4	50%	0%	25%	25%	25%	25%	25%	25%	25%	25%	0%	25%	0%																
Feb 18	2	0%	50%	0%	0%	0%	0%	0%	0%	0%	0%																			
Feb 19	2	0%	0%	0%	0%	0%	0%	0%	0%	0%																				

Figure 8-3: The percentage of users still logging with Coco per day after initial download.

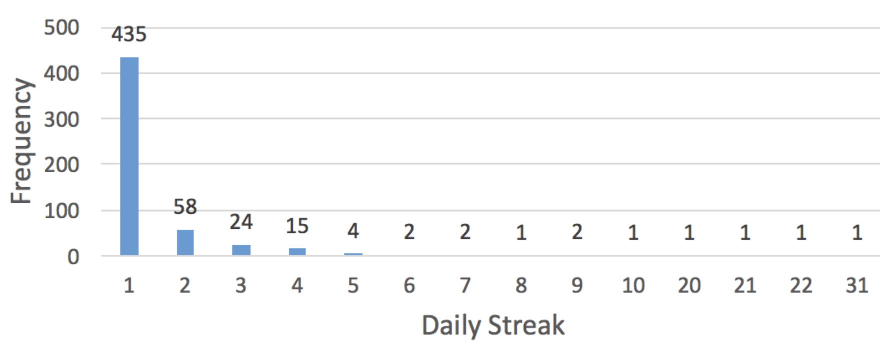


Figure 8-4: A histogram of all users' daily streaks. As expected, the majority of streaks are only one day long, but given how diet tracking is such a tedious process, it is still noteworthy that 10.2% of the streaks are three or more consecutive days, 20.8% are for two or more days, and four users logged their meals for 20 or more days in a row!

string matching algorithms, which we note both have lower recall than the food prediction with the learned embeddings approach (i.e., 85.8% top-1 quantity recall and only 80.3% number amount prediction, versus 92.3% top-1 food recall).

Number Impressions	121,056
Number Product Page Views	3,180
Conversion Rate	0.97%
Number App Units	980
Number Users	582
Number Spoken Recordings	1,895
Daily Average Sessions per Active Device	2.78
Top-1 Food Recall	92.3%
Top-5 Food Recall	97.1%
Top-1 Quantity Recall	85.8%
Amount Prediction Accuracy	80.3%

Table 8.1: Statistics on Coco app downloads and usage between launch date (1/24/19) and just over one month later (3/1/19), as well as food and quantity recall. Conversion rate is calculated as the number of app units divided by the number of unique device impressions. Note that in order to compute recall, we make the assumption that whatever food or quantity the user selects is the correct match (or a good enough estimate for them).

We also analyze the difference between spoken and typed food logs. Around 30% of the logs are spoken, and in Fig. 8-5 we see that the spoken logs have a longer tail, which indicates that speech is lengthier than writing. While most written logs tend to be only one, two, or three words long, the spoken logs' probability mass is more spread out and highest for four words long. This makes sense since people tend to be more descriptive and wordy when speaking because it is easier and more natural than typing. We can also check the food density for typed or spoken logs, as shown in Fig. 8-6, which indicates a similar pattern to that of the number of words per log. There seems to be a higher fraction of spoken logs with more than one logged food item, compared to typed logs (but for both, the majority of logs only contain one food item, compared to AMT meal descriptions that contain at least three foods). In addition, the spoken food logs contain fewer capitalized and punctuation characters than written (i.e., 3.95% of spoken characters are uppercase, compared to 6.47% of typed characters, and only 0.37% of spoken characters are punctuation, whereas 1.42% of typed characters are punctuation), which is reasonable since written logs tend to be more formal, and the ASR often omits punctuation.

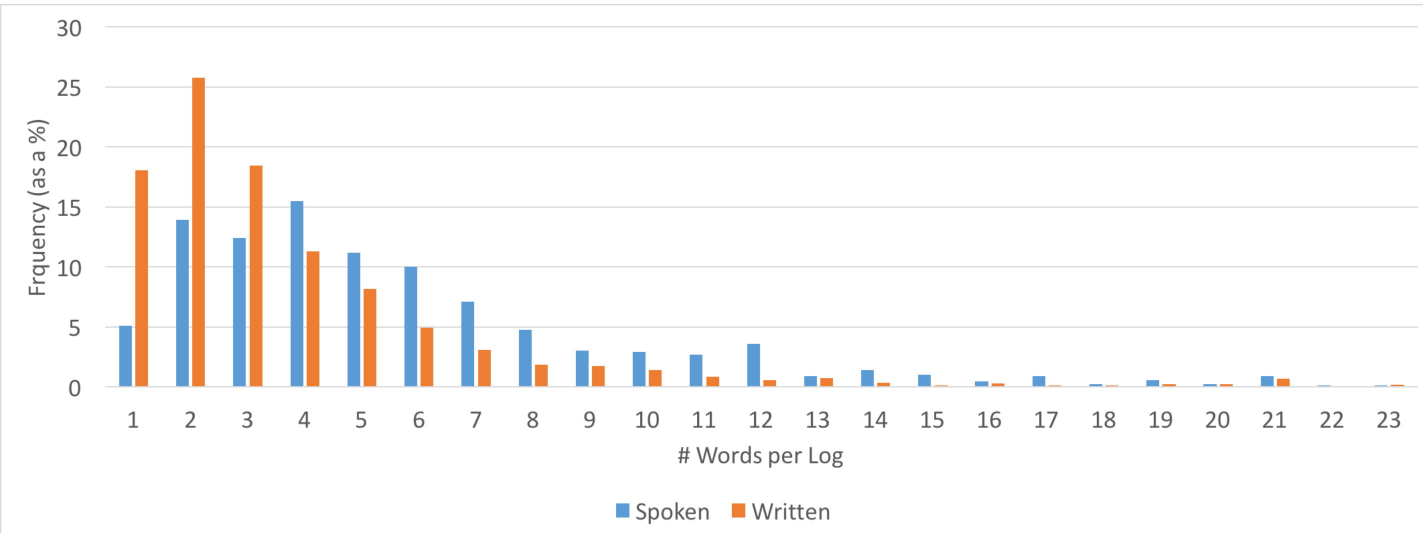


Figure 8-5: A histogram of the number of words per spoken or typed food log.

Finally, we can compute the perplexity of the written and spoken food logs, as well as the AMT meal descriptions, where perplexity is, intuitively, the number of words on average that can be selected next by a language model of the corpus (i.e., how confused the LM is about which word comes next in a test sentence). We computed perplexity as

$$PPL = 2^{-\frac{1}{N}\sum_{w=1}^{N}\log p(w)} \tag{8.1}$$

where N is the number of words in the corpus, and $p(w)$ is the probability of a given word according to a unigram LM computed over the whole corpus. The perplexity of each of the three sets of food logs we are interested in are shown in Table 8.2. We see

