POINTS OF INTEREST (POI) EXTRACTION FROM SOCIAL MEDIA

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

İSMAIL TALHA YILMAZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

MAY 2017

Approval of the thesis:

POINTS OF INTEREST (POI) EXTRACTION FROM SOCIAL MEDIA

submitted by **İSMAIL TALHA YILMAZ** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Adnan Yazıcı _____
Head of Department, **Computer Engineering**

Assoc. Prof. Dr. Pınar Karagöz _____
Supervisor, **Computer Engineering Department, METU**

Assoc. Prof. Dr. Yusuf Kavurucu _____
Co-supervisor, **Turkish Naval Research Center Pendik**

Examining Committee Members:

Prof. Dr. Halit Oğuztüzün
Computer Engineering Department, METU _____

Assoc. Prof. Dr. Pınar Karagöz
Computer Engineering Department, METU _____

Prof. Dr. Ahmet Coşar
Computer Engineering Department, METU _____

Prof. Dr. İsmail Hakkı Toroslu
Computer Engineering Department, METU _____

Assist. Prof. Dr. Mehmet Tan
Computer Engineering Department, TOBB University _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: İSMAIL TALHA YILMAZ

Signature :

ABSTRACT

POINTS OF INTEREST (POI) EXTRACTION FROM SOCIAL MEDIA

Yılmaz, İsmail Talha
M.S., Department of Computer Engineering
Supervisor : Assoc. Prof. Dr. Pınar Karagöz
Co-Supervisor : Assoc. Prof. Dr. Yusuf Kavurucu

May 2017, 71 pages

A point of interest (POI) is a particular location point that is useful or interesting for people such as restaurants, museums, parks and hotels. POIs are mostly used on location based social media applications, especially for place recommendation. Social media users share the places they like and discovering such new POIs has importance for understanding the taste and preference of city citizens and for understanding the city. Therefore, detecting which word can be POI in a social media message is an important problem. This process of retrieving POIs from a text is called POI extraction. In this work, we propose methods to extract POIs from microblogs. We explore both machine learning and artificial neural network based approaches. As machine learning approach, we use Conditional Random Fields (CRF) for sequential tagging. We investigate the effect of various additional features such as sentiment of tweets, POI density and population density of the location where the tweet was posted. We also use built-in features of CRF. As a hybrid approach, we generate word embeddings by Word2vec and apply K-Nearest Neighbors classification algorithm on the vectors constructed. Finally we construct a deep, feed-forward neural network to extract POIs from microblog text. These techniques are applied on a collection of tweets in Turkish, posted by users from Ankara. Experimental results show that CRF constructed with POI density feature outperforms CRF with other feature sets along with other neural network approaches in terms of POI extraction accuracy.

Keywords: Point of Interest Extraction, Microblogs, Machine Learning, Neural Networks

ÖZ

SOSYAL MEDYADAN İLGİ NOKTASI ÇIKARIMI

Yılmaz, İsmail Talha
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Doç. Dr. Pınar Karagöz
Ortak Tez Yöneticisi : Doç. Dr. Yusuf Kavurucu

Mayıs 2017 , 71 sayfa

İlgi noktası, kişiler tarafından ilginç ve faydalı bulunabilecek restoranlar, müzeler, parklar ve oteller gibi belirli konum noktalarıdır. İlgi noktaları çoğunlukla konum bazlı sosyal medya uygulamalarında lokasyon önerimi amacıyla kullanılmaktadır. Sosyal medya kullanıcıları beğendikleri lokasyonlar ile ilgili paylaşımda bulunurlar ve bu yeni ilgi noktası bilgilerine ulaşmak, kullanıcıların zevklerini öğrenebilmek ve şehir hakkında bilgi almak açılarından önemlidir. Bununla birlikte, cümlede hangi kelimenin ilgi noktası olduğunu tespit edebilmek te önemlidir. Cümleden bu noktaları tespit ederek çıkarma işlemine ilgi noktası çıkarımı denilmektedir. Bu tezde, tweetlerde geçen ilgi noktalarını çıkaracak yöntemler önermekteyiz. Bu amaçla Makine Öğrenmesi ve Sinir Ağları temelli yaklaşımlar kullanıldı. Makine Öğrenmesi yöntemi olarak Conditional Random Fields (CRF) uygulandı. Bu yöntemde kullanıcıların duygu analizi, çevredeki ilgi noktası yoğunluğu ve çevredeki popülasyon yoğunluğu gibi özelliklerin etkisi incelendi. Bununla birlikte CRF ile gelen bazı diğer özellikler de kullanıldı. İlgi noktası çıkarımı problemine hibrit yaklaşım olarak Word2vec kullanıldı ve K-Nearest Neighbor sınıflandırma algoritması ile oluşturulan vektörler üzerinden en yakın eşleşmelerin ilgi noktası olup olmadığı kontrol edildi. Son olarak İleri Beslemeli Sinir Ağları kullanılarak ilgi noktası çıkarımı yapıldı. Bahsedilen teknikler, Ankara'dan gönderilen Türkçe tweetler üzerine uygulandı. Yapılan çalışmalar sonucunda CRF tekniğinin yakınlardaki ilgi noktası yoğunluğu özelliği ile birlikte kullanımının diğer özelliklere, daha önce bu alanda yapılmış çalışmaya ve

