

# Spoken Language Understanding in a Nutrition Dialogue System

by

Mandy B. Korpusik

B.S., Franklin W. Olin College of Engineering (2013)

Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2015

© Massachusetts Institute of Technology 2015. All rights reserved.

Author .....  
Department of Electrical Engineering and Computer Science  
May 20, 2015

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

Accepted by .....  
Leslie A. Kolodziejcki  
Chairman, Department Committee on Graduate Theses



# Spoken Language Understanding in a Nutrition Dialogue System

by

Mandy B. Korpusik

Submitted to the Department of Electrical Engineering and Computer Science  
on May 20, 2015, in partial fulfillment of the  
requirements for the degree of  
Master of Science in Electrical Engineering and Computer Science

## Abstract

Existing approaches for the prevention and treatment of obesity are hampered by the lack of accurate, low-burden methods for self-assessment of food intake, especially for hard-to-reach, low-literate populations. For this reason, we propose a novel approach to diet tracking that utilizes speech understanding and dialogue technology in order to enable efficient self-assessment of energy and nutrient consumption. We are interested in studying whether speech can lower user workload compared to existing self-assessment methods, whether spoken language descriptions of meals can accurately quantify caloric and nutrient absorption, and whether dialogue can efficiently and effectively be used to ascertain and clarify food properties, perhaps in conjunction with other modalities. In this thesis, we explore the core innovation of our nutrition system: the language understanding component which relies on machine learning methods to automatically detect food concepts in a user's spoken meal description.

In particular, we investigate the performance of conditional random field (CRF) models for semantic labeling and segmentation of spoken meal descriptions. On a corpus of 10,000 meal descriptions, we achieve an average F1 test score of 90.7 for semantic tagging and 86.3 for associating foods with properties. In a study of users interacting with an initial prototype of the system, semantic tagging achieved an accuracy of 83%, which was sufficiently high to satisfy users.

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



## Acknowledgments

First, I would like to thank the Electrical Engineering and Computer Science department at MIT for accepting me into their graduate school Ph.D. program. I am incredibly grateful for this opportunity and am really enjoying every moment here. I could not have asked for a more welcoming, supportive community full of intelligent, passionate researchers who love learning, exploring, and asking questions. I have already learned so much here and cannot wait to see where this journey takes me.

Leslie and Janet, thank you for your warm welcome during our first semester here in the EECS Women's breakfast seminar—I will always remember the last morning when you made us delicious, homemade waffles (including a gluten-free option)!

Thank you to GW6 and GWAMIT for providing many opportunities to meet other graduate women at MIT and for organizing professional and social activities for women such as the Path of Professorship workshop, GWAMIT mentoring program (thank you, Kathy Huber, for being an awesome mentor!), empowerment and leadership conferences, and book clubs.

Life at MIT would not be nearly as much fun without my best friends—thank you Mengfei, Amy, Ramya, Jennifer, and Danielle for always being there for me, celebrating my birthday (especially our first year here when you planned a scavenger hunt and baked a gluten-free cake from scratch for me!), and of course, our girls' nights! Mengfei, thank you for being the most considerate, kind, and thoughtful roommate imaginable; Amy and Jennifer, thank you for being super intense, energetic exercise buddies; and Ramya and Danielle, thank you for being lovely study buddies!

Thank you, Uncle Peter and Aunt Patty, for everything you have done for me over the past six years. At first, I was terrified to move across the country, but your support (especially all the times you drove me to and from the airport and helped me move into my dorm rooms) eased the transition. It is truly a blessing to have had the opportunity to spend all these holidays with you and get to know you so well.

Of course, I would like to thank the Spoken Language Systems group for becoming my new family at MIT. I will always remember my first month here when you surprised

me the day before my birthday with what we now refer to as the “Mandy” cake (aka triple chocolate mousse from Flour). Thank you to my officemates for putting up with my repeating over and over, “This morning I had a bowl of cereal,” while I was testing the nutrition system. Jackie, it was an honor to help proofread your Ph.D. thesis and watch your defense—you have inspired me! Thank you, Jingjing, for co-advising me my first semester here—you were always very helpful and provided great ideas. Najim, thank you for always being the life of the party and for taking care of me and introducing me to so many people at SLT. Marcia, thank you for your sense of humor—you always brighten my day! Victor, thank you for always seeing the best in me and for being such a helpful graduate counselor. Stephen and Ekapol—you are like big brothers to me, and I am so grateful for all the advice you have given and for always listening to me and comforting me (especially during distributed systems). Scott, thank you for patiently answering my numerous questions and for all the help you have given me, both teaching me about SLU and helping search for internships.

Most of all, thank you, Jim, for being the best advisor imaginable. You are one of the nicest, most patient people I have ever met, and I am incredibly lucky to have the opportunity to work with you! Thank you for taking the time to teach me new concepts and to answer my questions. Thank you for always seeming like you have all the time in the world for each of your many students and for genuinely caring about us. If I become a professor one day, I hope to be like you!

Finally, thank you to my family, especially my parents, for being extremely caring and supportive throughout my entire life—I would not be here, finishing my Master’s thesis at MIT today, if not for you. You have shown me what it means to be an engineer and have always supported me in following my dreams. Your love and support mean so much to me! Thank you, Mom, for being my number one mentor and role model of a woman in computer science—I am truly grateful for everything you have done and continue to do for me, and I cannot thank you enough.

Thanks to undergraduate Calvin Huang for his work on the word vector features. This research was sponsored in part by a grant from Quanta Computing, Inc., and by the NIH.

# Contents

<b>1</b>	<b>Introduction</b>	<b>17</b>
1.1	Dialogue Systems . . . . .	17
1.2	Obesity in the United States . . . . .	18
1.3	A Nutrition Dialogue System . . . . .	18
1.3.1	Previous Work . . . . .	19
1.3.2	An Initial Prototype . . . . .	20
<b>2</b>	<b>Crowdsourcing the Data Collection</b>	<b>25</b>
2.1	Background and Related Work . . . . .	25
2.2	Nutrition Data Collection . . . . .	27
2.3	Data Statistics . . . . .	27
2.4	Inter-Annotator Agreement . . . . .	29
2.5	Challenges and Solutions . . . . .	31
2.5.1	Meal Diary Submission Prefiltering . . . . .	31
2.5.2	Property Labeling Prefiltering . . . . .	31
2.5.3	Inaccurate Food Labels . . . . .	32
2.5.4	Ongoing Data Collection . . . . .	33
<b>3</b>	<b>Conditional Random Fields (CRFs) for Semantic Tagging</b>	<b>35</b>
3.1	Background and Related Work . . . . .	35
3.1.1	Semantic Tagging . . . . .	36
3.2	Classifiers . . . . .	37
3.2.1	CRFs . . . . .	37

3.2.2	Semi-CRFs . . . . .	38
3.3	Features . . . . .	39
3.4	Results . . . . .	40
3.4.1	Labeling Results . . . . .	41
3.4.2	Error Analysis . . . . .	42
3.4.3	Effects of Noisy Data . . . . .	45
3.5	Conclusion . . . . .	46
<b>4</b>	<b>Distributional Semantics for Semantic Tagging</b>	<b>49</b>
4.1	Background and Related Work . . . . .	50
4.2	Generating Word Vectors . . . . .	50
4.2.1	Skip-gram Model . . . . .	51
4.2.2	Multiple-sense Embeddings Model . . . . .	53
4.2.3	GloVe Model . . . . .	53
4.3	Incorporating Word Vectors as Features . . . . .	53
4.3.1	Dense Embeddings . . . . .	54
4.3.2	Clustering Embeddings . . . . .	54
4.3.3	Distributional Prototypes . . . . .	55
4.4	Results . . . . .	56
4.4.1	Qualitatively Analyzing Vectors . . . . .	56
4.4.2	CRF Baseline . . . . .	58
4.4.3	Incorporating Vector Features . . . . .	59
4.5	Independent Questions of Inquiry . . . . .	60
4.5.1	Handling Unknown Vectors . . . . .	61
4.5.2	Performance with Varying Amounts of Data . . . . .	62
4.6	Conclusion . . . . .	63
<b>5</b>	<b>Associating Foods and Properties</b>	<b>65</b>
5.1	Segmenting Approaches . . . . .	66
5.1.1	Simple Rule . . . . .	66
5.1.2	Markov Model . . . . .	66



5.1.3	Transformation-Based Learning . . . . .	68
5.1.4	CRF . . . . .	68
5.2	Segmenting Results . . . . .	69
5.3	Food-Property Association with Word Vectors . . . . .	73
5.4	Comparing Segmentation to Classification . . . . .	76
5.5	Conclusion . . . . .	77
<b>6</b>	<b>The Nutrition System Prototype</b>	<b>79</b>
6.1	Related Work . . . . .	79
6.2	Ongoing Work . . . . .	81
6.2.1	Designing the User Interface . . . . .	81
6.2.2	Connecting It to a Database . . . . .	82
6.2.3	Engaging the User in Dialogue . . . . .	83
6.2.4	Context Resolution . . . . .	84
6.3	System Evaluation . . . . .	84
<b>7</b>	<b>Conclusion</b>	<b>87</b>
7.1	Contributions . . . . .	87
7.1.1	Data Collection . . . . .	88
7.1.2	Semantic Tagging . . . . .	88
7.1.3	Distributional Semantics . . . . .	88
7.1.4	Food-Property Association . . . . .	88
7.2	Directions for Future Research . . . . .	89
7.2.1	Semantic Tagging . . . . .	89
7.2.2	Distributional Semantics . . . . .	89
7.2.3	Food-Property Association . . . . .	90
7.3	Looking Forward . . . . .	90
<b>A</b>	<b>Training CRFs</b>	<b>93</b>
A.1	Inference . . . . .	93
A.2	Parameter Estimation . . . . .	95

<b>B</b>	<b>Measurements and Calculations</b>	<b>99</b>
B.1	Kappa Score . . . . .	99
B.2	F1 Score . . . . .	100
B.3	Significance Tests . . . . .	100

# List of Figures

1-1	A screenshot of the MyFitnessPal mobile application. . . . .	20
1-2	A screenshot of the PmEB mobile phone application where <b>a</b> is the main menu, <b>b</b> is the current caloric balance, <b>c</b> is the meal selection page, and <b>d</b> is the history. . . . .	22
1-3	A prototype of the nutrition system with an example food diary, where “oatmeal,” “banana,” and “milk” are identified as food items and “a bowl,” “a,” and “a glass” are quantities. . . . .	23
1-4	A diagram of the flow of the nutrition system. . . . .	24
2-1	The AMT task for labeling foods in a meal description. . . . .	28
2-2	The AMT task for labeling properties of foods. . . . .	28
2-3	Frequency of food items per meal description. . . . .	30
2-4	AMT task to collect unique, descriptive meals with the deployed system. . . . .	34
4-1	20 nearest neighbors of “bowl” and “cheese,” using 300-dimension word2vec embeddings trained on the Google News corpus and reduced to two dimensions through t-SNE. . . . .	57
4-2	20 nearest neighbors of “bowl” and “cheese,” using 300-dimension word2vec embeddings trained on nutrition data and reduced to two dimensions through t-SNE. . . . .	58
4-3	Semantic tagging average F1 score as a function of increasing data size, for two feature sets: baseline and baseline combined with raw vector values, raw similarity values, and shape. . . . .	64

5-1	A depiction of the food-property association task, in which the quantity “a bowl” is assigned to the food “cereal,” and “two cups” is associated with “milk.” . . . . .	65
5-2	A first-order Markov chain for the food description “I had a bowl of cereal.” . . . . .	67
6-1	A diagram of the nutrition system’s current architecture. . . . .	80
6-2	An AMT task for evaluating the system’s performance on the sentence “This morning for breakfast I had a bowl of oatmeal followed by a banana.” . . . . .	85
B-1	Probability density function of the Student’s $t$ distribution for varying degrees of freedom $v$ . . . . .	101
B-2	Probability mass function of the binomial distribution, where the tails are used for McNemar’s significance test [23]. . . . .	103

# List of Tables

2.1	Example AMT tasks (i.e., food labeling of meal descriptions) demonstrating inconsistencies in data due to gaming the system, mistakes, and confusion among Turkers. . . . .	26
2.2	Detailed information for each meal. . . . .	29
2.3	Labeling model’s 10-fold results (i.e., mean F1, variance, and standard deviation) for different thresholds of Turkers in the property labeling task. . . . .	29
2.4	Kappa scores for food and property labeling tasks. . . . .	30
3.1	Ten items from the food and brand lexicons. . . . .	40
3.2	Unigram, bigram, and POS features (10 each) learned by the semi-CRF. . . . .	40
3.3	CRF and semi-CRF cross-validation results for three different feature sets. The highest average F1 score is shown in bold. . . . .	41
3.4	CRF token-level performance on test data. . . . .	42
3.5	Semi-CRF concept-level performance on test data. . . . .	42
3.6	CRF versus semi-CRF errors, where $p < 0.01$ ( $p = 2.64 \times 10^{-401}$ ) according to McNemar’s significance test. $n_{00}$ is the number of tokens labeled correctly by both, $n_{11}$ is the number labeled incorrectly by both, $n_{10}$ is the number of tokens labeled correctly by the semi-CRF but not the standard CRF, and $n_{01}$ vice versa. . . . .	42
3.7	10 common CRF errors on the token level, along with the frequency of the error on the test data. . . . .	43

3.8	10 common semi-CRF errors on the token level, along with the frequency of the error on the test data. . . . .	43
3.9	An example meal description comparing the CRF and semi-CRF semantic tag predictions, with errors shown in bold. . . . .	44
3.10	10 most common CRF errors per label category. . . . .	45
3.11	10 most common semi-CRF errors per label category. . . . .	45
3.12	Number of tokens missed by the CRF and semi-CRF for each label category. . . . .	46
3.13	Semi-CRF semantic tagging performance on 38 randomly selected meal descriptions, using both AMT labels and expert hand labels. . . . .	46
3.14	Examples of Turker labeling mistakes. . . . .	47
4.1	Top eight prototype words for each category (except Other) selected using the NPMI metric, where out-of-vocabulary words were omitted. . . . .	55
4.2	Cross-validation average F1 scores across labels for three different similarity threshold values $\delta$ , where the best threshold for each label is in bold. These results were obtained on a smaller data set of 8,000 meal descriptions. . . . .	55
4.3	Cross-validation average F1 scores when using raw similarity value features with varying $m$ (i.e., the number of prototypes per label). These results were obtained on the most recent data set of 10,000 meal descriptions. The first row is the CRF baseline method from Chapter 3, as shown in the rightmost column of Table 3.4. . . . .	56
4.4	F1 scores per category when using all features (with vectors) trained on three different data sources. . . . .	58
4.5	Average F1 scores on the semantic tagging task with CRF++, CRF-suite, and Mallet on two feature sets: with and without raw word vector values (i.e., dense embeddings). . . . .	59

4.6	CRFsuite F1 scores per label in the semantic tagging task with incrementally complex feature sets: baseline n-grams, POS tags, and lexical features; dense embeddings; raw prototype similarities; shape; and clusters. The first row is the CRF baseline method from Chapter 3, as shown in the rightmost column of Table 3.4. . . . .	60
4.7	Semantic tagging F1 scores per label with only dense embedding features, for six prediction methods. The full system is taken from Section 4.4.3, where there is no OOV handling, but it performs better because it uses the full set of features. . . . .	63
4.8	Average F1 scores on the semantic tagging task for six vocabulary sizes with only dense embedding features, where prediction is done by averaging the top two predicted words of the basic language model trained on nutrition data. . . . .	63
5.1	Example of the food chunking classification problem, where a chunk label B, I, or O is assigned to each token, given its semi-CRF label (i.e., brand, quantity, description, or food). . . . .	69
5.2	A sample of five rules from the template for training the TBL model.	69
5.3	CRF++ template for learning features from tokens and property labels, where token <sub>0</sub> refers to the current token, and all other indices are relative to the current token. label <sub>-2</sub> ◦label <sub>-1</sub> indicates the combination of two features into a single feature. . . . .	70
5.4	Test performance of approaches to the food segmenting task, where accuracy is calculated at the token-level and precision, recall, and F1 are computed at the phrase-level. The CRF (shown in bold) achieved the best accuracy and F1 score. . . . .	71
5.5	BIO food chunking mistakes, where auto is the prediction and AMT is the gold standard annotation. . . . .	71

5.6	TBL (with simple rule baseline) high-scoring rules, where score is the number of improvements minus performance reductions. $C_i$ represents a chunk label at index $i$ (e.g., B, I, or O), and $L_i$ indicates the food or property label at $i$ . . . . .	72
5.7	Performance of CRF+TBL on segmentation task using three different label representations. . . . .	72
5.8	Oracle experiments on the segmenting task with six different methods, where AMT gold standard labels are used rather than semi-CRF predictions. . . . .	73
5.9	Food-property association with three different classifiers, for both gold standard semantic tags (i.e., oracle) and predicted tags. . . . .	75
5.10	Performance on the food-property association task using the prior approach of IOE segmenting with the CRF, the new food prediction method with the random forest classifier, and the union, evaluated on property tokens for all methods. . . . .	76
6.1	Examples of errors the system made in the AMT task for evaluating performance. There are five error types: substitutions (i.e., labeling a food as a non-food), insertions (labeling non-foods as food items), tagging (i.e., swapping property tags), quantity (i.e., predicting the incorrect database quantity), and USDA (i.e., selecting the incorrect USDA hit). . . . .	86
B.1	McNemar’s matrix of tokens labeled correctly or incorrectly by two methods. . . . .	103



# Chapter 1

## Introduction

Imagine you could track your diet simply by speaking to a mobile device about the meals you eat on a daily basis. You would not need to manually calculate calories, record foods one at a time, or select the correct match from a list of numerous options. Rather, you would describe your meal, and the device would automatically identify the foods and nutrition facts, asking followup questions as needed. You might even have a dialogue about healthy alternatives or foods you should eat in order to address the lack of a specific nutrient.

### 1.1 Dialogue Systems

Spoken dialogue systems like this one have become increasingly prevalent in today's society, especially with the advent of popular personal assistants such as Siri, Cortana, and Google Now on mobile devices. These systems enable a user to converse with a machine simply through speech, making tasks easier for users to accomplish. Although dialogue systems have recently made great advances, they are still an important area of active research due to unsolved challenges in each of their underlying components: speech recognition, language understanding, dialogue management, and response generation. First, the speech recognizer extracts the words in a user's query from the speech waveform. Then, the language understanding engine determines the user's meaning, or semantics, given the recognition output. Using the semantic

representation of the user’s query and the current state of the system, the dialogue manager selects the next action that the system should take. Finally, a response is generated using a text-to-speech mechanism. In this work, we investigate approaches for building the language understanding component of a nutrition system.

## 1.2 Obesity in the United States

A nutrition dialogue system has the potential to benefit society by addressing health concerns in the United States, especially the rising obesity rate. According to the work in [78], adult obesity increased from 13% to 32% between the 1960s and 2004, and it predicted that by 2015, 75% of adults would be overweight or obese, and 41% obese. More than one-third of American adults (i.e., 78.6 million) are obese [55], leading to health conditions such as heart disease, stroke, type 2 diabetes, and certain types of cancer.

The rise of obesity in the US is also associated with increased medical spending; for example, the estimated medical cost of obesity in 2008 was \$147 billion [19]. Research shows that, on average, obese patients cost Medicare over \$600 more per year than non-obese patients. From 1998 to 2006, the increase in spending costs attributable to an increase in obesity was statistically significant for all private payer services, ranging from \$420 for inpatient services (an 82% increase in adult per capita medical spending) to \$568 for prescription drugs (a 90% increase). Across all payers, the data indicate that obesity is associated with a 9.1% increase in annual medical spending, or as much as \$147 billion per year, with prescription drugs as the leading cost.

## 1.3 A Nutrition Dialogue System

In an effort to address the health concerns and medical costs caused by the rising obesity rate in the United States, researchers have begun to explore the application of dialogue systems to the medical domain. Prior work [43] has investigated the use of dialogue for discussing with users the potential side effects of drugs based on other

patients' online drug reviews. We plan to investigate whether dialogue would be effective in helping users track their nutrient intake. Existing methods for treating obesity through self-assessment of food intake are often too cumbersome and tedious for patients to use, especially for hard-to-reach, low-literate populations [57].

### 1.3.1 Previous Work

To overcome the barrier preventing many obese individuals from easily tracking their food intake, we propose building a spoken dialogue system with which users simply describe their meals, and the system automatically determines the nutrition content. Existing applications such as MyFitnessPal [33] (shown in Figure 1-1) for tracking nutrient and caloric intake require manually entering each eaten food by hand and selecting the correct item from a list of possibilities. Some apps such as CalorieCount [80] use speech recognition, but the user only records one food item at a time and selects the correct food from a list, rather than our novel approach of utilizing a dialogue system to automatically select the appropriate food item and attributes.

In addition to commercial applications such as MyFitnessPal and CalorieCount, academic research groups have also explored the benefits of incorporating technology into diet tracking. One group built a mobile phone application called Nutricam for recording dietary intake [64], which enabled recording a meal through both a photograph and speech. When used by adults with type 2 diabetes, Nutricam was found to be easier and faster to use than written food diaries, as well as acceptably accurate. The main difference between Nutricam and our nutrition system is that they asked a professional dietician to determine the nutrition facts, whereas our system automatically detects the nutrient content. Another group demonstrated the usability of a mobile phone application for diet tracking called PmEB (shown in Figure 1-2), which users preferred over recording meals on paper [74]. Again, this app required users to manually enter caloric and nutrient intake, whereas ours is automatic.