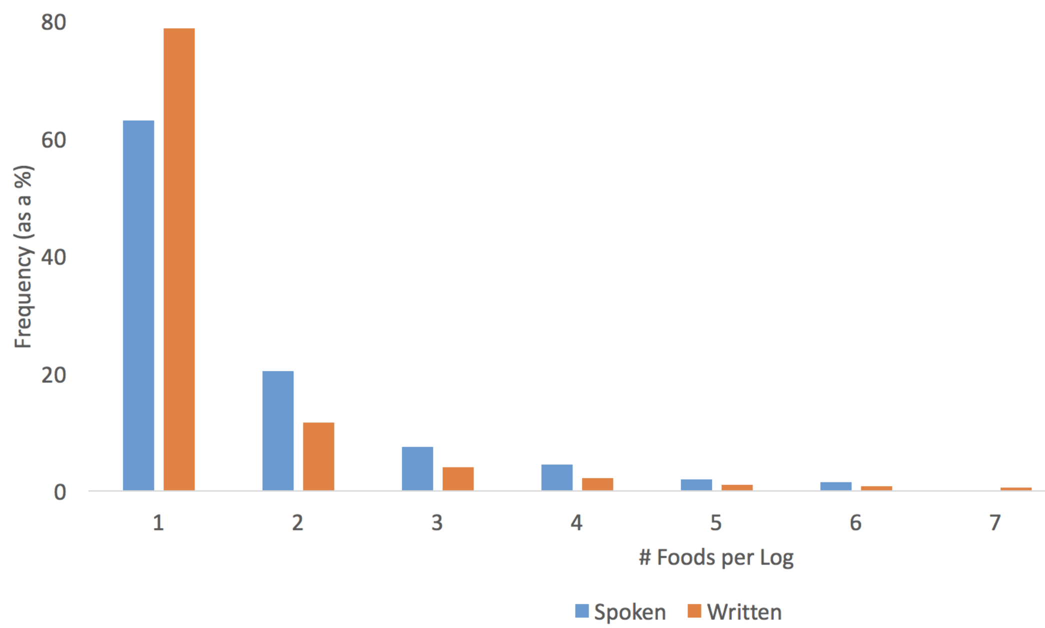


Figure 8-6: A histogram of the number of *foods* per spoken or typed food log.

that the spoken logs have the lowest perplexity, whereas both written corpora have higher perplexity. Interestingly, written food logs on *Coco* have much higher perplexity than that in AMT training data, which may indicate a mismatch between training and testing corpora and motivates our collection of data with the real-world system to re-train our models.

Corpus	**Perplexity**
AMT	241
Spoken	158
Typed	457

Table 8.2: Perplexity of three food log corpora: AMT training data, typed logs on *Coco*, and spoken logs on *Coco*. Note that we are computing the unigram probability separately for each corpus, rather than comparing different language models on a standard test set. The key takeaway is that written food logs have more variability than spoken descriptions.

8.2 Future Work

8.2.1 Combining Modalities

In the future, we aim to train our own nutrition speech recognizer, as well as combine the different modalities of language and vision. We are now collecting audio and transcripts from the Google and Siri recognizers when users log the food they have eaten, which would enable training our own in-house speech recognizer. As shown in Gür (2012), when doing speech recognition on clinical conversations with a recognizer trained on dictational data from doctors, the mismatch in training and testing data leads to poor performance, and can be improved by updating the default language model using a provided corpus. In our case, we need to be careful about ensuring the data our models and recognizer are trained on do not differ significantly from the meal descriptions recorded with the deployed application, and we can improve the language model based on our corpus of written meal descriptions. It would also be interesting to provide the predicted word lattice or N-best list from the recognizer to help handle speech recognition errors in downstream language understanding tasks. Many users have also requested the ability to take a photo of their meal, which may provide complementary information to that in speech and text. For example, if the user logs “pizza” with natural language, and a photo of their meal contains pepperoni slices, computer vision will disambiguate the type of pizza as pepperoni; likewise, if the photo of a meal is a glass of milk, the user might specify the percent milkfat verbally. People have trouble estimating how much they ate, so an image may provide quantity information without relying on the user.

8.2.2 Semantic Tagging and Database Mapping

We also plan to investigate jointly training the semantic tagger and database retrieval, as well as incorporating a feedback mechanism for learning online from users when it has made a tagging error. Our ultimate goal is to build a truly end-to-end diet tracking system that would use a single overarching model to carry on a conversation with a user about their diet choices, including the ability to acquire information, relate it to existing data, interpret it in terms of the user's goals and desires, and carry on a conversation to deliver requested information and advice to the user.

8.2.3 Dialogue

Finally, we aim to make the nutrition system more interactive, enabling users not only to log their meals, but also to ask nutrition questions (e.g., *Which has more iron, spinach or kale?*), and provide diet advice since many users have requested recommendations for a healthy snack to eat next that falls within their calorie goal, is tailored to the types of food they like to eat, and ideally is high in nutrients that they're missing for that day.

8.3 Summary

We believe the critical challenge of building dialogue systems is enabling them to converse *naturally*, like humans. Current production dialogue systems are often too tedious and unnatural for users, as may have been noted by anyone who has struggled to provide credit card information over the phone to automated banking systems. When conversational agents sound more natural, such as the new Google Duplex,[5] this allows them to communicate more easily with people. To this end, in this thesis, we have built deep

[5]https://ai.googleblog.com/2018/05/duplex-ai-system-for-natural-conversation.html

learning models that allow users to naturally log what they have eaten, and automatically retrieve the best matching food database items along with their nutrition facts. However, nutrition is only the first instantiation of this technology. We envision building personal AI digital assistants to help people in all aspects of life, from booking appointments and sending email, to logging the food they eat and how much they exercise, and even providing therapy and companionship. Obesity and diabetes are two major health problems facing America today, and we hope to address both. But healthcare is only the first step—we can have an even broader impact by incorporating conversational AI research into smart homes and appliances, self-driving cars, and disaster response.

Appendix A

Full Dialogue Sample Interactions

Here we show full sample dialogues with *Coco Nutritionist*, including several examples where the system cannot handle the user query properly. This illustrates the limitations of the current implementation, which we aim to address in future work. In Fig. A-1, we see that Coco correctly tags the user query "I had an apple with two tablespoons of peanut butter," labeling "an" and "two tablespoons" as `Quantity`, whereas "apple" and "peanut butter" are assigned the `Food` semantic tag. The correct food items are retrieved from the food database, and the correct quantities are found as well.

However, there are instances where the system performs poorly. For example, in Fig. A-2, the system incorrectly combines two different food items (i.e. "apple" and "toast") into a single food item. This is due to a mismatch between testing and training data. On AMT, workers were shown examples of meal descriptions, which they would tend to imitate closely (e.g. "I had a bowl of oatmeal with a cup of milk"), and were writing the meals instead of speaking. However, with the live system, users will not necessarily describe their meals in the same manner, and sometimes list ingredients one after the other in a less natural manner than if they were writing or talking to a friend.

