

**TAGGING AND ACTION GRAPH GENERATION FOR
RECIPES**

YEMEK TARİFLERİNİ ETİKETLEME VE ÇİZGE ÜRETİMİ

Mehmet ÖZGEN

Prof. Dr. Pınar Duygulu ŞAHİN

Supervisor

Asst. Prof. Dr. Gönenc ERCAN

Co-advisor

Submitted to Graduate School of Science and Engineering of
Hacettepe University
as a Partial Fulfillment to the Requirements
for the Award of the Degree of Master of Science
in Computer Engineering

2019

This work titled "Tagging and Action Graph Generation For Recipes" by MEHMET ÖZGEN has been approved as a thesis for the Degree of Master of Science in Computer Engineering by the below mentioned Examining Committee Members.

Prof. Dr. Pınar KARAGÖZ

Head



Prof. Dr. İlyas ÇİÇEKLI

Member



Prof. Dr. Pınar DUYGULU ŞAHİN

Supervisor



Asst. Prof. Dr. Burcu CAN BUĞLALILAR

Member



Assoc. Prof. Dr. Lale ÖZKAHYA

Member



This thesis has been approved as a thesis for the Degree of Master of Science in Computer Engineering by Board of Directors of the Institute for Graduate School of Science and Engineering on/.../2019.

Prof. Dr. Menemşe GÜMÜŞDERELİOĞLU

Director of the Institute of
Graduate School of Science and Engineering

ETHICS

In this thesis study, prepared in accordance with the spelling rules of Institute of Graduate School of Science and Engineering of Hacettepe University,

I declare that

- all the information and documents have been obtained in the base of the academic rules.
- all audio-visual and written information and results have been presented according to the rules of scientific ethics
- in case of using others works, related studies have been cited in accordance with the scientific standards
- all cited studies have been fully referenced
- I did not do any distortion in the dataset
- and any part of this thesis has not been presented as another thesis study at this or any other university.

09/09/2019



Mehmet ÖZGEN

YAYINLANMA FİKRİ MÜLKİYET HAKKLARI BEYANI

Enstitü tarafından onaylanan lisansüstü tezimin/raporumun tamamını veya herhangi bir kısmını, basılı (kağıt) ve elektronik formatta arşivleme ve aşağıda verilen koşullarla kullanıma açma iznini Hacettepe üniversitesine verdiğimi bildiririm. Bu izinle Üniversiteye verilen kullanım hakları dışındaki tüm fikri mülkiyet haklarım bende kalacak, tezimin tamamının ya da bir bölümünün gelecekteki çalışmalarda (makale, kitap, lisans ve patent vb.) kullanım hakları bana ait olacaktır.

Tezin kendi orijinal çalışmam olduğunu, başkalarının haklarını ihlal etmediğimi ve tezimin tek yetkili sahibi olduğumu beyan ve taahhüt ederim. Tezimde yer alan telif hakkı bulunan ve sahiplerinden yazılı izin alınarak kullanması zorunlu metinlerin yazılı izin alarak kullandığımı ve istenildiğinde suretlerini Üniversiteye teslim etmeyi taahhüt ederim.

Yükseköğretim Kurulu tarafından yayınlanan *“Lisansüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge”* kapsamında tezim aşağıda belirtilen koşullar haricince YÖK Ulusal Tez Merkezi / H. Ü. Kütüphaneleri Açık Erişim Sisteminde erişime açılır.

- Enstitü / Fakülte yönetim kurulu kararı ile tezimin erişime açılması mezuniyet tarihimden itibaren 2 yıl ertelenmiştir.
- Enstitü / Fakülte yönetim kurulu gerekçeli kararı ile tezimin erişime açılması mezuniyet tarihimden itibaren ay ertelenmiştir.
- Tezim ile ilgili gizlilik kararı verilmiştir.

09/09/2019


Mehmet ÖZGEN

ABSTRACT

Tagging and Action Graph Generation For Recipes

Mehmet ÖZGEN

Master of Science, Computer Engineering Department

Supervisor: Prof. Dr. Pınar Duygulu ŞAHİN

Co-advisor: Asst. Prof. Dr. Gönenç ERCAN

09/09/2019, 76 pages

Processing instructions is significant to accomplish daily tasks. Instructions can be found in many different forms for a variety tasks. Machine understanding of instructions, similarly, can be beneficial for artificial agents/robots to perform a task automatically. Building systems checking if a task is carried-out in conformity with the instructions is important for many mission critical tasks, for instance factories, workers who repair the electronic devices etc.

In this thesis, it is aimed to automatically extract the steps of a certain task with the aid of instructions. Instructions dataset is needed to train model and extract the steps of a task. Recipes are the examples of instructions that are easy to follow and can be found in large quantities. Understanding of how to cook a recipe step by step requires extraction of course of actions, ingredients, tools and, the relationships between each other through Natural Language Processing (NLP) Techniques.

Supervised and rule-based model is proposed to clarify and extract actions and components. Instead of a fully supervised method, NLP Techniques are used to find relations between components and actions in the text. The workflow of the recipes are finally produced by a rule-based method. When compared to a state-of-the-art unsupervised method which models

the task as a whole, the proposed method benefits from the output of smaller and well-studied NLP Techniques.

Keywords: text mining, part of speech tagging, word embedding, collocation, conditional random fields (CRF), semantic, long short-term memory (LSTM), deep learning

ÖZET

Yemek Tariflerinin Etiketlenmesi ve Çizge Üretimi

Mehmet ÖZGEN

Yüksek Lisans, Bilgisayar Mühendisliği

Danışman: Prof. Dr. Pınar Duygulu ŞAHİN

Yrd. Danışman: Yrd. Doç. Dr. Gönenç ERCAN

09/09/2019, 76 Sayfa

Hayatın içinde birçok farklı formda bulunan talimatlar günlük hayattaki işlerimizi yerine getirebilmek için önemlidir. Bu talimatların makinelerle öğretilerek otomatik olarak robotlara yaptırılması faydalı olabilir. Sistemlerin doğru işlediğini talimatlara göre kontrol edecek sistem tasarımları özellikle kritik görevler için çok önemlidir.

Bu tezde, bir görevin iş adımlarını talimatlardan faydalanarak otomatik olarak çıkarmak amaçlanmaktadır. Tarifler, takip edilmesi kolay ve çok miktarda bulunabilen talimatlara örnektir. Bir tarifi nasıl adım adım pişireceğinizi anlamak, doğal dil işleme teknikleri ile eylemlerin, işlemlerin, araçların ve aralarındaki ilişkilerin çıkarılmasını gerektirir.

Eylemleri ve bağlı bileşenleri netleştirmek ve çıkarmak için denetimli ve kural tabanlı bir model önerilmiştir. Tamamiyle denetimli öğrenme yerine, daha basit doğal dil işleme teknikleri kullanılarak eylemler ve bileşenleri arasında isim varlık ilişkisi bulunur ancak tariflerin iş akışı sonunda kural tabanlı bir yöntemle üretilir. Önerilen modeli denetimsiz öğrenme yöntemleriyle karşılaştırıldığında sistemin bütünü çözmek yerine daha küçük ve iyi çalışılmış doğal dil işleme yöntemlerinin çıktısını kullanmaktadır.

Keywords: metin madenciliđi, part of speech tagging, word2vec, NLTK, collocation finder, linear conditional random fields (crf), semantik, long short-term memory (LSTM), derin öğrenme

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my master thesis supervisor Prof. Dr. Pınar Duygulu ŞAHİN and my co-advisor Gönenç ERCAN for their precious advices, guidance and their everlasting patience. Since I started working with them, I have learned priceless information from them.

I would also like to thank my thesis committee members for accepting to be in my thesis committee.

In addition, I would like to thank my dear friends , Sinan GÖKER and Mehmet Mert ONAĞ for providing me with continuous encouragement throughout my years of study and through the process of researching and writing this thesis. I would also like to thank to my all friends and colleagues for good wishes.

I would finally like to express my appreciate to my beloved family for believing and supporting me for all my life.

CONTENTS

	<u>Page</u>
ABSTRACT	i
ÖZET	iii
ACKNOWLEDGMENTS	v
CONTENTS	vi
FIGURES	viii
TABLES	x
1. INTRODUCTION.....	1
1.1. Overview	1
1.2. Motivation	2
1.3. Research Questions	3
1.4. Thesis Structure	4
2. BACKGROUND AND RELATED WORK	5
2.1. Background.....	5
2.2. Related Work	13
3. METHODOLOGY AND IMPLEMENTATION	18
3.1. Ingredient Parsing	20
3.2. Directive Parsing	26
3.3. Recipe Parsing	30
3.4. Action Graph Generation	32
4. EXPERIMENTAL ANALYSIS	39
4.1. Datasets	39
4.2. Evaluation Metrics	40
4.3. Experiments	41
4.4. Error Analysis	43
5. CONCLUSION.....	45
5.1. Conclusion.....	45
5.2. Future Research	46

A APPENDIX RECIPE PARSER OUTPUTS	47
B APPENDIX, ALGORITHMS OF UPDATE TAGS	54
REFERENCES	56

FIGURES

	<u>Page</u>
1.1. General Framework of The Proposed Model	3
2.1. An example sentence for PoS tagging	6
2.2. Tagger Context [1]	7
2.3. Conditional Random Fields.....	9
2.4. Example of LSTM Cell	10
2.5. Example of LSTM Memory Cell.....	11
2.6. Example Implementations of Word2Vec.....	12
2.7. Models Of Word2Vec [1]	12
3.1. An Example Recipe Text	18
3.2. Action graph for the recipe given in Figure 3.1.....	19
3.3. Framework of The Proposed Model.....	20
3.4. Epoch/Loss Graph.....	25
3.5. Epoch/Accuracy Graph	25
3.6. The Example of Sentence Node	34
3.7. The Example of Real Sentence Node	34
3.8. The Example of An Action Graph	35
3.9. The Action Graph of Amish Meatloaf	38
1.1. Action Graph Of Easy Whole Banana Muffins	48
1.2. Napa Slaw	51
1.3. Action Graph Of Taco Soup	53

TABLES

2.1. Universal Part of Speech Tagset	6
3.1. Sample Sentence	20
3.2. Different Tagged Sample Sentence	21
3.3. Tag Definition of CRF/Bi-LSTM Model For Ingredients	21
3.4. Feature Functions Of "1 teaspoon sugar"	22
3.5. Hot Chicken Salad	27
3.6. Sample Sentence, Stopwords Filtered	27
3.7. Sample Sentence, Stopwords Not Filtered	27
3.8. Labeled Sample Sentence	28
3.9. Tag Definition of The Proposed Model	30
3.10. Bi LSTM/CRF Tags Updated Sentence	30
3.11. Example Results of Updating Tools	31
3.12. The Output sentence of the model I proposed	32
3.13. Tagged Sentence Example	32
3.14. The Shape Definitions of Action Graph Nodes	34
3.15. The Directive of Amish Meatloaf	35
3.16. Parsed Directive of Amish Meatloaf 1	36
3.17. Parsed Directive of Amish Meatloaf 2	37
4.1. The Details of the Greene Ingredient Dataset used for Training and Testing ...	39
4.2. Kiddon Dataset for Testing	39
4.3. Experiment Results of Parsing Ingredient	41
4.4. Parsed Ingredient Phrases	41
4.5. Experiment Results of Parsing Recipe	42
4.6. Experiment Results of Generating Action Graph	43
4.7. Wrong Parsed Sentences	44
1.1. Easy Whole Banana Muffins	47

1.2. Napa Slaw	49
1.3. Napa Slaw(continue).....	50
1.4. Taco Soup	52

1. INTRODUCTION

1.1. Overview

Using instructions which aid us in our daily life, is becoming increasingly more common and effective. One such domain is the maintenance of safety-critical hardware components, which require well defined step-by-step guides to be carried out. Because safety-critical hardware components must be maintained step by step according to technical orders. Extracting these guides, especially for tasks that require a particular order of steps is *sine qua non* for teaching a robot or a computer how to perform a given task.

In this long-term goal of learning how to perform a task from raw textual resources, cooking recipes can be considered as a stepping-stone because there are many resources available on the Internet. One big challenge for building a clear step-by-step model for recipes is the ambiguity in natural language. Even the most exact recipe can involve ambiguity that must be resolved by the reader. A clear recipe model should correctly segment the text into actionable steps. Then the dependencies between these steps, ingredients, and tools involved in each step should be identified. The resulting model showing all of these components is called an action graph, which is a machine-interpretable version. The accuracy of the action graph depends on the effectiveness of natural language processing methods for the segmentation and named entity recognition.

To extract how to cook a recipe step by step and to find named entity recognition, there are many NLP approaches. Supervised NLP approaches handle to find named entity recognition by using labeled data. However, like finding labeled data for generating action graph is a tough and complicated issue. Alternatively, unsupervised NLP approaches like Kiddons et.al.[2] model do not need any labeled data to find named entity recognition and to generate action graph. As a whole, an unsupervised model can be developed; still, it is difficult to learn because of too many parameters, and its performance can be lower than supervised approaches. As a solution to this, the unsupervised method can be divided into sub-problems that can be solved in supervised NLP approaches. In the cooking domain, the general point of view of tagging recipes, extracting ingredients, actions, tools, and the relations of themselves depends on only directive parsing. However, a recipe can be divided into two parts: ingredients and directive. Now, there are two sub-problems to solve: ingredient parsing and directive parsing. Ingredients can be parsed with supervised methods and labeled with

”NUMBER, UNIT, NAME, COMMENT and QUANTITY” with the help of labeled data[3]. Actions can be found with the most straightforward way which does not need any recipe dataset to train. Tools can be found by using a dictionary. After all, processes are finished, finding the relations of actions are utilized for generating consistent action graph.

1.2. Motivation

Attributing machine-readable format to an instruction entails many NLP processes. Directly using the unsupervised model, which is depended on probabilistic is a kind of executive way[2] that needs more parameters to solve and needs well stacked and clean data to process. However, supervised and rule-based hybrid model is another approach to overcome these handicaps more accurate and safe outcomes. If there are some labeled data for sub-problems of the model that can be found easily on the Internet, then the model can be designed with supervised approaches with the help of labeled data. The other part can be a rule-based model.

The hybrid proposed model should solve one of the main problems, which is named entity recognition, in a recipe. Finding tools and ingredients with the directive is not an efficient way. As Leonardo Da Vinci said, ”*Simplicity is the ultimate sophistication.*”[4], a recipe can be split into two part as ingredients and directive. With the division of a recipe, the problem is separated into two part; ingredient parsing and directive parsing which of them can be determined with a supervised approach. Rather than other implemented approaches, the proposed model does not need recipe dataset to train. PoS Tagging is a simple way of tagging words according to their lexical categories. However, this technique is not satisfying when it comes to finding name or entity recognition, in particular, finding tools and ingredients in a recipe, because standard PoS Tagging is trained in the different dataset such as news. Adapting PoS Tagging increases the results of the next named entity recognition processes correctly. Especially for directives, with the aid of labeled data, it is used Conditional Random Fields (CRF)[5] or Long Short-Term Memory(LSTM)[6] techniques to overcome named entity recognition problem for ingredients.

However, while parsing directive, some obstacles occur. One of the obstacles is finding action. At the end of everything, the model generates an action graph. If a sentence in the directive does not have any word tagged with ”VERB,” this action in the sentence will be

neglected. This is an undesirable situation. Collocation approach is implemented to solve this ambiguity. Another problem is finding tools which are related to the action. For finding tools, a self-made tool dictionary shall be used¹. Generating action graph, updating tools, ingredients, and actions is inefficient. Because the relationships between the actions are not still generated. Using the word embedding approach like word2Vec[7], helps to find relations between the actions. Thus, the information to be used in action graph generation is provided.