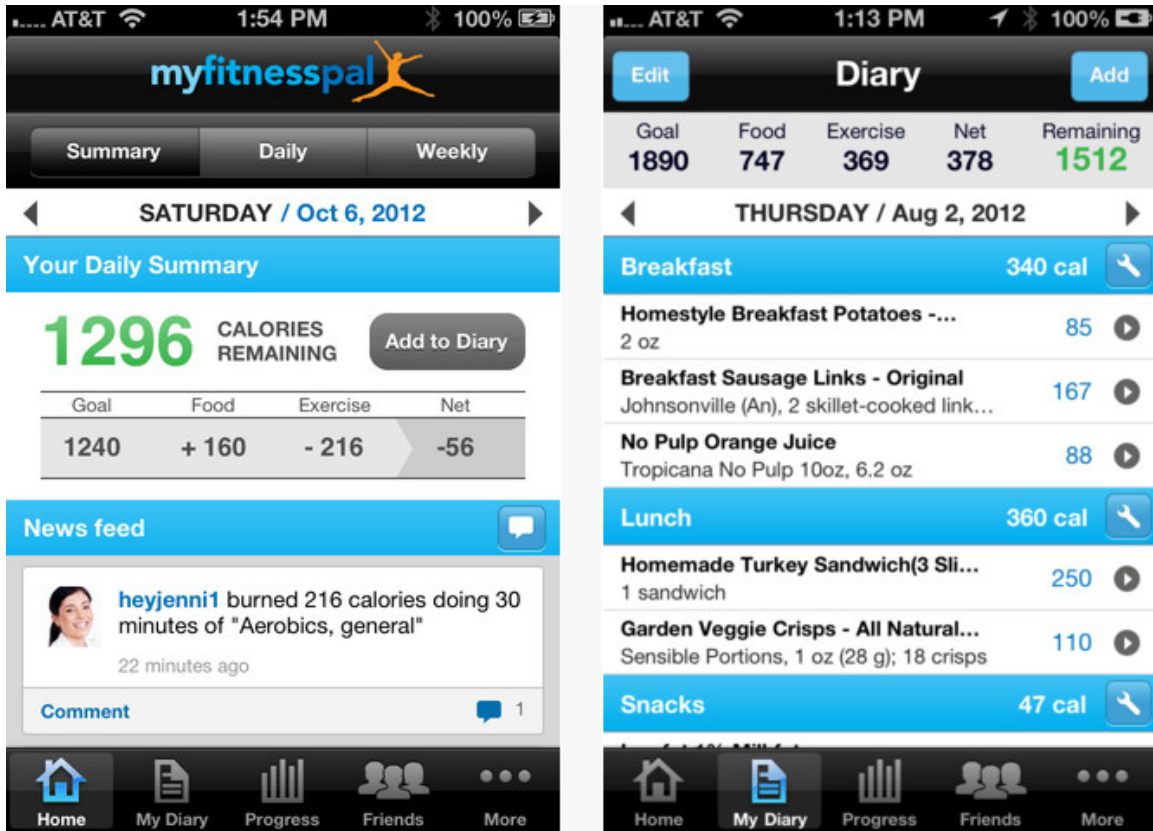


Figure 1-1: A screenshot of the MyFitnessPal mobile application.

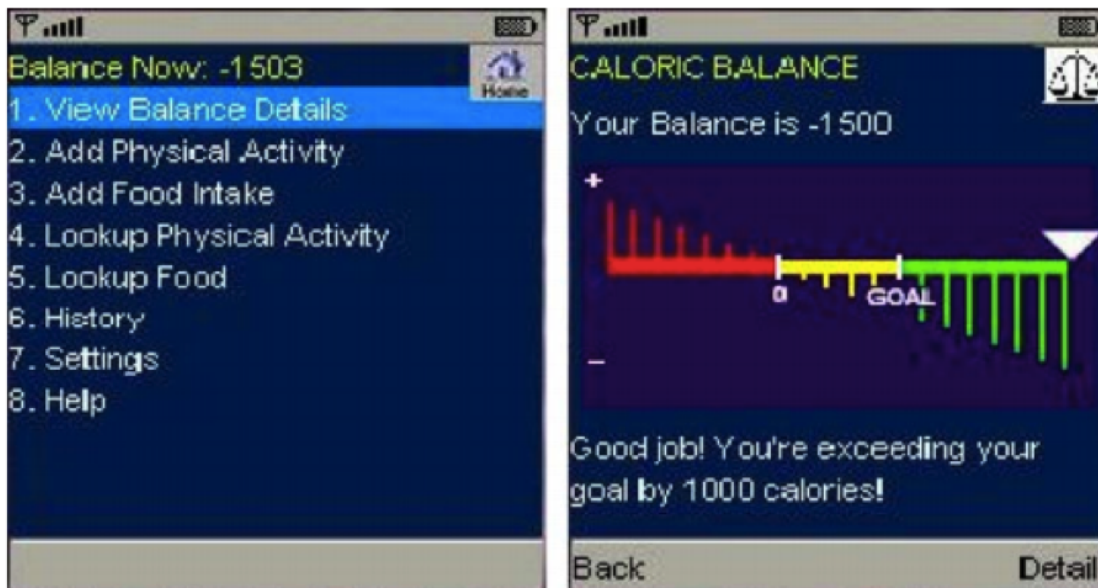
### 1.3.2 An Initial Prototype

So far, we have built a nutrition logging prototype whose current interface is shown in Figure 1-3. In this example, the user has said “I had a bowl of oatmeal followed by a banana and a glass of milk.” The display shows the output of a speech recognizer, along with color-coded semantic tags (e.g., quantity, brand, description, and food) associated with particular word sequences. The segmented food concepts are then shown in matrix form in a table along with potential matches to a nutritional database containing over 20,000 foods from the USDA and other sources.

The flow of the nutrition system is shown in Figure 1-4. After the user generates a meal description by typing or speaking (to a speech recognizer), the language understanding component processes the text, first by splitting the meal description into tokens (i.e., words or segments of words). Each meal description contains food and property tokens (e.g., brands, quantities, and descriptions), and each property is

associated with one (or more) of the foods. For example, “a bowl” is a quantity that could be associated with “cereal,” whereas “a cup” would be associated with “milk.” The language understanding component determines which tokens are foods and properties, and assigns properties to the foods which they describe (e.g., the quantity “2” is associated with the food “pancakes” in Figure 1-4). We applied conditional random field (CRF) models to the two language understanding tasks: semantic tagging (i.e., labeling tokens as foods or properties) and associating properties with foods.

In the remainder of this thesis, we begin by explaining the crowdsourcing methods we have developed and deployed on Amazon Mechanical Turk (AMT) for data collection and annotation [47]. Chapter 3 provides details on the conditional random field (CRF) models we have explored for language understanding, specifically the semantic tagging task, and reports experimental results. In Chapter 4, we investigate distributional semantics approaches to language understanding, and Chapter 5 applies CRFs and distributional semantics to the food-property association task. Finally, Chapter 6 presents an overview of the current system prototype, and Chapter 7 concludes and describes future work.

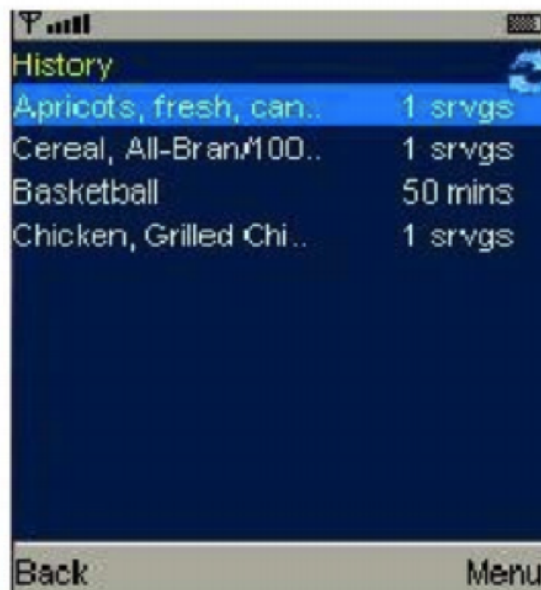


**a**

**b**



**c**



**d**

C. Tsai, G. Lee, F. Raab, G. Norman, T. Sohn, W. Griswold, and K. Patrick. Usability and feasibility of PmEB: A mobile phone application for monitoring real time caloric balance. *Mobile networks and applications*, 12(2-3):173-184,2007.

Figure 1-2: A screenshot of the PmEB mobile phone application where **a** is the main menu, **b** is the current caloric balance, **c** is the meal selection page, and **d** is the history.

## Record a nutrition log!



I had **a bowl** of **oatmeal** followed by **a banana** and **a glass** of **milk**




Food	Quantity	USDA Hits
 Oatmeal	Quantity: 1 <input type="text"/> cup	Cereals, oats, regular and quick, not fortified, dry, Calories: 307 Source: USDA • <a href="#">See more options</a>
 Banana	Quantity: 1 <input type="text"/> medium (7" to 7-7/8" long)	Bananas, raw, Calories: 105 Source: USDA • <a href="#">See more options</a>
 Milk	Quantity: 1 <input type="text"/> cup	Milk, whole, 3.25% milkfat, with added vitamin D, Calories: 149 Source: USDA • <a href="#">See more options</a>

Figure 1-3: A prototype of the nutrition system with an example food diary, where “oatmeal,” “banana,” and “milk” are identified as food items and “a bowl,” “a,” and “a glass” are quantities.

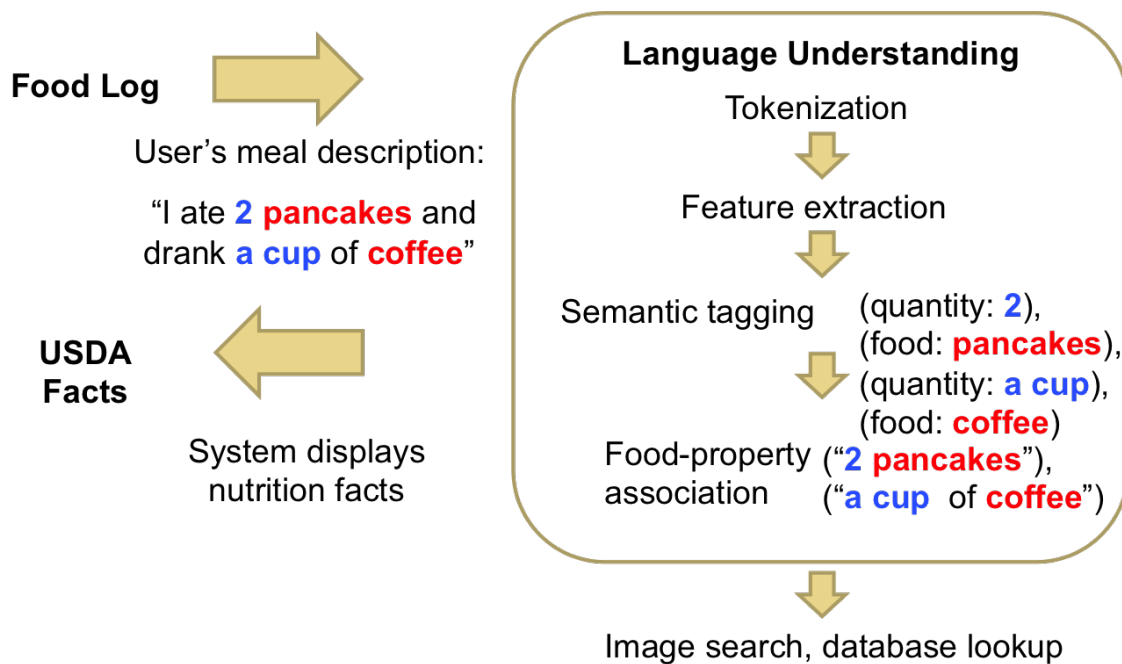


Figure 1-4: A diagram of the flow of the nutrition system.



# Chapter 2

## Crowdsourcing the Data Collection

Before training machine learning models to make predictions on new, unseen data, we must first provide the models with labeled (i.e., annotated) training data, as well as testing data for evaluating the models' performance. Since training our natural language processing models requires thousands of annotated data samples, it would be prohibitively time consuming for us to label all the data. Thus, we crowdsource data collection (e.g., on platforms such as Amazon Mechanical Turk) by paying people called "Turkers" to label small subsets of data. Since Turkers are not experts, we must implement techniques for creating data collection tasks that are simple enough for the average person to complete correctly. This chapter discusses the data collection subtasks we designed for training the language understanding models in the nutrition system, as well as challenges we encountered and our solutions.

### 2.1 Background and Related Work

There has been much prior work on crowdsourcing data collection and annotation, especially on Amazon Mechanical Turk (AMT). AMT data has been used for research ranging from linguistics and natural language processing (NLP) to computer vision, information retrieval, human computer interaction, and economics [9]. For example, crowdsourcing on AMT has been combined with NLP techniques for automatic poetry generation, since machines and humans excel at complementary tasks (in this case,

generating many possible phrases versus selecting meaningful, grammatically correct phrases) [10], as well as grammatical error correction for foreign language learners [59].

When designing AMT tasks, there are several factors that must be taken into consideration: settings, layout design (including task instructions), worker qualifications, and “gaming the system” prevention techniques. To illustrate the noisy nature of AMT data, Table 2.1 shows examples of Turker mistakes when labeling foods in meal descriptions. Settings include the number of workers per task, the reward amount for successfully completing a task, how long the task will be available to Turkers, and how long Turkers must wait before their work will be accepted or rejected; these settings affect how quickly and accurately Turkers complete tasks and need to be tuned to each data collection task. The layout of the task can significantly impact the quality of Turkers’ work, since the easier it is for Turkers to understand what they are expected to do, the more likely they are to produce high quality work. For additional quality control, properties can be set that require Turkers to be in a certain location (e.g., we require Turkers to be in the US so that they are more likely to speak English fluently) or have a rating above some threshold (e.g., we require Turkers to have had at least 80% of their previous work accepted). Finally, Turkers can be required to complete qualification tests prior to working on a task, both to select for Turkers who do quality work and to provide Turkers with practice problems where they can learn from their mistakes.

<b>Turker</b>	<b>Meal Description (Foods in Bold)</b>
1	For Breakfast I had Brown Sugar <b>Cinnamon Pop Tarts</b>
2	For Breakfast I had Brown Sugar Cinnamon <b>Pop Tarts</b>
3	For Breakfast I had <b>Brown Sugar Cinnamon Pop Tarts</b>
1	For lunch i had a Tostino’s <b>pepperoni Pizza</b>
2	For lunch i had a Tostino’s pepperoni <b>Pizza</b>
3	None
1	For dinner last night I had an Italian <b>sub</b> from a local sub shop
2	For dinner last night I had an <b>Italian sub</b> from a local sub shop
3	For dinner last night I had an Italian <b>sub</b> from a local <b>sub</b> shop

Table 2.1: Example AMT tasks (i.e., food labeling of meal descriptions) demonstrating inconsistencies in data due to gaming the system, mistakes, and confusion among Turkers.

Another challenge when collecting data on AMT is that many Turkers cheat in order to complete tasks as quickly as possible. For example, if a task contains 10 multiple-choice questions, they may try to select the first option from the list for every question in an effort to earn money more quickly. Prior work on AMT gaming the system prevention embedded a support vector machine classifier within a task for transcribing audio clips in order to automatically detect poor quality transcripts and alert Turkers to improve their transcription before enabling them to submit the task [38]. Other work has used a two-step collaboration process between translators and editors to improve AMT data quality for machine translation; in the first task, several Turkers translated a sentence from one language to another, and in the second task, native speakers of the second language edited the translations [82].

## 2.2 Nutrition Data Collection

We deployed three subtasks of experiments on AMT in order to crowdsource our data collection and annotation. Our goal was to collect meal descriptions, where each token was labeled (e.g., as a brand, quantity, etc.), and property tokens were assigned to food tokens (e.g., the quantity “bowl” was assigned to the food “cereal”). The first phase involved the collection of food diaries, where we prompted Turkers to write a description of a meal as they would imagine describing it orally. The diaries were then tokenized and used as input for the second phase, shown in Figure 2-1, where we asked users to label individual food items within the diaries. The third phase combined the meal descriptions with their food labels and prompted Turkers to label the concepts associated with a particular food item (see Figure 2-2).

## 2.3 Data Statistics

We collected and labeled 2,000 meal descriptions each of breakfast, lunch, dinner, and snacks on AMT, which we used to train our models. The data were tokenized on spaces, and if one of the resulting strings began or ended with a punctuation mark, we

**Instructions**

You will be presented with several descriptions of meals. For each description, label the individual food AND drink items. Select a phrase to label as a food or drink item by dragging from the first word to the last word, then select the category "Food" from the popup menu.

For example:

I had two eggs and 2 and a half strips of bacon this morning with two pieces of toast with margarine and Smucher 's strawberry jam on it . I had a cup of milk to drink with it .

Figure 2-1: The AMT task for labeling foods in a meal description.

<p><b>Categories &amp; Sample Phrases</b></p> <p><b>Brand</b> <i>Trader Joe's, Kellogg's, homemade...</i></p> <p><b>Quantity</b> <i>a cup, a large bowl, two [eggs]...</i></p> <p><b>Description</b> <i>black [coffee], nonfat [milk]...</i></p>	<p><b>Instructions</b></p> <p>You will be presented with several descriptions of meals. Within each description, one food item will be highlighted in red text. Categorize the words associated with that food item. Select a phrase to categorize by dragging from the first word to the last word, then select the relevant category from the popup menu.</p> <p><b>Please review all categories and sample phrases on the left before you begin!</b></p> <p>For example:</p> <p>I had a large bowl of Kellogg 's Frosted Flakes with about a cup of 2 % milk .</p>
--	---

Figure 2-2: The AMT task for labeling properties of foods.

further split the token on the punctuation (e.g., “She’s an MIT student.” ⇒ “She’s” “an” “MIT” “student” “.”); thus, commas and periods became separate tokens, but contractions and hyphenations retained their meaning. Five Turkers annotated each of the food labeling human intelligence tasks (HITs), and three Turkers annotated each of the property labeling HITs. A breakdown of the statistics for each meal are listed in Table 2.2. Labeling lunch and dinner was slightly more challenging than breakfast and snack, so Turkers earned more per HIT. Unfortunately, due to Turkers copying and pasting meal descriptions, not all 2,000 descriptions per meal were unique.

Initially, we used the AMT label for a token if at least four out of five Turkers labeled the token as a food item or if three out of five Turkers labeled the token as the same attribute. These thresholds were selected by comparing the performance of

Meal	HIT Price	Unique Data	Training Data	Testing Data
Breakfast	\$0.04	1,922	1,729	193
Lunch	\$0.05	1,963	1,766	197
Dinner	\$0.05	1,794	1,614	180
Snack	\$0.04	1,991	1,791	200

Table 2.2: Detailed information for each meal.

the resulting model trained on these labels, as shown in Table 2.3. However, it later became clear that as we collected more data, many food labels were missing, and thus the properties describing these foods were also missing, causing the resulting trained models to often miss entire food concepts. Thus, we decided to lower the threshold of Turkers for the food labeling task to one, since if even one Turker believed a token was a food, we counted it. However, we used a threshold of two out of three Turkers for the property labeling task because this improved the F1 score on the labeling task.

Every tenth query was added to the test set (for a total of 770 test queries), while all other queries were part of the training data (6,900 training queries in total). The histogram in Figure 2-3 shows that most food diaries contain two, three, or four foods. Turkers tend to have high agreement when labeling foods and quantities, but there are more conflicts among brands and descriptions.

Threshold	Mean F1	Variance	St. Dev.
1	78.75	2.21	1.49
2	84.05	2.38	1.54
3	<b>84.58</b>	2.81	1.68
4	83.74	1.01	1.01
5	76.80	2.78	1.67

Table 2.3: Labeling model’s 10-fold results (i.e., mean F1, variance, and standard deviation) for different thresholds of Turkers in the property labeling task.

## 2.4 Inter-Annotator Agreement

We measured the reliability of the data annotations by calculating the inter-annotator agreement among Turkers. Specifically, we calculated Fleiss’ kappa score (see Appendix B for more details). The kappa scores for the two labeling tasks are shown in

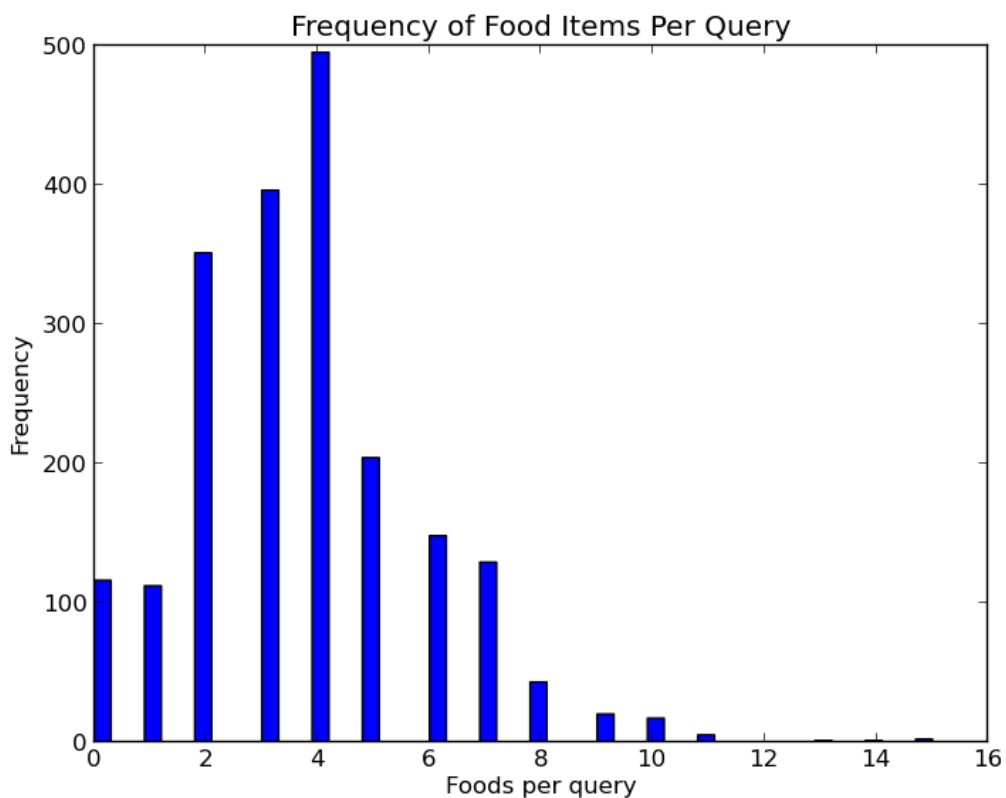


Figure 2-3: Frequency of food items per meal description.