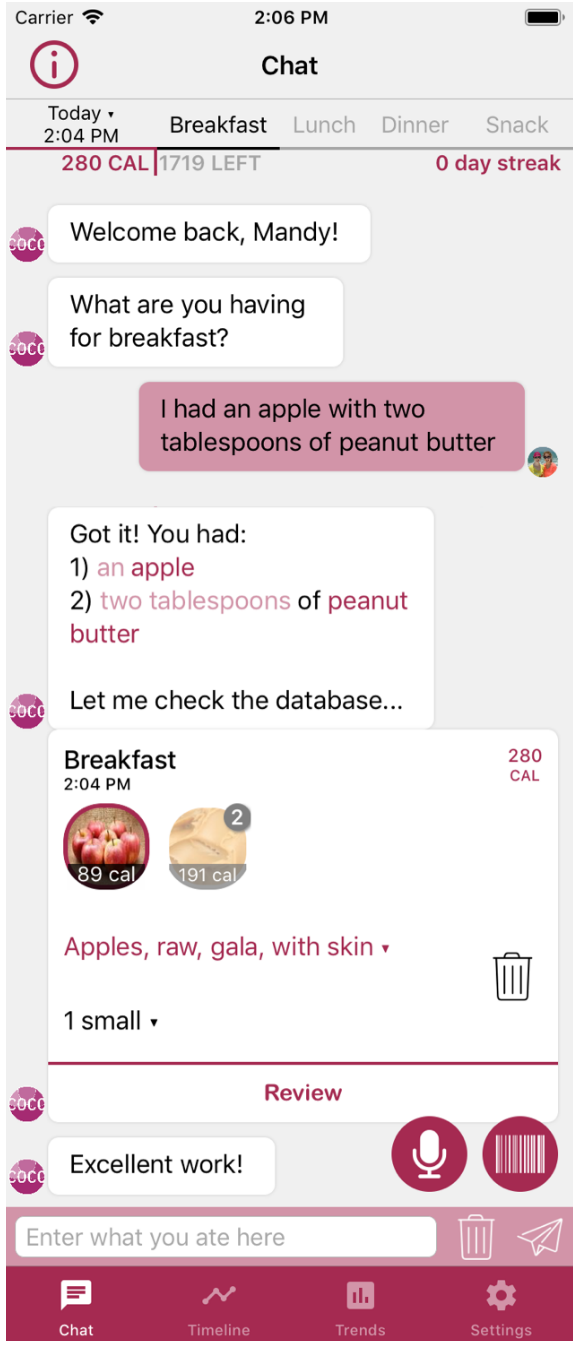


Figure A-1: A sample dialogue with *Coco Nutritionist*, where the system properly tags the user's meal description into foods and quantities, and retrieves the correct matching food database entries with their corresponding units and amounts.

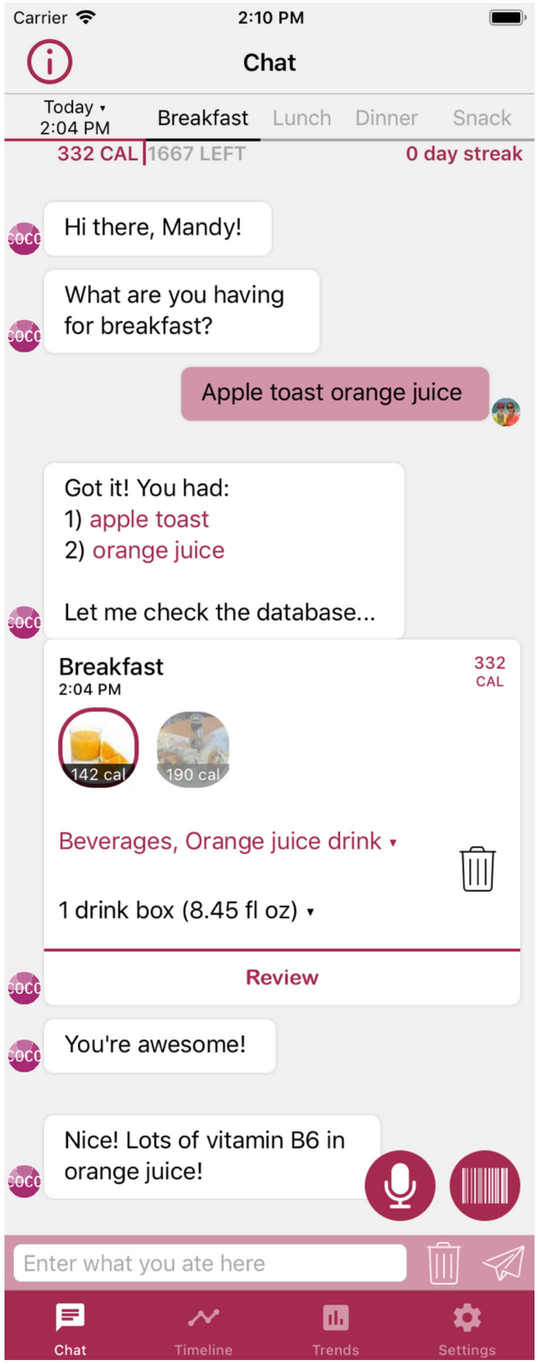


Figure A-2: A sample interaction where a tagging error (i.e., labeling "apple toast" as a single food item instead of two different foods) leads to retrieving incorrect food database matches. In this case, the user would delete the "apple toast" entry and try again.

Another limitation of the system is we are constrained to the food items that are in the database. This means we are missing many items on the menu at various restaurants, as well as other types of cuisines since the emphasis is on American food. In Fig. A-3, we see an example where the food item "chocolate perpetuem" does not exist in the USDA database. Thus, the predicted match for a mocha almond KIND bar is incorrect. One possibility for future work that could address this is to allow users to manually enter the nutrition facts for a missing food item. But since they might make typing mistakes, a better alternative might be to allow the user to take a picture of the nutrition facts label on the package, and automatically extract the nutrition facts with optical character recognition.

Another issue for food logging is composite food items. For example, sandwiches and salads are composed of many different ingredients. The system should understand the relationships between foods and ingredients, combining all the components into a single meal. In Fig. A-4, we see that the food description "a sandwich with Swiss cheese and ham" is incorrectly tagged as three different food items, rather than a single sandwich containing cheese and ham. We are currently implementing a feature to address this, where users can enter homemade meals and recipes, referring to these with speech. For example, in this case, the user may enter a new meal item named "Ham sandwich," which contains three ingredients (ham, bread, and Swiss cheese), and when they refer to this item, the system will be able to retrieve their personalized meal with ham and Swiss cheese.

Finally, the current system assumes every user query is a food description, and thus does not predict the user's intent. However, the user might be greeting or thanking Coco, or asking a nutrition question. Or there could be other subtleties, such as that shown in Fig. A-5, where the user is not logging a burger because they had a salad instead, but the system still tags both the burger and the salad as food items that were consumed. In the future, the system should be able to handle these cases, and others, such as if the user specifies "without mayonnaise," or corrects the system (e.g., "cooked, not raw").

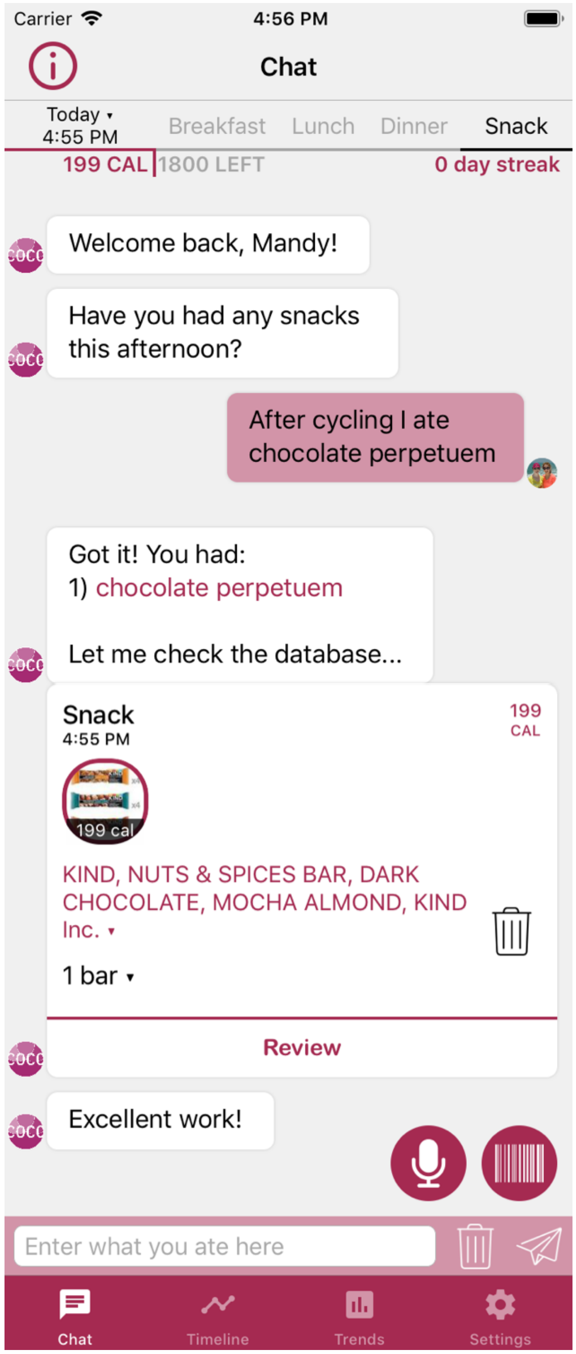


Figure A-3: A sample interaction with the system, where the user's logged food item does not exist in the USDA database.