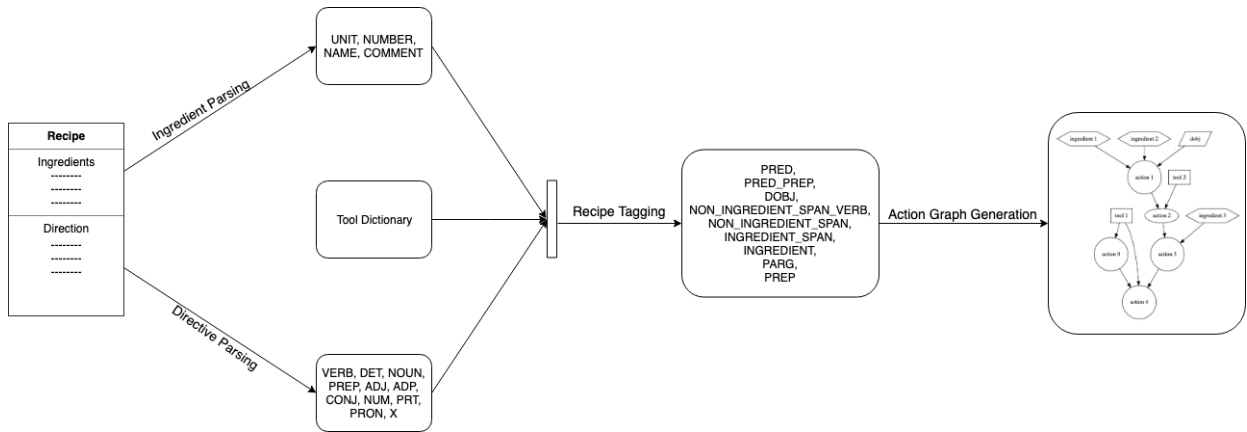


Figure 1.1. General Framework of The Proposed Model

This thesis introduces a model that labels recipe with the aid of parsing directive and ingredient finds relations of actions and generate action graph separately, which is a universal machine-interpretable format. All the techniques mentioned above are gathered to form this model whose framework is shown in Figure 1.1.. When a recipe is given to the model, the model generates an action graph.

1.3. Research Questions

Main research questions investigated in this thesis can be listed as below.

- Can we improve rule-based and supervised methods by stacking different NLP techniques like PoS tagging, named entity recognition and lexical similarity that are trained with out-of-domain training data for building recipe action graphs?
- Is the performance between Deep learning-based and CRF based named entity recognizers comparable for the cooking domain?

¹<http://www.enchantedlearning.com/wordlist/cookingtools.shtml>

- Is it possible to use neural word embedding to disambiguate the dependencies between recipe instructions and to generate the action graph?

1.4. Thesis Structure

The structure of the thesis is given as follows:

Chapter 2 describes background information of the proposed approaches from two different aspects of labeling methods: linguistic labeling methods and sequence labeling methods and reviews related works implemented in recipe parsing. This chapter also discusses the strong and weak sides of those studies.

Chapter 3 explains the algorithms of the proposed approaches, the steps and the methods of the model proposed in detail with examples.

Chapter 4 shows the evaluations of these approaches in detail. This chapter also includes a comparison of the proposed approaches with other necessary studies. This section is finished with error analysis.

Finally, **Chapter 5** concludes the thesis with a brief summary and discussion on the proposed ideas, contributions to literature, and discusses potential future work.

2. BACKGROUND AND RELATED WORK

This chapter is divided into two sections. Section 2.1., I give some information about basic NLP methods that are implemented to the proposed model. Section 2.2. reviews related works implemented in directive parsing and discuss about strong and weak sides of those works.

2.1. Background

In this thesis, studies of food domain which are made especially for finding named entity recognition and labelling are sum up the topic as Natural Language Processing (NLP) methods. In this section, some information are given about linguistic background of parsing directives which are PoS tagging and Collocation Finder. Labeling recipe is a kind of named entity recognition. In fact, it is further sequential labeling problem. Sequential labelling can be roughly defined as labelling the items/events in a sequence by categorical classes. For this reason, Conditional Random Fields (CRF) and Bi-Direction Long Short Term Memory (BiLSTM) which of them used in ingredient parsing are introduced. To understanding the whole proposed model, another NLP technique, Word2Vec which is utilized to find relations of the actions.

2.1.1. Natural Language Processing (NLP) Methods

This subsection gives information about techniques in detail that are used in the proposed model. These information are necessary to understand and comprehend my thesis.

2.1.1.1. Part Of Speech Tagging

Part of speech tagging (PoS tagging) or grammatical tagging or word-category disambiguation, is a process of marking a word as corresponding to a specific part of speech which is based on both its definition and its context. It is useful in the local context because of giving much more information about a word and its neighbors.

PoS tagging is a method that marks up of words in a text as corresponding to particular part of speech to describe the function of a particular word. In the English language, words are divided into parts of speech. Part-of-speech categories include noun, verb, article, adjective, preposition, pronoun, adverb and conjunction.

Table 2.1. Universal Part of Speech Tagset

Tags	Colors	Examples
ADJ	adjective	good, high, special etc.
ADP	ad-position	on, of, at, with, by, into etc.
ADV	adverb	really, already, still etc.
CONJ	conjunction	and, or, but etc.
DET	determiner	the, a, some, most etc.
NOUN	noun	year, home, costs etc.
NUM	numeral	fourth, 1991, 14:24 etc.
PRT	particle	at, on, out, over per etc.
PRON	pronoun	he, their, her, its etc.
VERB	verb	is, say, told etc.
.	punctuation marks	. , ; : etc.
X	others	esprit, dunno, gr8 etc.

Universal Tags and definitions are depicted on Table 2.1. that explains the tags NLTK Libraries simplified. Figure 2.1. shows an example sentence for part of speech tagging.

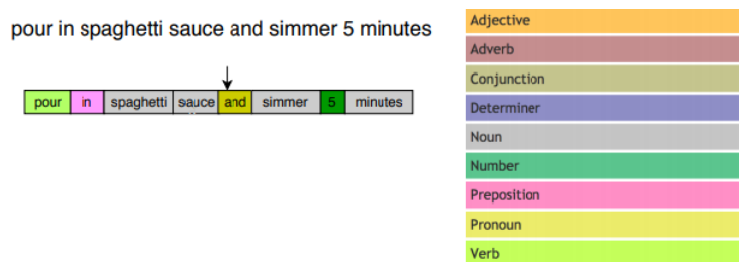


Figure 2.1. An example sentence for PoS tagging

N-grams are very powerful models and difficult to beat (at least for English) since frequently the short-distance context is most important. NLTK library is used n-gram tagging approach for PoS tagging. If the n-gram tagging is wanted to explain, firstly uni-gram tagging should be explained because the n-gram tagging is a general version of the uni-gram tagging. The uni-gram tagging is based on a simple statistical algorithm, for each token which is called as each individual occurrence of a word form, assign the tag that is most likely for that particular token. An n-gram tagger is a kind of a uni-gram tagger that is the current word together with the part-of-speech tags of the n-1 previous words[1] shown in Figure 2.2..

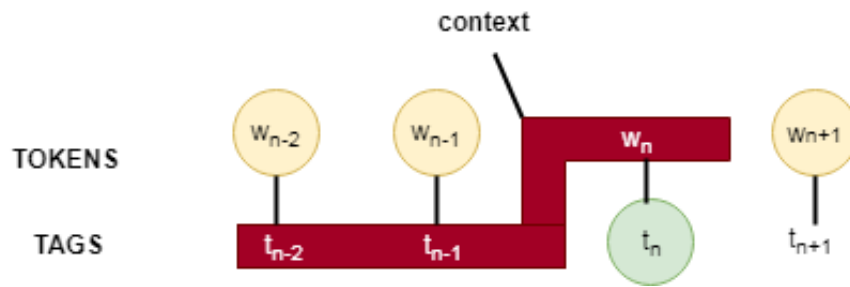


Figure 2.2. Tagger Context [1]

This method is implemented to the proposed model to find actions and noun phrases whose tags updates after this process is done.

2.1.1.2. Collocation Finder

Collocations are expressions of multiple words that frequently occur in common. A collocation is an expression that forms a specific meaning. A collocation can be noun phrase like *beautiful house*, phrasal verb like *make up*, idioms, cliché or technical phrases.

There are many approaches to find collocations in a text corpus. The significant ones are:

- Frequency
- Mean and Variance
- Hypothesis Testing
- Mutual Information

Frequency is a straightforward method for getting collocations in a text corpus. Selecting the most frequently occurring bigrams pass through the PoS tagging filter and it works well especially for fixed phrases. Collocations are found by counting the number of occurrences. Usually the collocation results in a lot of function word pairs that need to be filtered out. In order to fix this, candidate phrases pass through part-of-speech filter which only let pass those patterns that are most probably "phrases". However, many collocations consist of two words in more flexible relationships. The method of mean and variance can find this flexible relations. The method calculates the average and variance of the shift (marked distance) between two words in the corpus. If high frequency and low variance occur at the same

time, that can be accidental. It is desired to determine whether co-occurrence is random or more frequent than coincidence. That is called as *hypothesis*. Likelihood Ratios is simply a number that tells us how much more likely one hypothesis is than the other.

For collocation discovery, for instance, two alternative *hypothesis 1* - (H_1) and *hypothesis 2* - (H_2) can be examined for the occurrence frequency of a bigram w^1, w^2 :

- *hypothesis 1* : The occurrence of w^2 is independent of the previous occurrence of w^1 .
- *hypothesis 2* : The occurrence of w^2 is dependent of the previous occurrence of w^1 .

The log likelihood ratio is then shown on Equation 1.

$$\log \lambda = \log \frac{L(H_1)}{L(H_2)} \quad (1)$$

With the help of NLTK[8], the library's default is trained with Brown Corpus, which is related to the news. The collocations package in NLTK provides collocation finders which by default consider all n-grams in a text as candidate collocations. I utilize bigram scores of the words and get the likelihood ratio for spanning intervening words. Using NLTK gives bigram scores of multiple words and get the likelihood ratio for finding probable tags of words. With the light of these probable tags, if a sentence does not have any "VERB" tag after PoS tagging process, the word whose tag probably is "VERB" can be found.

2.1.1.3. Linear Chain Conditional Random Fields (CRF)

Linear Chain Conditional Random Fields is a kind of Conditional Random Fields that performs well on similar tasks such as PoS tagging and named-entity recognition[9].

$$P(y|x) \propto \prod_{n=1}^N \psi(y_n, y_{n-1}, x) \quad (2)$$

In Equation 2, let N be the number of words in the phrase x . The equation introduces a "potential" function ψ that takes two sequential labels, y_n and y_{n-1} , and the phrase, x . The

potential function is the weighted average of simple feature functions, each of which captures a single quality of tags and words [3].

On the other hand, Equation 3 is the weighted average of simple feature functions, and each of them gives information about a single attribute of the labels and words.

$$\psi(y_t, y_{t-1}, x) = \exp \sum_{n=1}^N w_n f_n(y_t, y_{t-1}, x) \quad (3)$$

I usually define feature functions to return 0 or 1. Each feature function, $f_k(y_n, \dots, x)$, is selected by the person who creates the model depending on which features are more useful for detecting the relations between words and labels.

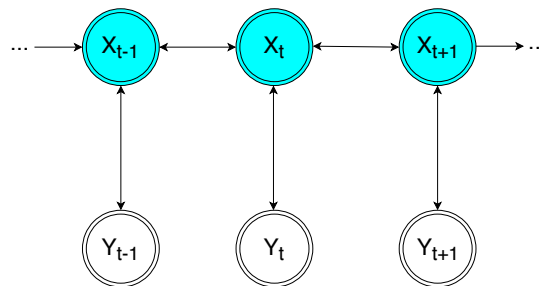


Figure 2.3. Conditional Random Fields

CRF++ is a basic, customized implementation of CRF for tagging sequential data[10]. It is written in C++ language. It is easy to implement and gives us to redefine feature clusters. As shown in Figure 2.3., it is a tool for tagging sequential data. This library is suitable to tag ingredient phrases.

2.1.1.4. Long-Short Term Memory (LSTM)

Long Short Term Memory networks are gated Recurrent Neural Networks, able to learn and memorize long-term dependencies in sequences. Initially proposed by Hochreiter & Schmidhuber [11], it is commonly used for building neural networks for natural language tasks. A common LSTM unit as shown in Figure 2.4..

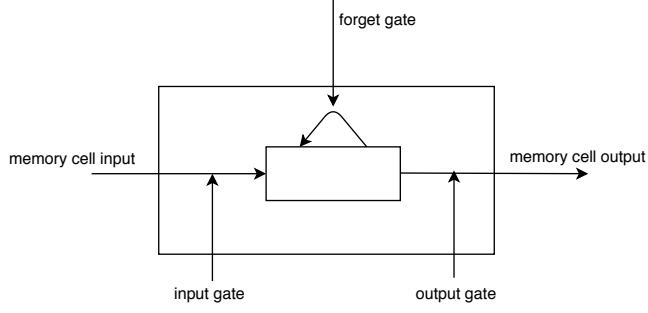


Figure 2.4. Example of LSTM Cell

LSTMs are clearly produced to avoid the vanishing and exploding gradient problem. They can selectively memorize information for a long time. The LSTM is carefully arranged by structures called gates and has the ability to add or append information to the cell state. The logistic function consists of the neural network layer and a multiplication operation.

$$\begin{aligned}
 f_t &= \sigma(w_f[h_{t-1} - x_t] + b_f) \\
 i_t &= \sigma(w_i[h_{t-1} - x_t] + b_i) \\
 o_t &= \sigma(w_o[h_{t-1} - x_t] + b_o)
 \end{aligned} \tag{4}$$

The equations for the gates in LSTM are shown in the Equation 4. f_t represents forget gate, i_t represents input gate, o_t represents output gate, σ represents sigmoid function, w_x represents weights for the respective gate(x) neuron. h_{t-1} represents output of the previous lstm block (at timestamp $(t - 1)$), x_t represents input at current timestamp and b_x represents biases for the respective gates(x).

$$\begin{aligned}
 \tilde{c}_t &= \tanh(w_c[h_{t-1} - x_t] + b_c) \\
 c_t &= f_t * c_{t-1} + i_t * \tilde{c}_t \\
 h_t &= o_t * \tanh c^t
 \end{aligned} \tag{5}$$

The example of LSTM memory cell is shown on the Figure 2.5. and the equations for the cell state, candidate cell state and the final output are shown in the Equation 5. C_t represents cell state (memory) at timestamp(t) and \tilde{c}_t represents candidate for cell state at timestamp(t).

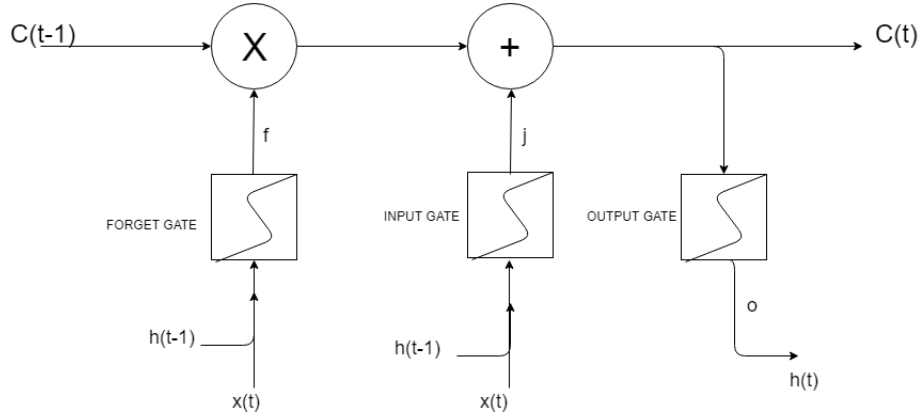


Figure 2.5. Example of LSTM Memory Cell

2.1.1.5. Expectation Maximization (EM)

Expectation Maximization is a kind of learning method optimizing the maximum-likelihood for the model parameters. EM can be used for incomplete data and/or latent variables. It is an iterative method to approach the maximum likelihood. Although proposed methods are not utilized EM, it is used in the algorithm proposed by Kiddon et al. [2], and its details are relevant for the following discussions.