Table 2.4. The score for the food labeling task is close to one and indicates substantial agreement. As expected, the score for the property labeling task is lower, since there were more categories and fewer annotations per task; in addition, distinguishing between brands and descriptions was challenging for Turkers. However, the score still indicates a fair amount of agreement [77].

AMT Task	Kappa
Food Labeling	0.766
Property Labeling	0.409

Table 2.4: Kappa scores for food and property labeling tasks.

## 2.5 Challenges and Solutions

After the initial round of data collection, we noted that Turkers were producing food diaries and annotations of lower quality than we desired. In order to improve the data, we required the HITs to pass a series of checks before submission. Our algorithms address several common trends we identified among low-quality annotations.

### 2.5.1 Meal Diary Submission Prefiltering

Often, a single food diary was submitted numerous times, resulting in semantically identical data. Our solution was to generate a corpus of submitted responses and disallow repeat submissions. In addition, low-quality descriptions often contained few words, so we required diaries to consist of at least four words. Another attempt to outwit the checker involved using repetition within a diary (e.g., “a a a a”). Our solution to this challenge was to prevent diaries from containing more than 60% repetition. Finally, due to extensive spelling errors, we required at least 60% English for submission.

### 2.5.2 Property Labeling Prefiltering

In addition to the checks we implemented for preventing Turkers from gaming the system when submitting diaries, 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 tokens were missing.

In order to automatically generate the hidden food and property labels, we used a trie matching algorithm 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, one node might contain the token “orange,” and one of its children nodes might contain the token “juice.” Then, the matching algorithm would find every matching entry from the USDA trie that is present in a meal description. For example, the meal description “I had a glass of orange juice” would yield three trie matches: “orange,” “juice,” and “orange juice.”

Since the USDA food entries often contain only the singular or plural form of a food token (e.g., “egg” but not “eggs”), we incorporated plural handling into the trie matching. We used the Evo Inflector library’s implementation of Conway’s English pluralization algorithm to convert tokens from singular to plural [14]. In addition, we tried using the Porter stemmer [62] to extract the stems of words, which provide the singular forms of plural words; however, the stems were often inaccurate or not foods (e.g., the stem of “topping” is “top,” and the singular of “greens” is “green,” neither of which are foods), so we did not use the stemmer in the deployed AMT tasks. To incorporate pluralization into the trie, for every food item in the USDA lexicon, we added the plural form as another pattern in the trie.

Sometimes this check introduced new problems, since not every token matching a USDA food word should be labeled. For example, in the meal description “Rather than drinking my usual coffee this morning, I had a cup of tea,” the token “coffee” was not actually consumed and thus should not be labeled; however, this requires a deeper understanding of the context, which we have not yet implemented. We manually fixed these tasks after Turkers reported difficulty submitting the task.

### **2.5.3 Inaccurate Food Labels**

Turkers occasionally mislabeled food items, which caused confusion in the final round of property labeling tasks. For example, in the meal description “I had a blueberry yogurt,” Turkers labeled both “blueberry” and “yogurt” as foods, even though “blueberry” is actually a description in this context. Thus, in the property labeling task, we added a button which enabled Turkers to specify that the token they were labeling properties of was not a food. This introduced a problem where Turkers clicked the



“Not a food!” button for drinks; we fixed this by modifying the instructions to clarify that drinks are foods.

In addition, we addressed the issue of mistakes in the food labeling round by combining adjacent food tokens into single food items if they were not comma-separated, and otherwise leaving them as separate foods. This combined tokens such as “blueberry yogurt” into one food and divided comma-separated phrases such as “veggies, chicken” into two. We experimented with the Stanford part-of-speech (POS) tagger [73] to identify which tokens were foods by keeping only the nouns; however, even though removing adjectives eliminated colors, it kept some tokens that were descriptions, not foods (e.g., “**grape** jam” and “**wheat** bread”).

#### 2.5.4 Ongoing Data Collection

We are currently collecting and integrating more data into our existing set of training and testing data for the labeling and food-property association models. Since Turkers are able to evade each new anti-gaming-the-system mechanism we implement with more subtle techniques (e.g., Turkers can evade the trie matching check by simply labeling the entire meal description), we plan to explore a data annotation refinement mechanism [38] where we will prevent submission when the number of labeled tokens is much lower or greater than what our model predicts.

In addition, we may ask Turkers to respond to food diaries (e.g., “That sounds delicious!”) and train a classifier on these responses to automatically give feedback to Turkers on their meal descriptions in real-time. We have already launched a new version of the initial meal diary collection task, shown in Figure 2-4, where we incorporated the deployed nutrition system rather than a simple text box. Turkers completed 500 HITs, in which we asked them to record any four meals (either spoken or written), and asked for their feedback on using the system. In this manner, we explored how real users interact with the system and gained useful feedback, such as the preference for users to have the ability to modify the semantic tags and recognized speech. This task yielded an additional 2,000 meals which we have incorporated into the data set for a total of 10,000 annotated meal descriptions. As before, we required

that each description was unique, descriptive, non-repetitive, and spelled correctly.

## Food Log

**IMPORTANT: You must ACCEPT the HIT before you can submit the answers.**

**Instructions**

Please record **four** different meals (e.g. what you ate for breakfast, lunch, dinner, or snack today or yesterday) using as much detail and accuracy as possible. Be creative--we will not accept repeat answers. Try to include as much additional information as you remember, such as brand names and quantities. Please press the record button below to describe your meal orally, or type in the textbox. It's more fun to record your meal verbally, but it requires using Chrome.

**Examples**

I had a boiled egg, a Thomas's english muffin, and an ounce of organic butter.

For lunch I ate a plate of spaghetti with a spinach salad and feta cheese.

**\*\*\*Note: you must record each meal one at a time, pressing Enter or stop recording at the end of each.\*\*\***

**System:** Hi! Please record or type your meal **one at a time** (hit 'Enter' when done):

**You:**

(Optional) Please let us know how we can make this HIT better.

Figure 2-4: AMT task to collect unique, descriptive meals with the deployed system.

Finally, we plan to address the limitation of the current approach for labeling the data, in which adjacent meal description tokens labeled as foods are combined into a single food item, with the last token representing the entire food phrase. This data representation results in poor performance when searching the nutrition database for a single-word food that in actuality should be a multi-word phrase. For example, “ice cream” is labeled as a description followed by a food, but since the word “cream” is a different food than “ice cream,” the database lookup will yield inaccurate results. A simple solution is to augment the property labeling task so that it can handle food items with multiple tokens, rather than only a single token.

# Chapter 3

## Conditional Random Fields (CRFs) for Semantic Tagging

The language understanding component of the nutrition system that we have implemented has two phases: semantically labeling the food concepts and properties in a meal description, and assigning attributes to the correct food items. In the next two chapters, we will first focus on the semantic tagging experiments and results. In this chapter, we will discuss the baseline method using CRFs with conventional linguistic features, and we will compare the standard CRF to its variation, the semi-CRF. In Chapter 4, we will take our best baseline CRF system and augment it with distributional semantics features. We will then extend our work to cases in which there is a limited vocabulary, and observe how the performance improves as we increase the amount of training data.

### 3.1 Background and Related Work

In a semantic tagging task, also referred to as slot filling, meaning is extracted from a user’s query by assigning values to pre-defined slots. For example, in our problem, the slot concepts are foods, brands, quantities, and descriptions; the associated values are words pulled directly from the query, such as “cereal” for the food slot. Several different approaches have been used in the past for semantic tagging, but here we will

explore two in particular: conditional random fields (CRFs) and semi-Markov CRFs.

### 3.1.1 Semantic Tagging

The CRF is a popular method for sequence modeling tasks such as semantic tagging. In the past, CRFs have been applied to many NLP problems, including Chinese word segmentation (i.e., determining whether each character is the beginning of a word or not) [87], sentence parsing [18], and determining string similarity [45]. CRFs have been a popular approach in other fields as well. For example, in computer vision, CRFs have been used for labeling images [30] in addition to foreground and shadow segmentation in images [79]. Prior research has shown the applicability of CRFs to the semantic tagging problem in particular [29], as well as the benefit of using semi-CRFs for intent detection [39] and semantic labeling of user queries [42].

Other recent work on language understanding has applied frame-semantic parsers [13], triangular CRFs [81], and neural networks [48, 16, 66] to the semantic tagging problem. The authors in [13] proposed a probabilistic frame-semantic parser for automatically inducing and filling semantic slots in an unsupervised manner. In this approach, a semantic parser trained on the linguistic resource FrameNet was used to extract frames (corresponding to slot candidates) and lexical units (corresponding to the values that fill the slots). They found that slots generated by their model aligned well with those created by domain experts.

Due to the increasing popularity of neural networks, among both the speech and language communities, several types of neural networks have been investigated for slot filling. In the work in [81], a convolutional neural network-based triangular CRF jointly modeled both the user intent (i.e., the goal of the user, given the query) as well as slot filling, achieving gains of 0.9-2.1% over approaches that independently model slot filling. The authors in [48] explored using a recurrent neural network (RNN) and found that both the Elman-type and Jordan-type RNNs outperformed the CRF baseline, but the bi-directional Jordan-type network worked best. Deep belief networks were employed in both [16] and [66], outperforming CRFs on the standard Airline Travel Information System (ATIS) test set, and outperforming maximum

entropy and boosting methods in a call routing domain.

## 3.2 Classifiers

For the first language understanding task in the system (i.e., labeling each token in a meal description as a food, quantity, brand or description), we applied a standard conditional random field (CRF) baseline, which we compared to a semi-Markov conditional random field (semi-CRF).

### 3.2.1 CRFs

Conditional random fields (CRFs) are useful models for natural language processing tasks such as semantic tagging and slot filling that involve sequential classification. In such a problem, we wish to predict a vector of output labels  $\mathbf{y} = \{y_0, y_1, \dots, y_T\}$  corresponding to a set of input feature vectors  $\mathbf{x} = \{\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_T\}$ . Due to complex dependencies, this multivariate prediction problem is challenging. We can use graphical models to represent a complex distribution over many variables more easily. The structure of the graphical model determines how the probability distribution factorizes, based on a set of conditional independence assumptions [71].

In the past, generative models, such as the naive Bayes classifier and hidden Markov models (HMMs), were popular. They describe how to “generate” values for features given the label. However, since they model a joint probability distribution  $p(\mathbf{y}, \mathbf{x})$ , these models can become intractable when there are complex dependencies. The CRF takes a discriminative approach, where the conditional distribution  $p(\mathbf{y}|\mathbf{x})$  is modeled directly. The linear-chain CRF has the form

$$Pr(\mathbf{y}|\mathbf{x}, \theta) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^T \exp\left\{\sum_{k=1}^K \theta_k f_k(y_t, y_{t-1}, \mathbf{x}_t)\right\} \quad (3.1)$$

where  $\theta_k$  is a weight parameter for feature function  $f_k(y_t, y_{t-1}, \mathbf{x}_t)$ , and  $Z(\mathbf{x})$  is the normalization factor  $\sum_{\mathbf{y}} \prod_{t=1}^T \exp\{\sum_{k=1}^K \theta_k f_k(y_t, y_{t-1}, \mathbf{x}_t)\}$ . More details on training CRFs can be found in Appendix A.

### 3.2.2 Semi-CRFs

To accomplish the first semantic tagging task (and to compare against the CRF baseline method), we utilized a variation of the standard CRF model, a semi-Markov conditional random field (semi-CRF). Rather than assigning an output label to each token, a semi-CRF assigns an output label to token *segments* [65].

Semi-CRFs can be viewed as the conditional, or discriminative, version of generative semi-Markov chain models, in which there is a segment of tokens from  $i$  to  $d_i$  where the behavior of the system may not be Markovian. The state  $s_i$  at token  $i$  persists until token  $d_i$ , at which point there is a transition to a new state  $s'$  which only depends on state  $s_i$ . Semi-CRFs were developed because it seemed as though they would perform better on segmenting tasks such as named entity recognition and noun-phrase (NP) chunking. In our case, the semi-CRF is a reasonable choice for a semantic tagging model because food items and properties are sequences of tokens. For example, a quantity might be the segment “a cup.”

Rather than modeling the conditional probability of output  $\mathbf{y}$  given input  $\mathbf{x}$ ,  $Pr(\mathbf{y}|\mathbf{x})$ , a semi-CRF models the probability of a segmentation  $\mathbf{s}$  given  $\mathbf{x}$ ,  $Pr(\mathbf{s}|\mathbf{x})$ , where each segment  $s_i \in \mathbf{s}$  consists of a start position  $t_j$ , an end position  $u_j$ , and a label  $y_j$ :  $s_i = \langle t_j, u_j, y_j \rangle$ . For example, the quantity segment “a cup,” appearing in the food log “I had a cup of milk,” would be represented as  $\langle 2, 3, Quantity \rangle$ , assuming zero-indexing, since “a” has index 2 and “cup” has index 3. Also, rather than using local feature functions  $\mathbf{f}$ , which correspond to output labels of individual elements in  $\mathbf{x}$ , semi-CRFs use segment feature functions which correspond to output segments of  $\mathbf{x}$ . We define each segment feature function  $g^k(j, \mathbf{x}, \mathbf{s}) = g^k(y_j, y_{j-1}, \mathbf{x}, t_j, u_j)$  according to the Markov assumption that a segment  $s_j$  depends only on the previous segment  $s_{j-1}$ . Then, if we let  $\mathbf{G}(\mathbf{x}, \mathbf{s}) = \sum_{j=1}^{|\mathbf{s}|} \mathbf{g}(j, \mathbf{x}, \mathbf{s})$ , a semi-CRF estimates the distribution

$$Pr(\mathbf{s}|\mathbf{x}, \mathbf{W}) = \frac{1}{Z(\mathbf{x})} \exp(\mathbf{W} \cdot \mathbf{G}(\mathbf{x}, \mathbf{s})) \quad (3.2)$$

where  $\mathbf{W}$  is a weight vector for  $\mathbf{G}$  and  $Z(\mathbf{x})$  is the normalization factor  $\sum \mathbf{s}' \exp(\mathbf{W} \cdot \mathbf{G}(\mathbf{x}, \mathbf{s}'))$ .

In addition, semi-CRFs provide the benefit of high-order CRFs without the associated computational cost.

### 3.3 Features

The baseline features we selected for the food-property labeling task include n-grams (unigrams, bigrams, trigrams, and 4-grams), lexicon features (e.g., the segment matches an item in a lexicon of USDA food products), and part-of-speech (POS) tags. We used Stanford’s open source tagger to generate POS tags [15].

The n-gram features are commonly used as baseline features for NLP classifiers. A unigram feature checks whether the token is an exact match of a unigram seen in the training data (e.g., whether the current token is the unigram “cup”), bigram features check two consecutive tokens, trigrams consist of three consecutive tokens, and so on. In the semi-CRF, the frequency cutoff is one, which means the n-gram must appear at least twice (in order to avoid learning many sparse features for n-grams that only appear once in the whole training data set). In addition, for the semi-CRF, n-grams are contained within token segments. For example, the segment “shredded cheddar cheese” would have two bigrams: “shredded cheddar” and “cheddar cheese.”

Lexicon features indicate whether the current token (in the case of the standard CRF) or the current segment (for the semi-CRF) appear within a list of foods or brands. Table 3.1 shows ten food items in the lexicon obtained from the USDA National Nutrient Database for Standard Reference (containing 2,636 foods total), as well as ten brand names from both the USDA and Wikipedia (containing 2,174 brands total). We would expect these features to boost food and brand detection. One important distinction between the CRF and the semi-CRF with respect to the lexicon features is that the semi-CRF can easily compare segments of tokens to foods or brands that consist of multiple words, whereas it is more challenging to match consecutive tokens against multi-word entries with the standard CRF.

The POS tag features indicate the part-of-speech of the current token (or segment of tokens, in the semi-CRF). For example, foods are often nouns and would be assigned

<b>Foods</b>	<b>Brands</b>
mackerel	1-2-3 Gluten Free
apple-raspberry	3 Musketeers
carrots	5-hour Energy
figs	7 Up
fritter	7 Whole Grain
bacon	A&W
millet	All-Bran
sausage sandwich	Almond Dream
nachos with cheese	Altoids
soybean curd cheese	Ancient Quinoa Harvest

Table 3.1: Ten items from the food and brand lexicons.

the POS tag “NN” for singular nouns, “NNS” for plural nouns, “NNP” for singular proper nouns, and “NNPS” for plural proper nouns. Descriptions and brands are more likely to be adjectives, and as such would be assigned the POS tag “JJ.” Table 3.2 shows 30 different n-gram and POS tag features learned by the semi-CRF on the training data.

<b>Unigrams</b>	<b>Bigrams</b>	<b>POS Tags</b>
16	I_drunk	JJ (Adjective)
oz	I_ate	VBZ (Verb, 3rd person singular present)
bottle	6_oz	VBG (Verb, gerund)
of	8oz_of	VBP (Verb, non-3rd person singular present)
WinCo	1_orange	CD (Cardinal number)
Purified	had_16	IN (Preposition)
Drinking	with_some	DT (Determiner)
Water	2_tablespoons	NNS (Noun, plural)
and	banana_and	NN (Noun, singular)
Granny	of_Blue	NNP (Proper noun, singular)

Table 3.2: Unigram, bigram, and POS features (10 each) learned by the semi-CRF.

### 3.4 Results

To evaluate our methods for labeling and associating foods and properties, we split the AMT data into training and test sets (with 9,000 and 1,000 meal descriptions each) and computed the precision (i.e., the fraction of predicted labels that were correct),



recall (i.e., the fraction of gold standard labels that were predicted), and F1 (i.e., the harmonic mean of precision and recall) scores for each approach (see Appendix B for more details). We then measured statistical significance in performance differences among several approaches using McNemar’s significant test, as described in detail in Appendix B.

We begin by presenting results for the standard CRF (we used the Python CRF-suite implementation [56]), and then compare its results to that of the semi-CRF.

### 3.4.1 Labeling Results

In order to select the best set of baseline features for the CRF and semi-CRF, we ran cross-validation experiments with three different sets of features, as shown in Table 3.3. We selected the combination of n-grams, food and brand lexicon features, and POS tags for our final baseline feature set because they yielded the highest average F1 score across all label categories for the CRF and semi-CRF with cross-validation.

Features	CRF Average F1	Semi-CRF Average F1
N-grams	90.0	84.6
+ POS tags	90.1	85.1
+ POS tags + Lexicon	<b>90.4</b>	<b>85.9</b>

Table 3.3: CRF and semi-CRF cross-validation results for three different feature sets. The highest average F1 score is shown in bold.

We measured the performance of the CRF and semi-CRF trained on our selected feature set using the test data, as shown in Tables 3.4 and 3.5 respectively. We evaluated the semi-CRF at the concept level as opposed to the token level so that a concept is considered correct if the semi-CRF labels the concept correctly, even if a word within the concept is labeled incorrectly (i.e., the current token’s label is the same as either the previous or the next token’s label, and both the previous and next tokens’ labels are correct). For example, “a bowl” would be counted as correct even if “a” is labeled incorrectly, as long as “bowl” is labeled correctly.

We compared the CRF and the semi-CRF performance token by token, testing for significance using McNemar’s significance test. As shown in the 2x2 matrix in

Label	Precision	Recall	F1
Food	94.2	94.4	94.3
Brand	84.0	97.6	80.7
Quantity	88.9	95.0	91.8
Description	87.3	89.2	88.2
Other	96.1	94.0	95.0
Overall	90.1	94.0	90.0

Table 3.4: CRF token-level performance on test data.

Label	Precision	Recall	F1
Food	94.4	78.4	85.7
Brand	74.6	67.3	70.8
Quantity	90.2	90.0	90.1
Description	77.2	70.5	73.7
Other	82.4	92.3	87.1
Overall	84.6	77.0	80.6

Table 3.5: Semi-CRF concept-level performance on test data.

Table 3.6, we found that the difference was statistically significant, where  $p < 0.01$  ( $p = 2.64 \times 10^{-401}$ ); thus, although the semi-CRF handles segmenting in a more intuitive manner than do standard CRFs, we chose to use the CRF because the performance was significantly better than that of the semi-CRF.

	Semi-CRF Correct	Semi-CRF Incorrect
<b>CRF Correct</b>	$n_{00} = 30,830$	$n_{01} = 2,584$
<b>CRF Incorrect</b>	$n_{10} = 380$	$n_{11} = 2,300$

Table 3.6: CRF versus semi-CRF errors, where  $p < 0.01$  ( $p = 2.64 \times 10^{-401}$ ) according to McNemar’s significance test.  $n_{00}$  is the number of tokens labeled correctly by both,  $n_{11}$  is the number labeled incorrectly by both,  $n_{10}$  is the number of tokens labeled correctly by the semi-CRF but not the standard CRF, and  $n_{01}$  vice versa.

### 3.4.2 Error Analysis

In Tables 3.7 and 3.8, we see 10 of the most common errors made by the CRF and semi-CRF (excluding punctuation and function words). The semi-CRF appears to miss many obvious foods, such as “eggs,” “onions,” “pizza,” and “bread.” These errors could be due to a tokenization mismatch between the tokens in the AMT data

and the tokens generated by the POS tagger within the semi-CRF. Table 3.9 shows an example meal description from the test data set and the corresponding CRF and semi-CRF semantic tags. Both models tag most of the tokens correctly; however, the semi-CRF incorrectly labels “toast using Aunt Millies” as a description, whereas the CRF labels this phrase correctly. They both mistakenly tag “Oreo” as a food instead of a brand.