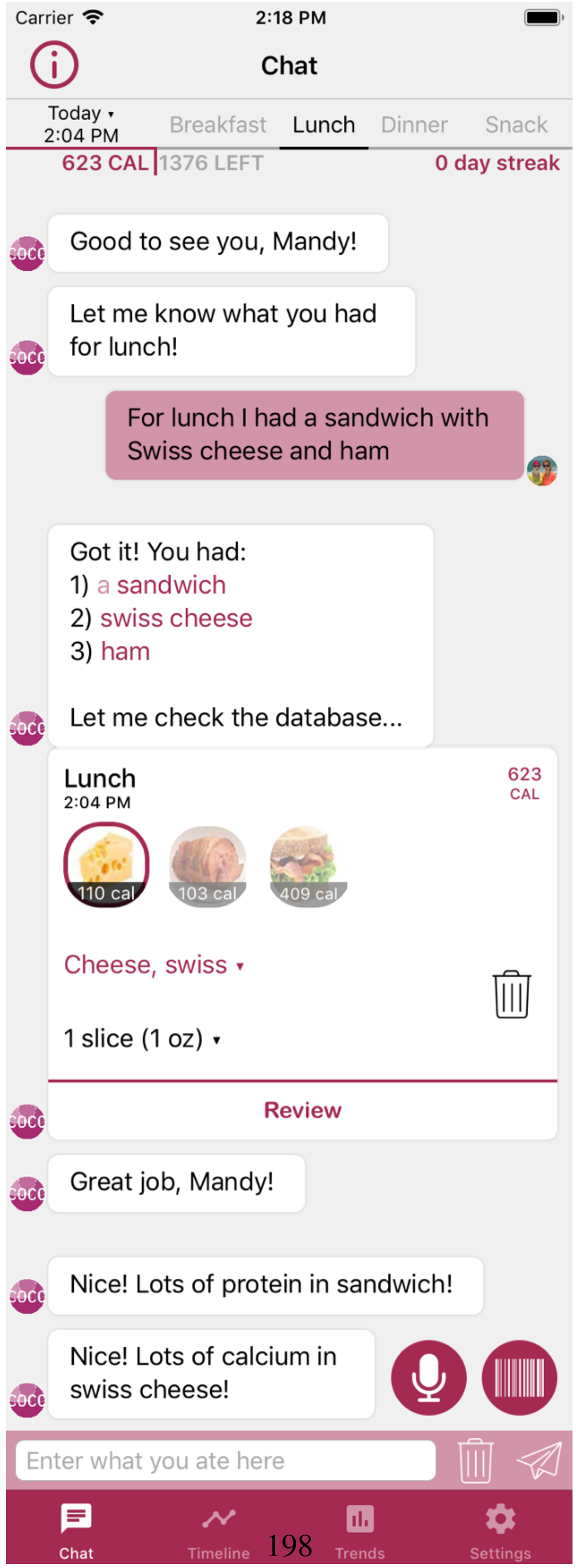


Figure A-4: Example meal description for a composite food item (i.e., a sandwich), that is incorrectly tagged as three separate food items.

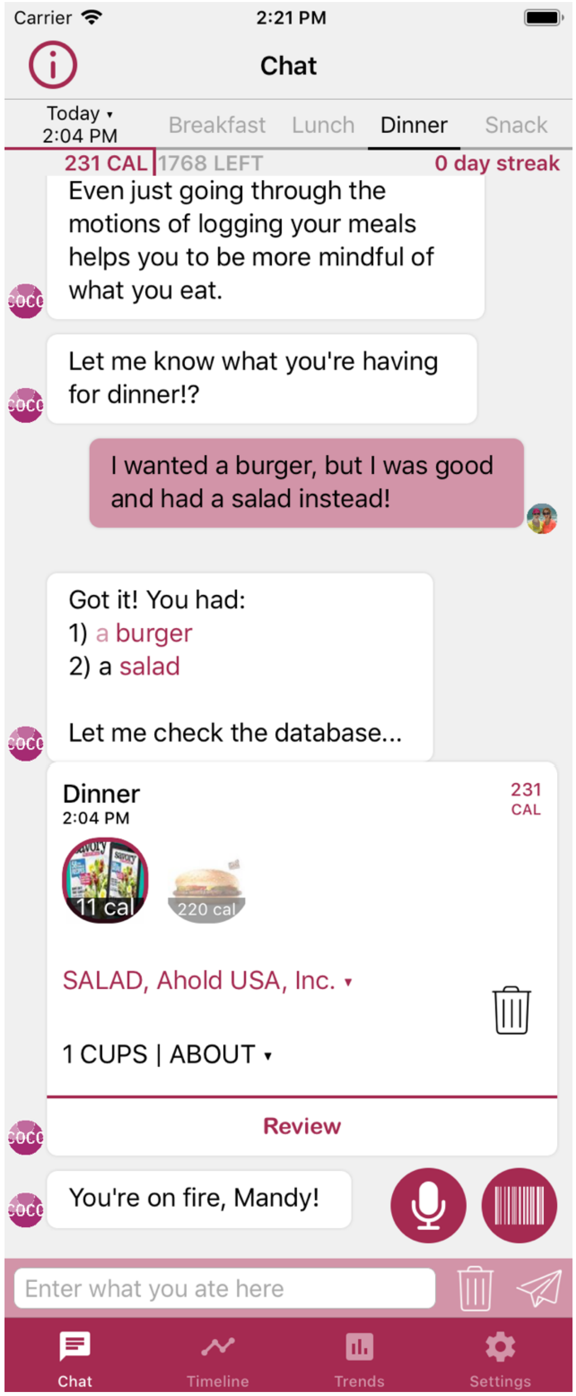


Figure A-5: An example interaction illustrating the importance of intent detection, which we do not currently handle, and requires a deeper understanding of the user's meaning.