It is given a probabilistic model with some incomplete or latent variables, EM targets to find the best parameters yielding the maximum likelihood for the complete dataset. Naturally, as the number of parameters increases, it is more difficult to find the optimum model parameters. Starting from random initial values for the model parameters, EM iteratively updates the parameters to maximize the likelihood function.

New values are used to create a better estimation for the first set, and the process continues until the algorithm approaches a local or global maximum. Let the set of observed variable be the set M , missing random variables as N and the vector of unknown parameters is β , then likelihood function is $L(\beta; M, N) = p(M, N; \beta)$, the maximum likelihood estimate (MLE) of the unknown parameters is determined by maximizing the marginal likelihood of the observed data, Equation 6.

$$L(\beta; M) = p(M|\beta) = \int p(M, N|\beta)dN \quad (6)$$

However, Equation 6, especially if N is a sequence of events, the number of parameters to solve grows exponentially with the sequence length. Namely, the EM algorithm can be very slow depending on the model.

2.1.1.6. Word2Vec

Word2Vec is an algorithm and a two-layer neural network that processes text. Text and set of vectors are the input and output of the algorithm, respectively. After training the neural network with raw text, it can list the semantically close words to a given query word. It is a representation learning technique, building vectors to represent the words in a language. As seen in Figure 2.6., analogies between words can also be queried with Word2Vec.

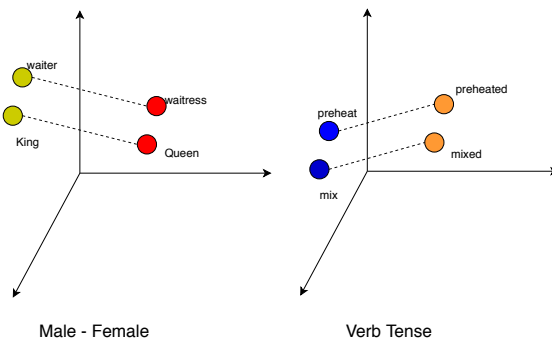


Figure 2.6. Example Implementations of Word2Vec

Mikolov [12] proposes two different methods to learn the word vectors, namely continuous bag-words, and skip-gram architectures.

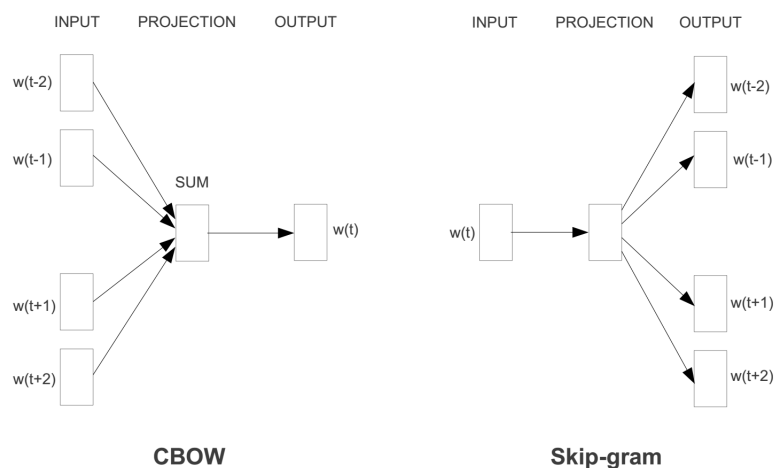


Figure 2.7. Models Of Word2Vec [1]

The graphics of the models are shown on Figure 2.7., in both models there are three layers: *Input layer*, *Projection layer* and *Output layer*. Graphically, contexts, $\{w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}\}$, are given for the CBOW model to predict w_t , while skip-gram model is defined w_t to guess the context, $\{w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}\}$.

According to Mikolov, skip-gram performance well with a small amount of the training data performs well on even rare words or phrases. However, CBOW is several times faster to train than the skip-gram, lightly better accuracy for the frequent words. By implementing word2Vec, the similarity between words can be estimated.

2.2. Related Work

In this subsection, several studies related to named-entity recognition and action graph generation for the cooking domain are reviewed. Within this scope, different approaches have been improved in directive named-entity recognition and graph generation. The literature is briefly explained in two subsections as supervised and unsupervised methods below.

Algorithmic processing of instructional texts and commands expressed in natural language have long been an interesting question [13]. In this problem, the artificial intelligence agent (such as a robot) is trying to learn the operations corresponding to symbols that are expressed in a domain. The system is expected to automatically interact with definitions in the physical world as symbols to interact with objects.

Chen [14] have developed a system that can act according to the expressions expressed in natural language in the system. Based on reinforcement learning methods, the system can translate the sentences into natural language by using the information in the location (such as the objects on the wall) and the state flow that is modeled. As can be seen from this study in the literature, reinforcement learning techniques are used in problems where the current situation can be modeled more clearly.

For a slightly different problem, a similar method for solving mathematical problems expressed in natural language is developed [15]. According to this method, the system is mapping the problem described in the text to a set of equations and solving them to produce the result. Equations that can be created in the system can be given or can be learned only when the last answer is given. In the method they developed, the solution is defined as diagrams,

and a probability model that maps the information given in the text to the diagram is developed. Of course, it is necessary that the structure of the questionnaire is similar and that all of them can be transferred to the equation diagram.

Directives can also be considered as a set of steps that must be followed. From this point of view, there are common aspects to the problems described above. Converting directives to a computational model requires a system to create an action graph to capture the workflow with clear steps and identifying the attributes of the steps. Cooking procedure systems are used for a variety of applications for the cooking domain, including directive search, summary, task scheduling, visualization, and field enrichment. Recognizing the critical attributes (ingredients, tools, etc.) and their relationship can be an essential prerequisite step for finding the workflow. Extracting the named-entities from directives is a challenging task. To overcome this, supervised and unsupervised approaches are proposed, the next subsection will briefly overview such approaches.

2.2.1. Literature Review on Supervised Methods

Mori[16] created a labeling tool to build a corpus of graphs indicating the workflow of cooking procedures. The Japanese cooking directives are first segmented to words. Then the named entities are labeled, and finally, the predicate-argument structures are constructed. Mori[16] represents directives as acyclic graphs where each leaf is related to a food or a cooking tool, and each intermediate node represents a cooking action, and the root node is the final cooking action. The model he proposed used dependency parser to find noun phrases and actions.

Jermurawong[17] also defined a tree based representation that stores food items and their relationships over the description steps. This representation could show which step of the directives was related to what steps and the materials used in that step. He made a model which is dependency tree data structure, Simplified Ingredient Merging Map in Recipes(SIMMR), to find ingredients and represent recipes. The extracted ingredients were used to create a tree data structure. The relationships between the actions were found using Linear SVM Rank [18]. Two contributions of this model were the ingredient-instruction dependency tree representation of recipe structure and the cooking recipe parser that maps text recipes into the proposed ingredient-instruction tree structures. Although they found each ingredient which action used with, they did not find the state of ingredients like melted cheese.

Malmaud [19] modeled the texts in the directives with a Markov Decision Process problem by first finding their semantic role labels. The model he proposed maintains the relationship between the process and the material. The purpose of this relationship is to highlight the processes in training. The relationship means the association between actions and ingredients. A directive model based on step-by-step analysis was created. According to the model, semantically, a sentence is divided into three parts: arguments (contents or implicit objects such as mixtures), actions (contents and verb relations) and control structure (sequential conditions, alternatives (chopping or puree)). The aim of this model was convincing; however, the model needed videos or images to contribute parsing results and finding latent actions in the recipe is another problem for the model. On the other hand, they found relations of actions and components in a recipe, representation of the relations still were needed to work on.

In order to address the ambiguities in the contents and label ingredients with specific tags, Greene et. al.[3] created a probabilistic model for tagging sequential data. The model uses 171,224 tagged data to find out a pattern that can predict the label sequence for a sentence they gave it. Greene considered only the ingredient parts of the recipes, not the whole instructive sentence. Ingredients were tagged with number, quantity, unit, name, or comment with the help of CRF algorithms. Their point of view for tagging ingredient was a sequential labeling problem that represented a different perspective.

2.2.2. Literature Review on Unsupervised Methods

Salvador [20] proposed a model for matching images to recipes. In this model, Word2Vec is used to find the ingredients in the directive and for the parsing directive sentences. A skip-instruction method (i.e., decoding/predicting previous and next sentences) is used, which is inspired by skip-thought vectors[21]. After parsing the directive, the model matches the images which are related to the parsed data. For cooking instructions, he used a two-stage LSTM model which is designed to encode a sequence of sequences. In order to feed his fixed-length joint embedding model, each sentence in the recipe is converted to a skip-instruction vector that is jointly trained by the LSTM model. This model is satisfying to match recipe to images, yet it is inadequate for modeling recipe as a machine-interpretable format.

To obtain information from the recipes in a machine-interpretable format, Bertini et al. [22] used the Semantic Role Labeling model (SRL) based on Maximum Entropy Classification.

Nedovic[23] used a similar method to define a method of learning materials for different types of food. Latent Dirichlet Allocation (LDA) and Deep Belief Networks (DBN) are used in the method. It has been seen that the output of the system can group materials according to the foods in different kitchens. With the result of the model, he found which cuisine used the ingredients of this result. However, he focused only on the ingredient list of each recipe and analyzed it in a bag of words methods.

In 2015, Kiddon[2] has developed a method using unsupervised learning for processing recipes. Unlike previous supervised works, this method learns to interpret recipes using text alone with dependency parser[24]. After the interpretation, the EM method is used. They calculate to learn the features of the whole probability of a recipe. Prior to estimating the parameters of the model, dependency parser tags are required.

Moreover, the generated data model has a high number of implicit variables. Because it defines each probability of relation between verbs and contents as one of the parameters of the model, when such high number of parameters are unknown, in the high dimensional space, the number of local maximums can be high preventing finding the optimum parameters for the model. They generate action graphs from instructional recipes with the help of unsupervised EM approach.

These methods are inspired by semantic spaces used in text mining and meaning spaces that show word meaning relations. By using these material spaces, it is possible to determine which materials can be replaced instead of which one of them.

2.2.3. Discussion

This subsection reviews the previous studies on name-entity recognition and action graph generation on food domain in two sub-sections as supervised and unsupervised approaches. There are countless studies on recipe name-entity recognition and action graph generation. Most of the work is evaluated on standard corpora like Recipe 1M [25], food 101[26] etc. Although there are a small number of studies targeting to generate action graphs from recipes, The data is collected for building a new corpus. Each recipe has its title, ingredients list, and the recipe text.

Neural approaches are dominating the methods developed for NLP in recent years. The advantages of Artificial Neural Networks (ANN) are also utilized to learn and model non-linear

and complex relationships from the directives. Neural networks approaches have improved the state-of-the-art in common NLP tasks like PoS tagging and named-entity recognition.

In supervised approaches, large quantities of tagged data are required for a successful learning process. Finding labeled data for tagging directives and ingredients require a lot of manual effort for labeling the corpora. Especially for ingredients, with the help of Greene et.al.[3] study, a dataset which is already labeled is always helpful for people who work in this area. Unsupervised methods do not require annotated corpora. However, the dataset for unsupervised approaches must be clean and well stacked. The results are dependable for implemented methods, data type, and the format of data included in the dataset play an essential role in getting good results.

In unsupervised approaches or models divide directives into sentences, and each sentence is divided into ingredients, tools, actions, or implicit objects. These approaches or models which depend on semantic types as Kiddon [2], performs better than a sequential baseline. But the errors of the model are caused by missing or incorrect decisions in the segmentation stage. The computational time of the algorithm is also a concern as the model must learn many parameters.

The two-stage LSTM model [20] is favorable for categorizing the skip-instructions. However, it is inadequate for generating action graphs as it does not explicitly identify the attributes in the directives. Even though vectors can match directives to images, The meticulous data is needed for the action graph generation.

The shortcomings of current literature are summarized in the next sentences. Firstly, they try to label action and ingredients only using direction. Implemented methods from the literature need recipe dataset to train. Also, the general point of view of these methods, labeling recipe is a kind of named entity recognition. In fact, it is a further sequential labeling problem.

The recipe parsing is the only question to seeks. However, the problem can be divided into sub-problems. Rather than extracting whole relations from directions, each recipe is categorized into two parts: ingredients and direction. The proposed model, with the light of the literature, aims to utilize: rather than labeling ingredients in the direction, tagging ingredients and direction separately, seeking actions with the most straightforward way which does not need any recipe dataset to train, finding the relations of actions for generating consistent action graph.

3. METHODOLOGY AND IMPLEMENTATION

In this chapter, the approaches that are proposed in this thesis for recipe-entity recognition are explained in detail. For labeling ingredients; CRF and LSTM, for finding a course of actions; PoS tagging and Collocation Finder, for generating action graph; word2vec are the approaches to create the proposed model.

	Ingredients
Preheat oven to 350 degrees F (175 degrees C).	2 1/2 cups chopped, cooked chicken meat
Combine the chicken, celery, almonds, bell pepper, onion, pimento, salt, lemon juice, and mayonnaise.	2 cups chopped celery
	1/2 cup chopped salted almonds
Mix well and pour into a 1 1/2 quart casserole dish.	1/4 cup chopped green bell pepper
	2 tablespoons minced onion
Top with grated cheese and the crushed potato chips.	2 tablespoons chopped pimento peppers
	3/4 teaspoon salt
Bake for 25 minutes or until cheese is melted.	2 tablespoons lemon juice
	1/2 cup mayonnaise
	1/3 cup shredded Swiss cheese
	3 cups crushed potato chips

Figure 3.1. An Example Recipe Text

Converting a recipe requires a system to build an action graph with clear steps and the attributes used in these steps. Identifying attributes such as ingredients, tools, and action performed in a recipe is fundamental for building the model of a given recipe. Figure 3.1. depicts the text of an example recipe from the corpus. That is, a recipe has already been stored as two parts. The system should address the attributes and flow of the recipe, as shown in Figure 3.2.. The steps denoted by S_i are presented on a swimlane with edges showing the associated attributes and the sequence of the steps. For example, *preheat* step S_1 is linked to S_5 *bake for* step indicating that steps S_2 to S_4 are independent from S_1 and can be carried out in parallel. Along with the circle action nodes, ingredients are related with the step in two different levels, main ingredients which are tagged with only *INGREDIENT* and the other ingredients which are tagged with *INGREDIENT_SPAN* used during the step.