Token	Predicted Label	AMT Label	Frequency
large	Quantity	Description	8
small	Quantity	Description	6
medium	Quantity	Description	6
cheese	Description	Food	5
brand	Other	Brand	5
no	Quantity	Other	4
butter	Food	Description	4
glass	Quantity	Other	4
wheat	Description	Food	3
salt	Food	Other	3

Table 3.7: 10 common CRF errors on the token level, along with the frequency of the error on the test data.

Token	Predicted Label	AMT Label	Frequency
some	Other	Quantity	23
2	Other	Quantity	18
eggs	Other	Food	17
onions	Other	Food	14
drink	Food	Other	13
sauce	Other	Food	11
pizza	Other	Food	11
scrambled	Other	Description	10
bread	Other	Food	10
tomatoes	Quantity	Food	10

Table 3.8: 10 common semi-CRF errors on the token level, along with the frequency of the error on the test data.

More examples of the specific types of errors made by the CRF and semi-CRF are shown in Tables 3.10 and 3.11. Foods, brands, and quantities are often mistakenly labeled as other or as a description. Descriptions and brands are often swapped

Token	CRF Label	Semi-CRF Label	AMT Label
I	Other	Other	Other
had	Other	Other	Other
three	Quantity	Quantity	Quantity
pieces	Quantity	Quantity	Quantity
of	Other	Other	Other
toast	Food	<b>Description</b>	Food
using	Other	<b>Description</b>	Other
Aunt	Brand	<b>Description</b>	Brand
Millies	Brand	<b>Description</b>	Brand
Cracked	Description	Description	Description
Wheat	Description	Description	Description
Bread	Food	Food	Food
.	Other	Other	Other
I	Other	Other	Other
had	Other	Other	Other
10	Quantity	Quantity	Quantity
Double	Description	Description	Description
Stuffed	Description	Description	Description
Oreo	<b>Food</b>	<b>Food</b>	Brand
cookies	Food	Food	Food

Table 3.9: An example meal description comparing the CRF and semi-CRF semantic tag predictions, with errors shown in bold.

or omitted altogether. From these errors, as well as the error statistics shown in Table 3.12, we infer that both models identify foods, quantities, and other much more easily than brands or descriptions, which reflects the high Turker disagreement for the brand and description categories. In addition, some brands may not be seen in training data (i.e., out-of-vocabulary words). To address these issues, we may need to revise the AMT tasks to enable Turkers to more easily differentiate between brands and descriptions. We have already begun to address this by creating a brand lexicon using the nutritional database and Wikipedia. In the future, the nutrition system may learn new brands or foods through dialogue with a user.

AMT Label	Predicted Label	Frequency
Food	Description	103
Food	Other	55
Brand	Description	117
Brand	Other	55
Quantity	Other	108
Description	Other	136
Description	Food	87
Description	Brand	61
Other	Quantity	301
Other	Description	135

Table 3.10: 10 most common CRF errors per label category.

AMT Label	Predicted Label	Frequency
Food	Other	802
Food	Description	327
Brand	Description	165
Brand	Other	154
Quantity	Other	346
Description	Description	589
Description	Brand	156
Other	Quantity	266
Other	Description	233
Other	Food	214

Table 3.11: 10 most common semi-CRF errors per label category.

### 3.4.3 Effects of Noisy Data

In order to identify the effects of noisy AMT data on the performance of the semi-CRF, we compared the number of mistakes made on a random sample of meal descriptions (i.e., 38 out of 771) labeled with AMT to the number of mistakes on the same sample, but using more carefully hand-labeled gold standard tags. As shown in Table 3.13, we found that the F1 score increased by 7%, from 82.6 to 88.6, when using the hand-labeled data rather than AMT, which indicates that many of the semi-CRF errors are actually due to inaccurate Turker labels. Some examples of Turker labeling mistakes are shown in Table 3.14.

Label	Num CRF Missed	Num semi-CRF Missed	Num Total
Food	178 (6%)	1,274 (40%)	3,155
Brand	202 (22%)	390 (43%)	902
Quantity	141 (5%)	430 (15%)	2,838
Description	315 (11%)	1,006 (35%)	2,917
Other	554 (6%)	774 (8%)	9,179

Table 3.12: Number of tokens missed by the CRF and semi-CRF for each label category.

Labels	Precision	Recall	F1
AMT	83.0	82.3	82.6
Expert	<b>88.9</b>	<b>88.3</b>	<b>88.6</b>

Table 3.13: Semi-CRF semantic tagging performance on 38 randomly selected meal descriptions, using both AMT labels and expert hand labels.

### 3.5 Conclusion

In this chapter, we have presented the first task of the system’s language understanding component: semantic tagging of foods and properties. We explored a baseline approach of CRFs with conventional linguistic features. We compared the performance of a standard CRF to that of a semi-CRF, which outputs segments (rather than individual tokens) of foods and attributes. Since the performance of the CRF was significantly better than that of the semi-CRF, we will only use the standard CRF in the following chapter.

<b>Token</b>	<b>AMT Label</b>	<b>Expert Label</b>
Whole Foods	Other	Brand
some	Other	Quantity
tangerine	Food	Description
a	Other	Quantity
homemade	Other	Description
graham	Food	Description

Table 3.14: Examples of Turker labeling mistakes.





# Chapter 4

## Distributional Semantics for Semantic Tagging

According to the theory of distributional semantics [68, 52], words or phrases with similar meaning will be located in nearby regions within the continuous vector space. Thus, the vector representation of words can improve machine learning algorithms by enabling models to determine whether two words or phrases have similar semantics. Recent work [3] has shown that using word embeddings learned from neural networks as classifier features improves performance in many natural language processing tasks.

In this chapter, we will explore the application of distributional semantics to the task of semantic labeling. Specifically, we will investigate three approaches for incorporating word embedding features into a CRF model for semantic tagging: using vector values directly (i.e., dense embeddings), measuring the cosine similarity between tokens and “prototypes” (i.e., words most representative of a category, such as “bread” for the food category), and assigning embeddings to clusters. We will then examine the effects of unknown words in situations where there is a limited vocabulary, and explore methods for predicting vectors. Finally, we will analyze the effects of varying the amount of training data. We will begin with the best baseline CRF model found in Chapter 3 (i.e., CRFsuite with n-gram, POS tag, and lexicon features), since the CRF performed significantly better than the semi-CRF.

## 4.1 Background and Related Work

One application in which word vectors are useful is in analogy tasks [51], such as predicting “king is to man as queen is to  $\mathbf{x}$ .” The correct answer, “woman,” is determined by selecting the word in the vocabulary with the largest cosine similarity to the vector  $\mathbf{x}$ , where  $\mathbf{x} = \text{vec}(\text{“man”}) - \text{vec}(\text{“king”}) + \text{vec}(\text{“queen”})$ , and the cosine distance is defined as

$$\text{dist}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}. \quad (4.1)$$

Other applications of word embeddings include using them as the sole features for part-of-speech taggers in multiple languages [1], tailoring them for dependency parsing [2], using them to enrich spoken user queries within conversational interfaces on smartphones [11], question answering [34], and semantic parsing [5]. Recent work has also incorporated distributional semantics into spoken language understanding tasks such as slot filling and semantic tagging. For example, in the work in [8], embeddings were enriched with context (i.e., with entities and relations between subjects and objects from a knowledge graph) before being used as CRF features for semantic tagging in the movie domain. The authors in [12] induced semantic slots from unlabeled speech data by augmenting a frame semantic parser with word embeddings, and [26] jointly trained a recursive neural network for domain detection, intent determination, and slot filling; each word (i.e., leaf node), as well as each internal node in the tree, was associated with a continuous vector representation.

## 4.2 Generating Word Vectors

There are several different approaches to generating word vectors. A popular method is through learning weights in a feed-forward or recurrent neural network language model and saving the weights as vectors after the network is fully trained. This is the approach taken by Mikolov in the skip-gram [50] and continuous bag-of-words [49] models, which we used in this work. An alternative method, which we did not in-

investigate, is that of the GloVe model [61]: a global log-bilinear regression model. Finally, the multiple-sense skip-gram model (MSSG) [54] extends Mikolov’s original skip-gram model by incorporating multiple meanings, or senses, of words. Although we have not yet explored this model, we believe the MSSG is an interesting direction for future work.

### 4.2.1 Skip-gram Model

One commonly used implementation of the neural network approach to learning embeddings is Mikolov’s skip-gram model [50], implemented and released publicly as the word2vec toolkit<sup>1</sup>, which learns word vector representations that best predict the context surrounding the word in a sentence or document. Given training words  $w_1, w_2, \dots, w_T$ , the objective is to maximize the average log probability

$$\frac{1}{T} \sum_{t=1}^T \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j} | w_t) \quad (4.2)$$

where  $c$  is the size of the context window surrounding the center word  $w_t$ . The probability  $p(w_{t+j} | w_t)$  is defined using the softmax function

$$p(w_O | w_I) = \frac{\exp(v_{w_O}^\top v_{w_I})}{\sum_{w'_O=1}^W \exp(v_{w'_O}^\top v_{w_I})} \quad (4.3)$$

where  $v_{w_I}$  and  $v_{w_O}$  are the input and output vectors of  $w$ , and  $W$  is the vocabulary size.

Unfortunately, this is computationally expensive due to the summation over the entire vocabulary in the denominator. Therefore, we can apply approximation algorithms such as hierarchical softmax or negative sampling. The hierarchical softmax reduces the computation over  $W$  output nodes to only  $\log(W)$  nodes by using a binary tree representation, where there are  $W$  leaves and each word  $w$  in the vocabulary can be reached by a path from the root to that leaf node. If we let  $n(w, j)$  be the  $j$ -th node on the path from the root to word  $w$ ,  $L(w)$  be the path length,  $\text{ch}(n)$  be an

---

<sup>1</sup><https://code.google.com/p/word2vec/>

arbitrary fixed child of  $n$ ,  $n(w, 1)$  be the root,  $n(w, L(w))$  be the leaf  $w$ , and  $\llbracket x \rrbracket$  be 1 if  $x$  is true and -1 otherwise, then the hierarchical softmax defines  $p(w_O|w_I)$  as

$$p(w_O|w_I) = \prod_{j=1}^{L(w_O)-1} \sigma(\llbracket n(w_O, j+1) = \text{ch}(n(w_O, j)) \rrbracket) \cdot v_{n(w_O, j)}^\top v_{w_I} \quad (4.4)$$

where  $\sigma(x) = 1/(1 + \exp(-x))$ . The hierarchical softmax is computed by a random walk over each inner node on the path from the root to the leaf word  $w_O$ . At each step (i.e., each term of the product), a dot product is taken between  $v_{w_I}$  and the current tree node  $v_{n(w, j)}$ , with a weight of 1 if the node’s arbitrarily chosen child  $\text{ch}(n(w, j))$  matches the next node on the path to  $w_O$ , and a weight of  $-1$  otherwise. Since the maximum path length of a binary tree is  $\log(W)$ , the computational cost is proportional to  $\log(W)$ .

An alternative method to the hierarchical softmax is negative sampling, in which we distinguish between the target word  $w_O$  and  $k$  negative samples from the noise distribution  $P_n(w)$  using logistic regression, resulting in the objective

$$\log \sigma(v_{w_O}^\top v_{w_I}) + \sum_{i=1}^k \mathbb{E}_{w_i \sim P_n(w)} [\log \sigma(-v_{w_i}^\top v_{w_I})], \quad (4.5)$$

which is used as a replacement for every  $\log P(w_O|w_I)$  term in the skip-gram objective.

Since frequent words such as “the” co-occur with many words, they often provide less information than rare words. Therefore, we can apply a subsampling approach in order to avoid the imbalance between rare and frequent words. We discard each word  $w_i$  in the training data with probability

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}} \quad (4.6)$$

where  $f(w_i)$  is the frequency with which word  $w_i$  appears in the data, and  $t$  is a threshold that can be tuned on a development set. In our experiments, we trained the vectors with the continuous bag-of-words (CBOW) approach, which predicts the current word based on the context [49].

### 4.2.2 Multiple-sense Embeddings Model

The multiple-sense skip-gram (MSSG) [54] was built as an extension of Mikolov’s original skip-gram model in order to address the problem of representing each of the different meanings of a word with only a single vector. For example, in the vector space, “bowl” might lie close to food-related words such as “cereal,” as well as sports-related words such as “football” or “game.” Thus, it would be useful to extract multiple embeddings per word, one for each sense. A non-parametric version, NP-MSSG, automatically determines the correct number of embeddings to assign to each word. We have not explored MSSG yet, but it is an area for future work.

### 4.2.3 GloVe Model

An alternative open source word vector toolkit is GloVe (for “Global Vectors”) [61], which relies on a novel global log-bilinear regression model that combines the advantages of two families of methods: local context window models, such as word2vec, and latent semantic analysis, which leverages global co-occurrence statistics but performs poorly on word analogy tasks. GloVe reaches 75% accuracy on an analogy task and outperforms several other methods on similarity and named entity recognition (NER) tasks. We did not use GloVe in our experiments, but it would be interesting to evaluate how well GloVe vectors perform compared to word2vec’s embeddings.

## 4.3 Incorporating Word Vectors as Features

Since words with similar meaning lie near each other in vector space, we can use the pre-trained vectors of tokens as new features in the CRF tagging and segmenting models in order to indicate whether certain words (e.g., “bread” and “cereal”) have related semantics. There are several methods for incorporating such vectors as features: assigning each dimension of the vector to a new feature and using the float values directly, clustering the embeddings and using the cluster which a token is closest to as the feature, and defining prototypes for each label category from which

we use the similarity between tokens and prototypes as the features [27].

### 4.3.1 Dense Embeddings

First, we directly used vector component values as features for each of the 300 dimensions of the pre-trained word vectors from the Google News corpus, which has a three million word vocabulary from about 100 billion words total (available on the word2vec website<sup>2</sup>). For these experiments, we employed the CRFsuite [56] implementation rather than CRF++ [37] (although performance was similar) for two reasons: faster running time and the ability to use vector float values directly. Unfortunately, using the continuous, dense embeddings as features in linear models such as CRFs is not effective because these architectures perform better in high-dimensional, discrete feature space. Thus, we explored two methods for discretizing the embedding features.

### 4.3.2 Clustering Embeddings

One approach to make embeddings suitable for linear models is to cluster the embeddings and measure the Euclidean distance between words and clusters

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^n (q_i - p_i)^2}, \quad (4.7)$$

where  $\mathbf{p}$  is the vector of the word,  $\mathbf{q}$  is the mean of the embeddings of the words within the cluster, and  $n$  is the number of dimensions. Each token  $w_i$  is assigned to the cluster which it is nearest to, and the feature is simply that cluster  $c_i$ . Compound cluster features can be created by combining the cluster for the current token with the clusters assigned to the previous or following tokens:  $c_i \circ c_{i+1}$ ,  $c_{i-1} \circ c_i$ , and  $c_{i-1} \circ c_{i+1}$ . We used the k-means clustering algorithm ( $k = 500$ ), as provided by the word2vec toolkit.

---

<sup>2</sup><https://code.google.com/p/word2vec/>

### 4.3.3 Distributional Prototypes

A more sophisticated approach to discretizing the continuous word embeddings is to represent each label category with a prototype word, as shown in Table 4.1 (e.g., “bread” could represent food, and “Kellogg’s” could represent brands) and to use the similarity between a token and prototypes as features [28]. We experimented both with features representing the similarity between a token and individual prototypes, as well as the average similarity between a token and all the prototypes in a category. In addition, we explored binary features which fired when the similarity was below a threshold  $\delta$  tuned with cross-validation (see Table 4.2). The similarity was calculated with cosine distance, and the prototypes were selected through normalized pointwise mutual information (NPMI)

$$\lambda_n(\text{label}, \text{word}) = \frac{\lambda(\text{label}, \text{word})}{-\ln p(\text{label}, \text{word})}, \quad (4.8)$$

where  $\lambda(\text{label}, \text{word})$  is the standard PMI

$$\lambda(\text{label}, \text{word}) = \ln \frac{p(\text{label}, \text{word})}{p(\text{label})p(\text{word})}. \quad (4.9)$$

Label	Prototypes
Food	water, milk, sauce, coffee, tea, juice, sandwich, bread
Brand	Kraft, Trader, Great, Kroger, Joe’s, Value, Farms, Quaker
Quantity	cup, two, glass, oz, one, 2, 1, 8
Description	white, green, peanut, black, whole, fresh, red, wheat

Table 4.1: Top eight prototype words for each category (except Other) selected using the NPMI metric, where out-of-vocabulary words were omitted.

$\delta$	Food	Brand	Quantity	Description
0.25	85.68	<b>86.56</b>	85.91	85.67
0.3	<b>86.56</b>	86.29	<b>86.56</b>	<b>86.56</b>
0.4	86.38	85.78	86.14	86.28

Table 4.2: Cross-validation average F1 scores across labels for three different similarity threshold values  $\delta$ , where the best threshold for each label is in bold. These results were obtained on a smaller data set of 8,000 meal descriptions.

For each label category (i.e., food, brand, quantity, and description), the NPMI was computed for every word in the vocabulary. The top  $m$  words were chosen as prototypes for each label, where the value of  $m = 50$  was selected through cross-validation (see Table 4.3).

Model	Food	Brand	Quantity	Description	Other	Overall
Baseline	94.3	80.7	91.8	88.2	95.0	90.0
$m = 40$	94.7	81.8	92.0	88.7	95.2	90.5
$m = 50$	94.7	<b>82.0</b>	<b>92.0</b>	<b>88.8</b>	<b>95.3</b>	<b>90.5</b>
$m = 60$	<b>94.8</b>	81.8	92.0	88.6	95.2	90.5

Table 4.3: Cross-validation average F1 scores when using raw similarity value features with varying  $m$  (i.e., the number of prototypes per label). These results were obtained on the most recent data set of 10,000 meal descriptions. The first row is the CRF baseline method from Chapter 3, as shown in the rightmost column of Table 3.4.

## 4.4 Results

In this section, we describe our results using the CRF with baseline features and show how incorporating word embedding features improves performance. We begin with the best CRF baseline model found in Chapter 3, and iteratively add more sophisticated features: dense embeddings, prototype similarity, shape, and clusters.

### 4.4.1 Qualitatively Analyzing Vectors

First, in order to qualitatively examine how well the word vectors represented our data, we measured the cosine similarity between pairs of word embeddings to find their nearest neighbors and applied t-Distributed Stochastic Neighbor Embedding (t-SNE) [76] before plotting the 300-dimensional vectors in two-dimensional space. As expected, in Figure 4-1, the closest neighbors of “cheese” tend to be cheese-related words such as “mozzarella” and “cheddar” and are closer to each other than to those most similar to “bowl,” which are in another cluster further away. However, due to the generic content of the Google news corpus, many of the words nearest to “bowl” involve sports and championships (e.g., “Rose Bowl” and “NCAA Tournament”)





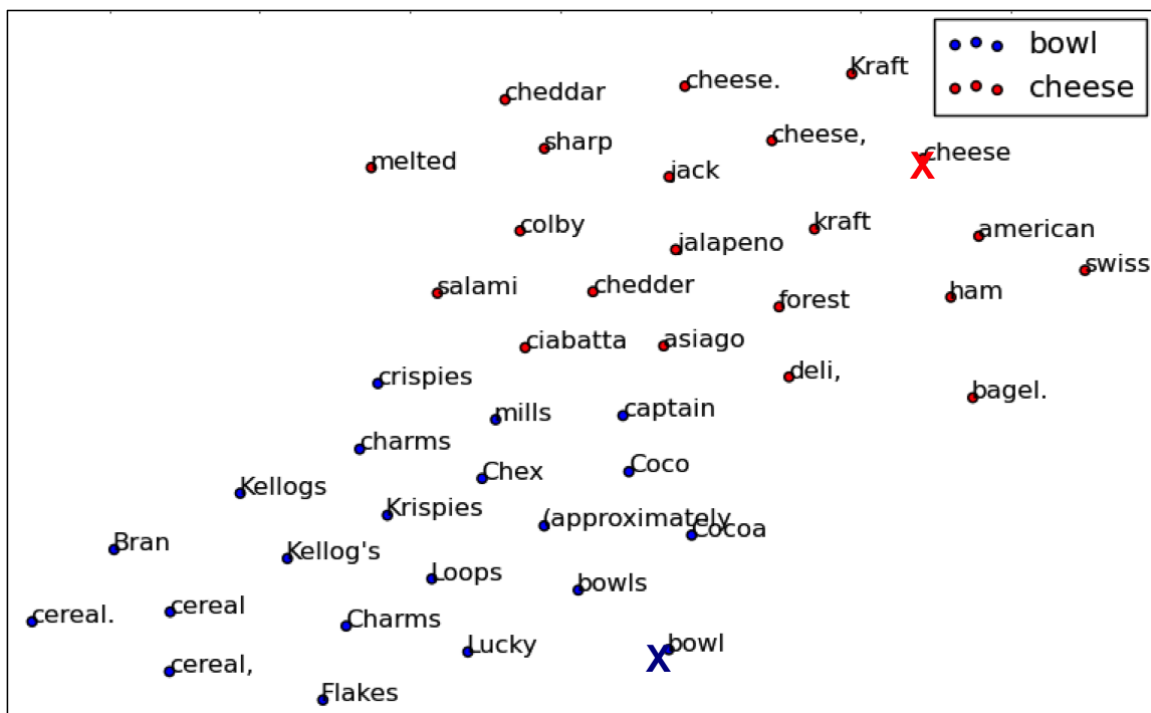


Figure 4-2: 20 nearest neighbors of “bowl” and “cheese,” using 300-dimension word2vec embeddings trained on nutrition data and reduced to two dimensions through t-SNE.

Data	Food	Brand	Quantity	Description	Other	Overall
Nutrition	94.5	81.3	<b>91.9</b>	88.3	95.1	90.2
Wikipedia	94.9	81.6	91.8	88.0	95.1	90.3
Google	<b>95.0</b>	<b>82.7</b>	91.7	<b>88.9</b>	<b>95.2</b>	<b>90.7</b>

Table 4.4: F1 scores per category when using all features (with vectors) trained on three different data sources.