diğer temel algoritmalara göre daha isabetli sonuçlar sağladığı gözlendi.

Anahtar Kelimeler: İlgi Noktası Çıkarımı, Mikroblog, Makine Öğrenmesi, Sinir Ağları

To my family with deepest love...

ACKNOWLEDGMENTS

I would like to give my deepest thanks to my advisor Assoc. Prof. Dr. Pınar Karagöz for all of her support and guidance throughout this study.

I would like to thank to Meryem Sağcan for sharing her model to make comparisons with this work.

I also would like to thank to Güral Vural for sharing his sentiment dictionary, which is used to calculate sentiment feature.

Finally, I have to express my gratefulness to my mother Gülnur Yılmaz, my father Süleyman Yılmaz, my sister Esra Yılmaz and my brother Ahmet Burak Yılmaz for all of their encouragement and love during that study. Without their support, I could not be at this point.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

POI	Point of Interest
NLP	Natural Language Processing
NER	Named Entity Recognition
CRF	Conditional Random Fields
K-NN	K-Nearest Neighbor
POS	Part-Of Speech
HMM	Hidden Markov Model
CBOW	Current Bag of Words
MLP	Multi-Layer Perceptrons
RELU	Rectified Linear Activation Function
RNN	Recurrent Neural Network
CNN	Convolutional Neural Network
BILOU	Beginning Inside Last Outside Unit
VGI	Volunteered Graphic Information
TF-IDF	Term Frequency - Inverse Document Frequency
BIO	Beginning Inside Outside

CHAPTER 1

INTRODUCTION

1.1 Problem Definition

Recently, social media has become an inseparable part of daily life. With increasing mobility of technology, people share their instant activities, emotions and much more information about themselves with the people they may not even know. Twitter is an attractive social media platform for people to share ideas, moods and activities through messages limited to 140 characters. Foursquare is another popular social media platform with the purpose of sharing locations and activities on those locations. These social media platforms can also be used one within the other; Foursquare check-ins can be shared on Twitter through tweets. Besides Twitter and Foursquare, there are many other social media platforms serving different purposes. This wide variety of social media platforms and intensive usage among people generates huge collections of data that can be used for different aims.

A POI is a specific location that is attractive or important for people. Restaurants, museums, parks and hotels are examples to POIs. A POI is different from a location as it addresses to a specific point, while a location can be any point. Thus, locations such as cities, districts and even roads can not be considered as POI. We can conclude that every POI is a location but every location is not necessarily a POI. People share the information about POIs they have been through social media such as the places they visit, activities they do and their emotions about that place. POI extraction is discovering POI names from text. In this thesis work, we use Twitter data i.e. user tweets to extract POIs. In addition to that purpose, we investigate some features

in order to find out what makes it a POI. Firstly, we consider relationship between locations and emotions of people for them. We perform sentiment analysis over user tweets to detect their emotions at those places. We also search for the effects of population density at that POI's neighborhood on a specific POI. Finally, we compare POI density around a location to detect its effects on a POI.

There are some challenges in POI extraction. First of all, it is a challenge to disambiguate POIs from other text mentions as there may be conflicts between some words that are not POI but its name mentions a POI, or vice versa. Another challenge is disambiguation of POI names from location names as every location name is not necessarily be a POI.