In order to detect and differentiate the recipe specific entities, instead of incurring a manual labeling effort, an existing recipe named entity recognition corpora is utilized¹. Rather than annotating the recipes, the model is acquired using heuristics built on top of PoS tags. During

¹<https://open.blogs.nytimes.com/2016/04/27/structured-ingredients-data-tagging/>

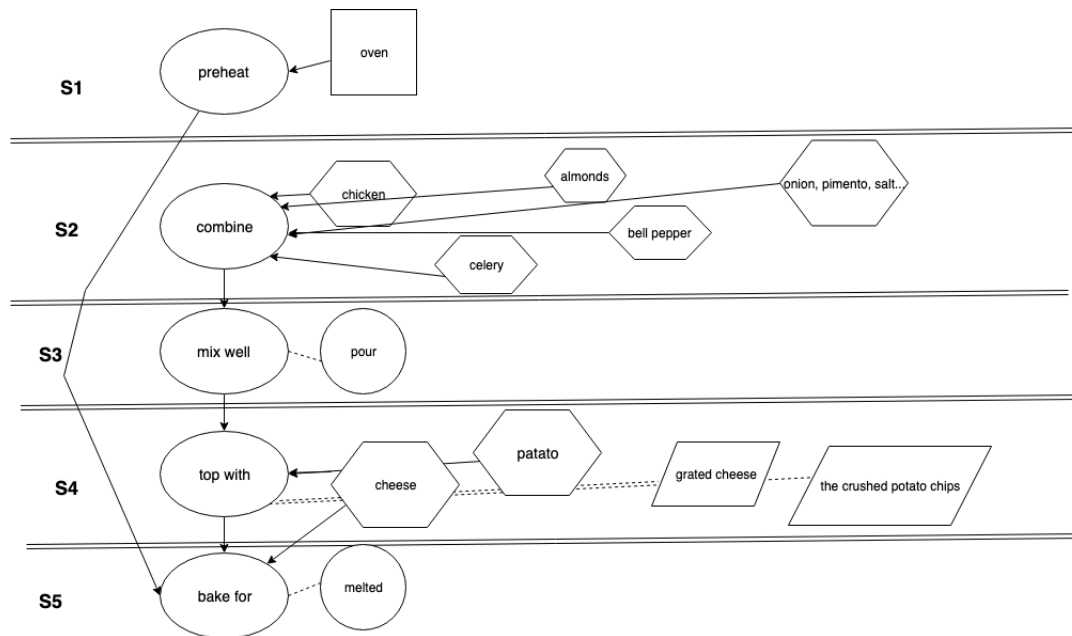


Figure 3.2. Action graph for the recipe given in Figure 3.1.

PoS tagging, thanks to word2vec, similarities of the actions are calculated. Also, probable action whose sentence is not parsed properly by PoS tagger is found with Collocation Finder, and the word tag is updated as "VERB". After PoS tagging, the proposed model depicted in Figure 3.3. is updated according to the rule-based model. The tags of the model are shown in Table 3.9.. Furthermore, a dictionary is extracted from a cooking website² in order to find tools in the sentence. Using this dictionary words tagged as *NOUN* or *ADV* (adverb) are checked and are replaced with the *TOOL* tag. Updated tags are eligible to generate an action graph.

As shown on Figure 3.3., this section divided into four parts:

- Ingredient Parsing, which is explained in Section 3.1..
- Directive Parsing which is explained in Section 3.2..
- Recipe Parsing which is explained in Section 3.3..
- Action Graph Generation which is explained in Section 3.4..

²<http://www.enchantedlearning.com/wordlist/cookingtools.shtml>

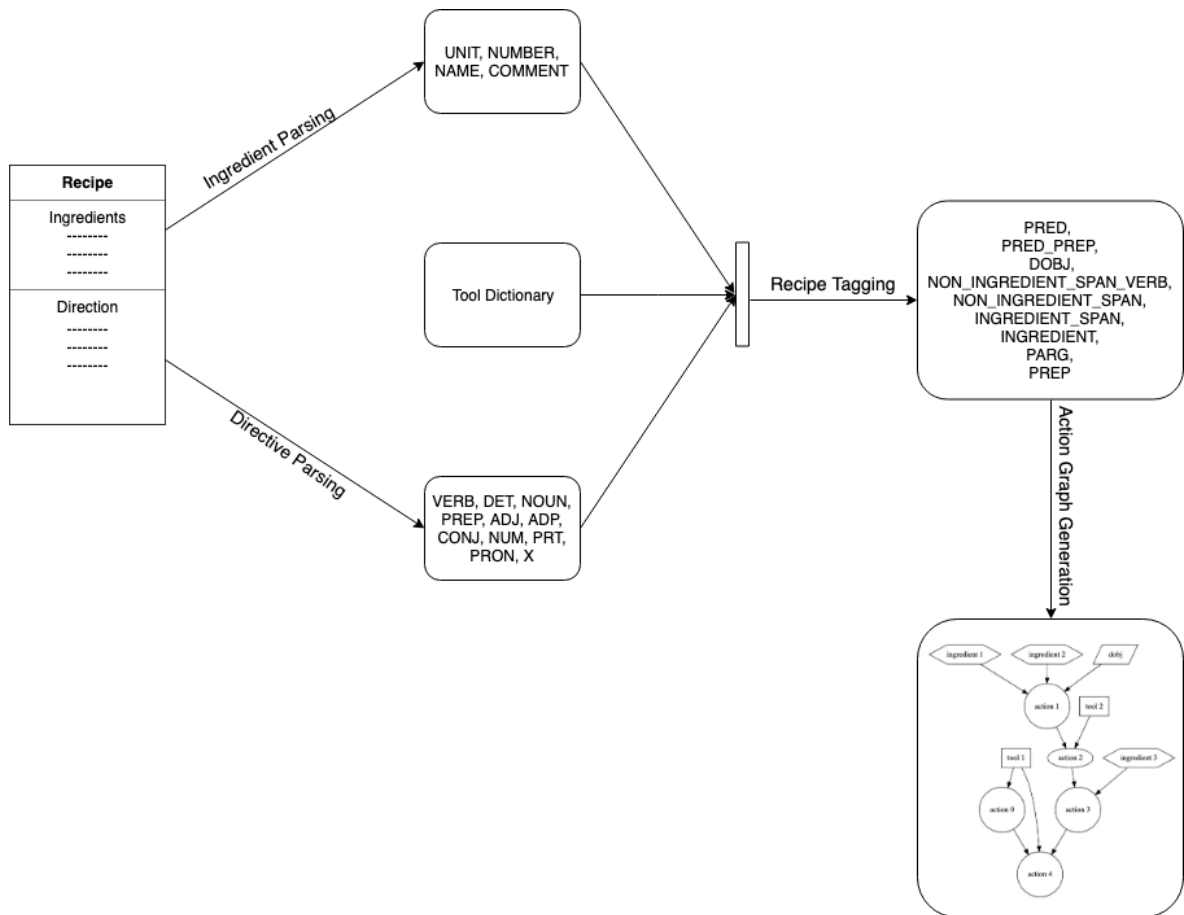


Figure 3.3. Framework of The Proposed Model

3.1. Ingredient Parsing

PoS tagging is inadequate for finding ingredients in a sentence in the direction. This thesis investigates if only PoS tags can be used to identify the ingredients. While the PoS tags signify the grammatical category of the word in the sentence, the ingredients that can be formed of multiple words, tools and quantities are also expressed with nouns. Thus, it is not possible to distinguish these attributes using only the PoS tags. Consider the example ingredient "1/4 cup grated Parmesan cheese", in the sentence "stir the mixture and 1/4 cup of Parmesan cheese", whose PoS tags are shown in Table 3.1.. These tags do not give any clues to show which of words are relevant to ingredients.

Table 3.1. Sample Sentence

stir	the	mixture	and	1/4	cup	of	parmesan	cheese
VB	DET	NN	CC	CD	NN	IN	NNP	NN

The sentence with the aid of PoS tagging, which looks for the word’s position and word’s task in the sentence is parsed. These tags are not efficient while finding which ingredient is related to this sentence. Table 3.2. shows more efficient tags rather than Table 3.1..

Table 3.2. Different Tagged Sample Sentence

stir	the	mixture	and	1/4	cup	of	parmesan	cheese
VB	DET	NN	CC	NUMBER	UNIT	IN	NAME	NAME

Considering the different sentence structures in the directive sentences, instead of a heuristic-based tagging method, a supervised method is preferred to achieve better results in the labeling task. An already existing labeled data from the work of Greene et al. [3] whose tags are depicted in Table 3.3. is used to label the ingredients and quantities.

Table 3.3. Tag Definition of CRF/Bi-LSTM Model For Ingredients

Tags	Meanings
NAME	defines ingredients: onion, sugar etc.
UNIT	defines quantity of uncountable ingredients: teaspoon, pinch, etc
QUANTITY	defines quantity of countable ingredients: 1, 2, 5, 1/2
COMMENT	defines ingredient status: melted, sliced etc.

3.1.1. Conditional Random Fields (CRF) Based Ingredient Parsing

The toughest aspect of the recipe parsing problem is the task of labeling ingredient components from the ingredient phrases[3]. Recipes showing ingredients like “ 1 teaspoon brown sugar”, should be further segmented and labeled to (brown sugar) - name, (1) - number, (teaspoon) - unit. This kind of difficulty is referred to as a structured prediction problem for trying to predict a structure or sequence of tags. In order to solve this problem, Greene et.al.[3] used CRF for sequence labeling.

Let $\{ x^1, x^2, \dots, x^N \}$ be the list of ingredient phrases where each x^i is an ordered list of words. Related with each x^i is a list of tags, y^i . The aim is to utilize the data to find a model that predicts the label sequence for the components. Greene et.al.[3] attack this task by modeling the conditional probability of a set of tags using the given p label (which represents the tag sequence) or the above notation $p(y|x)$ which is shown on Equation 2.

For example, the ingredient phrase is “1 teaspoon sugar” then all possible sequences of 3 tags should be scored.

$$\begin{aligned}
&P(\text{UNIT}, \text{UNIT}, \text{UNIT} \mid 1, \text{teaspoon}, \text{sugar}) \\
&P(\text{QUANTITY}, \text{UNIT}, \text{UNIT} \mid 1, \text{teaspoon}, \text{sugar}) \\
&P(\text{UNIT}, \text{QUANTITY}, \text{UNIT} \mid 1, \text{teaspoon}, \text{sugar}) \\
&P(\text{UNIT}, \text{UNIT}, \text{QUANTITY} \mid 1, \text{teaspoon}, \text{sugar}) \\
&P(\text{UNIT}, \text{QUANTITY}, \text{QUANTITY} \mid 1, \text{teaspoon}, \text{sugar}) \\
&P(\text{UNIT}, \text{QUANTITY}, \text{NAME} \mid 1, \text{teaspoon}, \text{sugar})
\end{aligned}$$

...

Sample feature functions on Equation 3 for the ingredient phrase are "1 teaspoon sugar" used for this problem are shown on Table 3.4..

Sequential Functions	Conditions If	Results
$f_1(y_t, y_{t-1}, x)$	x_t is capitalized and y_t is NAME	1
$f_2(y_t, y_{t-1}, x)$	x_t is "1" and y_t is QUANTITY	1
$f_3(y_t, y_{t-1}, x)$	x_t is "teaspoon" and y_t is QUANTITY	1
$f_4(y_t, y_{t-1}, x)$	x_t is "sugar" and y_t is QUANTITY	1
$f_5(y_t, y_{t-1}, x)$	x_t is fraction and y_t is QUANTITY	1
$f_6(y_t, y_{t-1}, x)$	y_t is QUANTITY and y_{t-1} is UNIT	1
$f_7(y_t, y_{t-1}, x)$	y_t is NAME and y_{t-1} is UNIT	1
All	otherwise	0

Table 3.4. Feature Functions Of "1 teaspoon sugar"

CRF, similar to logistic regression, learns large positive weights on features that capture highly likely in the training instances, large negative weights for features that capture highly unrelated patterns in the training instances. For example, f_2 describes a likely pattern in the data ("1" is likely a quantity), f_4 describes an unlikely pattern in the data (the word "sugar" is almost never a quantity), and f_1 does not extract a typical pattern. In this situation, w_2 should have a large positive number, w_4 should have a substantial negative number, and w_1 should be almost 0.

Although CRF is used by Greene et al. [3], state-of-the-art results in named entity recognition are achieved by LSTM based models. A deep learning-based model is also used for identifying the ingredients in the recipes to investigate this.

3.1.2. Long-Short Term Memory (LSTM) Based Ingredient Parsing

In order to predict sequences of tags from ingredients phrases, Bidirectional LSTM models achieving excellent results in named entity recognition tasks are evaluated [6]. Bidirectional LSTM uses two LSTM layers to process the sequence from both directions of the text [27]. The input sequence is fed to one LSTM layer and also the sequence reversed and fed to a second LSTM layer. This provides the model to observe both the right and left context when predicting the labels of a token in the sequence.

The Bidirectional LSTM based named entity recognizer is implemented using the Keras framework. Keras is a framework, a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano backends[28].

The first layer defined in the model is the word embedding layer, which maintains the intense relative meaning of the words [29]. This layer does not be pre-trained.

The second layer is a bidirectional LSTM layer with Time Distributed wrapper layer, which is used around the output layer and using a recurrent neural network for the tagging task. Thus, a value can always be estimated when an entire array is provided as an input. This requires the LSTM hidden layer to return a value sequence (one per time) instead of a single value for the entire input sequence. Recurrent dropout regularization method is used to impede over-fitting training data, which is lessen their predictive abilities. Recurrent dropout parameter is selected as $recurrentdropout = 0.1$.

Softmax activation function is the normalized exponential function. It calculates probabilities of n different categories, which are the labels shown in 3.3.. In other words this function estimates the probability of each target over all targets. General formula of *Softmax* activation function for $i = 1 \dots N$ is shown on Equation 7:

$$Softmax(x)_i = \frac{e^{z_i}}{\sum_{n=1}^N e^{z_n}} \quad (7)$$

The loss function is used for calculating the error of the model. The loss function, *Categorical Cross Entropy* to solve the multi-class classification problem is used. Although intuitive, Categorical Cross-Entropy loss and Softmax loss works better for Binary Cross-Entropy loss in multiple-label classification problems. As illustrated in Equation 8, M is the number of

classes, y is the binary indicator (0 or 1) if class tag c is the true result of the classification for observation n and p means that predicted probability observation n is of class c .

$$- \sum_n^M y_{n,c} \log(p_{n,c}) \quad (8)$$

An optimizer algorithm is used to learn the parameters of the neural network. During the training period, it updates a parameter for each input and target pair. [30]. SGD (Stochastic Gradient Descent), Adam (Adaptive Moment Estimation) [31], Adadelta [32] and RMSProp (Root Mean Square Propagation) [33] algorithms have also been experimented.

3.1.2.1. Model Configuration

Configuration of the Bidirectional LSTM is depicted below:

- **Unit**, used to select the dimension of the memory cells in an LSTM. In this study, the unit is 100.
- **Activation function**, used to compute hidden output: In this study, *Softmax* is selected.
- **Loss function**, used to compute the error of the model: In this study, *Categorical crossentropy* is selected.
- **Optimizer function**, used to compute offset the loss function: In this study, *RMSProp* is selected.

The other parameters are as follows:

- **Epoch** used to define the number of iteration for training: In this study, the epoch is 6.
- **Batch size** used to define the number of samples in training data in training: In this study, the batch size is 32.