#### 4.4.2 CRF Baseline

We evaluated the performance of feature sets with and without semantic vectors for three CRF software implementations: CRFsuite [56], CRF++ [37], and Mallet [46]. As the baseline from Chapter 3, we used n-gram features (as-is and all-lowercase), POS tags [44], and presence in USDA food and brand lexicons [21]. We found that CRF++ and CRFsuite were both significantly better than Mallet (see Table 4.5), where  $p < 0.01$ .

The shape features, which indicated whether or not the token was in titlecase, lowercase, uppercase, a number, a piece of punctuation, or other, improved upon the

baseline features. Although this feature is not as helpful for speech as it is for text (since there is almost no punctuation in recognized speech), shape features do provide some important information, such as capitalization for brands.

Model	No Vectors	With Vectors
CRFsuite	<b>90.0</b>	<b>90.5</b>
CRF++	90.0	90.0
Mallet	78.9	85.6

Table 4.5: Average F1 scores on the semantic tagging task with CRF++, CRFsuite, and Mallet on two feature sets: with and without raw word vector values (i.e., dense embeddings).

### 4.4.3 Incorporating Vector Features

To improve upon the baseline even further by using word vectors, first (as in Section 4.3.1), we added features corresponding to the values of the tokens’ word2vec embeddings, which increased the average F1 score by 0.5, as shown in Table 4.6. Since the continuous, dense embeddings are not as effective as discrete features in linear models such as CRFs, we explored a distributional prototype method for utilizing vectors in a manner more suited to CRFs: we incorporated features corresponding to the similarity between a token and distributional prototypes. When high scoring prototypes did not appear in the corpus, we selected the next best prototype with a vector representation.

We did not include prototypes for the Other category because this lowered the average F1 score. We also found that averaging the similarities, as opposed to using a unique feature for each prototype’s similarity, resulted in lower F1 scores. This could be due to differences in meaning among the various prototypes within a category. For example, drinks such as “juice” are considered foods, as is “bread;” however, a token such as “tea” would be similar to drinks, but not to foods like “bread.” Therefore, the token “tea” will have a large similarity value for the prototype “juice,” but low similarity value for the prototype “bread,” so the average similarity value would be a less informative feature than the individual similarity values for identifying “juice”

as a food item.

Model	Food	Brand	Quantity	Description	Other	Overall
Baseline	94.3	80.7	91.8	88.2	95.0	90.0
+ Dense	94.6	82.2	<b>92.1</b>	88.5	95.2	90.5
+ Protos	94.7	82.0	92.0	88.8	<b>95.3</b>	90.5
+ Shape	95.0	82.6	91.6	88.9	95.1	90.6
+ Clusters	<b>95.0</b>	<b>82.7</b>	91.7	<b>88.9</b>	95.2	<b>90.7</b>

Table 4.6: CRFsuite F1 scores per label in the semantic tagging task with incrementally complex feature sets: baseline n-grams, POS tags, and lexical features; dense embeddings; raw prototype similarities; shape; and clusters. The first row is the CRF baseline method from Chapter 3, as shown in the rightmost column of Table 3.4.

We experimented with two similarity features: a binary number indicating whether or not the similarity was below a threshold  $\delta$  determined through cross-validation (i.e., 0.3 for foods, quantities, and descriptions, and 0.25 for brands) and the raw value of the similarity. The raw similarity value features improved upon the baseline combined with raw word vector values, but the binary similarity did not yield further improvement. This might be because the CRF is able to determine the relative importance of each similarity feature, whereas the binary similarity weights the different prototypes’ similarities equally.

Finally, the addition of cluster features using word2vec’s built-in k-means clustering (500 classes) improved performance. Thus, the baseline combined with shape features, raw vector and similarity values, and clusters yielded the highest gain, with an average F1 score of 90.7. The improvement of the word vector, shape, and cluster features relative to the baseline was significant, where  $p < 0.01$  ( $p = 0.003$ ).

## 4.5 Independent Questions of Inquiry

Now that we have determined the best model and feature set for the semantic tagging task, we wish to analyze the performance of our approach under two different conditions: with a limited vocabulary, and with varying amounts of data. In the first case, we explore methods for predicting unknown words’ vectors in order to boost performance. We then show that increasing the training data improves results.

### 4.5.1 Handling Unknown Vectors

A challenge that speech recognition and language understanding systems often face is handling words that are unknown, or out-of-vocabulary (OOV), since the training data cannot contain every possible word a user might say [4]. OOV words are problematic when using embeddings as CRF features because if a word is unknown, then it has no corresponding vector representation. So far, we have adopted the simplistic approach of omitting these features by setting the unknown vectors to zero. In this section, we investigate two alternative methods for handling unknown words: a language model-based prediction of the OOV word, and a context-based representation of the OOV word.

For the first approach, we used a bigram language model to predict an in-vocabulary word to represent the OOV word. The predicted word is the most likely word given the previous word (i.e., the language model predicts a token  $w$  that maximizes the probability  $P(w|c)$ , where  $c$  is the token that appears prior to  $w$ ). The predicted in-vocabulary word is then used to generate the word embedding features for the unknown word. We experimented with two types of language models: a simple bigram language model generated from bigram counts on the nutrition data, as well as the language model generated by word2vec. We also predicted the top  $N$  words (where  $N = 1, 2, \text{ or } 3$ ) and averaged their embeddings to yield a single predicted embedding.

For the second approach, we explored a context-based representation of OOV words, where we generated word vectors by summing the context words surrounding the unknown word. We experimented with two context window sizes (e.g., one and two), where a context window of size one includes the previous word, as well as the subsequent word.

Since performance was already high, there was no significant gain from predicting vectors for unknown words, where  $p < 0.01$ ; in addition, it is difficult to determine which feature sets benefit from OOV handling. Thus, to isolate the effect of word vector prediction on the raw vector value features, we investigated the performance of six OOV handling methods when using only dense embedding features. As shown in Ta-

ble 4.7, using the basic bigram language model yielded better performance than using the word2vec language model, and the language model-based prediction method performed better than the context-based representation method (where a context window of size one was better than a window of two). In addition, for the language model-based prediction method, averaging the top two predicted words' vectors, rather than using only the top one, resulted in higher scores. Overall, the full system without OOV handling performed best because it incorporated more features than only dense embeddings.

Since the whole data set has an out-of-vocabulary rate of only 2% (with 24% of meal descriptions containing at least one OOV word), we evaluated whether the improvement would be more significant with a data set that has a higher OOV rate. For this experiment, we declared only the top 20 prototype words for each of the five categories (i.e., out of 100 words in total) to be in the vocabulary, and set the remaining words as OOV. This resulted in an OOV word rate of 56% and 99.9% OOV meal descriptions. As expected (see Table 4.8), while predicting unknown words did not help on the original data set with 2% OOV rate, there was a significant improvement of 20% when we limited the vocabulary to 100 words.

In the future, we will collect more data for continued testing, which may show even more improvement when predicting unknown vectors; in addition, if we were to use an ASR system trained on the nutrition corpus, recognition errors and nutrition-specific terminology may cause the OOV rate to increase, in which case predicting unknown words' vectors could provide greater gains.

## 4.5.2 Performance with Varying Amounts of Data

We evaluated the semantic tagging performance of the CRF on both the baseline and highest performing feature sets as a function of the amount of training data, as shown in Figure 4-3. As expected, increasing the amount of training data from 2,000 to 10,000 meal descriptions significantly improved the semantic tagging performance with both the baseline features and the best performing feature set, where  $p < 0.01$ . The average improvement with the baseline features was 2.3 points, or 2.7%, when

OOV Handling	Food	Brand	Quantity	Descr.	Other	Overall
Sum context (cw = 1)	79.9	54.2	78.6	71.3	85.3	73.9
Sum context (cw = 2)	<b>80.1</b>	49.6	15.1	<b>71.8</b>	82.2	69.7
Basic lang model (top 1)	77.1	<b>58.2</b>	77.5	71.5	84.9	73.8
Basic lang model (top 2)	79.8	53.9	<b>83.7</b>	70.9	<b>86.3</b>	<b>74.9</b>
Basic lang model (top 3)	80.1	54.2	78.7	71.3	85.4	73.9
word2vec lang model (top 2)	79.9	54.2	78.4	71.2	85.2	73.8
Full System (no handling)	<b>95.0</b>	<b>82.7</b>	91.7	<b>88.9</b>	<b>95.2</b>	<b>90.7</b>
Full System (with handling)	94.7	81.8	<b>91.9</b>	88.5	95.2	90.4

Table 4.7: Semantic tagging F1 scores per label with only dense embedding features, for six prediction methods. The full system is taken from Section 4.4.3, where there is no OOV handling, but it performs better because it uses the full set of features.

Vocabulary Size	Word OOV Rate	No Prediction	With Prediction
50 words	62%	41.6	59.4
100 words	56%	50.1	60.4
200 words	50%	59.3	62.8
400 words	45%	66.5	63.8
1000 words	38%	71.4	67.5
All words	2%	79.3	74.9

Table 4.8: Average F1 scores on the semantic tagging task for six vocabulary sizes with only dense embedding features, where prediction is done by averaging the top two predicted words of the basic language model trained on nutrition data.

using five times as many training samples; the average improvement with the best feature set was 2.5 points, or 2.8%, which is a 0.2% relative improvement over the baseline.

## 4.6 Conclusion

In this chapter, we have discussed more sophisticated features for CRF models in the language understanding component of a nutrition dialogue system using word embeddings. We explored three approaches for incorporating word embedding features: using the dense embedding values directly, measuring the raw and binary similarity between a token and prototypes for each category (except Other), and assigning vectors to clusters. We found that the best feature set consisted of baseline n-grams, POS tags, lexicon, and word shape features; raw vector values; raw similarity values

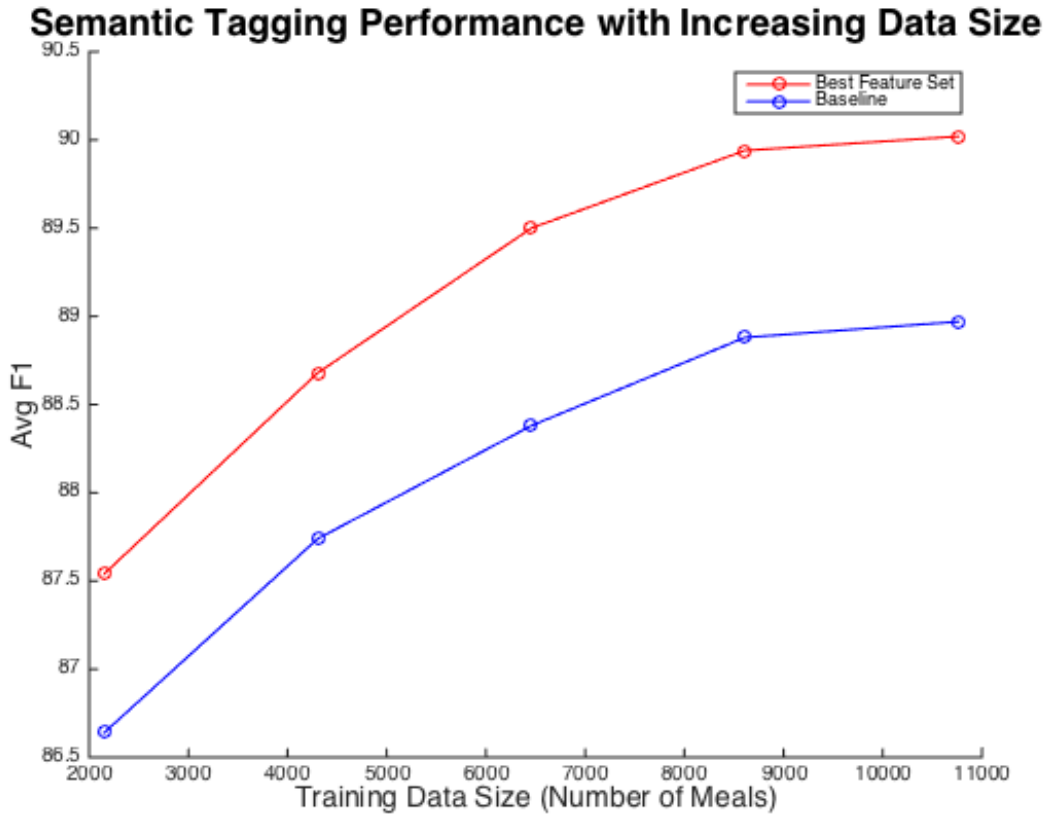


Figure 4-3: Semantic tagging average F1 score as a function of increasing data size, for two feature sets: baseline and baseline combined with raw vector values, raw similarity values, and shape.

to individual prototypes; and clusters, yielding an average F1 score of 90.7.

Predicting out-of-vocabulary word vectors by summing the top two predicted tokens from a basic language model trained on nutrition data did not improve performance with the full model; however, it provided a significant gain of 20% when using only dense embedding features and a limited vocabulary of 100 words. Finally, as expected, performance improved as the size of the training data set increased.



# Chapter 5

## Associating Foods and Properties

In the nutrition system, after the user describes his or her meal, the language understanding component must not only identify the foods and properties (i.e., semantic tagging), but also determine which foods are associated with which properties, as shown in Figure 5-1. There are two alternative approaches for accomplishing this food-property association task: segmenting the meal description tokens into food chunks (each containing a food item and its associated properties), and predicting the most likely food for each property.

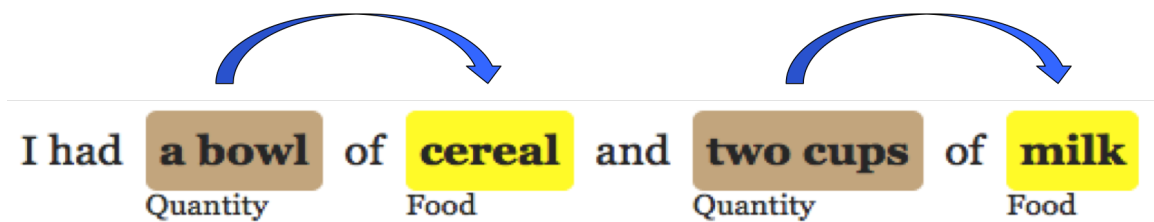


Figure 5-1: A depiction of the food-property association task, in which the quantity “a bowl” is assigned to the food “cereal,” and “two cups” is associated with “milk.”

We investigated three approaches for segmenting the meal descriptions (using semantic tags from the semi-CRF classifier because this was our default model at the time of running these experiments): a Markov model (MM), transformation-based learning, and a CRF classifier. However, due to issues arising with this segmentation method, we considered conducting the food-property association independently of the temporal methods. We explored three different classifiers for directly predicting the

food with the highest probability of being associated with each property.

## 5.1 Segmenting Approaches

The first four methods we investigated for associating foods with properties all involved segmenting meal descriptions into food chunks, where all properties within a chunk would be assigned to the food item that appears within the same chunk. For example, the meal description “I ate two pancakes and drank a glass of milk” would be segmented into two chunks: “two pancakes” and “a glass of milk.”

### 5.1.1 Simple Rule

As our baseline, since on average 86% of the attributes in the data appear prior to their corresponding food item, we defined a simple rule which assigns properties to the subsequent food. For example, in the description “I had a cup of milk with a handful of blueberries,” the quantity “a cup” would be associated with “milk,” and the quantity “a handful” with “blueberries.” In the case where an attribute appears after the last food item in the description, the attribute is assigned to the last food.

### 5.1.2 Markov Model

To improve upon the baseline method, we took advantage of the sequential nature of the food description data (e.g., a food item may be more likely to appear after a brand than a quantity) by modeling it probabilistically. We defined a first-order Markov chain, in which each observation  $x_i$  depends only on the previous observation  $x_{i-1}$ . The joint distribution for a sequence of  $n$  observations under this generative model is

$$p(x_1, \dots, x_n) = p(x_1) \prod_{i=2}^n p(x_i | x_{i-1}), \quad (5.1)$$

which leads to the conditional distribution for observation  $x_i$ , given all previous observations, of

$$p(x_i|x_1, \dots, x_{i-1}) = p(x_i|x_{i-1}), \quad (5.2)$$

since, by the Markov assumption,  $x_i$  depends only on the previous observation  $x_{i-1}$  [6, 70]. In our case, we let each observation in the Markov chain represent an attribute or food item. For example, in the meal description “I had a bowl of cereal,” the semi-CRF would label “a bowl” as a quantity and “cereal” as a food, resulting in the Markov chain in Figure 5-2.

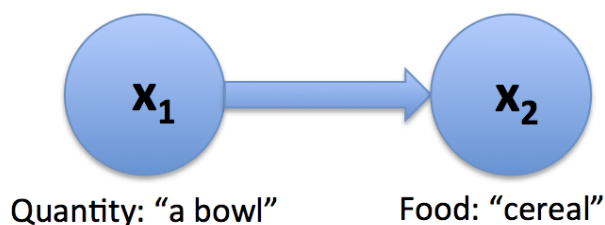


Figure 5-2: A first-order Markov chain for the food description “I had a bowl of cereal.”

We implemented this Markov model with a finite state transducer (FST), which transduces an input string into an output string [31]. Like a Markov model, an FST has states and transitions with associated weights; we let each state represent a possible food or property semi-CRF label and added start and end states. The input to the FST is a string of food or property labels in a food diary, and the output is the same string segmented by “#” such that each food item and its associated properties are within the same segment. For example, the diary “I had a bowl of cereal with milk” would correspond to the input string “Q F F” and would generate the output “Q F # F.” This indicates that “a bowl” is a quantity describing “cereal,” and “milk” has no attributes. With this approach, properties that appear after a food item may be within the same segment (e.g., “Q F B # F”), and a single segment may contain multiple food items (e.g., “F F” and “F # F” are both valid).

We used the frequency of label patterns (e.g., “Q B F D F”) that occur in the training data to calculate the initial state distribution (e.g.,  $P(Q) = 0.62$ ) and the state transition probabilities (e.g.,  $P(F|D) = 0.74$ ). In addition, using the histogram in Figure 2-3, we calculated the probability that a subsequent food follows the current

food. The transition weights in the FST behave like negative log probabilities.

### 5.1.3 Transformation-Based Learning

We further improved upon both the simple rule and the Markov model approaches by applying a transformation-based learning (TBL) algorithm. This method starts with an initial solution to a problem (e.g., the simple rule baseline) and iteratively applies transformations, selecting those which improve the performance most. We used the Fast TBL toolkit developed at Johns Hopkins [20].

In order to adapt TBL to the food-property association problem, we framed it as a classification task. To do this, we modeled it after the NP chunking problem, a well-known natural language processing (NLP) task. In NP chunking, each word in a sentence belongs to one of three classes: B (the start of an NP), I (inside an NP), or O (outside an NP). For the food chunking problem, we used the same three classes to label each word as belonging to a food chunk or not. The features used by the classifier are composed of a token and its semi-CRF label (i.e., food, quantity, brand, description, or other). An example food diary is shown in Table 5.1. We also defined general rule templates (see Table 5.2) from which the model learns specific rules that may be applied in order to improve the system’s performance. For example, the rule template “ $\text{chunk}_0 \text{ chunk}_1 \text{ label}_0 \Rightarrow \text{chunk}$ ” implies that given the current chunk label, the next chunk label, and the current semi-CRF label, the current chunk label should be transformed to that specified by the rule.

### 5.1.4 CRF

As an alternative to the TBL algorithm, we investigated the CRF. Since it is a discriminative classifier as well, the CRF was trained on the same data as the TBL model. However, rather than defining a set of rule templates, we provided feature templates corresponding to unigrams and bigrams of output tags that appear with certain combinations of tokens and food or property labels, as shown in Table 5.3. We used the CRF++ toolkit [37]. We also trained a TBL model using the CRF as

Token	CRF Label	Chunk
I	Other	O
had	Other	O
a	Quantity	B
bowl	Quantity	I
of	Other	I
cereal	Food	I
from	Other	I
Trader	Brand	I
Joe’s	Brand	I
and	Other	O
a	Quantity	B
glass	Quantity	I
of	Other	I
milk	Food	I

Table 5.1: Example of the food chunking classification problem, where a chunk label B, I, or O is assigned to each token, given its semi-CRF label (i.e., brand, quantity, description, or food).

Rule
$Label_{-1}, Label_0, Label_1 \Rightarrow Chunk$
$Chunk_{-1}, Chunk_0, Word_0 \Rightarrow Chunk$
$Chunk_{-1}, Chunk_0, Label_0 \Rightarrow Chunk$
$Chunk_{-1}, Chunk_0, Label_{-1}, Label_0 \Rightarrow Chunk$
$Chunk_0, Chunk_1, Word_1, Word_2 \Rightarrow Chunk$

Table 5.2: A sample of five rules from the template for training the TBL model.

the initial solution.

## 5.2 Segmenting Results

In Table 5.4, we present the performance of six approaches to the food-property association task. We trained and evaluated the models on a set of 8,000 annotated meal descriptions. The TBL algorithm significantly improved upon the simple rule and Markov model (MM) baselines, but not the CRF, where  $p < 0.01$ , although it did provide higher precision. The CRF was significantly better than both the simple rule and the Markov model, as well as the models with TBL on top of the simple rule and Markov model. Overall, the CRF model performed best, achieving a token-level

Feature Rule Template
token <sub>-2</sub>
token <sub>-1</sub>
token <sub>0</sub>
token <sub>1</sub>
token <sub>2</sub>
token <sub>-1</sub> ◦ token <sub>0</sub>
token <sub>0</sub> ◦ token <sub>1</sub>
label <sub>-2</sub>
label <sub>-1</sub>
label <sub>0</sub>
label <sub>1</sub>
label <sub>2</sub>
label <sub>-2</sub> ◦ label <sub>-1</sub>
label <sub>-1</sub> ◦ label <sub>0</sub>
label <sub>0</sub> ◦ label <sub>1</sub>
label <sub>1</sub> ◦ label <sub>2</sub>
label <sub>-2</sub> ◦ label <sub>-1</sub> ◦ label <sub>0</sub>
label <sub>-1</sub> ◦ label <sub>0</sub> ◦ label <sub>1</sub>
label <sub>0</sub> ◦ label <sub>1</sub> ◦ label <sub>2</sub>

Table 5.3: CRF++ template for learning features from tokens and property labels, where token<sub>0</sub> refers to the current token, and all other indices are relative to the current token. label<sub>-2</sub> ◦ label<sub>-1</sub> indicates the combination of two features into a single feature.

accuracy of 86.9% and a phrase-level F1 score of 61.9.