Natural Language Processing (NLP) is a field of Machine Learning concerned with the interaction between natural language of human and computer. Named Entity Recognition (NER) is one of the areas in NLP, which aims to find some specified words in text and categorize them into some predefined classes. Neural Networks are structures that can be used on different areas of computer science such as image processing and NLP. These structures work like brain; they take inputs, process information passing over layers through neurons and make some calculations to produce an output. Deep Neural Networks are specified type of Neural Networks that contain one or more hidden layers besides input layer, which improves the accuracy and performance of network. In this study, we aim to make comparisons for POI extraction problem using different NER approaches. We use Conditional Random Fields (CRF) for sequential tagging, that handle words as sequences and categorize words on predefined classes. We use Mallet as CRF tool. Using different features with CRF, we have experiments and observe the results. After that, we study on neural networks for NLP. We construct a Shallow Neural Network and a Deep Neural Network model; we approach NER problem by calculating accuracy results. We use shallow network containing two layers with Word2vec algorithm, created by Google. We use deep learning methodology with four layers and make experiments for that NER task. We use Deeplearning4j for shallow neural network model and Tensorflow as the deep learning tool. Finally, we compare all results with baseline algorithms such as Naive Bayes and Decision Tree.

1.2 Motivation and Contributions

Social media contains huge amount of data that are provided by millions of users from all around the world. From that data, information about locations can be gained. Extraction of POI becomes important as user comments can vary from different places and information about places can be retrieved from those comments. As information are gained directly from users, they are more accurate because of user experiments on those places, which makes POI extraction very important. Moreover, using social media data, new locations that are becoming popular can be detected. Those facts make the extraction of POI crucial. The main motivation in this study is to add new features to the existing works on the POI extraction problem and compare different machine learning methods and artificial neural network approaches to improve accuracy.

In the literature, there exists several works and different approaches for the problem of POI extraction. Our proposed system is based on NER using CRFs as in [23] and [12]. However our approach is mainly different from [23] as we extract POIs from text, on the other hand they extract toponyms that may not be necessarily POI. Moreover we apply three different feature extraction methods which makes our work different from [23] and [12]. In [12], tweets sent time is considered as a feature; aiming to predict whether user has been in that place past, present or future. In our methodology, we calculate sentiment scores for each tweet and use it as a feature. Sentiment analysis performed considering three attitudes of user; positive, negative and neutral. In our work, we examine the effects of sentiment on a POI. Secondly, we fetch the coordinates of tweets where posted and with that location, we get the population density at that neighborhood. We use Turkish Statistical Institute (TUIK) data and with that data, we categorize density in three groups as low, medium and high. With that feature, we examine the effect of population around a location. Finally we work on the effects of density of POIs around a location. Whether a location with lots of POI around makes it a POI or not is the main question behind that feature. We use Google's location service and categorize results into three groups that are low, medium and high.

One of the main differences of our system from existing works is the way we retrieve data. Gazetteer based approach used in [21], where POI names retrieved directly

from Wikipedia and Foursquare. Moreover after data retrieval, they increase their training data by seeding extracted POI names into search engines. Our approach is slightly different as we take user comments into consideration. Although Twitter is a free format social media platform, most Foursquare check-in tweets do not contain any sentiment information. Most of the time, only location information occur in such tweets. We retrieve only useful check-in tweets using format checking in data retrieval part of our system. As a result, all of the data we get include user comments that possibly contain sentiment information. We use that sentiment information as a feature in CRF. After that step, data with candidate sentiment information is retrieved. However that data needs to be cleaned up so that meaningless words are eliminated and feature extraction performed over clean data. To that aim, we delete unnecessary characters, but we keep emojis as they contain sentiment information.

As another part of this study, we work on neural network architectures. Firstly, we construct a shallow neural network model with two layers containing an input layer and an output layer. On that model we apply Word2vec, which is an algorithm that process huge amounts of text and turn them into feature vectors in multidimensional vector space. Using those word embeddings, we apply K-Nearest Neighbor (K-NN) classification algorithm; i.e. we retrieve closest n number of words for each word in training dataset. Then we compare those closest words with target word if they are classified correctly and measure accuracy. In addition to shallow network, we build multilayer feed-forward deep neural network that contains an input layer, two hidden layers and an output layer. We add hidden layers to shallow network which makes the model deep. As a result of experiments, we evaluate the performance effect of deep and shallow neural networks to POI extraction problem.