Glossary

ABCNN Attention-based convolutional neural network (ABCNN) is a convolutional neural network that models relationships between pairs of sentences through an attention mechanism. 86

Adam An adaptive optimization method, used as a newer alternative to stochastic gradient descent. 69, 70, 90, 106, 128, 150, 160, 171

AMT Amazon Mechanical Turk (AMT) crowdsourcing platform is where researchers pay workers, also called Turkers, to complete a set of tasks for data collection and annotation. 37, 56, 61, 62, 75, 78, 129, 132, 182, 184, 187–189

ANN Artificial neural network (ANN) is a powerful type of machine learning model inspired by neural networks in the human brain. 42

ASR Automatic speech recognition (ASR) is the transcription of speech into text by computers. 71, 87, 155, 182, 185, 188

ATIS Air Travel Information System (ATIS) is an automated flight booking dialogue system from the nineties. 35, 66, 67, 77, 78, 82

attention Mechanism for aligning source and target inputs in seq2seq models. 47, 48, 69, 70, 83, 86, 87, 169, 180

BERT Bidirectional Encoder Representations from Transformers (BERT) is Google's deep, attention-based neural network model pre-trained on a large text dataset for learning contextual word representations. 66, 68–71, 76–78, 82

charCNN Character-based convolutional neural network (charCNN) is a CNN with character embeddings at the bottom layer instead of words. 91, 93, 94

CNN Convolutional neural network (CNN) is a type of neural network commonly used in computer vision, where a learned convolutional filter is slid over patches of an input image, or windows of words for text input. 38, 41, 46, 47, 52, 67, 69, 75, 76, 78, 82, 84–90, 94, 99–101, 103, 105–110, 112, 114, 115, 117, 122, 123, 141–143, 147, 151, 154, 156–161, 163–167, 169, 171–175, 179

CRF Conditional random field (CRF) models are statistical models that were previously state-of-the-art for semantic tagging and require hand-crafted feature engineering. 41, 66–71, 78, 82, 100, 142, 160

DNN Deep neural network (DNN) refers to an artificial neural network with multiple stacked hidden layers. 62, 87

dropout Method for regularizing neural networks, by setting random nodes to zero during training. 89, 106, 149, 155, 171

ELMo Embeddings from Language Models (ELMo) is another approach to learning contextual word representations. 68

F_1 An evaluation metric defined as the harmonic mean of precision and recall. 71

FF Feed-forward neural network (FF) is the simplest neural network, where every node in the previous layer is connected to every node in the subsequent layer. 42, 44, 69

GPR Generative Pre-trained Transformer (GPT) is OpenAI's deep, attention-based neural network model for learning contextual word representations. 68

GRU Gated recurrent unit (GRU) is a simpler version of the LSTM. 44, 46, 69, 85, 121

highway Type of connection in neural networks. 91, 92

HMM Hidden Markov model (HMM) is a statistical model with unobservable states that follow the Markov assumption, in which each state only depends on the previous state. 62

LM A language model (LM) predicts which word should come next in a sequence, given the previous context of words. 68, 188

LSTM Long short-term memory (LSTM) is a type of recurrent neural network with gating mechanisms for remembering longer history. 44, 67, 68, 85, 88, 99, 121, 122, 142, 151, 157, 158, 160, 164, 167, 169, 172, 173, 175, 180

MFCC Mel-frequency cepstral coefficient (MFCC) is a widely used feature in speech recognition models. 62

ML Machine learning (ML) is a sub-field of Computer Science that trains models on large amounts of labeled data for making predictions at test time. 42, 52, 54

MLP Multi-layer perceptron (MLP) is a standard feed-forward neural network. 42

MTL Multitask learning (MTL) involves training a single model to perform multiple related tasks, ideally improving generalizability and performance on all tasks by sharing information. 87, 113

NER Named entity recognition (NER), closely related to semantic tagging, involves labeling entities in sentences, such as people or locations. 67, 68, 87

NLP Natural language Processing (NLP) is the intelligent understanding of human language, either written or spoken, by computers. 31, 33, 41, 46–48, 50, 53, 54, 68, 86, 87, 120

OOV Out-of-vocabulary (OOV) words are unseen during training, and difficult for systems to handle at test time. 90, 91, 93, 94

ReLU Rectified linear unit (ReLU) is a piecewise activation function commonly used in neural networks. 42, 89, 106, 128, 149, 160, 161, 171

RNN Recurrent neural network (RNN) is a type of neural network commonly used in language tasks since it preserves the sequential order of words and sounds. 35, 41, 44, 67, 69, 142

SDS A spoken dialogue system (SDS) is an automated agent for tasks such as booking flights or restaurants. 33, 35, 156

seq2seq Sequence-to-sequence (seq2seq) models are an encoder-decoder style neural network, commonly used in machine translation and caption generation to generate text given some input. 48, 87, 167

sigmoid Final output prediction layer used in binary classification tasks. 43, 44, 62, 90, 105–107, 109, 110, 112, 145, 147, 158, 169, 171

SLU Spoken language understanding (SLU) involves natural language understanding, or semantic tagging, on spoken language. 65–67, 139–141

softmax Normalization layer used as the final output prediction layer in neural networks for classification tasks with multiple classes. 43, 62, 68, 70, 73, 90, 105, 107–110, 112–114, 121, 126, 145–147, 149, 161, 168, 169

SR Standard Reference (SR) is the smaller USDA food database of 7,793 basic foods, including fruits, vegetables, meat, and dairy. 55

SRL Semantic role labeling (SRL) is an NLP task that assigns labels to words or phrases in a sentence that indicate their semantic role, such as agent or predicate. 87

SVM Support vector machine (SVM) is a common supervised machine learning algorithm that finds a hyperplane separating different classes. 158, 164

Transformer A multi-head attention-based encoder-decoder neural network model. 47, 68

USDA United States Department of Agriculture (USDA) provides an online food database widely used by nutritionists. 36, 55, 56, 60, 62, 83–95, 97–103, 105–107, 109, 113–115, 118, 119, 123, 125, 126, 128–133, 179

WER Word error rate (WER) is a commonly used metric for measuring the performance of speech recognition. 62

Bibliography

Adi, Y., Kermany, E., Belinkov, Y., Lavi, O., and Goldberg, Y. (2016). Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *arXiv preprint arXiv:1608.04207*.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bai, Z., Yu, B., Chen, G., Wang, B., and Wang, Z. (2017). Modeling conversations to learn responding policies of e2e task-oriented dialog system. *Dial. Syst. Technol. Challenges*, 6.

Bingel, J. and Søgaard, A. (2017). Identifying beneficial task relations for multi-task learning in deep neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 164–169.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Bordes, A. and Weston, J. (2016). Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*.

Boureau, Y., Bordes, A., and Perez, J. (2017). Dialog state tracking challenge 6 end-to-end goal-oriented dialog track. Technical report, Tech. Rep.

Chen, Y., Hakkani-Tür, D., Tür, G., Gao, J., and Deng, L. (2016). End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding. In *Interspeech*, pages 3245–3249.

Chollet, F. (2015). keras. `https://github.com/fchollet/keras`.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 160–167. ACM.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Conneau, A., Schwenk, H., Lecun, Y., and Barrault, L. (2017). Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, page 11071116.

Conway, D. (1998). An algorithmic approach to English pluralization. In *Proceedings of the Second Annual Perl Conference. C. Salzenberg. San Jose, CA, O'Reilly*.

Craswell, N., Zaragoza, H., and Robertson, S. (2005). Microsoft cambridge at TREC-14: Enterprise track.

Dahl, D., Bates, M., Brown, M., Fisher, W., Hunicke-Smith, K., Pallett, D., Pao, C., Rudnicky, A., and Shriberg, E. (1994). Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Proceedings of the workshop on Human Language Technology*, pages 43–48. Association for Computational Linguistics.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dhingra, B., Li, L., Li, X., Gao, J., Chen, Y., Ahmed, F., and Deng, L. (2016). End-to-end reinforcement learning of dialogue agents for information access. *arXiv preprint arXiv:1609.00777*.