A few examples of errors made by the different models are shown in Table 5.5. Since brands and descriptions occasionally appear after a food, the simple rule incorrectly assigns attributes in these cases. For example, in the food diary “2 uncured hot dogs from Trader Joes,” the simple rule mistakenly begins a new food chunk with the brand “Trader Joes,” since “Trader Joes” is actually part of the food chunk corresponding to “2 uncured hot dogs.” Although TBL corrects these errors, it occasionally overcompensates by incorporating unrelated tokens into the food segment (e.g., it predicts that “turned on the gas stove” belongs to the food segment “vegetable oil”). MM errors occur when it incorrectly segments the labels. For example, it segments the phrase “a small iced coffee from Dunkin’ Donuts with cream” into the output string “Q D F # B D F,” which incorrectly associates the brand “Dunkin’

Approach	Acc	Prec	Recall	F1
Simple Rule	76.7	48.0	48.0	48.0
Simple + TBL	84.9	57.5	57.4	57.5
MM	75.3	49.2	49.2	49.2
MM + TBL	84.8	56.4	56.5	56.4
CRF	<b>86.9</b>	60.6	<b>63.3</b>	<b>61.9</b>
CRF + TBL	86.9	<b>63.3</b>	60.6	61.9

Table 5.4: Test performance of approaches to the food segmenting task, where accuracy is calculated at the token-level and precision, recall, and F1 are computed at the phrase-level. The CRF (shown in bold) achieved the best accuracy and F1 score.

Donuts” with “cream” instead of “coffee.” TBL is unable to correct some of the CRF errors, which could be the reason it does not improve on the CRF’s performance.

Method	Text	Auto	AMT
Simple	2 uncured hot dogs from Trader Joes	BIII OBI	BIIIIII
Simple	a hot chocolate from the local convenience store	IIIOOBII	OBIIIII
+ TBL	2 uncured hot dogs from Trader Joes	BIIIIII	BIIIIII
+ TBL	a hot chocolate from the local convenience store	BIIIIII	OBIIIII
MM	a small iced coffee from Dunkin’ Donuts with cream	BIII OBIII	BIIIIIOB
MM	a 7 layer burrito from Taco Bell	OBI OBI	BIIIIII
+ TBL	two pop tarts in each pack	BIIIIII	BIIOOO
+ TBL	rotisserie chicken and rice	BIII	BIOB
CRF	garlic powder and cayenne pepper	BIIII	BIOBI
CRF	Stonyfield whole-milk plain yogurt, ice	IIII	BIIIB
+ TBL	garlic powder and cayenne pepper	BIIII	BIOBI
+ TBL	Stonyfield whole-milk plain yogurt, ice	IIII	BIIIB

Table 5.5: BIO food chunking mistakes, where auto is the prediction and AMT is the gold standard annotation.

The TBL toolkit outputs a list of successful rules, as shown in Table 5.6, where the score is defined as the number of improvements minus performance reductions. This allows us to observe where the baseline method made errors and which rules were used to fix those mistakes. For example, the model learned that if the current token is labeled Description and the previous token is labeled Food, then the current chunk label should change from B to I. In addition, if the current token is the word “I”, then the current chunk label should change from I to O, since new phrases or sentences often start with with word “I.” Some rules specify that tokens labeled Brand

should have the chunk label I, which would fix the simple rule’s mistake of assigning brands that appear after food items to the subsequent food rather than the prior food. Finally, the word “from” is learned to be part of a food segment, whereas the word “with” is often outside of any food segments.

Rule	Score
$C_0 = B, L_{-1} = \text{Food}, L_0 = \text{Description} \Rightarrow C = I$	849
$C_0 = I, \text{word}_0 = \text{“I”} \Rightarrow C = O$	829
$C_{-1} = I, C_0 = B, L_0 = \text{Brand} \Rightarrow C = I$	344
$C_0 = O, \text{word}_0 = \text{“from”} \Rightarrow C = I$	300
$C_- = I, C_0 = I, \text{word}_0 = \text{“with”} \Rightarrow C = O$	288

Table 5.6: TBL (with simple rule baseline) high-scoring rules, where score is the number of improvements minus performance reductions.  $C_i$  represents a chunk label at index  $i$  (e.g., B, I, or O), and  $L_i$  indicates the food or property label at  $i$ .

In addition to IOB labeling, there are two other representations for chunking tasks, IOE and IOBES. In IOE, rather than marking the first token in a chunk with B and the last with I, we mark the first token with I and the last with E (i.e., the “end” token). In IOBES, we represent the first token with B and the last with E; however, if a chunk consists of only a single token, we represent it with S. We experimented with these other class types using the TBL algorithm on top of the CRF classifier, since it was the best model under the BIO representation. As can be seen in Table 5.7, the accuracies of the IOB and IOE representations are not significantly different, where  $p < 0.01$ , but IOE has the highest F1 score.

Type	Accuracy	Precision	Recall	F1
IOB	86.9	63.3	60.6	61.9
IOE	<b>87.2</b>	63.2	<b>65.3</b>	<b>64.2</b>
IOBES	83.0	<b>64.0</b>	62.7	63.4

Table 5.7: Performance of CRF+TBL on segmentation task using three different label representations.

We also explored whether incorporating the IOE segmentation labels as features in the initial labeling task would improve the semantic tagging performance. In this iterative two-step process, we used the results from the best method for the second task (i.e., CRF+TBL with IOE label representation) to redo the first task. However,



the performance with the additional IOE label feature only slightly increased from 85.9% accuracy to 86.0%. Thus, we can conclude that adding IOE labels does not significantly improve the semantic tagging task, where  $p < 0.01$ .

Finally, we conducted oracle experiments in order to observe how well the models performed on the segmenting task when using the gold standard AMT labels directly, rather than the semi-CRF predictions (we used the semi-CRF for semantic tagging in this analysis because it was the baseline model we used at the time of the experiments). Since the semi-CRF labeling errors are compounded when fed into the segmenting task, we investigated how much the segmenter improved when provided with correct labels. As shown in Table 5.8, the F1 scores for all six methods in the oracle experiments were significantly higher (by McNemar’s test) than those from the non-oracle experiments, as expected. This time, TBL significantly improved upon the simple rule and the Markov model, but not the CRF. Therefore, although the CRF with TBL had the highest recall, the CRF had the overall best accuracy and F1 score.

Approach	Acc	Prec	Recall	F1
Simple Rule	90.1	67.5	74.6	70.9
Simple + TBL	93.1	77.8	78.1	77.9
MM	87.7	69.1	75.2	72.0
MM + TBL	91.7	74.8	73.5	74.2
CRF	<b>93.7</b>	<b>78.9</b>	77.7	<b>78.3</b>
CRF + TBL	93.6	77.7	<b>78.7</b>	78.2

Table 5.8: Oracle experiments on the segmenting task with six different methods, where AMT gold standard labels are used rather than semi-CRF predictions.

### 5.3 Food-Property Association with Word Vectors

As an alternative to the four segmenting methods we have explored so far in this chapter, we trained a classifier for assigning properties to foods. This approach indirectly incorporated word embeddings into the food-property association task by using the predicted semantic tags from the CRF trained on word vectors. Again, we com-

pared oracle experiments with gold standard tags to experiments using predicted tags, where we used the model with the best feature combination from Section 4.4.3 (i.e. n-grams, POS, lexicons, shape, raw vector values, and prototype similarity values).

One drawback to using the IOE labeling (i.e., segmenting) scheme proposed in Section 5.1.3 is that the representation assumes properties appear either directly before or after the food with which they are associated, neglecting potential long-range dependencies. For example, in the meal description “I had two eggs and some cheese from Trader Joe’s,” the brand “Trader Joe’s” should be assigned to both “eggs” and “cheese;” however, with the chunking scheme, it is impossible to associate “Trader Joe’s” with “eggs” without also assigning the quantity “two” to “cheese.” In addition, converting the labeled AMT data to IOE format requires making assumptions where some information (e.g., these long-range dependencies) is omitted. Thus, we investigated an alternative method for food-property association where we trained a classifier to predict which food a property describes. The challenge with this approach is determining whether a property describes more than one food. However, since only 3% of property tokens in the labeled data describe multiple tokens, missing some foods should not hurt the accuracy by a large margin.

In our approach, given a tagged meal description, for each of the property tokens, the classifier determines with which food it is associated. Given a property token  $t_i$ , we iterate through each food token  $f_j$  in the meal description and generate features for each  $(t_i, f_j)$  pair. For each pair, the classifier outputs a probability that  $f_j$  is the corresponding food item for  $t_i$ . Then, for each  $t_i$ , the  $f_j$  with maximal probability is selected as the corresponding food item. Note that this does not allow a property to be associated with more than one food, but we consider this a first step and will explore association of properties with multiple foods in future work.

The classification is done using three features: whether or not the food token comes before or after the property token, the distance between the two tokens, and the semantic tag of the first token. We explored three different classifiers, using the Scikit-learn toolkit’s implementation for Python [60]: a random forest (i.e., a collection of decision tree classifiers trained on a random sample of training data) [7],

logistic regression, and a naive Bayes classifier. We used the spaCy NLP toolkit<sup>1</sup> in Python for dependency parsing, tokenizing, and tagging because it is fast and provides shape features (e.g., capitalization, numbers, etc.) that improved performance over our manually defined shape features. Performance was evaluated using precision, recall, and F1 scores for property tokens only. In the oracle experiments, since we use the gold standard semantic tags, the number of actual property tokens equals the number of predicted property tokens, so the precision, recall, and F1 scores are all equivalent.

As shown in Table 5.9, the random forest classifier performed best. As expected, the performance is significantly better in the oracle experiments than when using predicted tags, where  $p < 0.01$ ; this is partly due to the semantic tagging model predicting properties which are not actually properties, and so will always be marked incorrect during evaluation. The random forest classifier constructs an ensemble of decision trees by sampling with replacement from the training set. The decision trees are constructed from splits made on random subsets of features, which increases the bias, but decreases the variance due to averaging, which yields a better model overall. The trees are combined by averaging their probabilistic predictions, and the trees are constructed by finding features at each node that yield the largest information gain.

Model	Precision	Recall	F1
Naive Bayes (Oracle)	94.6	94.6	94.6
Logistic Regression (Oracle)	95.2	95.2	95.2
Random Forest (Oracle)	<b>96.2</b>	<b>96.2</b>	<b>96.2</b>
Naive Bayes (Predicted)	84.1	87.3	85.7
Logistic Regression (Predicted)	84.2	87.4	85.7
Random Forest (Predicted)	<b>84.7</b>	<b>87.9</b>	<b>86.3</b>

Table 5.9: Food-property association with three different classifiers, for both gold standard semantic tags (i.e., oracle) and predicted tags.

---

<sup>1</sup><https://honnibal.github.io/spaCy/>

## 5.4 Comparing Segmentation to Classification

In order to compare the performance of this food-property association approach (i.e., classification) to that of the BIO chunking with the CRF (i.e., segmentation), we converted the IOE labels from the CRF method to the predicted associations format for both oracle and non-oracle experiments (see Table 5.10). These results show that the new association approach using a random forest classifier yields a significantly higher F1 score than the CRF, where  $p < 0.01$ , when evaluated on property tokens. For the CRF method, the number of gold property tokens with associated foods is greater than the number of property tokens with predicted foods, which indicates that some properties were missed in the IOE chunking scheme and therefore were not assigned any foods.

Model	Precision	Recall	F1
Segmenting (Oracle)	87.9	83.9	85.9
Classifying (Oracle)	96.2	96.2	96.2
Combined (Oracle)	<b>96.5</b>	<b>96.5</b>	<b>96.5</b>
Segmenting (Predicted)	<b>86.2</b>	81.0	83.5
Classifying (Predicted)	84.7	87.9	86.3
Combined (Predicted)	84.9	<b>88.2</b>	<b>86.5</b>

Table 5.10: Performance on the food-property association task using the prior approach of IOE segmenting with the CRF, the new food prediction method with the random forest classifier, and the union, evaluated on property tokens for all methods.

We also investigated whether the IOE labels from the CRF were complementary to the food-property classification approach by incorporating the predicted IOE labels as new features in the random forest classifier. As shown in the last row of both sections of Table 5.10, the addition of the IOE labels does improve the classification performance for both oracle and non-oracle experiments. The combination of the random forest classifier with IOE labels relative to the CRF segmenter on its own significantly improved the performance, where  $p < 0.01$  ( $p = 0.002$ ).

## 5.5 Conclusion

In this chapter, we have presented the second task of the system’s language understanding component: association of attributes with foods. We evaluated two approaches: segmentation and classification. For the segmenting approach, we explored three methods for assigning properties to foods: a Markov model that segments meal descriptions into food chunks, a TBL algorithm that iteratively learns rules to correct the baseline’s errors, and a CRF classifier that outperforms the other methods.

Since the majority of attributes appear prior to their corresponding food items, the baseline simple rule which associates foods with prior attributes performs similarly to the Markov model approach. However, the simple rule makes mistakes which the Markov model does not. For example, brands and descriptions may appear after a food; in the diary “I had eggs from Trader Joe’s with bread,” “eggs” is the food item and “Trader Joe’s” is its brand, but the simple rule would assign “Trader Joe’s” to the food “bread.” Even though the Markov model is likely to segment these diaries correctly, the TBL algorithm shows greater improvement by directly correcting these errors through the use of transformation rules.

Transformation-based learning also improves upon the Markov model by learning rules that fix typical errors made by this model. TBL and the CRF contain more information than the Markov model by incorporating tokens, not just the food and property labels, as well as tokens that are labeled Other. The CRF model (using IOE labels) is the best, labeling the test data with the highest F1 score of 61.9 [36].

As an alternative to the segmenting methods, we investigated using classifiers to predict the food with which a property is associated with the highest probability. We used three features: the distance between a food and the property, the semantic tag of the property, and whether the property is before or after the food token. We found that the random forest classifier performed better than logistic regression and a naive Bayes classifier, yielding an F1 score of 86.3 when using semantic tags predicted by the CRFsuite model trained on word vector features. We also discovered that the food prediction method is complementary to the IOE segmenting approach, since

incorporating IOE labels as additional features increased the F1 score to 86.5.

# Chapter 6

## The Nutrition System Prototype

The NLP methods we have explored thus far ultimately fit into the larger nutrition system with the goal of enabling users to easily and efficiently track their dietary intake. Although the language understanding component is a critical aspect of the overall system, there are four other important areas we are currently investigating: the user interface, connecting to nutrition databases, conversing with and responding to the user through dialogue, and the initial speech recognition. The entire system architecture is shown in the diagram in Figure 6-1. In this chapter, we describe previous work in spoken dialogue systems, ongoing work on our nutrition system, and evaluation of the overall system performance.

### 6.1 Related Work

There are two general areas into which dialogue systems are classified: task-oriented [58, 63, 17] and nontask-oriented [25, 35]. In the first category, the user has a specific goal they wish to accomplish by interacting with the system, and there are a series of commands available for instructing the system to complete a desired task. Nontask-oriented systems, on the other hand, are more conversational in nature. There is no specific task the user wishes to accomplish; rather, the system acts as a conversational partner and ideally should be able to respond to any query in a natural, human-like manner. While our proposed nutrition system is currently task-oriented, we envision

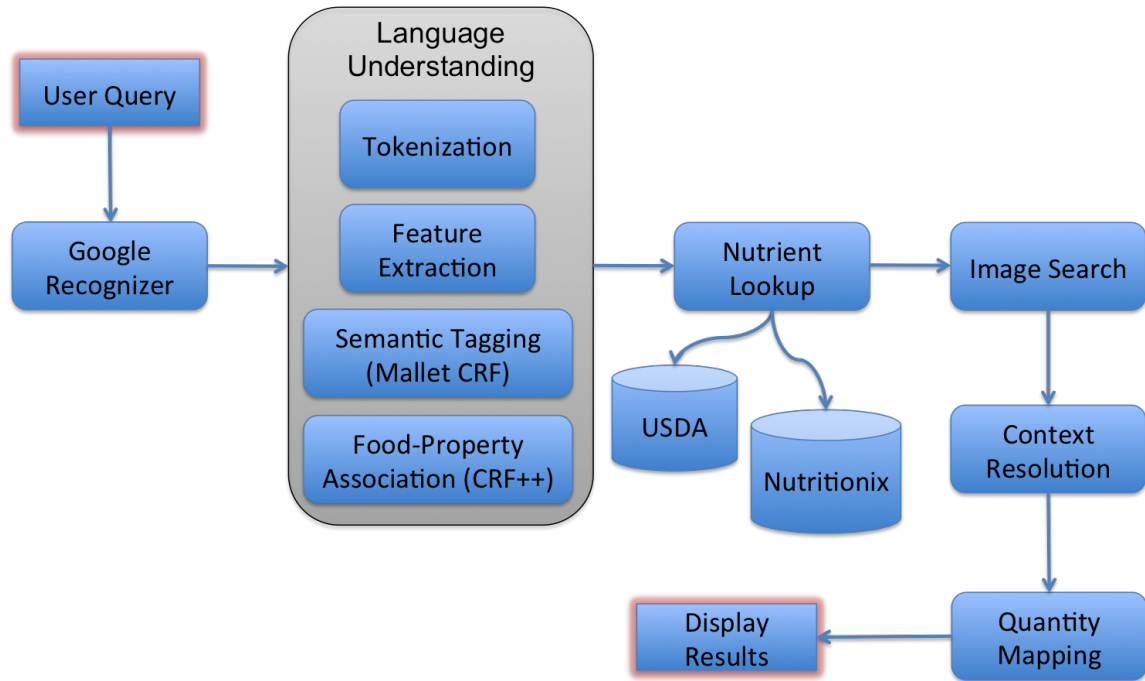


Figure 6-1: A diagram of the nutrition system’s current architecture.

expanding the queries to which the system can respond in order to make it more conversational.

One of the first commercially deployed dialogue systems was the AT&T How May I Help You? (HMIHY) system [24], which is a call routing system that classifies customer’s calls and routes them to the appropriate destination. Two other early dialogue systems developed at MIT were the JUPITER system [88] for obtaining worldwide weather forecasts and the Mercury automatic flight booking system [69], which provides telephone access to an online flight database. All three systems are task-oriented; however, while dialogues in HMIHY are guided solely by the system after the initial user query, in JUPITER and Mercury, the technology is more complex because the user can ask questions to which the system must adapt. Whereas in the past, dialogue systems have mostly been useful for answering weather queries, providing summaries of restaurant or movie reviews, and booking flights, we propose applying dialogue systems to the nutrition domain.

In the past, dialogue managers were often composed of rules constructed by hand [69]. This approach works well when users strictly follow the system-defined



dialogue pattern; however, this model suffers when users direct the course of the conversation themselves, or move outside the realm of the hand-crafted slots. The rule-based approach is also expensive, requires experts, and becomes increasingly complex as more slots and actions are incorporated into the set of dialogue states. As an alternative, recent implementations of dialogue managers are more statistical in nature [72, 41]. Dialogue managers built by [86] use a Partially Observable Markov Decision Process (POMDP), in which a reward based on the state of the environment allows the policy to be optimized through reinforcement learning. Thus, a POMDP models uncertainty directly, as opposed to the rule-based approach of representing uncertainty with additional hand-crafted dialogue states. Although we have not done any of this in our current prototype, in the future we plan to investigate a combination of rule-based and statistical methods for dialogue management in the nutrition system.

## **6.2 Ongoing Work**

Although we have built an initial prototype of the system and connected all the pieces, there is still a great deal to accomplish before the system is ready for real-world use. We are moving beyond the language understanding component now, by connecting the USDA nutrition database to the system and designing a user interface for both the web and mobile devices. In the future, we plan to add a bar code scanning feature and employ computer vision object recognition to automatically detect foods and quantities.

### **6.2.1 Designing the User Interface**

A critical component of the nutrition dialogue system is the user interface. As shown in Figure 1-3, the current web prototype simply displays the user's query, highlighted with color-coded semantic tags, and a table with the corresponding food items, quantities, and USDA database hits. Master of Engineering student Patricia Saylor is building a framework for leveraging web technologies in websites such as the nutri-

tion system, including the microphone icon for displaying the audio level [67]. Our undergraduate SuperUROP researchers are currently building a fun, efficient, and user-friendly interface for both the web and tablet. The goal of the interface is to make the system easy to use and rewarding so that users are motivated to log their meals.

### 6.2.2 Connecting It to a Database

Another important component of the nutrition dialogue system, in addition to language understanding and the user interface, is connecting the automatically extracted food concepts to a nutrition database in order to extract relevant nutrition information. This requires determining which database hits match the user’s spoken food concepts, selecting the most similar hit for each food item, transforming the user’s spoken quantity into the standard quantity used by the database, and obtaining the nutrition facts.

In the current system prototype, Master of Engineering student Rachael Naphtal has used the USDA Nutrient Database for Standard Reference and the Nutritionix REST API<sup>1</sup> in order to return a list of potentially matching food items from the database. In her algorithm [53], a series of binary decisions are made to narrow down the search to a short list of matches. If a valid brand is specified, then hits with that brand are more likely to be selected.

One challenge we are addressing is narrowing down a list of the top database hits to one matching food item. If several hits contain similar nutrition information (e.g., a raw apple with skin vs. a raw apple without skin), we simply select one arbitrarily.