To summarize, our contributions are as follows:

- We propose a method to identify a POI from given tweets. The methodology contains new features in order to detect what makes it a POI.

- The first feature is about the sentiment analysis of tweets. With that feature, we check whether user opinions affect the status of a location as a POI. We categorize tweet sentiments in three classes; positive, negative and neutral.

- As the second feature, we investigate the effect of population density at a location; whether it affects the location to be a POI or not. To that aim, we use publicly available population values for 2016 that gives the human density at all of the districts in Turkey [1].

- As the third feature we check whether the number of POIs in some specific region affects being a POI.

- Additionally, we use two neural models for POI extraction. In the first one, we built a shallow neural network with Word2vec. Using the word embeddings created by Word2vec, we apply K-NN classifier to detect whether a word denotes a POI or not. In the second one, we construct a deep neural network for POI extraction problem.

- For experiments, we collect a data set of tweets in Turkish posted in Ankara. In order to guarantee that the data set contains candidate POI mentions, we retrieve Foursquare check-in tweets containing user comments.

1.3 Thesis Organization

The next chapter focuses on techniques used in this thesis work such as CRF, Word2vec and neural networks along with the tools used such as Mallet, Deeplearning4j and Tensorflow. Chapter 3 discusses previous studies related to location extraction and POI extraction problems. In Chapter 4, methodologies used in this thesis work are defined in detail. Experiments and their results, along with comparison of baseline algorithms and previous work are presented in Chapter 5. In the last chapter, overview of this thesis work is given and future works are described.

[1] www.tuik.gov.tr

CHAPTER 2

BACKGROUND

In this chapter, techniques, algorithms and tools used in this thesis work are explained in detail. First of all, techniques used in this proposed work which are Conditional Random Fields, Word2vec and Neural Networks are described. Then, the tools used along with techniques are investigated; which are Mallet, Deeplearning4j and Tensorflow.

2.1 Conditional Random Fields

Named Entity Recognition can be done in three different ways. As the first method, rule based techniques such as regular expressions can be used. This helps to find some patterns in text and categorize them accordingly. Secondly, machine learning techniques which do not depend on text structure can be used. Different linear classification methods and Conditional Random Fields (CRF) are examples to such techniques. Finally both rule based and machine learning based techniques can be used together. In this study, some experiments contain only CRF. However there are some experiments, where both rule based methods used with machine learning based methods. In that context, there are some experiments where regular expressions along with CRF is used.

CRFs are statistical models that are mostly used in machine learning, that takes word sequences into account. CRFs are probabilistic models implemented as undirected graph. Lafferty et. al, who are presenters of CRF, define it as a framework for building probabilistic models to segment and label sequence data in [10]. On the one hand,

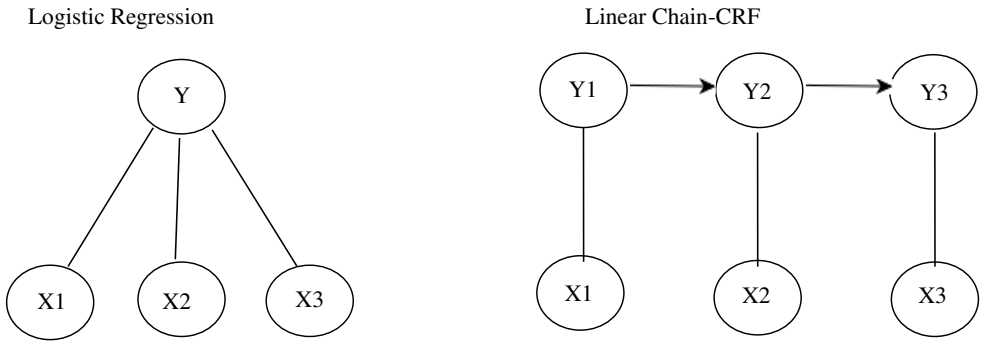

Figure 2.1: Difference between Logistic regression model and Linear-chain CRF. Features are defined as X1, X2, X3 and target variables are Y, Y1, Y2, Y3

discrete classifiers deal with words only by themselves. CRF deals with word and neighboring words using different features. CRFs are mostly used in Part-Of Speech (POS) tagging, Named Entity Recognition (NER) and computer vision areas.