Eric, M. and Manning, C. (2017). Key-value retrieval networks for task-oriented dialogue. *arXiv preprint arXiv:1705.05414*.

Finkelstein, E., Trogdon, J., Cohen, J., and Dietz, W. (2009). Annual medical spending attributable to obesity: Payer-and service-specific estimates. *Health affairs*, 28(5):w822–w831.

Gers, F., Schraudolph, N., and Schmidhuber, J. (2003). Learning precise timing with LSTM recurrent networks. *The Journal of Machine Learning Research*, 3:115–143.

Glass, J., Flammia, G., Goodine, D., Phillips, M., Polifroni, J., Sakai, S., Seneff, S., and Zue, V. (1995). Multilingual spoken-language understanding in the MIT voyager system. *Speech communication*, 17(1-2):1–18.

Glass, J., Hazen, T., and Hetherington, I. (1999). Real-time telephone-based speech recognition in the jupiter domain. In *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No. 99CH36258)*, volume 1, pages 61–64. IEEE.

Goo, C., Gao, G., Hsu, Y., Huo, C., Chen, T., Hsu, K., and Chen, Y. (2018). Slot-gated modeling for joint slot filling and intent prediction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), Volume 2 (Short Papers)*, volume 2, pages 753–757.

Gorry, G., Kassirer, J., Essig, A., and Schwartz, W. (1973). Decision analysis as the basis for computer-aided management of acute renal failure. *The American journal of medicine*, 55(4):473–484.

Gür, B. (2012). Improving speech recognition accuracy for clinical conversations. Master's thesis, Massachusetts Institute of Technology.

Haghani, P., Narayanan, A., Bacchiani, M., Chuang, G., Gaur, N., Moreno, P., Prabhavalkar, R., Qu, Z., and Waters, A. (2018). From audio to semantics: Approaches to end-to-end spoken language understanding. *arXiv preprint arXiv:1809.09190*.

Hakkani-Tür, D., Tür, G., Celikyilmaz, A., Chen, Y., Gao, J., Deng, L., and Wang, Y. (2016). Multi-domain joint semantic frame parsing using bi-directional RNN-LSTM. In *Interspeech*, pages 715–719.

Ham, J., Lim, S., and Kim, K. (2017). Extended hybrid code networks for dstc6 fair dialog dataset. *Dial. Syst. Technol. Challenges*, 6.

Harwath, D., Torralba, A., and Glass, J. (2016). Unsupervised learning of spoken language with visual context. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NIPS)*, pages 1858–1866.

He, J., Chen, J., He, X., Gao, J., Li, L., Deng, L., and Ostendorf, M. (2016). Deep reinforcement learning with an action space defined by natural language. In *Workshop Track of the International Conference on Learning Representations (ICLR)*.

He, Y. and Young, S. (2003). A data-driven spoken language understanding system. In *Proceedings of 2003 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 583–588. IEEE.

Heintze, S., Baumann, T., and Schlangen, D. (2010). Comparing local and sequential models for statistical incremental natural language understanding. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 9–16. Association for Computational Linguistics.

Hemphill, C., Godfrey, J., Doddington, G., et al. (1990). The ATIS spoken language systems pilot corpus. In *Proceedings of the DARPA speech and natural language workshop*, pages 96–101.

Henderson, J., Lemon, O., and Georgila, K. (2005). Hybrid reinforcement/supervised learning for dialogue policies from communicator data. In *IJCAI workshop on knowledge and reasoning in practical dialogue systems*, pages 68–75.

Henderson, M., Thomson, B., and Williams, J. (2014a). The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272.

Henderson, M., Thomson, B., and Young, S. (2014b). Robust dialog state tracking using delexicalised recurrent neural networks and unsupervised adaptation. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 360–365. IEEE.

Henderson, M., Thomson, B., and Young, S. (2014c). Word-based dialog state tracking with recurrent neural networks. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 292–299.

Hermann, K. and Blunsom, P. (2014). Multilingual models for compositional distributed semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hu, B., Lu, Z., Li, H., and Chen, Q. (2014). Convolutional neural network architectures for matching natural language sentences. In *Proceedings of Advances in neural information processing systems (NIPS)*, pages 2042–2050.

Hu, H., Zheng, K., Wang, X., and Zhou, A. (2015). Gfilter: A general gram filter for string similarity search. *IEEE Transactions on Knowledge and Data Engineering*, 27(4):1005–1018.

Huang, P., He, X., Gao, J., Deng, L., Acero, A., and Heck, L. (2013). Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2333–2338. ACM.

Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Juang, B. and Rabiner, L. (1991). Hidden Markov models for speech recognition. *Technometrics*, 33(3):251–272.

Karpathy, A. and Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3128–3137.

Ketkar, N. (2017). Introduction to pytorch. In *Deep learning with python*, pages 195–208. Springer.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kiros, R., Zhu, Y., Salakhutdinov, R., Zemel, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Klerke, S., Goldberg, Y., and Søgaard, A. (2016). Improving sentence compression by learning to predict gaze. In *Proceedings of the Human Language Technologies: The 2016 Annual Conference of the North American Chapter of the ACL (HLT-NAACL)*, pages 1528–1533.

Knuth, D., Morris, J., and Pratt, V. (1977). Fast pattern matching in strings. *SIAM journal on computing*, 6(2):323–350.

Korpusik, M. (2015). Spoken language understanding in a nutrition dialogue system. Master's thesis, Massachusetts Institute of Technology.

Korpusik, M., Collins, Z., and Glass, J. (2017a). Semantic mapping of natural language input to database entries via convolutional neural networks. *Proceedings of IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5685–5689.

Korpusik, M. and Glass (2019a). Convolutional neural encoder for the 7th dialogue system technology challenge. In *Proceedings of 2019 AAAI 7th Dialogue System Technology Challenge Workshop (DSTC7)*. AAAI.

Korpusik, M. and Glass, J. (2017). Spoken language understanding for a nutrition dialogue system. *IEEE Transactions on Audio, Speech, and Language Processing*, 25:1450–1461.

Korpusik, M. and Glass, J. (2018a). Convolutional neural networks and multitask strategies for semantic mapping of natural language input to a structured database. In *Proceedings of IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6174–6178. IEEE.

Korpusik, M. and Glass, J. (2018b). Convolutional neural networks for dialogue state tracking without pre-trained word vectors or semantic dictionaries. In *Proceedings of 2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 884–891. IEEE.

Korpusik, M. and Glass, J. (2019b). Dialogue state tracking with convolutional semantic taggers. In *Proceedings of IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE.

Korpusik, M., Huang, C., Price, M., and Glass, J. (2016a). Distributional semantics for understanding spoken meal descriptions. *Proceedings of 2016 IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6070–6074.

Korpusik, M., Sakaki, S., Chen, F., and Chen, Y. (2016b). Recurrent neural networks for customer purchase prediction on twitter. In *CBRecSys@ RecSys*, pages 47–50.

Korpusik, M., Schmidt, N., Drexler, J., Cyphers, S., and Glass, J. (2014). Data collection and language understanding of food descriptions. *Proceedings of 2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 560–565.

Korpusik, M., Z. Collins, and Glass, J. (2017b). Character-based embedding models and reranking strategies for understanding natural language meal descriptions. *Proceedings of Interspeech*, pages 3320–3324.