The gold standard for machine learning model evaluation is k-fold cross validation [34]. It provides a robust prediction of a model on unseen data. It does this by dividing the training

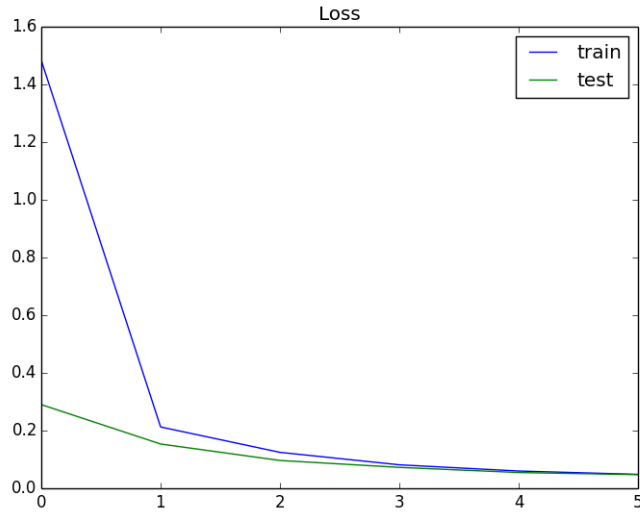


Figure 3.4. Epoch/Loss Graph

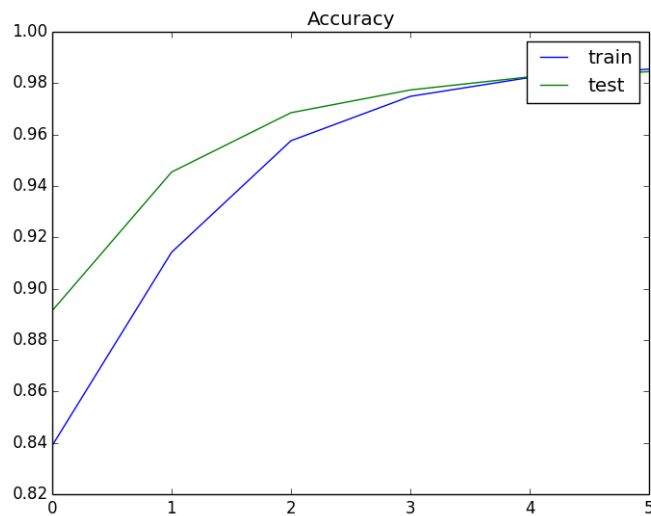


Figure 3.5. Epoch/Accuracy Graph

dataset into k sub-data and takes turns training models on all sub-data except one which is held out and assessing model performance on the held-out validation dataset. K-fold cross-validation ($k=10$), is used for this model and Figure 3.4. summarizes history for average loss, and Figure 3.5. summarizes history for average accuracy.

3.2. Directive Parsing

Directive or recipe is a paragraph which has just imperative sentences. To parse an imperative sentence with linguistic labeling methods and to find action and other components, without data preprocessing, PoS tagging is inefficient. Each sentence is generally related to ingredients. Before PoS tagging, data should be ready to get the result more accurate.

In general, the algorithm for directive parsing is shown in Algorithm 1. As shown in Algorithm 1, "*taggedDirective*" is a list that each element is a sentence's word-tag tuple list which is derived from PoS tagging and preprocessing operations. If each sentence of the *taggedDirective* does not have any action, *updateVerbs* function runs and finds probable verb in the sentence. Then, *updateTools* function finds tools in the sentence and update the tags. At the end of the for loop, *findRelatedVerbsPair* function runs and finds similarity between actions in the directive.

Algorithm 1: Pos Tagging Overview

```
Function updateTags taggedDirective
  forall sentence in taggedDirective do
    updateVerbs(sentence)
    updateTools(sentence)
    updateIngredients(sentence)
  findRelatedVerbsPair(taggedDirective)
```

This section is divided into three part to get better results:

- Data Preprocessing
- Directive Tagging
- Recipe Tagging

3.2.1. Data Preprocessing

This process contains segmentation, punctuation removal, and normalization. Each directive is segmented into sentences, and also, each sentence is divided into words. An example directive paragraph is separated in sentences which are depicted in Table 3.5..

Table 3.5. Hot Chicken Salad

No	Directive
1	Preheat oven to 350 degrees F (175 degrees C).
2	Combine the chicken, celery, almonds, bell pepper, onion, pimento, salt, lemon juice, and mayonnaise.
3	Mix well and put into a 1 1/2 quart casserole dish.
4	Top with grated cheese and the crushed potato chips.
5	Bake for 25 minutes or until cheese is melted.

Then punctuation is removed from each sentence. This is done to avoid errors in tokenization. Also, every capital letter is changed to lowercase. These are the requirement for labeling words correctly. However, stopwords are not to be filtered. If stopwords are removed, some positive results caused by PoS tagging can be lost. In particular, *"Mix well and put into 1 1/2 quart casserole dish"*, this phrase has 2 separate sentences, *"Mix well"* and *"put into 1 1/2 quart casserole dish"*. PoS tagging result of sample sentence whose stopwords are filtered is depicted on Table 3.6., and labels do not have any clues to divide the two directives correctly. However, if a sentence has conjunction and next tag of the conjunction is "VERB", it can be considered as two sentences, thus, with the help of conjunction, the sentences are divided into two parts, and the process is continued.

Table 3.6. Sample Sentence, Stopwords Filtered

Mix	well	put	into	1/2	quart	casserole	dish
NNP	DET	VP	IN	NUM	NN	NN	NN

Table 3.7. Sample Sentence, Stopwords Not Filtered

mix	well	and	put	into	1/2	quart	casserole	dish
VP	DET	CC	VP	IN	NUM	NN	NN	NN

The words whose length are 2 or less are filtered, except stopwords. These words might cause a mislabeling.

3.2.2. Directive Tagging

After data preprocessing is done, the directive is fed to the proposed model, shown on Algorithm 1. There is a problem in PoS tagging for imperative sentences. To label an imperative sentence accurately, *"I would"* phrase is added in front of the sentence. As the PoS tagger is

typically trained on news articles, it is not trained with imperative sentences and produces errors in imperative sentences. Adding the phrase "I would" reduces the errors and adapts the imperative sentences to a form that PoS tagger can process more accurately.

As a result of the labeling made, *I would chop green chile peppers*, in the table of contents is labeled as below in Table 3.8..

Table 3.8. Labeled Sample Sentence

chop	green	chile	peppers
VERB	NOUN	NOUN	NOUN

Due to PoS tagging errors actions in some of the sentences might not be labeled as a verb. During PoS tagging, if a sentence in the directive does not have a "VERB" tag, most probable action in the sentence is tried to be found with the help of bigram collocations, which are expressions of multiple words which commonly co-occurs. This process is inevitable for generating action graph. Because each sentence must have at least one action. To solve missing action problem in the recipe which relies on PoS tagging model, Collocation Finder is a reasonable approach to solve. If the sentence does not have any action after PoS tagging process, Collocation Finder finds probable action in the sentence with the help of the likelihood ratio. Firstly, words tagged with "NOUN and ADVERB" are filtered from the sentence, then, Collocation Finder seeks the best word which is the most frequently tagged with "VERB" in the dataset. In other words, collocation searches the word which is most frequently tagged with the verb in the default dataset, which is Brown Corpus. Not only, NLTK[8] default trained models are used in this study for PoS tagging. But also, the default trained model for collocation is used.

On the other hand, generating an action graph is based on the action of each sentence. The sequence of the identified steps is mainly determined by the action in that step. If a sentence does not have any ingredient tags, the action may not be related to the next action or may not be relevant with the ingredients which are used before action. This action is generally applied regardless of the order in the directive. In order to generate the graph, the relations between the actions are found.

Each action node is generated and linked with related ingredients and tools in its sentence, when the action node generation is finished, linking actions process begins. Actions are generally sequential in the recipe. However, some actions are related to a specific action or

relevant to the next action in the recipe. "preheat" is the action related to the "bake for" action. Because only after the "preheat the oven" task, "bake for the mixture" can be done. This "preheat" action should be performed before the "bake for" action. "preheat" action's result is related to the "bake for" action. In order to find this relation and similarity about whole actions in the directive, the word embedding (word2vec) is used. If the similarity is bigger than a predetermined threshold value ($threshold = 0.95$), the pairs and the similarity values are held in the list. Then the list is used for linking the action nodes in the graph. Using the word2Vec [7] actions are converted into 300-dimensional vectors. If creating the word embedding from domain-specific corpus improves the results or not, the domain-specific word embedding is trained using the recipes crawled for this research.

The configurations of the proposed word2Vec Model is depicted as below:

- **num_features** , used to define dimensionality of the resulting word vectors. In this study, the num_features is 300.
- **min_word_count** , is minimum word count threshold: In this study, 3 is selected.
- **num_workers** , is the number of threads to run in parallel: In this study, 8 is selected.
- **downsampling** , used to downsample setting for frequent words: In this study, 0.00001 is selected.
- **sg**, is a number that if the value is 1, the skip-gram algorithm otherwise CBOW algorithm is run: In this study, 1 is selected.
- **context_size** , used to define context window length: In this study, 2 is selected.

Using Word2Vec, cosine similarity between action words are calculated, as shown in Equation 9. The similarity is calculated between the action which does not have any relations with ingredients and other actions in the directive.

$$\theta = \operatorname{argmax}(\cos(\operatorname{word2Vec}(A_k, V_w))) \quad (9)$$

A_k is the vector representation of the action word. If A_k is a phrasal verb such as "put into", "bake for" etc., each word is converted to vector and averaged. V_w represents the vector array

of whole verbs in the directive. The similarity of cosine is calculated between the average vector and each vector of V_w in the recipe.

θ is the maximum value between the action A_k that does not have any ingredients in its sentence, and each action of the whole verbs of action array V_w in the recipe. That value gives information about which action in the array V_w is most likely related with action A_k .

3.3. Recipe Parsing

When all sentences in the directive are tagged, tags are updated according to the model depicted in Figure 3.3.. Definition of each tag is shown on Table 3.9.. The tagged sentence is updated with the help of PoS tagger, Ingredient Tagger, and the dictionary.

Table 3.9. Tag Definition of The Proposed Model

Tags	Meanings	Source
PRED	verb tag	PoS tagger
PRED_PREP	verb tag with adverb	PoS tagger
DOBJ	whole ingredients in the sentence	CRF/Bi-LSTM
NON_INGREDIENT_SPAN	object is not related with ing.	PoS tagger
NON_INGREDIENT_SPAN_VERB	verb that is not related with ing.	PoS tagger
INGREDIENT_SPAN	object that is related with ingredient	CRF/Bi-LSTM
INGREDIENTS	pure ingredient	CRF/Bi-LSTM
PARG	object that is related with tool	The Dictionary
PREP	preposition	PoS tagger

Using Bi-LSTM and CRF for parsing ingredient, sequential relations between words and tags of the words are found in the ingredient list of a recipe. Then with the help of relations, ingredients in each sentence of a directive are updated as depicted on Table 3.10..

Table 3.10. Bi LSTM/CRF Tags Updated Sentence

put	the	peppers	into	the	pan
VB	DET	NAME	ADP	DET	NOUN

Another problem that needs to be solved is finding tools in a directive. If tagged data is available, supervised approaches can be used. Extracting tool is another searching field automatically. Although the proposed model catches noun phrases, it cannot decide whether it is a tool or an implicit object. After all, if the whole tool list can be generated as a dictionary,

this problem can be converted as finding the word in the dictionary. Thanks to the website³, the dictionary is generated. If the words whose tags are "NOUN" and "ADV" are found in the dictionary, the tags of the words will be updated as "TOOL" which is shown in Table 3.11..

Table 3.11. Example Results of Updating Tools

put	the	peppers	into	the	pan
VB	DET	NAME	ADP	DET	TOOLS

Tags of the sentence are merged by adhering to defined rules. Defined rules are:

- If "VERB" tag's next tag is in ["ADV", "ADP", "ADJ", "PRT"], the words are merged, tagged with PRED_PREP and add to the list. Else "VERB" tag is updated with "PRED" and is added to the list.
- If "NAME" tag's next or previous tag is in ["NUM", "COMMENT", "QTY", "ADP", "DET", "UNIT", "ADJ"], the words are merged and tagged with "INGREDIENT_SPAN" and each "NAME" tag is updated as "INGREDIENTS" and each of them is added to the list.
- If "TOOL" tag's next or previous tag is in ["ADP", "DET", "NOUN", "NUM", "ADJ", "COMMENT"], the words are merged and tagged with "NON_INGREDIENT_SPAN" and each "TOOLS" tag is updated as "PARG" and each of them is added to the list
- If the sentence has two or more "INGREDIENT_SPAN", also these words are concatenated, and each of them is added to the list.
- If the sentence has second "VERB" tag without a conjunction, also if "VERB" tag's next tag is in ["ADV", "ADP", "ADJ", "PRT"] or not, the words are merged, tagged with NON_INGREDIENT_SPAN_VERB and add to the list.

Thus, tags are updated according to neighbor tags. For instance, if the tag is "NAME" and the neighbor tag is "DET" (determiners), two words are concatenated by the model and tagged by "INGREDIENTS". That is depicted in the Table 3.11. and Table 3.12..

A tagged and processed sentence in the directive, shown in Table 3.13., is more readable for computer and ready to use for the graph generation.

³<http://www.enchantedlearning.com/wordlist/cookingtools.shtml>

Table 3.12. The Output sentence of the model I proposed

put	the peppers	into the pan
PRED	INGREDIENTS	NON_INGREDIENT_SPAN

Table 3.13. Tagged Sentence Example

The Example of Tagged Sentence
SENT : combine the chicken , celery , almonds , bell pepper, onion , pimento , salt , lemon juice , and mayonnaise . PRED : combine INGREDIENTS : chicken INGREDIENTS : celery INGREDIENTS : almonds INGREDIENTS : bell pepper INGREDIENTS : onion INGREDIENTS : pimento INGREDIENTS : salt INGREDIENTS : lemon juice INGREDIENTS : mayonnaise DOBJ : the chicken

3.4. Action Graph Generation

A graph is a set of nodes connected by edges. Each node is called a vertex [35]. Action graph is a kind of diagram that shows the relations between the actions. If an action is linked with another action, action has a direct relation with the other. Also, an action in the graph, which has one outward link with action is the beginning action. The last action in the graph is that it is not linked to any other action. While generally actions follow the flow of the recipe and are linked to the consecutive actions in the text, some are challenging as these actions are independent and can be done in parallel.

To generate the graph visualization with, Graphviz [36] library is used, drawing “hierarchical” or layered graphics. The placement algorithm targets edge in the same directive (from top to bottom or from left to right) and tries to avoid edge transitions and reduce edge length. This library supports many file extensions as, “.dot”, “.neato”, “.fdp”, “.sfdp”, etc. The graph visualization method has been implemented, generating “.dot” files for the recipe models with Graphviz library.

In the Action Graph Model, two main tasks must be done for creating the action graph:

- to create sentence nodes
- to create edges between sentence nodes

The segmented recipe, along with the resolved action links can be used to construct the graph. A connection gives information about the origin of a given the word or word array as either the output of a previous action or as a new ingredient or entity being introduced into the recipe. Each node is defined as a tuple as $(word, tag, index, isVerb)$. If the word is not a verb, the word will be connected to the verb node in the sentence. S represents sentence, C is the connection function between verb and words in the sentence, K represents counts of tuples in the sentence, and W is node list without verb node of the sentence S . (V_S) is the verb of the sentence S .

$$N_s = C_{k=1}^K(w_k, V_S) \quad (10)$$

Each sentence node (N_s) is a tuple as $(action\ node, verb, index)$. After Sentence Nodes (N_s) are created with Equation 10, edges between sentence nodes are created. For this process, action relation list (L_R) is used. L_R list's each element is a tuple that $(verb_i, verb_j, S_c)$. S_c is the value of cosine similarity between $(verb_i)$ and $(verb_j)$ which is calculated with the aid of word2Vec [7]. Action graph of a recipe A_r is generated with the Equation 11.

$$A_r = N_{k=1}^K, L_{R_{n=1}}^N \begin{cases} \text{if } S_i || S_{(i+1)} \text{ has ingredients, } C(N_k, N_{(k+1)}) \\ \text{otherwise } S_c > (S_{c_1}, S_{c_2}, \dots, S_{c_k}) C(N_k, N_n) \end{cases} \quad (11)$$

K is the action node indexes in a recipe. If a sentence does not have any ingredient tags, the model consults the related verb list (L_R) . If the similarity of cosine (S_c) is greater than others, the model creates an edge between sentence nodes, otherwise, behaves like a sequential and creates an edge between the sentence node and the next indexed sentence node. Also, information about the shapes of nodes in the action graph is depicted in Table 3.14..