In the future, we may explore a re-ranking method for re-ordering the list of hits returned by the database according to how relevant the hit is; thus, the first food product in the re-ranked list is most likely to be the correct match. By presenting the user with the top few hits, the system can learn whether the re-ranked list was correct or whether it made a mistake and must show hits further down in the list. Through such active learning, the system may learn specific users’ preferences as well

---

<sup>1</sup><http://www.nutritionix.com/api>

as improve its re-ranking algorithm in general.

### 6.2.3 Engaging the User in Dialogue

Another critical task that remains is determining how to translate user-described quantities into database quantities. The current solution relies on matching regular expressions to the phrases tagged as quantities in the meal description, as well as the user manually updating the quantity amount and units in the table. To map their quantities automatically, we plan to implement a dialogue manager. For example, if the user says, “This morning I ate a stack of pancakes smothered in butter,” the system must translate “smothered” into tablespoons before the nutrition facts can be retrieved from the database. Thus, the system may ask a follow-up clarification question such as, “How many tablespoons of butter did you eat?”

In a similar manner, dialogue may enable the system to select the correct match from the list of possible food items returned by the nutrition database. For example, if the user neglects to specify a brand for his or her cereal, but the database returns a list of several items that all have different brands (e.g., “Kellogg’s” or “General Mills”), then the system could ask the user to specify a brand.

In addition, the system could learn new out-of-vocabulary words (e.g., pastrami) by asking the user whether an unknown word is a food or brand. Then, if the system learns a new food, it can add it to a food lexicon used by the semantic tagging model so that it will correctly identify the food in the future.

Finally, through dialogue, we can also personalize the system for each user by constructing personal knowledge graphs [40]. For example, the system might ask the user whether they eat the same kind of cereal or milk every day, and if so, save that information so that subsequently the user need not describe their breakfast in as much detail as the first time. Or the system may ask the user whether they have any dietary restrictions, such as vegetarian or gluten-free, which would narrow down the database hits without requiring the user to specify such a description of their food every time they record a meal. The system might even have a conversation with the user, discussing healthier alternatives or suggesting food options that contain

nutrients in which their meals are lacking. Such a system would ultimately enable users, especially patients with obesity or diabetes, to more easily track their meals and improve their health.

#### **6.2.4 Context Resolution**

In this work, we assumed that each user query is a new meal description. However, this will not be the case when real users interact with the dialogue system. They will add followup corrections and refinements (e.g., “No, I had potatoes, not tomatoes”), or even ask open-ended questions (e.g., “How many calories are in an apple?”), and the system will need to be able to handle these queries.

For context resolution, there are two aspects of correctly interpreting followup queries: correctly tagging and segmenting the query, and correctly merging new information with old information. For the first aspect, the language understanding component is able to tag and segment followup queries as is. For the second aspect, we have begun handling followup refinements to previously recorded foods with rule-based heuristics. For example, if the user says, “I drank a glass of milk,” and the default database match is whole milk, the user may follow up with the refinement, “It was almond milk.” For now, we assume that if the subsequent recordings following a meal description contain the same food item that was previously recorded (e.g., “milk”), the table will update the description (e.g., “almond” will replace “whole”).

However, this simplistic approach does not include queries with intents other than recording a new meal description or refining a previously recorded meal. Thus, in the future, we may add a step to the language understanding component where we classify the user intent (e.g., record a new meal, refine a description, correct the system, etc.) and handle each intent differently.

### **6.3 System Evaluation**

In order to evaluate the system’s overall performance on real users, we launched an AMT task where Turkers rated how well the system performed on three separate

tasks: semantic tagging, quantity matching, and correctly identifying USDA (Nutrient Database for Standard Reference)<sup>2</sup> hits for matching foods. Each task cost \$0.10. We asked Turkers to record two meal descriptions each and to interact with the system by revising the quantities and narrowing down the USDA hits to a single option. As shown in Figure 6-2, we added three extra columns to each row with a checkbox for Turkers to check if the system completed the task successfully. In addition, we added two more checks underneath the table for Turkers to indicate whether the semantic tagging component made any substitutions (i.e., labeled a food as something other than a food) or insertions (i.e., labeled a non-food token as a food).



Food	Quantity	USDA Hits	Are the color-coded labels for this food (shown above) correct?	Is the quantity correct?	Is the USDA hit correct?
<p>Oatmeal</p> 	<p>Quantity: <input type="text" value="1"/> <input type="text" value="cup"/></p>	<p>Cereals, oats, regular and quick, not fortified, dry, Calories: 307 Source: USDA • <a href="#">See more options</a></p>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<p>Banana</p> 	<p>Quantity: <input type="text" value="1"/> <input 7-7="" 8"="" long)"="" to="" type="text" value="medium (7"/></p>	<p>Bananas, raw, Calories: 105 Source: USDA • <a href="#">See more options</a></p>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Figure 6-2: An AMT task for evaluating the system’s performance on the sentence “This morning for breakfast I had a bowl of oatmeal followed by a banana.”

The results from 437 meal descriptions containing a total of 975 food concepts indicated that 83% of the semantic tags were correct, 78% of the quantities were

<sup>2</sup><http://ndb.nal.usda.gov/ndb/search>

correct, and 71% of the USDA HITs were correct matches. There were only 34 insertions and 96 substitutions. 355 meal descriptions containing 885 food concepts were omitted due to cheating Turkers who neglected to rate the system on any of the tasks. Table 6.1 gives examples of some of the common errors the system made, such as deleting foods by incorrectly labeling them as descriptions, labeling descriptions as brands, incorrectly mapping user-described quantities to quantities in the table, and failing to find USDA matches for a food (e.g., Lay’s potato chips has no matching hit in the USDA database).

Meal Description	Error	Predict	Correct
2 22 oz <i>beers</i> , a Johnsonville bratwurst with ...	Subst.	Descr.	Food
I had a cup of coffee with no <i>sugar</i>	Insertion	Food	Other
Isagenix IsaLezn Bar - <i>Chocolate Cream Crisp</i>	Tag	Brand	Descr.
16.9 ounces of Trader Joe’s Natural ... Water	Quantity	1 serving	16.9 oz
Lay’s Salt and Vinegar Chips	USDA	None	N/A

Table 6.1: Examples of errors the system made in the AMT task for evaluating performance. There are five error types: substitutions (i.e., labeling a food as a non-food), insertions (labeling non-foods as food items), tagging (i.e., swapping property tags), quantity (i.e., predicting the incorrect database quantity), and USDA (i.e., selecting the incorrect USDA hit).

# Chapter 7

## Conclusion

In this thesis, we have examined the language understanding component of a novel nutrition dialogue system that allows obesity patients to monitor their caloric and nutrient intake more easily and efficiently than existing self-assessment methods. The initial prototype of the system is composed of five parts: a speech recognizer that converts the user’s spoken meal description into text, a language understanding engine that extracts food concepts, a database search that finds the matching foods and their corresponding nutrition facts, and a response generator that responds to the user through text and images. In the future, we plan to incorporate a dialogue manager to enable the system to interact with the user more deeply, by asking follow-up clarification questions and building a repository of personalized knowledge about each user.

### 7.1 Contributions

The methods presented here focused on the data collection and language understanding methods for two components: semantic tagging and food-property association. The primary contributions are as follows.

### **7.1.1 Data Collection**

We gathered 10,000 labeled meal descriptions for training and testing data via Amazon Mechanical Turk (AMT), which we discussed in Chapter 2. The AMT data collection consisted of three tasks: writing a meal as if they were speaking, labeling each food item in a meal, and labeling all the properties describing a specific food in a diary. We identified challenges and designed and implemented novel solutions for cheating prevention. In addition, we launched a task with the deployed nutrition system so that Turkers could interact with the prototype and provide feedback on their user experience.

### **7.1.2 Semantic Tagging**

In Chapter 3, we showed that semi-CRF and standard CRF models are both viable methods for semantic tagging of spoken meal descriptions. Our evaluation of the system’s performance on Amazon Mechanical Turk demonstrated that semantic tagging in the deployed system is reasonably accurate when tested by real users.

### **7.1.3 Distributional Semantics**

We verified in Chapter 4 that incorporating neural network trained dense word embedding features, as well as prototype similarity values, in the CRF classifier improves semantic tagging performance, since according to the theory of distributional semantics, vectors with similar meaning should lie near each other in vector space. In addition, we investigated a novel approach to predicting vectors for unknown words. Our work suggests that with a limited vocabulary, predicting vectors for out-of-vocabulary words improves semantic labeling performance when using dense embedding features.

### **7.1.4 Food-Property Association**

We explored two approaches to the food-property association task in Chapter 5: segmenting meal descriptions into food chunks using an IOE labeling scheme, and



predicting foods for each property using a classifier. In the segmenting approach, we discovered that although transformation-based learning improved upon the Markov model and simple rule baselines, it did not provide gains over the CRF model; rather, the CRF model (with an IOE label representation) yielded the best F1 scores. For the classification approach, we found that the random forest performed better than logistic regression or a naive Bayes classifier. Finally, we found that the segmenting and classification methods are complementary, since incorporating the CRF’s IOE labels as features improved the performance of the random forest classifier in predicting the most likely food for each property.

## 7.2 Directions for Future Research

There are many interesting directions we could take to improve and refine the techniques of the language understanding components in the nutrition system. Here, we describe a few possible extensions of our work in semantic tagging, distributional semantics, and associating foods with properties in meal descriptions.

### 7.2.1 Semantic Tagging

In the future, we may apply coreference resolution techniques or relation extraction methods to the problem of identifying words which refer to the same food. We may also investigate whether combining the CRF with the semi-CRF provides any performance gains. In addition, we will experiment with training our models on all the AMT data annotations, rather than only the majority.

### 7.2.2 Distributional Semantics

In the future, we plan to compare the performance of word embedding features and k-means clusters of vectors to that of Brown clusters, a hierarchical clustering algorithm that has been applied successfully to many NLP tasks [75]. In addition, we would like to explore multiple-sense embeddings [54], since many words have several meanings,

which are not all captured when using a single vector per word. Rather than using word vectors, we could use letter trigram vectors [85], which may help avoid the issue of out-of-vocabulary words. Finally, we could investigate the use of recurrent neural networks (RNNs) [48, 84] or recurrent CRFs [83] as alternatives to the standard CRF.

### 7.2.3 Food-Property Association

In the future, we may jointly model the labeling and segmenting tasks, rather than the three-step process of labeling, then segmenting, and finally using the segmenting results to redo the labeling task. We could also reverse the order of these two phases such that it matches the AMT annotation setup, where food concepts are identified first, and only afterward are the fine-grained properties labeled. Finally, we plan to enable association of properties with more than one food since with the random forest classification approach, only one food is assigned to each property. In the data, 3% of the properties describe multiple foods, so it may be beneficial to select a threshold such that all foods predicted with a probability above the threshold are assigned to the same property.

## 7.3 Looking Forward

In this thesis, we have explored the novel application of spoken language technology to the challenge of reducing user burden for tracking food and nutrient intake, especially for obese individuals who find existing diet applications too tedious. As speech recognition and spoken language understanding research advances, dialogue systems are being applied to more and more domains, including health and education, which can have a great impact on society. Personal assistants such as Siri are even able to handle multiple domains at once. We envision a future where we will be able to converse with computers in every aspect of our lives, from simple flight booking and financial transactions to diet tracking and medical diagnosis, where each dialogue will be tailored to a specific user, and information from previous conversations will be stored for future reference. The future of technology is exciting, and speech and

language understanding will be at the forefront.



# Appendix A

## Training CRFs

### A.1 Inference

Training a CRF involves two parts: parameter estimation (i.e., finding the set of parameters  $\theta$  such that the resulting distribution  $Pr(\mathbf{y}|\mathbf{x}, \theta)$  best fits the training data) and probabilistic inference (i.e., computing the marginal distributions  $Pr(\mathbf{y}_a|\mathbf{x}, \theta)$  over a subset of output variables, as well as computing the most likely labeling  $\mathbf{y}^* = \operatorname{argmax}_{\mathbf{y}} Pr(\mathbf{y}|\mathbf{x}, \theta)$  of a new input  $\mathbf{x}$ ). Here, we will focus on linear-chain CRFs, since they are simple enough that inference can be exact, and they are used for sequence labeling tasks such as our food and property semantic tagging task.

Computing the node and edge marginals is important because they are used in the parameter estimation calculation. The forward-backward algorithm is used for computing marginals, and the Viterbi algorithm is used for computing the most likely output labels for a new input sequence.

First, we introduce new notation for a CRF, where we define transition weights

$$\Psi(y_t, y_{t-1}, \mathbf{x}_t) = \exp\left\{\sum_{k=1}^K \theta_k f_k(y_t, y_{t-1}, \mathbf{x}_t)\right\}. \quad (\text{A.1})$$

The idea behind the forward-backward algorithm is that we can save computation by caching inner sums in order to compute the outer sums for the probability  $p(\mathbf{x})$  of the observed data. Plugging in the transition weights, we define forward variables

(representing intermediate sums) as

$$\alpha_t(j) = \sum_{\mathbf{y}_{\langle 1 \dots t-1 \rangle}} \Psi_t(j, y_{t-1}, x_t) \prod_{t'=1}^{t-1} \Psi_{t'}(y_{t'}, y_{t'-1}, x_{t'}) \quad (\text{A.2})$$

which is calculated recursively

$$= \sum_{i \in S} \Psi_t(j, i, x_t) \alpha_{t-1}(i). \quad (\text{A.3})$$

The backward weights are defined similarly, except that the summations are in the reverse order

$$\beta_t(i) = \sum_{\mathbf{y}_{\langle t+1 \dots T \rangle}} \prod_{t'=t+1}^T \Psi_{t'}(y_{t'}, y_{t'-1}, x_{t'}) \quad (\text{A.4})$$

which is again calculated recursively

$$= \sum_{j \in S} \Psi_{t+1}(j, i, x_{t+1}) \beta_{t+1}(j). \quad (\text{A.5})$$

After computing the forward and backward variables  $\alpha$  and  $\beta$ , the marginals are calculated as follows

$$Pr(y_{t-1}, y_t | \mathbf{x}) = \frac{\alpha_{t-1}(y_{t-1}) \Psi_t(y_t, y_{t-1}, x_t) \beta_t(y_t)}{Z(\mathbf{x})} \quad (\text{A.6})$$

$$Pr(y_t | \mathbf{x}) = \frac{\alpha_t(y_t) \beta_t(y_t)}{Z(\mathbf{x})}. \quad (\text{A.7})$$

Finally, we can compute the optimal assignment  $\mathbf{y}^* = \operatorname{argmax}_{\mathbf{y}} Pr(\mathbf{y} | \mathbf{x}, \theta)$  using the Viterbi algorithm by simply replacing the summations in the forward-backward algorithm with maximizations. Instead of  $\alpha$ , we have the analogous

$$\delta_t(j) = \max_{\mathbf{y}_{\langle 1 \dots t-1 \rangle}} \Psi_t(j, y_{t-1}, x_t) \prod_{t'=1}^{t-1} \Psi_{t'}(y_{t'}, y_{t'-1}, x_{t'}) \quad (\text{A.8})$$

which is similarly calculated recursively

$$\delta_t(j) = \max_{i \in S} \Psi_t(j, i, x_t) \delta_{t-1}(i). \quad (\text{A.9})$$

Then, the maximizing assignment is computed through a backwards recursion

$$y_T^* = \operatorname{argmax}_{i \in S} \delta_T(i) \quad (\text{A.10})$$

and

$$y_t^* = \operatorname{argmax}_{i \in S} \Psi_t(y_{t+1}^*, i, x_{t+1}) \delta_t(i) \quad (\text{A.11})$$

for  $t < T$ .

The most commonly used exact inference algorithm is the junction tree algorithm, which successively groups variables until the graph becomes a tree and tree-specific inference algorithms can be applied. In practice, though, approximate inference algorithms are often used because parameter estimation requires performing inference many times, so an efficient training algorithm is crucial. Two types of approximation algorithms are used: Monte Carlo and variational. Monte Carlo algorithms are stochastic algorithms that produce samples from the distribution of interest. Variational algorithms convert inference into an optimization problem by attempting to find an approximation of the marginals of interest. Although variational algorithms are biased, they are also fast, which makes them more useful for CRFs.

## A.2 Parameter Estimation

In linear-chain CRFs, parameter estimation is computed using numerical optimization methods. We are given independent and identically distributed (iid) data  $\mathcal{D} = \{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^N$ , where  $\mathbf{x}^{(i)} = \{\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}, \dots, \mathbf{x}_T^{(i)}\}$  is a sequence of inputs, and  $\{\mathbf{y}^{(i)} = \{y_1^{(i)}, y_2^{(i)}, \dots, y_T^{(i)}\}$  is a sequence of predicted output labels. If we set an objective function  $l(\theta)$ , then we can find the optimal parameter setting by calculating the maximum likelihood  $\hat{\theta}_{ML}$ , which corresponds to the parameters under which the observed data

is most likely. As our objective function, we use the conditional log likelihood

$$l(\theta) = \sum_{i=1}^N \log \text{Pr}(\mathbf{y}^{(i)} | \mathbf{x}^{(i)}; \theta). \quad (\text{A.12})$$

To compute the maximum likelihood estimate, we maximize  $l(\theta)$ . To do this, we first substitute the CRF model in equation 3.1 into the likelihood:

$$l(\theta) = \sum_{i=1}^N \log \left[ \frac{1}{Z(\mathbf{x}^{(i)})} \prod_{t=1}^T \exp \left\{ \sum_{k=1}^K \theta_k f_k(y_t^{(i)}, y_{t-1}^{(i)}, \mathbf{x}_t^{(i)}) \right\} \right] \quad (\text{A.13})$$

$$= \sum_{i=1}^N \log \left[ \prod_{t=1}^T \exp \left\{ \sum_{k=1}^K \theta_k f_k(y_t^{(i)}, y_{t-1}^{(i)}, \mathbf{x}_t^{(i)}) \right\} \right] + \sum_{i=1}^N \log \frac{1}{Z(\mathbf{x}^{(i)})} \quad (\text{A.14})$$

$$= \sum_{i=1}^N \sum_{t=1}^T \sum_{k=1}^K \theta_k f_k(y_t^{(i)}, y_{t-1}^{(i)}, \mathbf{x}_t^{(i)}) - \sum_{i=1}^N \log Z(\mathbf{x}^{(i)}). \quad (\text{A.15})$$

Next we introduce a regularization term, which prevents the model from overfitting when there are many parameters. A penalty based on the Euclidean norm of  $\theta$  is applied, and a regularization parameter  $\frac{1}{2\sigma^2}$  determines the strength of the penalty:

$$l(\theta) = \sum_{i=1}^N \sum_{t=1}^T \sum_{k=1}^K \theta_k f_k(y_t^{(i)}, y_{t-1}^{(i)}, \mathbf{x}_t^{(i)}) - \sum_{i=1}^N \log Z(\mathbf{x}^{(i)}) - \sum_{k=1}^K \frac{\theta_k^2}{2\sigma^2}. \quad (\text{A.16})$$

Next we take the partial derivatives of the likelihood with respect to each  $\theta_k$ , plugging in for the partition function  $Z$ :

$$\frac{\partial l}{\partial \theta_k} = \sum_{i=1}^N \sum_{t=1}^T f_k(y_t^{(i)}, y_{t-1}^{(i)}, \mathbf{x}_t^{(i)}) - \frac{\partial}{\partial \theta_k} \sum_{i=1}^N \log \sum_{\mathbf{y}} \prod_{t=1}^T \exp \left\{ \sum_{k=1}^K \theta_k f_k(y_t^{(i)}, y_{t-1}^{(i)}, \mathbf{x}_t^{(i)}) \right\} - \frac{\theta_k}{\sigma^2} \quad (\text{A.17})$$

$$= \sum_{i=1}^N \sum_{t=1}^T f_k(y_t^{(i)}, y_{t-1}^{(i)}, \mathbf{x}_t^{(i)}) - \sum_{i=1}^N \sum_{\mathbf{y}} \log \prod_{t=1}^T \exp \{ f_k(y_t^{(i)}, y_{t-1}^{(i)}, \mathbf{x}_t^{(i)}) \} - \frac{\theta_k}{\sigma^2} \quad (\text{A.18})$$



$$= \sum_{i=1}^N \sum_{t=1}^T f_k(y_t^{(i)}, y_{t-1}^{(i)}, \mathbf{x}_t^{(i)}) - \sum_{i=1}^N \sum_{t=1}^T \sum_{y, y'} f_k(y, y', \mathbf{x}_t^{(i)}) Pr(y, y' | \mathbf{x}^{(i)}) - \frac{\theta_k}{\sigma^2}. \quad (\text{A.19})$$

Since the partial derivatives contain marginal distributions, we must run inference for each training sample every time the likelihood is computed. Since this objective function is strictly concave, there is exactly one global optimum. Common optimization techniques such as conjugate gradient and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm approximate second-order information.



# Appendix B

## Measurements and Calculations

### B.1 Kappa Score

Cohen’s kappa score is used for determining the agreement between two annotators, whereas Fleiss’ kappa score is a statistical measure of the agreement among any fixed number of raters giving categorical ratings [77]. It also assumes different items are rated by different individuals. Fleiss’ kappa is more useful in our case because multiple Turkers annotated each meal description (five each for food labeling and three each for property labeling), assigning the categories “Food” and “Other” in the food labeling task and the categories “Brand,” “Quantity,” “Description,” and “Other” in the property labeling task.

Fleiss’ kappa, which expresses the amount of agreement among the annotators exceeding the agreement that would be present if the annotations were all made randomly, is calculated as

$$\kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e} \quad (\text{B.1})$$

where  $1 - \bar{P}_e$  is the possible agreement above chance, and  $\bar{P} - \bar{P}_e$  is the actually observed agreement above chance. A kappa score of one indicates perfect agreement, whereas a score below zero indicates less than chance agreement.  $\bar{P}$  is the mean of each  $P_i$ , where  $P_i$  is the agreement for word  $i$  and is computed as

$$P_i = \frac{1}{n(n-1)} \sum_{j=1}^k n_{ij}(n_{ij} - 1). \quad (\text{B.2})$$

$\bar{P}_e$  is the sum of each  $P_j$  squared, where  $P_j$  is the proportion of words assigned to category  $j$  and is calculated as

$$P_j = \frac{1}{Nn} \sum_{i=1}^N n_{ij}. \quad (\text{B.3})$$

$N$  is the total number of words that were labeled by Turkers,  $n$  is the number of ratings per word, and  $k$  is the number of possible categories.