Kummerfeld, J., Gouravajhala, S., Peper, J., Athreya, V., Gunasekara, C., Ganhotra, J., Patel, S., Polymenakos, L., and Lasecki, W. (2018). Analyzing assumptions in conversation disentanglement research through the lens of a new dataset and model. *arXiv preprint arXiv:1810.11118*.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

LeCun, Y., Bengio, Y., et al. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

Lee, J., Kim, D., Sarikaya, R., and Kim, Y. (2018). Coupled representation learning for domains, intents and slots in spoken language understanding. *arXiv preprint arXiv:1812.06083*.

Lemon, O. and Pietquin, O. (2012). *Data-driven Methods for Adaptive Spoken Dialogue Systems: Computational Learning for Conversational Interfaces*. Springer Science & Business Media.

Levin, E., Pieraccini, R., and Eckert, W. (1998). Using Markov decision process for learning dialogue strategies. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*, volume 1, pages 201–204. IEEE.

Li, J., Luong, M., and Jurafsky, D. (2015). A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*.

Li, J., Miller, A., Chopra, S., Ranzato, M., and Weston, J. (2016a). Learning through dialogue interactions. *arXiv preprint arXiv:1612.04936*.

Li, J., Monroe, W., Ritter, A., Galley, M., Gao, J., and Jurafsky, D. (2016b). Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*.

Li, J., Monroe, W., Shi, T., Jean, S., Ritter, A., and Jurafsky, D. (2017a). Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*.

Li, X., Chen, Y., Li, L., and Gao, J. (2017b). End-to-end task-completion neural dialogue systems. *arXiv preprint arXiv:1703.01008*.

Lin, D. and Wu, X. (2009). Phrase clustering for discriminative learning. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1030–1038. Association for Computational Linguistics.

Liu, B. and Lane, I. (2016). Attention-based recurrent neural network models for joint intent detection and slot filling. *arXiv preprint arXiv:1609.01454*.

Liu, B., Tur, G., Hakkani-Tur, D., Shah, P., and Heck, L. (2018). Dialogue learning with human teaching and feedback in end-to-end trainable task-oriented dialogue systems. *arXiv preprint arXiv:1804.06512*.

Liu, J., Pasupat, P., Cyphers, S., and Glass, J. (2013a). Asgard: A portable architecture for multilingual dialogue systems. In *Proceedings of IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8386–8390. IEEE.

Liu, J., Pasupat, P., Wang, Y., Cyphers, S., and Glass, J. (2013b). Query understanding enhanced by hierarchical parsing structures. In *Proceedings of 2013 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 72–77. IEEE.

Liu, X., Gao, J., He, X., Deng, L., Duh, K., and Wang, Y. (2015). Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proceedings of the Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL (HLT-NAACL)*, pages 912–921.

Lowe, R., Pow, N., Serban, I., and Pineau, J. (2015). The Ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909*.

Luong, M., Le, Q., Sutskever, I., Vinyals, O., and Kaiser, L. (2016). Multi-task sequence to sequence learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Ma, M., Zhao, K., Huang, L., Xiang, B., and Zhou, B. (2017). Jointly trained sequential labeling and classification by sparse attention neural networks. *arXiv preprint arXiv:1709.10191*.

Marzal, A. and Vidal, E. (1993). Computation of normalized edit distance and applications. *IEEE transactions on pattern analysis and machine intelligence*, 15(9):926–932.

Mesnil, G., Dauphin, Y., Yao, K., Bengio, Y., Deng, L., Hakkani-Tur, D., He, X., Heck, L., Tur, G., Yu, D., et al. (2015). Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

Mesnil, G., He, X., Deng, L., and Bengio, Y. (2013a). Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Proceedings of Fourteenth International Conference on Spoken Language Processing (Interspeech)*, pages 3771–3775.

Mesnil, G., He, X., Deng, L., and Bengio, Y. (2013b). Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Proceedings of the Fourteenth International Conference on Spoken Language Processing (Interspeech)*, pages 3771–3775.

Meza-Ruiz, I., Riedel, S., and Lemon, O. (2008). Accurate statistical spoken language understanding from limited development resources. In *Proceedings of 2008 IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5021–5024. IEEE.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Chen, K., Corrado, G., Dean, J., Sutskever, L., and Zweig, G. (2013b). word2vec. *URL https://code. google. com/p/word2vec*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013c). Distributed representations of words and phrases and their compositionality. In *Proceedings of Neural Information Processing Systems (NIPS)*, pages 3111–3119.

Miller, A., Fisch, A., Dodge, J., Karimi, A., Bordes, A., and Weston, J. (2016). Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*.

Misra, I., Shrivastava, A., Gupta, A., and Hebert, M. (2016). Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3994–4003.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mo, K., Li, S., Zhang, Y., Li, J., and Yang, Q. (2016). Personalizing a dialogue system with transfer learning. *arXiv preprint arXiv:1610.02891*.

Mrkšić, N., Séaghdha, D., Thomson, B., Gašić, M., Su, P., Vandyke, D., Wen, T., and Young, S. (2015). Multi-domain dialog state tracking using recurrent neural networks. *arXiv preprint arXiv:1506.07190*.

Mrkšić, N., Séaghdha, D., Wen, T., Thomson, B., and Young, S. (2016). Neural belief tracker: Data-driven dialogue state tracking. *arXiv preprint arXiv:1606.03777*.

Nadeau, D. and Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Naphtal, R. (2015). Natural language processing based nutritional application. Master's thesis, Massachusetts Institute of Technology.

Narasimhan, K., Kulkarni, T., and Barzilay, R. (2015). Language understanding for text-based games using deep reinforcement learning. *arXiv preprint arXiv:1506.08941*.

Ogden, C., Carroll, M., Kit, B., and Flegal, K. (2014). Prevalence of childhood and adult obesity in the United States, 2011-2012. *Jama*, 311(8):806–814.

Pang, L., Lan, Y., Guo, J., Xu, J., Wan, S., and Cheng, X. (2016). Text matching as image recognition. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2793–2799.

Passos, A., Kumar, V., and McCallum, A. (2014). Lexicon infused phrase embeddings for named entity resolution. *arXiv preprint arXiv:1404.5367*.

Pellom, B., Ward, W., and Pradhan, S. (2000). The CU communicator: An architecture for dialogue systems. In *Sixth International Conference on Spoken Language Processing*.

Peng, B., Li, X., Li, L., Gao, J., Celikyilmaz, A., Lee, S., and Wong, K. (2017). Composite task-completion dialogue system via hierarchical deep reinforcement learning. *arXiv preprint arXiv:1704.03084*.

Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global vectors for word representation. *Proceedings of 2014 Conference on Empirical Methods on Natural Language (EMNLP)*, 12.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G., and Vesely, K. (2011). The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding (ASRU)*.

Qiu, M., Li, F., Wang, S., Gao, X., Chen, Y., Zhao, W., Chen, H., Huang, J., and Chu, W. (2017). Alime chat: A sequence to sequence and rerank based chatbot engine. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 498–503.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*.

Rastogi, A., Hakkani-Tür, D., and Heck, L. (2017). Scalable multi-domain dialogue state tracking. In *Proceedings of 2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 561–568. IEEE.

Raymond, C. and Riccardi, G. (2007). Generative and discriminative algorithms for spoken language understanding. In *Proceedings of the Eighth International Conference on Spoken Language Processing (Interspeech)*, pages 1605–1608.

Sakaki, S., Chen, F., Korpusik, M., and Chen, Y. (2016). Corpus for customer purchase behavior prediction in social media. In *LREC*.

Saylor, P. (2015). Spoke: A framework for building speech-enabled websites. Master's thesis, Massachusetts Institute of Technology.