To express this process step by step, real sentence node example is shown in the Figure 3.7.. Also simple sentence node example shown on Figure 3.6.

Table 3.14. The Shape Definitions of Action Graph Nodes

Tags	Colors
PRED	circle
PRED_PREP	circle
DOBJ	parallelogram
NON_INGREDIENT_SPAN	box
NON_INGREDIENT_SPAN_VERB	ellipse
INGREDIENT_SPAN	hexagon
INGREDIENTS	hexagon
PARG	box

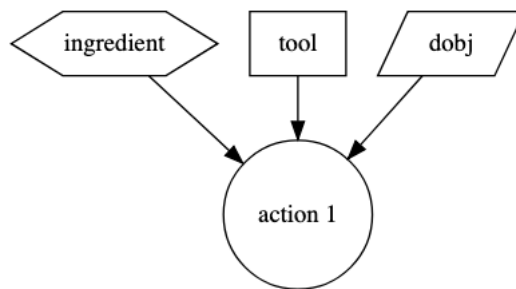


Figure 3.6. The Example of Sentence Node

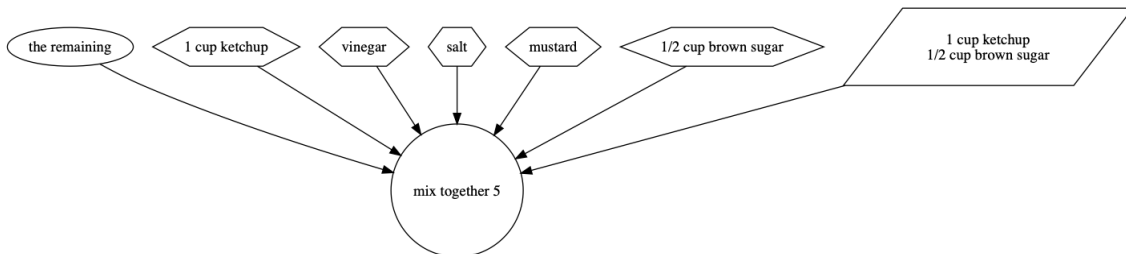


Figure 3.7. The Example of Real Sentence Node

Each sentence node has one main action in order to link with other action. If an action does not have any ingredient nodes, to link this action, related verb list as expressed before is used. The whole directive of the Amish Meatloaf Recipe is shown in Table 3.15.. Because the first sentence does not have any ingredients, the action graph starts with the second sentence. With the help of related verb list, the model link *preheat* action with *bake for* action.

The whole results of the parsed directive according to the proposed model is shown in Table 3.16. and Table 3.17. that first sentence is *preheat the oven to 350 degrees f (175 degrees c)* but action graph starts with *mix together* action. Because *preheat* action does not have any ingredients in its sentence. Thus, according to the model, the first action must have at least

Table 3.15. The Directive of Amish Meatloaf

No	Directive
1	preheat the oven to 350 degrees f (175 degrees c).
2	in a medium bowl , mix together ground beef , crushed crackers , onion , eggs , 3/4 cup ketchup and 1/4 cup brown sugar until well blended.
3	press into a 9x5 inch loaf pan.
4	lay the two slices of bacon over the top .
5	bake for 1 hmy in the preheated oven , or until cooked through .
6	while the loaf bakes , mix together the remaining 1 cup ketchup , vinegar , salt , mustard and 1/2 cup brown sugar.
7	spread over the top of the meatloaf for the last 15 minutes.

one ingredient in the sentence.

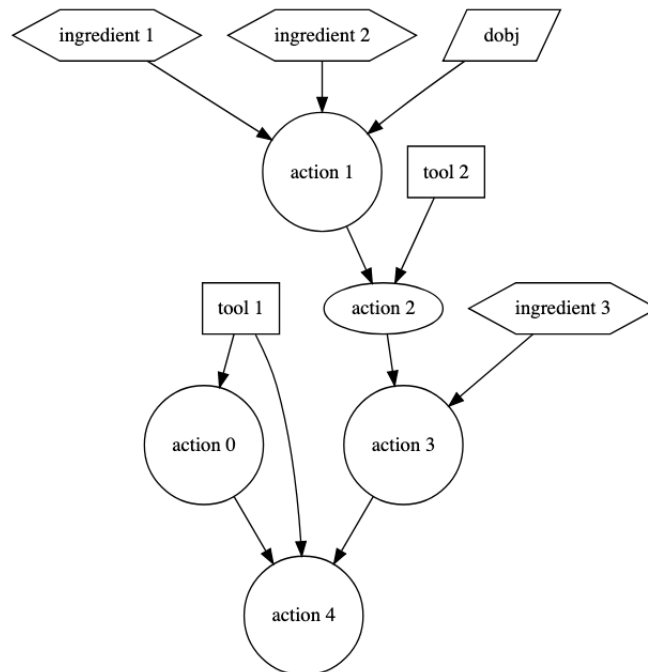


Figure 3.8. The Example of An Action Graph

The whole result of the action graph is depicted on Figure 3.9. and general example of an action graph is shown on Figure 3.8.. On Figure 3.8., action 's number is related to the indexes of sentences in the recipe. Also the algorithm looks for tools relation between actions, if one of the action node has a tool node. For instance, *preheat* sentence and *bake for* sentence have same tool, *oven*. Cosine similarity between *preheat* and *bake for* is over 95 percentages. These consequences prove that *bake for* action is related with *preheat* action. After whole linking process is done, the model is saved as a dot file.

Table 3.16. Parsed Directive of Amish Meatloaf 1

No	Directive
1	<pre> preheat the oven to 350 degrees f (175 degrees c). PRED : preheat PARG : oven NON_INGREDIENT_SPAN : the oven </pre>
2	<pre> in a medium bowl , mix together ground beef , crushed crackers , onion , eggs , 3/4 cup ketchup and 1/4 cup brown sugar until well blended. PRED : mix PRED_PREP : mix together NON_INGREDIENT_SPAN_VERB : until blended INGREDIENTS : beef INGREDIENT_SPAN : ground beef INGREDIENTS : crackers INGREDIENT_SPAN : crushed crackers INGREDIENTS : onion INGREDIENTS : eggs INGREDIENTS : ketchup INGREDIENT_SPAN : 3/4 cup ketchup INGREDIENTS : brown sugar INGREDIENT_SPAN : 1/4 cup brown sugar PARG : bowl NON_INGREDIENT_SPAN : in a medium bowl DOBJ : ground beef - crushed crackers - 3/4 cup ketchup - 1/4 cup brown sugar </pre>
3	<pre> press into a 9x5 inch loaf pan. PRED : press PRED_PREP : press into PARG : pan NON_INGREDIENT_SPAN : into pan NON_INGREDIENT_SPAN : 9x5 inch loaf pan </pre>
4	<pre> lay the two slices of bacon over the top . PRED : lay INGREDIENTS : bacon INGREDIENT_SPAN : the two slices of bacon DOBJ : the two slices of bacon </pre>

Table 3.17. Parsed Directive of Amish Meatloaf 2

No	Directive
5	bake for 1 hmy in the preheated oven , or until cooked through . PRED : bake PRED_PREP : bake for NON_INGREDIENT_SPAN_VERB : until cooked PARG : oven NON_INGREDIENT_SPAN : for hmy in the preheated oven
6	while the loaf bakes , mix together the remaining 1 cup ketchup , vinegar , salt , mustard and 1/2 cup brown sugar. PRED : mix PRED_PREP : mix together NON_INGREDIENT_SPAN_VERB : the remaining INGREDIENTS : ketchup INGREDIENT_SPAN : 1 cup ketchup INGREDIENTS : vinegar INGREDIENTS : salt INGREDIENTS : mustard INGREDIENTS : brown sugar INGREDIENT_SPAN : 1/2 cup brown sugar DOBJ : 1 cup ketchup - 1/2 cup brown sugar
7	spread over the top of the meat loaf for the last 15 minutes. PRED : spread PRED_PREP : spread over

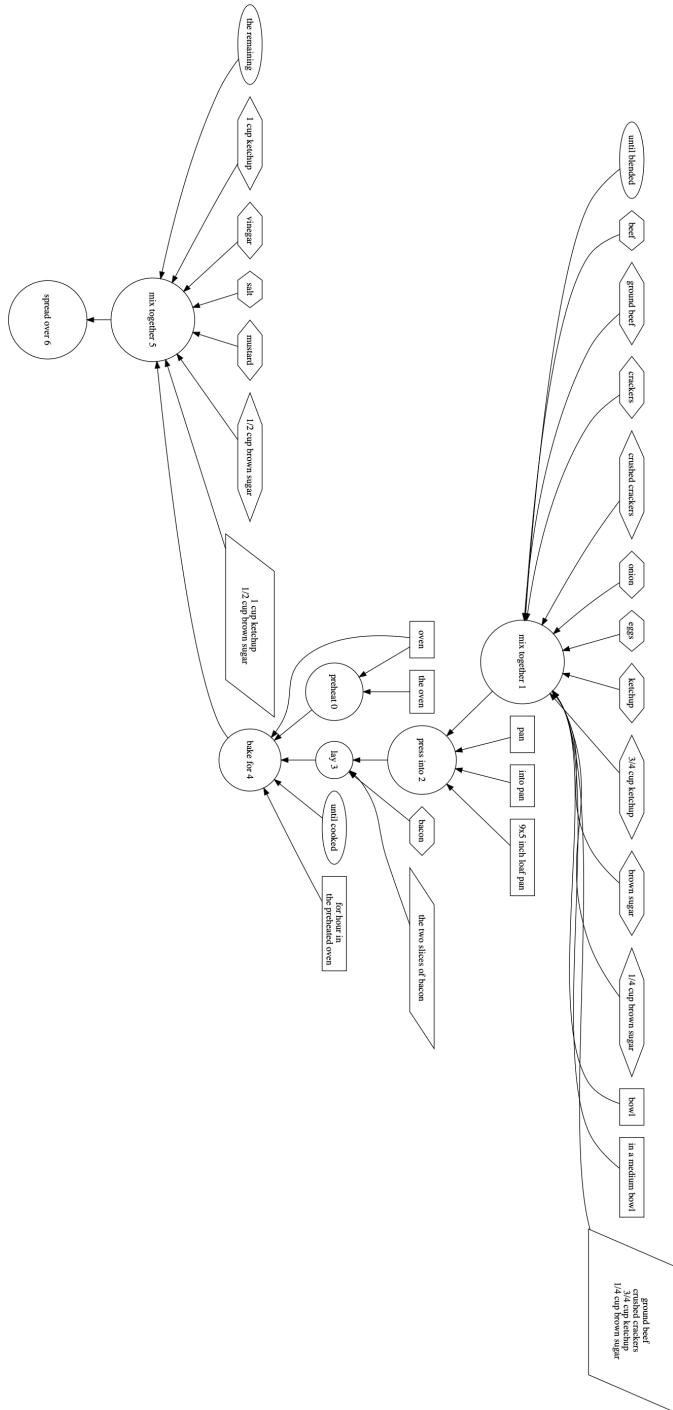


Figure 3.9. The Action Graph of Amish Meatloaf

4. EXPERIMENTAL ANALYSIS

4.1. Datasets

Two different corpora are used for the experiments. The first corpus is used to evaluate the final action graphs. The second corpora are used to learn and evaluate the named entity recognition task for the recipes and recognizing the ingredients.

The named entity recognition corpus is formed of 180,000 manually labeled ingredients [3]. For the CRF[5] algorithm and Bi-LSTM algorithm, 120,000 ingredients are used for training and 60,000 for testing.

On the other hand, to evaluate parsing directive, Kiddon [2] corpus, which has 2456 directive, are used. This corpus is used only for segmentation evaluation. Also, Kiddon generates their action graphs with the help of hand-labeled 33 recipes, which are gold standard annotated. 34,600 recipes from www.allrecipes.com and www.cooking.nytimes.com are collected, which are not utilized in the experiment.

Table 4.1. The Details of the Greene Ingredient Dataset used for Training and Testing

Data Sets	Training Set	Test Set
CRF/Bi LSTM	120,000	60,000

Because the NY Times corpus consists of NY Times recipes' ingredients, the corpus used for both CRF and Bi LSTM algorithm, which is also shown in Table 4.1.. On Table 4.2., corpora of Kiddon at. al. [2] used for evaluating action graph results is shown.

Table 4.2. Kiddon Dataset for Testing

Data Sets	Action Graph Gen.	Directive Parsing
Test Set	33	2456

4.2. Evaluation Metrics

In this section, evaluation metrics are defined. Accuracy, precision, recall, and F1 score are used for evaluation. The definition of those metrics follows:

- **True positives (tp):** The situation that I predict is right, and it is actually right.
- **True negatives (tn):** The situation that I predict is wrong, and it is actually wrong.
- **False positives (fp):** The situation that I predict is right, but it is actually wrong.
- **False negatives (fn):** The situation that I predict is wrong, but it is actually right.

Here are the details of the evaluation metrics that the study is used:

- **Accuracy** is the ratio of correct predictions to whole inputs.

$$accuracy = \frac{tp + tn}{total} \quad (12)$$

- **Precision** is the ratio of true positive predictions to all positive predictions. For parsing, verb, ingredients, and tools which are segmented correctly are divided with entities which are segmented mistakenly correct and entities which are segmented correctly.

$$precision = \frac{tp}{tp + fp} \quad (13)$$

- **Recall** is the ratio of true positive predictions to the sum of true positive and false negative. For parsing, true labeled words are divided by the sum of true labeled words and wrong labeled words.

$$recall = \frac{tp}{tp + fn} \quad (14)$$

- **F1 score** is the mean of precision and recall. It is used as a statistical measure to evaluate performance.

$$f1 = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}} \quad (15)$$

4.3. Experiments

The proposed model consists of 3 sub-processes. So, three processes are compared:

- To compare ingredient parsing
- To compare directive parsing
- To compare action graphs

The ingredient parsing approaches, which are Bi LSTM approach and the CRF model, are compared. The model firstly uses CRF approach for parsing ingredients. The result is 89 percentages for accuracy. However, the result of Bi-LSTM is much better than the CRF approach that is shown in Table 4.3..

Table 4.3. Experiment Results of Parsing Ingredient

	Precision	Recall	F1	Accuracy
Bi LSTM	97.59	96.56	97.07	96,91
CRF	89.6	85.03	88.8	89.1

Some examples of ingredient named-entity results are shown on Table 4.4..

Table 4.4. Parsed Ingredient Phrases

1	teaspoon	brown	sugar	-	-	-
NUM	UNIT	NAME	NAME	-	-	-
1/2	cup	dry	bread	crumbs	-	-
NUM	UNIT	NAME	NAME	NAME	-	-
1/2	cup	dry	bread	crumbs	-	-
NUM	UNIT	COMMENT	NAME	NAME	-	-
3/4	pound	chicken	breast	meat	finely	chopped
NUM	UNIT	NAME	NAME	NAME	COMMENT	COMMENT

Connected with the result of ingredient labeling, the directive parsing model is compared with Kiddon [2] segmentation model. Segmentation means that directive is divided into its sentences, and each sentence is divided into its actions, ingredients, tools, etc. Kiddon’s automatic segmentation (AS) result is a state-of-the-art unsupervised method for recipes.