## B.2 F1 Score

For evaluation, we calculated precision (i.e., the fraction of predicted labels that were correct), recall (i.e., the fraction of gold standard labels that were predicted), and F1 (i.e., the harmonic mean of precision and recall) scores for each approach:

$$\text{precision} = \frac{\text{numCorrect}}{\text{numPredict}} \quad (\text{B.4})$$

$$\text{recall} = \frac{\text{numCorrect}}{\text{numGold}} \quad (\text{B.5})$$

$$\text{F1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad (\text{B.6})$$

## B.3 Significance Tests

When comparing the performance of two methods, it is important to use statistical tests in order to test whether the difference is significant; that is, we aim to show that the difference in performance indicates that the two methods are indeed unequal and thus that one yields better performance than the other. We call the case where two methods are equal the null hypothesis  $H_0$ , and the alternate hypothesis (that the two

methods are unequal)  $H_1$ . If we can show the probability that the null hypothesis holds is smaller than some threshold  $\alpha$ , then we may reject it in favor of  $H_1$ .

One common approach for testing the significance of the difference between two methods is the Student's  $t$  test, which is a likelihood ratio test based on the variable  $t^2$  of the Student's  $t$  distribution (shown in Figure B-1). The likelihood ratio test allows us to reject  $H_0$  if the ratio of the likelihood of the observed result given  $H_0$ ,  $L(X|H_0)$ , over the likelihood of the observation given  $H_1$ ,  $L(X|H_1)$ , is small enough [32]. More formally, the likelihood ratio  $\lambda = \frac{L(X|H_0)}{L(X|H_1)}$ , where we reject  $H_0$  if  $\lambda \leq \lambda_0$ , and the threshold  $\lambda_0$  is chosen such that  $P(\lambda \leq \lambda_0|H_0) \leq \alpha$ .

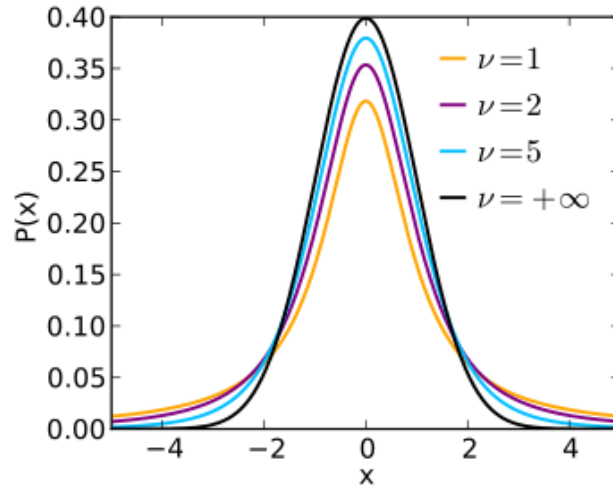


Figure B-1: Probability density function of the Student's  $t$  distribution for varying degrees of freedom  $v$ .

In the Student's  $t$  test, the likelihood ratio is equivalent to a test based on the variable  $t^2$  of the Student's  $t$  distribution, so we can compute the  $t$  statistic in order to test significance. The critical region where we reject the null hypothesis consists of the two tail regions of the  $t$  distribution with  $n - 1$  degrees of freedom, where  $n$  is the number of samples, such that  $|t| > t_0$  and  $t_0$  is chosen so that  $P(t > t_0) = \alpha/2$ . We can calculate the  $t$  statistic for several different scenarios, including equal or unequal sample sizes and equal or unequal variances. In general, the equation for  $t$  is the difference between the two means, divided by the standard error of this difference.

For unequal sample sizes and variances, this becomes

$$t = \frac{\bar{X}_1 - \bar{X}_2}{s_{\bar{X}_1 - \bar{X}_2}}, \quad (\text{B.7})$$

where  $\bar{X}_1$  and  $\bar{X}_2$  are the means and  $s_{\bar{X}_1 - \bar{X}_2}$  is the standard error

$$s_{\bar{X}_1 - \bar{X}_2} = \sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}, \quad (\text{B.8})$$

where  $n_1$  and  $n_2$  are the sample sizes, and  $s_1^2$  and  $s_2^2$  are the variances.

An example from natural language processing (NLP) where we might apply the paired  $t$  test is an information retrieval task. The goal is, given a user’s search query, to retrieve the most relevant document containing the answer to the user’s query. We can measure the success of the task by assigning a precision score to each query; we can do this for several methods and compare their performance using the  $t$  significance test. In this case, the  $t$  test is a reasonable choice because each method results in a set of  $n$  precision scores, where  $n$  is the number of sample queries. We can then calculate the mean and standard deviation of the precision scores for different approaches and measure the significance using the  $t$  statistic.

However, in our case, the semantic tagging task consists of assigning a label to each token in a set of food diaries, and each label is either correct or incorrect. The  $t$  test does not apply on the token level because there is no set of values from which to take the mean and standard deviation. It could apply if we were to calculate the accuracy of labels per food diary, since we could then use those scores to get a mean and standard deviation. However, this is not as fine-grained an approach as McNemar’s significance test, which allows us to compare the labeling performance on each token directly [22].

In McNemar’s significance test, we evaluate the performance of various methods on each token, checking whether or not each method labeled the token correctly or incorrectly. The algorithm collapses the results down to a 2x2 matrix, as shown in Table B.1, which lists the number of tokens labeled correctly or incorrectly by both

methods along the diagonal ( $n_{00}$  and  $n_{11}$  respectively) and the number of tokens labeled correctly by one method, but incorrectly by the other, on the off-diagonal (i.e.,  $n_{01}$  and  $n_{10}$ ).

Method 1/Method 2	Correct	Incorrect
Correct	$n_{00}$	$n_{01}$
Incorrect	$n_{10}$	$n_{11}$

Table B.1: McNemar’s matrix of tokens labeled correctly or incorrectly by two methods.

Using the values in this 2x2 matrix, we can test the null hypothesis  $H_0$  that the off-diagonals are equal, i.e.,  $n_{01} = n_{10}$ , or equivalently, whether  $\frac{n_{01}}{n_{01}+n_{10}} = \frac{1}{2}$ . Given  $H_0$ , the probability of observing  $k$  tokens classified asymmetrically (i.e., classified correctly by one method but not the other) has a binomial probability mass function (PMF) as shown in Fig. B-2

$$P(k) = \binom{n}{k} \left(\frac{1}{2}\right)^n, \tag{B.9}$$

where there are  $n = n_{01} + n_{10}$  tokens in total. We then test for significance by checking whether  $P < \alpha$ . We can compute the probability  $P$  by summing the tails of the PMF distribution

$$P = \sum_{k=0}^l P(k) + \sum_{k=m}^n P(k), \tag{B.10}$$

where  $l = \min(n_{01}, n_{10})$  and  $m = \max(n_{01}, n_{10})$ .

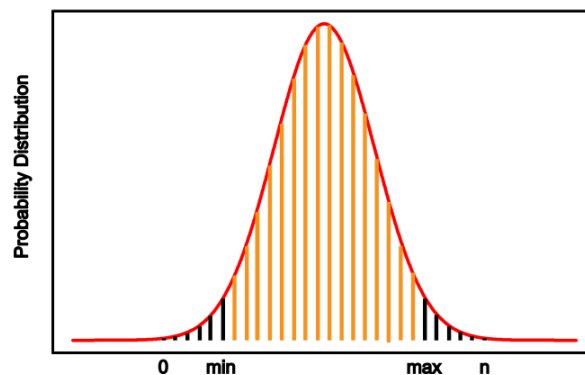


Figure B-2: Probability mass function of the binomial distribution, where the tails are used for McNemar’s significance test [23].





# Bibliography

- [1] R. Al-Rfou, B. Perozzi, and S. Skiena. Polyglot: Distributed word representations for multilingual NLP. *arXiv preprint arXiv:1307.1662*, 2013.
- [2] M. Bansal, K. Gimpel, and K. Livescu. Tailoring continuous word representations for dependency parsing. In *Proc. ACL*, 2014.
- [3] M. Baroni, G. Dinu, and G. Kruszewski. Dont count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proc. ACL*, volume 1, pages 238–247, 2014.
- [4] I. Bazzi. *Modelling Out-of-vocabulary Words for Robust Speech Recognition*. PhD thesis, Massachusetts Institute of Technology, 2002.
- [5] J. Berant and P. Liang. Semantic parsing via paraphrasing. In *Proc. ACL*, volume 7, page 92, 2014.
- [6] C. Bishop et al. *Pattern Recognition and Machine Learning*, volume 1. Springer New York, 2006.
- [7] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
- [8] A. Celikyilmaz, D. Hakkani-Tur, P. Pasupat, and R. Sarikaya. Enriching word embeddings using knowledge graph for semantic tagging in conversational dialog systems. *genre*, 2010.
- [9] J. Chen, N. Menezes, A. Bradley, and T. North. Opportunities for crowdsourcing research on Amazon Mechanical Turk. *Interfaces*, 5(3), 2011.
- [10] Q. Chen, C. Lei, W. Xu, E. Pavlick, and C. Callison-Burch. Poetry of the crowd: A human computation algorithm to convert prose into rhyming verse. In *Second AAAI Conference on Human Computation and Crowdsourcing*, 2014.
- [11] Y. Chen and A. Rudnicky. Dynamically supporting unexplored domains in conversational interactions by enriching semantics with neural word embeddings. *Proc. SLT*, 2014.
- [12] Y. Chen, W. Wang, and A. Rudnicky. Leveraging frame semantics and distributional semantics for unsupervised semantic slot induction in spoken dialogue systems.

- [13] Y. Chen, W. Wang, and A. Rudnicky. Unsupervised induction and filling of semantic slots for spoken dialogue systems using frame-semantic parsing. In *Proc. ASRU*, pages 120–125, 2013.
- [14] DM Conway. An algorithmic approach to English pluralization. In *Proceedings of the Second Annual Perl Conference. C. Salzenberg. San Jose, CA, O’Reilly*, 1998.
- [15] M. De Marneffe, B. MacCartney, C. Manning, et al. Generating typed dependency parses from phrase structure parses. In *Proc. LREC*, volume 6, pages 449–454, 2006.
- [16] A. Deoras and R. Sarikaya. Deep belief network based semantic taggers for spoken language understanding. In *Proc. INTERSPEECH*, pages 2713–2717, 2013.
- [17] L. Devillers, L. Lamel, and I. Vasilescu. Emotion detection in task-oriented spoken dialogues. In *Multimedia and Expo, 2003. ICME’03. Proceedings. 2003 International Conference on*, volume 3, pages III–549. IEEE, 2003.
- [18] J. Finkel, A. Kleeman, and C. Manning. Efficient, feature-based, conditional random field parsing. In *ACL*, volume 46, pages 959–967, 2008.
- [19] E. Finkelstein, J. Trogdon, J. Cohen, and W. Dietz. Annual medical spending attributable to obesity: Payer-and service-specific estimates. *Health affairs*, 28(5):w822–w831, 2009.
- [20] R. Florian and G. Ngai. Fast transformation-based learning toolkit. *Johns Hopkins University, USA*, 2001.
- [21] S. Gebhardt, L. Lemar, D. Haytowitz, P. Pehrsson, M. Nickle, B. Showell, R. Thomas, J. Exler, and J. Holden. USDA national nutrient database for standard reference, release 21. 2008.
- [22] L. Gillick and S. Cox. Some statistical issues in the comparison of speech recognition algorithms. In *Proc. ICASSP*, pages 532–535, 1989.
- [23] J. Glass and V. Zue. MIT open courseware: Automatic speech recognition (6.345), 2003.
- [24] A. Gorin, G. Riccardi, and J. Wright. How may I help you? *Speech communication*, 23(1):113–127, 1997.
- [25] A. Graesser, K. VanLehn, C. Rosé, P. Jordan, and D. Harter. Intelligent tutoring systems with conversational dialogue. *AI magazine*, 22(4):39, 2001.
- [26] D. Guo, G. Tur, W. Yih, and G. Zweig. Joint semantic utterance classification and slot filling with recursive neural networks.

- [27] J. Guo, W. Che, H. Wang, and T. Liu. Revisiting embedding features for simple semi-supervised learning. In *Proc. EMNLP*, pages 110–120, 2014.
- [28] J. Guo, W. Che, H. Wang, and T. Liu. Revisiting embedding features for simple semi-supervised learning. In *Proc. EMNLP*, pages 110–120, 2014.
- [29] S. Hahn, P. Lehnen, C. Raymond, and H. Ney. A comparison of various methods for concept tagging for spoken language understanding. In *LREC*, 2008.
- [30] X. He, R. Zemel, and M. Carreira-Perpindn. Multiscale conditional random fields for image labeling. In *Computer vision and pattern recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE computer society conference on*, volume 2, pages II–695. IEEE, 2004.
- [31] I. Hetherington. The MIT finite-state transducer toolkit for speech and language processing. In *Proc. INTERSPEECH*, 2004.
- [32] P. Hoel, S. Port, and C. Stone. *Introduction to Statistical Theory*. Houghton Mifflin Boston, 1971.
- [33] J. Ingber. My fitness pal: A guide to an accessible fitness tool. 2014.
- [34] M. Iyyer, J. Boyd-Graber, L. Claudino, R. Socher, and H. III. A neural network for factoid question answering over paragraphs. In *Proc. EMNLP*, pages 633–644, 2014.
- [35] K. Jokinen and M. McTear. Spoken dialogue systems. *Synthesis Lectures on Human Language Technologies*, 2(1):1–151, 2009.
- [36] M. Korpusik, N. Schmidt, J. Drexler, S. Cyphers, and J. Glass. Data collection and language understanding of food descriptions. *Proc. SLT*, 2014.
- [37] T. Kudo. CRF++: Yet another CRF toolkit. *Software available at <http://crfpp.sourceforge.net>*, 2005.
- [38] C. Lee and J. Glass. A transcription task for crowdsourcing with automatic quality control. In *Interspeech*, pages 3041–3044. Citeseer, 2011.
- [39] X. Li. Understanding the semantic structure of noun phrase queries. In *Proc. ACL*, pages 1337–1345. Association for Computational Linguistics, 2010.
- [40] X. Li, G. Tur, D. Hakkani-Tur, and Q. Li. Personal knowledge graph population from user utterances in conversational understanding. In *Proc. SLT*, pages 224–229. IEEE, 2014.
- [41] J. Liu, P. Pasupat, S. Cyphers, and J. Glass. Asgard: A portable architecture for multilingual dialogue systems. In *Proc. ICASSP*, pages 8386–8390, 2013.

- [42] J. Liu, P. Pasupat, Y. Wang, S. Cyphers, and J. Glass. Query understanding enhanced by hierarchical parsing structures. In *Proc. ASRU*, pages 72–77. IEEE, 2013.
- [43] J. Liu and S. Seneff. A dialogue system for accessing drug reviews. In *Proc. ASRU*, pages 324–329, 2011.
- [44] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky. The stanford CoreNLP natural language processing toolkit. In *Proc. ACL: System Demonstrations*, pages 55–60, 2014.
- [45] A. McCallum, K. Bellare, and F. Pereira. A conditional random field for discriminatively-trained finite-state string edit distance. *arXiv preprint arXiv:1207.1406*, 2012.
- [46] Andrew Kachites McCallum. MALLET: A machine learning for language toolkit. <http://mallet.cs.umass.edu>, 2002.
- [47] I. McGraw, S. Cyphers, P. Pasupat, J. Liu, and J. Glass. Automating crowd-supervised learning for spoken language systems. In *Proc. INTERSPEECH*, 2012.
- [48] G. Mesnil, X. He, L. Deng, and Y. Bengio. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Proc. INTERSPEECH*, pages 3771–3775, 2013.
- [49] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
- [50] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.
- [51] T. Mikolov, W. Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751, 2013.
- [52] G. Miller and W. Charles. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28, 1991.
- [53] R. Naphtal. *Natural Language Processing Based Nutritional Application*. PhD thesis, Massachusetts Institute of Technology, 2015.
- [54] A. Neelakantan, J. Shankar, A. Passos, and A. McCallum. Efficient nonparametric estimation of multiple embeddings per word in vector space. In *Proc. EMNLP*, 2014.
- [55] C. Ogden, M. Carroll, B. Kit, and K. Flegal. Prevalence of childhood and adult obesity in the United States, 2011-2012. *Jama*, 311(8):806–814, 2014.

- [56] Naoaki Okazaki. CRFsuite: A fast implementation of conditional random fields (CRFs), 2007.
- [57] World Health Organization. *Obesity: Preventing and Managing the Global Epidemic*. Number 894. World Health Organization, 2000.
- [58] K. Papineni, S. Roukos, and R. Ward. Natural language task-oriented dialog manager and method, June 12 2001. US Patent 6,246,981.
- [59] E. Pavlick, R. Yan, and C. Callison-Burch. Crowdsourcing for grammatical error correction. In *Proceedings of the companion publication of the 17th ACM conference on Computer supported cooperative work & social computing*, pages 209–212. ACM, 2014.
- [60] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [61] J. Pennington, R. Socher, and C. Manning. GloVe: Global vectors for word representation. *Proc. EMNLP*, 12, 2014.
- [62] M. Porter. An algorithm for suffix stripping. *Program: Electronic Library and Information Systems*, 14(3):130–137, 1980.
- [63] A. Raux and M. Eskenazi. Using task-oriented spoken dialogue systems for language learning: Potential, practical applications and challenges. In *INSTIL/ICALL Symposium 2004*, 2004.
- [64] M. Rollo, S. Ash, P. Lyons-Wall, and A. Russell. Trial of a mobile phone method for recording dietary intake in adults with type 2 diabetes: Evaluation and implications for future applications. *Journal of telemedicine and telecare*, 17(6):318–323, 2011.
- [65] S. Sarawagi and W. Cohen. Semi-markov conditional random fields for information extraction. In *Proc. NIPS*, pages 1185–1192, 2004.
- [66] R. Sarikaya, G. Hinton, and B. Ramabhadran. Deep belief nets for natural language call-routing. In *Proc. ICASSP*, pages 5680–5683, 2011.
- [67] P. Saylor. *Spoke: A Framework for Building Speech-Enabled Websites*. PhD thesis, Massachusetts Institute of Technology, 2015.
- [68] H. Schütze. Word space. In *Advances in Neural Information Processing Systems 5*. Citeseer, 1993.
- [69] S. Seneff and J. Polifroni. Dialogue management in the Mercury flight reservation system. In *Proc. ANLP/NAACL Workshop on Conversational systems-Volume 3*, pages 11–16. Association for Computational Linguistics, 2000.

- [70] C. Sutton and A. McCallum. An introduction to conditional random fields for relational learning. *Introduction to statistical relational learning*, pages 93–128, 2006.
- [71] C. Sutton and A. McCallum. An introduction to conditional random fields. *arXiv preprint arXiv:1011.4088*, 2010.
- [72] B. Thomson. *Statistical Methods for Spoken Dialogue Management*. Springer, 2013.
- [73] K. Toutanova, D. Klein, C. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics, 2003.
- [74] C. Tsai, G. Lee, F. Raab, G. Norman, T. Sohn, W. Griswold, and K. Patrick. Usability and feasibility of PmEB: A mobile phone application for monitoring real time caloric balance. *Mobile networks and applications*, 12(2-3):173–184, 2007.
- [75] J. Turian, L. Ratinov, and Y. Bengio. Word representations: A simple and general method for semi-supervised learning. In *Proc. ACL*, pages 384–394. Association for Computational Linguistics, 2010.
- [76] L. Van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.
- [77] A. Viera, J. Garrett, et al. Understanding interobserver agreement: The kappa statistic. *Family Medicine*, 37(5):360–363, 2005.
- [78] Y. Wang and M. Beydoun. 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, 2007.
- [79] Y. Wang, K. Loe, and J. Wu. A dynamic conditional random field model for foreground and shadow segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(2):279–289, 2006.
- [80] M. Worsham. Calorie count iPhone application offers first speech recognition tool to the health and wellness industry, along with redesigned food-logging platform. 2011.
- [81] P. Xu and R. Sarikaya. Convolutional neural network based triangular CRF for joint intent detection and slot filling. In *Proc. ASRU*, pages 78–83, 2013.
- [82] R. Yan, M. Gao, E. Pavlick, and C. Callison-Burch. Are two heads better than one? Crowdsourced translation via a two-step collaboration of non-professional translators and editors. In *The 52nd Annual Meeting of the Association of Computational Linguistics*, 2014.

- [83] K. Yao, B. Peng, G. Zweig, D. Yu, X. Li, and F. Gao. Recurrent conditional random field for language understanding. In *Proc. ICASSP*, pages 4077–4081. IEEE, 2014.
- [84] K. Yao, G. Zweig, M. Hwang, Y. Shi, and D. Yu. Recurrent neural networks for language understanding. In *INTERSPEECH*, pages 2524–2528, 2013.
- [85] W. Yih, X. He, and C. Meek. Semantic parsing for single-relation question answering. In *Proc. ACL*, 2014.
- [86] S. Young, M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu. The hidden information state model: A practical framework for POMDP-based spoken dialogue management. *Computer Speech & Language*, 24(2):150–174, 2010.
- [87] H. Zhao, C. Huang, and M. Li. An improved Chinese word segmentation system with conditional random field. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, volume 1082117. Sydney: July, 2006.
- [88] V. Zue, S. Seneff, J. Glass, J. Polifroni, C. Pao, T. Hazen, and L. Hetherington. JUPITER: A telephone-based conversational interface for weather information. *Speech and Audio Processing, IEEE Transactions on*, 8(1):85–96, 2000.