Shah, P., Hakkani-Tür, D., Tür, G., Rastogi, A., Bapna, A., Nayak, N., and Heck, L. (2018). Building a conversational agent overnight with dialogue self-play. *arXiv preprint arXiv:1801.04871*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Srivastava, R., Greff, K., and Schmidhuber, J. (2015). Highway networks. *arXiv preprint arXiv:1505.00387*.

Sun, K., Chen, L., Zhu, S., and Yu, K. (2014). The SJTU system for dialog state tracking challenge 2. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 318–326.

Sutskever, I., Vinyals, O., and Le, Q. (2014). Sequence to sequence learning with neural networks. In *Proceedings of Advances in neural information processing systems (NIPS)*, pages 3104–3112.

Sutton, C., McCallum, A., et al. (2012). An introduction to conditional random fields. *Foundations and Trends® in Machine Learning*, 4(4):267–373.

Thomson, B. and Young, S. (2010). Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech & Language*, 24(4):562–588.

Toshniwal, S., Tang, H., Lu, L., and Livescu, K. (2017). Multitask learning with low-level auxiliary tasks for encoder-decoder based speech recognition. *arXiv preprint arXiv:1704.01631*.

Tur, G., Hakkani-Tür, D., and Heck, L. (2010). What is left to be understood in ATIS? In *Proceedings of the 2010 IEEE Workshop on Spoken Language Technology (SLT)*, pages 19–24. IEEE.

Tur, G., Hakkani-Tür, D., Heck, L., and Parthasarathy, S. (2011). Sentence simplification for spoken language understanding. In *Proceedings of the 2011 IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5628–5631. IEEE.

Varges, S., Quarteroni, S., Riccardi, G., and Ivanov, A. (2010). Investigating clarification strategies in a hybrid POMDP dialog manager. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 213–216. Association for Computational Linguistics.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5998–6008.

Vergin, R., O'shaughnessy, D., and Farhat, A. (1999). Generalized mel frequency cepstral coefficients for large-vocabulary speaker-independent continuous-speech recognition. *IEEE Transactions on Speech and Audio Processing*, 7(5):525–532.

Viera, A., Garrett, J., et al. (2005). Understanding interobserver agreement: The kappa statistic. *Family Medicine*, 37(5):360–363.

Wang, F., Zuo, W., Lin, L., Zhang, D., and Zhang, L. (2016a). Joint learning of single-image and cross-image representations for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1288–1296.

Wang, Y., Acero, A., Mahajan, M., and Lee, J. (2006). Combining statistical and knowledge-based spoken language understanding in conditional models. In *Proceedings* *of the COLING/ACL on Main conference poster sessions*, pages 882–889. Association for Computational Linguistics.

Wang, Y. and Beydoun, M. (2007). The obesity epidemic in the United States–gender, age, socioeconomic, racial/ethnic, and geographic characteristics: A systematic review and meta-regression analysis. *Epidemiologic reviews*, 29(1):6–28.

Wang, Z. and Lemon, O. (2013). A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information. In *Proceedings of the SIGDIAL 2013 Conference*, pages 423–432.

Wang, Z., Mi, H., and Ittycheriah, A. (2016b). Sentence similarity learning by lexical decomposition and composition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1340–1349.

Wen, T., Vandyke, D., Mrksic, N., Gasic, M., Rojas-Barahona, L., Su, P., Ultes, S., and Young, S. (2016). A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*.

Weston, J., Bengio, S., and Usunier, N. (2011). Wsabie: Scaling up to large vocabulary image annotation. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence (IJCAI)*, volume 11, pages 2764–2770.

Williams, J. (2014). Web-style ranking and SLU combination for dialog state tracking. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 282–291.

Williams, J., Raux, A., Ramachandran, D., and Black, A. (2013). The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413.

Williams, J. and Zweig, G. (2016). End-to-end LSTM-based dialog control optimized with supervised and reinforcement learning. *arXiv preprint arXiv:1606.01269*.

Wu, Y., Wu, W., Xing, C., Zhou, M., and Li, Z. (2016). Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. *arXiv preprint arXiv:1612.01627*.

Xiao, Y. and Cho, K. (2017). Efficient character-level document classification by combining convolution and recurrent layers. In *Proceedings of the Thirtieth International Florida Artificial Intelligence Research Society Conference*, pages 353–358.

Xing, C., Wu, W., Wu, Y., Liu, J., Huang, Y., Zhou, M., and Ma, W. (2017). Topic aware neural response generation. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Xu, P. and Sarikaya, R. (2013). Convolutional neural network based triangular CRF for joint intent detection and slot filling. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 78–83. IEEE.

Xu, Y. and Seneff, S. (2010). Dialogue management based on entities and constraints. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue (Sigdial)*, pages 87–90. Association for Computational Linguistics.

Yao, K., Peng, B., Zhang, Y., Yu, D., Zweig, G., and Shi, Y. (2014a). Spoken language understanding using long short-term memory neural networks. In *Proceedings of the 2014 IEEE Workshop on Spoken Language Technology (SLT)*, pages 189–194. IEEE.

Yao, K., Peng, B., Zweig, G., Yu, D., Li, X., and Gao, F. (2014b). Recurrent conditional random field for language understanding. In *Proceedings of 2014 IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4077–4081. IEEE.

Yao, K., Zweig, G., Hwang, M., Shi, Y., and Yu, D. (2013). Recurrent neural networks for language understanding. In *Proceedings of Fourteenth International Conference on Spoken Language Processing (Interspeech)*, pages 2524–2528.

Yin, W., Ebert, S., and Schütze, H. (2016a). Attention-based convolutional neural network for machine comprehension. In *Proceedings of 2016 NAACL Human-Computer Question Answering Workshop*, pages 15–21.

Yin, W., Schütze, H., Xiang, B., and Zhou, B. (2016b). ABCNN: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272.

Yoshino, K., Hori, C., Perez, J., D'Haro, L. F., Polymenakos, L., Gunasekara, C., Lasecki, W. S., Kummerfeld, J., Galley, M., Brockett, C., Gao, J., Dolan, B., Gao, S., Marks, T. K., Parikh, D., and Batra, D. (2018). The 7th dialog system technology challenge. *arXiv preprint*.

Young, S., Gašić, M., Keizer, S., Mairesse, F., Schatzmann, J., Thomson, B., and Yu, K. (2010). The hidden information state model: A practical framework for POMDP-based spoken dialogue management. *Computer Speech & Language*, 24(2):150–174.

Young, S., Gašić, M., Thomson, B., and Williams, J. (2013). POMDP-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. In *Proceedings of Advances in neural information processing systems (NIPS)*, pages 649–657.

Zhao, T. and Eskenazi, M. (2016). Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. *Proceedings of SIGDIAL*.

Zhong, V., Xiong, C., and Socher, R. (2018). Global-locally self-attentive dialogue state tracker. *arXiv preprint arXiv:1805.09655*.

Zhou, X., Li, L., Dong, D., Liu, Y., Chen, Y., Zhao, W., Yu, D., and Wu, H. (2018). Multi-turn response selection for chatbots with deep attention matching network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1118–1127.

Zilka, L. and Jurcicek, F. (2015). Incremental LSTM-based dialog state tracker. In *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*, pages 757–762. IEEE.

Zue, V., Seneff, S., Polifroni, J., Phillips, M., Pao, C., Goodine, D., Goddeau, D., and Glass, J. (1994). Pegasus: A spoken dialogue interface for on-line air travel planning. *Speech Communication*, 15(3-4):331–340.