However, the proposed model appears to be more productive. Because Kiddon’s segmentation model is depended on Dependency parser, Standford Natural Language Processing Group. With the help of dependency parser, they find related words for action. Also, they implement Expectation Maximization, which is defined in Section 2.1.1.5. to their model. This makes the model more complicated and prolongs computational time while training. In order to find relations for actions in a recipe, This study tries to use a simpler subtask like PoS tagging rather than a more involved Dependency parsing component which might be susceptible to more errors.

On the other hand, the results are better than Kiddon’s result, which is shown in Table 4.5.. Three main problems for parsing directives: finding actions, finding ingredients, and finding tools. To find tools, creating a dictionary is a convenient option and easy way because whole utensils used in the kitchen are given on the website ¹ in the alphabetical order.

Table 4.5. Experiment Results of Parsing Recipe

	Precision	Recall	F1
Sequential baseline w/ ingredients(AS)	60.4	57.2	58.8
Kiddon et. al (AS)	68.7	65.0	66.8
The proposed model	88.64	79.47	82.10

The Table 4.5. shows sequential baseline. Since most connections are sequential, that is, most of the expansion of the arguments depends on the output of the previous action - sequential connections provide a strong foundation. Generally, the model links the output of each predicate to the first predicate-argument range of the predicate below, which is expressed in Section 3.4..

Table 4.6., shows results that evaluating relation with the help of Google word2Vec model and the word2Vec model. By the aid of Gold Standard Annotated Dataset [2], the results indicate that it is improved with approximately 10 points by the model. However, Google word2Vec Corpus result is superior to the recipe Word2Vec corpus, which one thousand times smaller than Google’s. On the other hand, when word2Vec algorithm running on the ram is considered, the performance of the proposed word2Vec is much faster than Google’s, which needs around 6 GB memory.

¹<http://www.enchantedlearning.com/wordlist/cookingtools.shtml>

Table 4.6. Experiment Results of Generating Action Graph

Models	Precision	Recall	F1
The Proposed word2Vec Model Result	79.2	65.9	72.0
Google word2Vec Model Result	80.4	69.2	75.2
Kiddon Result After EM	68.2	65.0	66.8

The more the directive is parsed correctly, the more successful and logical action graph is created. In order to evaluate the action graphs, this thesis utilizes gold standard annotations dataset from Kiddon [2], which contains 33 hand-labeled recipes and graph files. The accuracy of the model which is around 80 percentages from that dataset is calculated. Also Table 4.5. shows the results of action graphs. In this case, this thesis accepts that every action is sequential, which means actions applied in the order of the directive.

4.4. Error Analysis

An error source appears to be in finding the actions in, especially long sentences. In order to find the action from imperative sentences, *I would* is added before the sentences are parsed. Some sentences like "**I would** when crisp and brown, remove to paper towels.", it is not a better solution for this. Because PoS tagger needs to be trained with imperative sentences. However, this kind of dataset is not found. The output also shown on Table 4.7.. In the first sentence, the model can not find *crisp*, and *brown* actions which effects *remove* action.

Table 4.7. Wrong Parsed Sentences

No	Sentence
1	when crisp and brown , remove to paper towels. PRED : remove INGREDIENTS : to
2	in a skillet over medium heat , brown ground beef with onion PRED : heat INGREDIENTS : beef INGREDIENT_SPAN : brown ground beef INGREDIENTS : onion INGREDIENT_SPAN : with onion PARG : skillet NON_INGREDIENT_SPAN : in a skillet DOBJ : brown ground beef - with onion

On the other hand, The model finds *heat* as an action, but the right action is *brown*. the model tags *brown* as an ingredient, as *brown ground beef* is in the ingredient list. Collocation finder is used for finding action in the sentence. This gives us that the probable verb for the sentence is *heat*. But the correct answer is *brown*.

Since the model looks at related verbs to link the actions with which does not have any ingredients, this can also cause errors because some actions which have ingredient entities cannot have any relation for the next actions.

If any verbs in the sentence are not found, collocation finder finds the probable verbs in the sentence. As seen on the second row of the Table 4.7., sometimes it gives the wrong word to tag as an action. Because verb tag of *heat*'s frequency is bigger than verb tag of *brown*'s frequency, if the dataset is much more extensive, the results might be different.

5. CONCLUSION

5.1. Conclusion

In this thesis, this study presents supervised and rule-based approaches for segmenting recipes to tag actions and ingredients. The proposed model uses Bi LSTM models to identify ingredient tags, PoS tagging to find actions and Word2Vec [7] model to identify the ordering of actions when constructing the action graph. This graph can be converted to a computer-readable format and used by other methods as a model for the tracing the steps of carrying out a recipe.

The proposed model performs outstanding from other models. According to Sequential baseline, our result is 4 points better than Kiddon's result whose model is complicated and needs more computational time. EM model [2] calculates probabilities of a word and tag pairs. To handle this problem, before parsing directive ingredients which is more comfortable and lighter is parsed. While parsing directive with PoS tagging, the tags of words related to the ingredients are updated. Thus, this process reduces the processing time on the computer. The recipes are labeled with a specific tag. Actions in the directive and similarity with actions that do not have any ingredients in the sentence are found. After all these processes, action graphs are generated, which are convertible and understandable by the computer.

To answer the research questions, the study shows that the supervised and rule-based hybrid method is improved by stacking different NLP techniques like PoS tagging (Brown Corpus), named entity recognition and lexical similarity that are trained with out-of-domain training data for building recipe action graphs. Bi LSTM approach is a successful neural network method for finding named entity recognition in ingredients rather than CRF based approach in the cooking domain. On the other hand, parsing directive can be done with PoS tagging and Collocation Finder. Because actions in the directive from a recipe can not be found, and these actions sufficient to generate an action graph. Collocation Finder approach is a way to find missing actions in the sentence of a directive. Word2Vec is an approach to find relationships between actions to create action graphs. Generating action graph from a recipe can be done with the help of word2Vec to find cosine similarity between actions in the vector domain.

5.2. Future Research

Future work includes learning a more comprehensive model of locations (e.g., identifying nested locations such as an oven and a pan in the oven), enriching action graphs with more excellent semantic coverage (e.g., durations, tools, amounts), and training and evaluating on more massive datasets. It is also planned to use the techniques to support related tasks, such as instructional recipe generation.

Time words also are significant for actions. Time of action given from the sentence gives more information about creating edges. If time words are found by the model, time-based action graph can be producible.

Not only recipe texts are crawled, but also videos from recipes are collected. With the help of this dataset, a model for generating action graph from videos to compare the results of the model produced in this thesis can be created.

These techniques implemented in this study also can be used to support related tasks, such as instructional recipe generation.

A APPENDIX RECIPE PARSER OUTPUTS

Table 1.1. Easy Whole Banana Muffins

No	Direction
1	SENTENCE : preheat oven to 350 degrees f (175 degrees c). PRED : preheat PARG : oven
2	SENTENCE : mix bananas , salad dressing , and sugar in a large bowl until smooth. PRED : mix INGREDIENTS : bananas INGREDIENTS : salad INGREDIENTS : sugar INGREDIENT_SPAN : dressing sugar PARG : bowl NON_INGREDIENT_SPAN : in large bowl DOBJ : dressing sugar
3	SENTENCE : stir flour , baking soda , and salt into banana mixture until batter is just moistened. PRED : stir NON_INGREDIENT_SPAN_VERB : until is NON_INGREDIENT_SPAN_VERB : moistened INGREDIENTS : flour INGREDIENTS : baking soda INGREDIENTS : salt INGREDIENTS : banana INGREDIENT_SPAN : into banana DOBJ : into banana
4	SENTENCE : divide batter evenly into 24 muffin cups . PRED : divide PRED_PREP : divide batter
5	SENTENCE :bake in the preheated oven until a toothpick inserted into the center comes out clean, about 17 minutes. PRED : bake PRED_PREP : bake in NON_INGREDIENT_SPAN_VERB : until inserted NON_INGREDIENT_SPAN_VERB : into the comes PARG : oven NON_INGREDIENT_SPAN : in the preheated oven

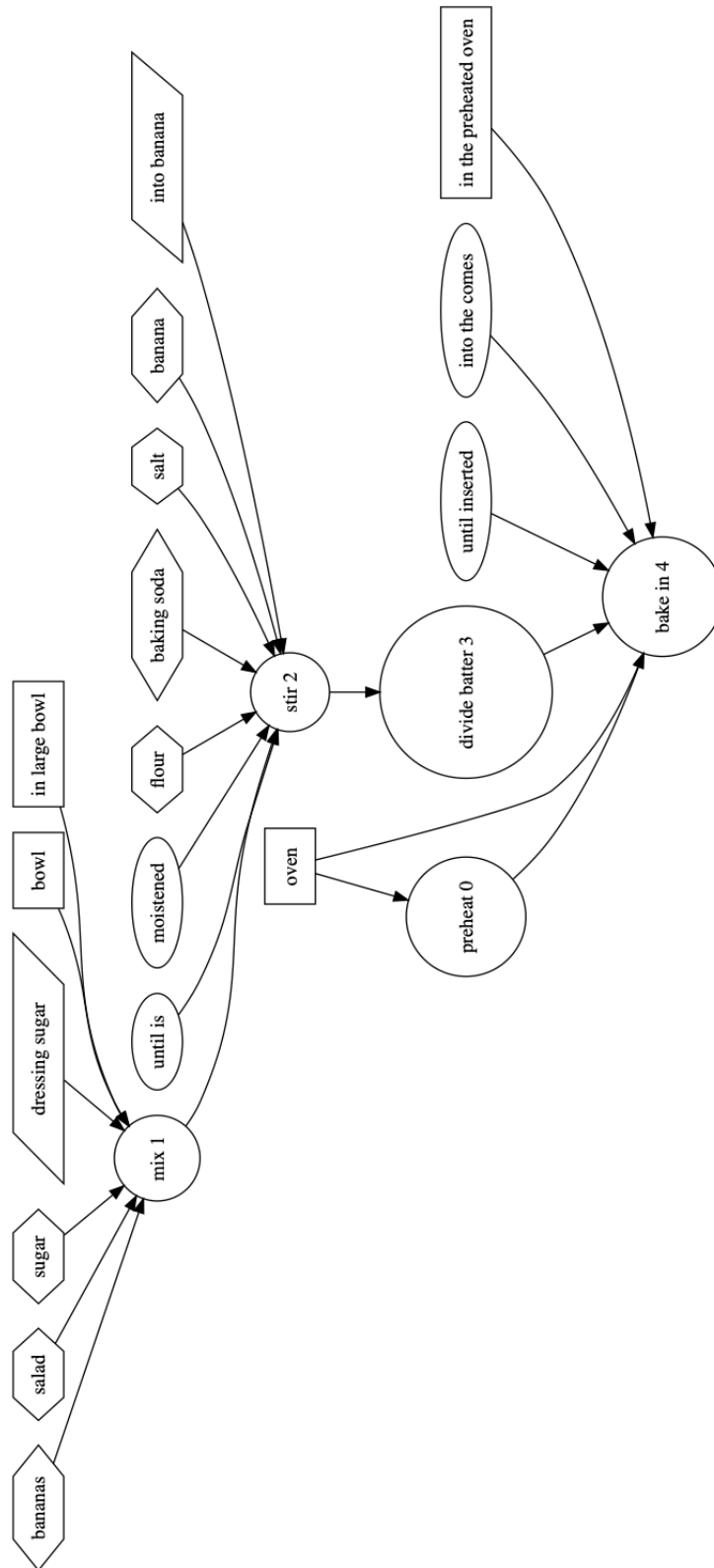


Figure 1.1. Action Graph Of Easy Whole Banana Muffins

Table 1.2. Napa Slaw

No	Direction
1	SENTENCE : melt butter in a large skillet over medium heat PRED : melt INGREDIENTS : butter PARG : skillet NON_INGREDIENT_SPAN : in large skillet
2	SENTENCE : cook and stir sunflower kernels and ramen noodles in hot butter until sunflower kernels are lightly browned and fragrant, 3 to 5 minutes . PRED : cook NON_INGREDIENT_SPAN_VERB : are NON_INGREDIENT_SPAN_VERB : browned INGREDIENTS : sunflower INGREDIENT_SPAN : until sunflower INGREDIENTS : kernels INGREDIENTS : sunflower kernels INGREDIENTS : ramen noodles INGREDIENTS : butter INGREDIENT_SPAN : in hot butter DOBJ : until sunflower - in hot butter
3	SENTENCE : remove from heat and transfer into a bowl to cool . PRED : remove PRED_PREP : remove from NON_INGREDIENT_SPAN_VERB : from transfer NON_INGREDIENT_SPAN_VERB : cool INGREDIENTS : into PARG : bowl NON_INGREDIENT_SPAN : a bowl
4	SENTENCE : stir sunflower seeds and ramen noodles with napa cabbage and spring onions in a large salad bowl . PRED : stir INGREDIENTS : sunflower INGREDIENTS : ramen noodles INGREDIENTS : cabbage INGREDIENT_SPAN : with napa cabbage INGREDIENTS : spring onions PARG : salad bowl NON_INGREDIENT_SPAN : in salad bowl PARG : bowl DOBJ : with napa cabbage

Table 1.3. Napa Slaw(continue)

No	Direction
5	SENTENCE : whisk vinegar , vegetable oil , sugar , and soy sauce together in a separate bowl until sugar has dissolved PRED : whisk NON_INGREDIENT_SPAN_VERB : has NON_INGREDIENT_SPAN_VERB : dissolved INGREDIENTS : vinegar INGREDIENTS : vegetable oil INGREDIENTS : sugar INGREDIENT_SPAN : until sugar INGREDIENTS : soy sauce PARG : bowl NON_INGREDIENT_SPAN : in separate bowl DOBJ : until sugar
6	SENTENCE : pour dressing over salad and toss . PRED : pour NON_INGREDIENT_SPAN_VERB : dressing

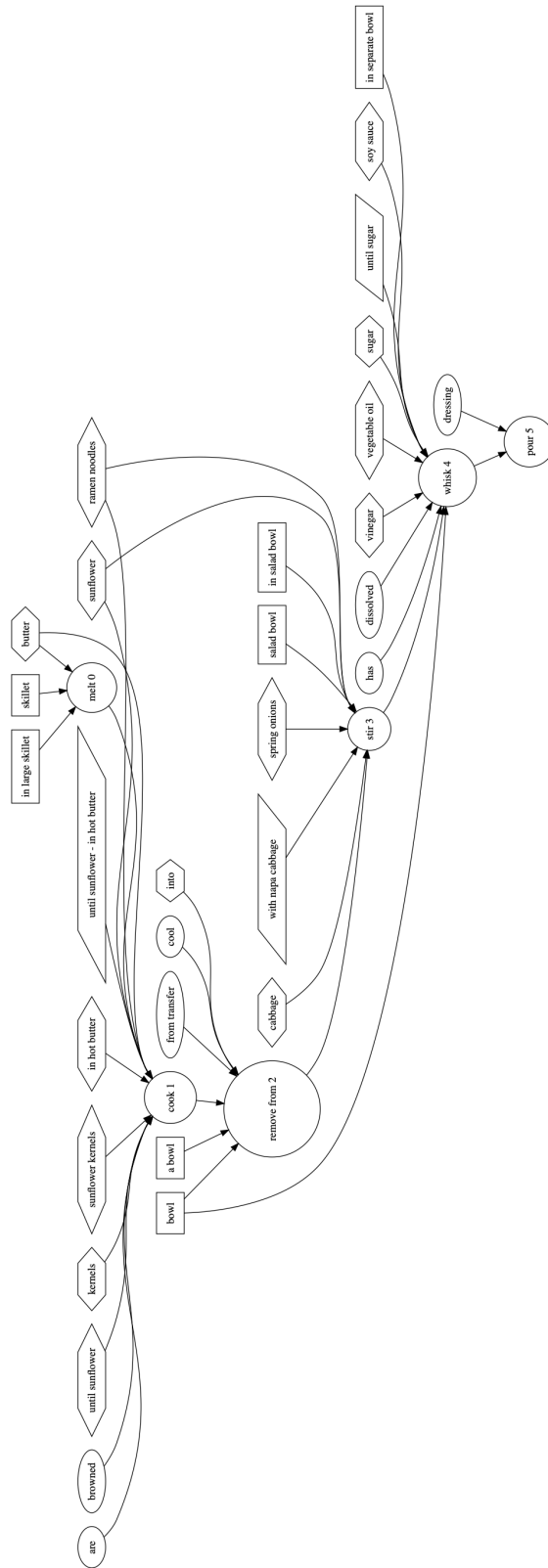


Figure 1.2. Action Graph Of Napa Slaw

Table 1.4. Taco Soup

No	Direction
1	SENTENCE : in a medium stock pot brown beef and onion , about 10 minutes. PRED : stock INGREDIENTS : beef INGREDIENT_SPAN : brown beef INGREDIENTS : onion INGREDIENT_SPAN : and onion PARG : pot DOBJ : brown beef - and onion
2	SENTENCE : drain grease if needed. PRED : drain NON_INGREDIENT_SPAN_VERB : if needed
3	SENTENCE : add tomatoes , tomato sauce , water , beans , corn and taco seasoning. PRED : add INGREDIENTS : tomato INGREDIENTS : sauce INGREDIENTS : water INGREDIENTS : beans INGREDIENTS : corn INGREDIENTS : taco seasoning INGREDIENT_SPAN : and taco seasoning DOBJ : and taco seasoning
4	SENTENCE : bring to boil , reduce heat and simmer for 5 minutes . PRED : bring PRED_PREP : bring to NON_INGREDIENT_SPAN_VERB : boil NON_INGREDIENT_SPAN_VERB : reduce
5	SENTENCE : top with cheese , corn chips , sour cream and olives . PRED : top PRED_PREP : top with INGREDIENTS : corn INGREDIENT_SPAN : with cheese corn DOBJ : with cheese corn

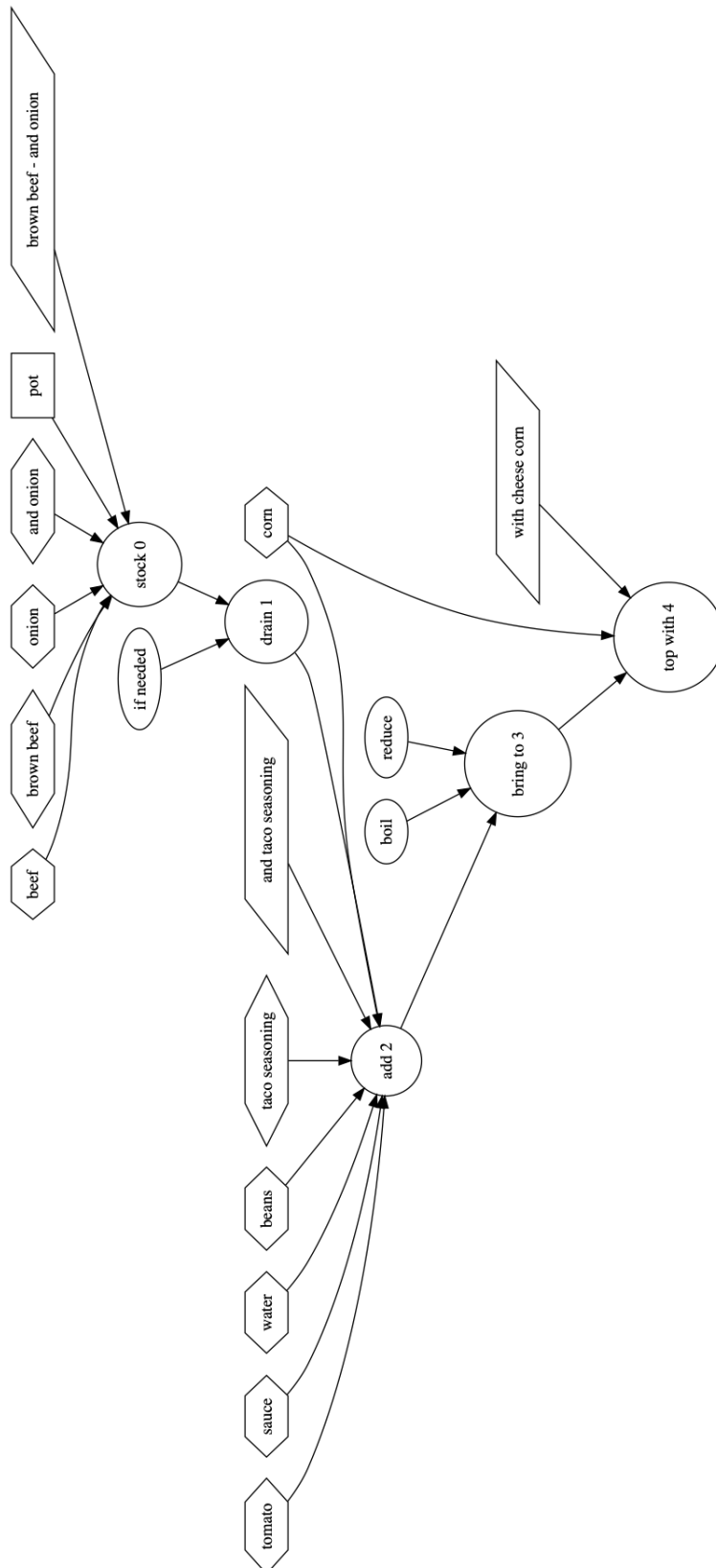


Figure 1.3. Action Graph Of Taco Soup

B APPENDIX, ALGORITHMS OF UPDATE TAGS

Algorithm 2: The Algorithm for Preparation of Directive

- 1: (ingredients, directive) = readDataFromCSVFile()
- 2: ingredientsNewTag = calculateTAGwithCRForLSTM(ingredients)
- 3: tokenizedDirective = tokenizeSentence(directive)
- 4: taggedDirective = calculateTagswithPartOfSpeechTagger(tokenizedDirective)
- 5: taggedDirective = updateTagAfterCRForLSTM(taggedDirective, ingredientsNewTag)

Input: tagged directive's sentence list taggedDirective

Input: non ingredient sentence's verb list default empty nonIngredientVerbs

Input: directive verb list verbs

Input: tool dictionary list toolList

Output: tagged directive sentences taggedDirective

- 6: **for** *sentence* \in taggedDirective **do**
 - 7: **for** (*word*, *tag*) \in sentence **do**
 - 8: **if** *tag* is *NOUN* or *ADV* and *word* \in toolList **then**
 - 9: update word tag to TOOL
 - 10: **end if**
 - 11: **if** *tag* is *VERB* **then**
 - 12: push *word* \rightarrow verbs
 - 13: **end if**
 - 14: **end for**
 - 15: **if** *sentence* has a verb but does not have any ingredient tags **then**
 - 16: push *word* \rightarrow nonIngredientVerbs
 - 17: **end if**
 - 18: **if** *sentence* does not have any *VERB* tag **then**
 - 19: find the most probable word which can be verb with collocation finder
 - 20: update word tag to VERB
 - 21: **end if**
 - 22: **end for**
-

Algorithm 3: The Algorithm for After Preparation of Directive

Require: non ingredient sentence's verb list *nonIngredientVerbs*

Require: directive verb list *verbs*

for *sentence* \in *taggedDirective* **do**

id is the number which holds the word's order in the sentence

for (*word*, *tag*, *id*) \in *sentence* **do**

if *tag* is *VERB*, check pre tag and after tag if the tag is *ADP* or *DET* **then**

 union words and push to *newTaggedDirective*

end if

if

tag is *NAME*, check pre tag and after tag if the tag is *NUM*, *COMMENT*, *QTY*, *ADP*, *DET*, *UNIT*

then

 union words and push to *newTaggedDirective*

end if

if *tag* is *TOOL*, check pre tag and after tag if the tag is *ADP*, *DET*, *NUM* **then**

 union words and push to *newTaggedDirective*

end if

end for

end for

data type of each object (*word1*, *word2*, similarity precision)

generateRelatedVerbList(nonIngredientVerbs, verbs)

REFERENCES

- [1] Word2vec model image. <https://rohanvarma.me/Word2Vec/>. Accessed: 2018-01-17.
- [2] C Kiddon, GT Ponnuraj, L Zettlemoyer, and Y Choi. Mise en place: Unsupervised interpretation of instructional recipes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 982–992. **2015**.
- [3] Our tagged ingredients data is now on github. <https://open.blogs.nytimes.com/author/erica-greene/>. Accessed: 2018-01-17.
- [4] Leonardo da vinci quotes. <https://www.goodreads.com/quotes/9010638-simplicity-is-the-ultimate-sophistication-when-once-you>. Accessed: 2018-01-17.
- [5] JD. Lafferty, A McCallum, and FCN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning, ICML '01*, pages 282–289. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, **2001**. ISBN 1-55860-778-1.
- [6] IH Witten, E Frank, MA Hall, and CJ Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, **2016**.
- [7] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method, **2014**. Cite arxiv:1402.3722.
- [8] E Loper and S Bird. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1, ETMTNLP '02*, pages 63–70. Association for Computational Linguistics, Stroudsburg, PA, USA, **2002**. doi:10.3115/1118108.1118117.
- [9] C Sutton, A McCallum, et al. An introduction to conditional random fields. *Foundations and Trends® in Machine Learning*, 4(4):267–373, **2012**.
- [10] Crf++. <https://taku910.github.io/crfpp/>. Accessed: 2018-07-23.

- [11] S Hochreiter and J Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, **1997**.
- [12] T Mikolov, I Sutskever, K Chen, GS Corrado, and J Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119. **2013**.
- [13] Stevan Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42:335–346, **1990**. doi:10.1016/0167-2789(90)90087-6.
- [14] David L Chen and Raymond J Mooney. Learning to interpret natural language navigation instructions from observations. In *Learning to Interpret Natural Language Navigation Instructions from Observations*. **2011**.
- [15] M Javad Hosseini, H Hajishirzi, O Etzioni, and N Kushman. Learning to solve arithmetic word problems with verb categorization. In *EMNLP*, pages 523–533. **2014**.
- [16] S Mori, H Maeta, Y Yamakata, and T Sasada. Flow graph corpus from recipe texts. In *LREC*, pages 2370–2377. **2014**.
- [17] J Jermsurawong and N Habash. Predicting the structure of cooking recipes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 781–786. **2015**.
- [18] Linear svm rank. https://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html#References/. Accessed: 2019-05-12.
- [19] J Malmaud, E Wagner, N Chang, and K Murphy. Cooking with semantics. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, pages 33–38. **2014**.
- [20] A Salvador, N Hynes, Y Aytar, J Marin, F Ofli, I Weber, and A Torralba. Learning cross-modal embeddings for cooking recipes and food images. *Training*, 720(619-508):2, **2017**.
- [21] R Kiros, Y Zhu, RR Salakhutdinov, R Zemel, R Urtasun, A Torralba, and S Fidler. Skip-thought vectors. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3294–3302. Curran Associates, Inc., **2015**.

- [22] E Bertini, N Elmqvist, and T Wischgoll. Classic techniques in new domains: An alternative recipe. *Eurographics Conference on Visualization*, **2016**.
- [23] V Nedovic. Learning recipe ingredient space using generative probabilistic models. In *Proceedings of Cooking with Computers Workshop (CwC)*, volume 1, pages 13–18. **2013**.
- [24] Stanford dependency parser documentation. <https://nlp.stanford.edu/software/nndep.shtml/>. Accessed: 2018-07-23.
- [25] Recipe 1 m. <http://pic2recipe.csail.mit.edu/>. Accessed: 2018-01-17.
- [26] Food 101 dataset. <http://visiir.lip6.fr/explore>. Accessed: 2018-01-17.
- [27] KM Hermann, T Kocisky, E Grefenstette, L Espeholt, W Kay, M Suleyman, and P Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701. **2015**.
- [28] Keras documentation. <https://keras.io/>. Accessed: 2018-07-23.
- [29] JPC Chiu and E Nichols. Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308*, **2015**.
- [30] I Goodfellow, Y Bengio, and A Courville. *Deep Learning*. MIT Press, **2016**. <http://www.deeplearningbook.org>.
- [31] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, **2014**.
- [32] MD Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, **2012**.
- [33] T Tieleman and G Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, **2012**.
- [34] Payam Refaeilzadeh, Lei Tang, and Huan Liu. *Cross-Validation*, pages 532–538. Springer US, Boston, MA, **2009**. ISBN 978-0-387-39940-9. doi:10.1007/978-0-387-39940-9_565.

- [35] Graph. <https://xlinux.nist.gov/dads/HTML/graph.html>. Accessed: 2018-07-23.
- [36] Graphviz documentation. <https://www.graphviz.org/about/>. Accessed: 2018-07-23.



HACETTEPE UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING
THESIS/DISSERTATION ORIGINALITY REPORT

HACETTEPE UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING
TO THE DEPARTMENT OF ~~COMPUTER ENGINEERING~~

Date: 12/09/2019

Thesis Title / Topic: Tagging and Action Graph Generation for Recipes

According to the originality report obtained by myself/my thesis advisor by using the *Turnitin* plagiarism detection software and by applying the filtering options stated below on 09/09/2019 for the total of 48 pages including the a) Title Page, b) Introduction, c) Main Chapters, d) Conclusion sections of my thesis entitled as above, the similarity index of my thesis is 10%.

Filtering options applied:

1. Bibliography/Works Cited excluded
2. Quotes excluded
3. Match size up to 5 words excluded

I declare that I have carefully read Hacettepe University Graduate School of Science and Engineering Guidelines for Obtaining and Using Thesis Originality Reports; that according to the maximum similarity index values specified in the Guidelines, my thesis does not include any form of plagiarism; that in any future detection of possible infringement of the regulations I accept all legal responsibility; and that all the information I have provided is correct to the best of my knowledge.

I respectfully submit this for approval.

12.09.2019
Date and Signature

Name Surname: Mehmet Özgen

Student No: N14223406


Department: Computer Engineering

Program:

Status: Masters Ph.D. Integrated Ph.D.

ADVISOR APPROVAL

APPROVED.


Prof. Dr. Pinar DUYGULU SAHİN
(Title, Name Surname, Signature)

CURRICULUM VITAE

Credentials

Name,Surname: Mehmet ÖZGEN
Place of Birth: Manisa, Turkey
Marital Status: Single
E-mail: ozgenmehmett@gmail.com
Address: Computer Engineering Dept., Hacettepe University
Beytepe-ANKARA

Education

BSc. : Electronic Engineering Dept., Turkish Airforce Academy, Turkey
MSc. : Computer Engineering Dept., Hacettepe University, Turkey

Foreign Languages

English

Work Experience

Electronic Engineer at 3rd Main Maintenance Base (2012-2015)
Software Engineer at Havelsan(...) (2015-Present)

Areas of Experiences

NLP, Machine Learning, Text Mining,
Unsupervised Learning, Semi-supervised Learning

Project and Budgets

-

Oral and Poster Presentations

Mapping Recipe Instructions to Action Graph, *BYOYO2017*

PUBLICATIONS

"Mapping Recipe Instructions to Action Graph" *Tubitak Electrical and Computer Science Journal, Manuscript Code:ELK-1908-111, Status : In Review , 2019.*