Here the basic idea is to define a conditional probability distribution over labels given words rather than a joint distribution over both labels and words. Suppose we have target class variable defined as Y and features as X. With CRFs, conditional probability distribution P(Y|X). This probability distribution can be computed without knowing the relations of features. Aim is to find the label sequence that has maximum conditional probability. However with logistic regression, which is also another discriminative model, conditional probability distribution calculated. The difference is that on the one hand logistic regression is a model for classification, on the other hand CRF is a model for sequential labeling. This difference between CRF and logistic regression is shown in Figure 2.1.

CRFs are mostly compared with Hidden Markov Models (HMM), which is another type of graphical model that is also presented as dynamic bayesian network. HMMs are generative models where joint distribution is calculated with P(Y,X). This requires the relations between features. CRF outperforms HMM on some cases as explained in [20]. As conditional probabilistic nature of CRFs, more information can be learned using CRF model. Moreover CRF prevents from label bias problem, whereas HMM models cannot as they are directed graphical models.

2.2 Word2vec

Input of artificial neural networks need to be converted to different format for using in tasks such as NER for text mining and NLP. In image processing tasks, input data consist of images. Whereas in natural language processing or named entity recognition tasks, input data consist of plain texts. As it is possible to infer from the name, Word2vec simply converts words into vectors. To give a more detailed definition, Word2vec is an unsupervised neural network based algorithm that create models to produce word embeddings [6]. Word embeddings are projections of texts into vectors in vectoral space which typically contains hundreds of dimensions. In other words with embeddings, each word is represented by a vector. Vectors are grouped such that the similar words at the corpus are positioned close in the vector space.

An important point arises here about the motivation behind learning the embeddings of words. First and foremost reason is that neural networks simply do not work with plain texts; they need to have some numbers to make computations. Moreover, similarities of words have crucial role and it is possible to calculate similarities using mathematics, which word embeddings make it possible.

Word2vec is created by developers in Google. Mikolov et. al. offer a Word2vec model that contains two different architectures that are skip-gram and Current Bag of Words (CBOW) in [16]. An important point is that Word2vec is not deep learning; both architectures that Word2vec implements are shallow. Proposed architectures are very similar except the main difference. CBOW predicts target words from source context words; whereas skip-gram does the reverse by predicting context words from target words.

CBOW is based on predicting the main word by taking consideration of its neighbors. Below example sentence, the target word that is to be predicted is taken into parentheses and all neighboring words that are used to predict target word are called as context.

Ankara bir gecede doğalgazı kesilmiş (öğrenci) evine döndü
(Ankara turned to a (student) home whose gas was cut off in a night)

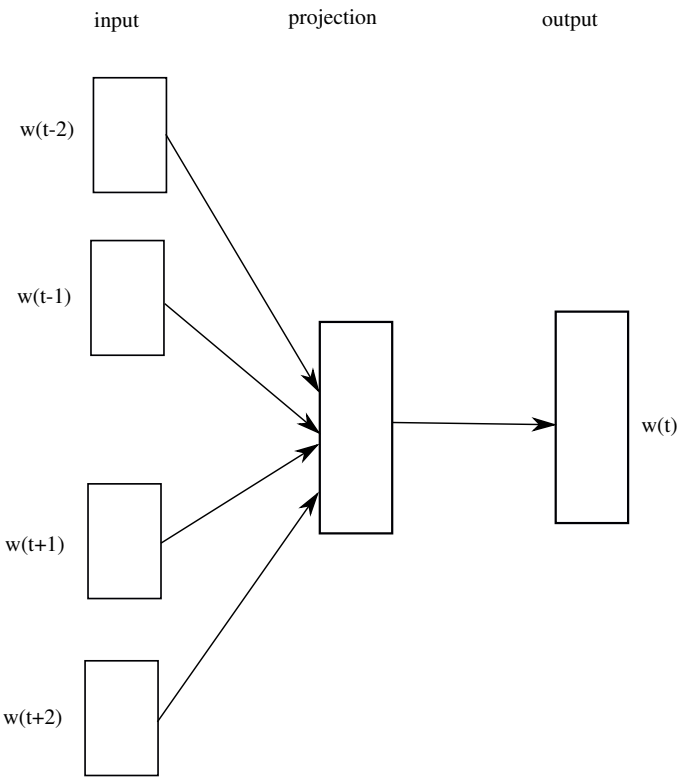

Figure 2.2: How CBOW works is explained. Input values are the context values of word a position t, that are the words at position t-2, t-1, t+1, t+2. As output, main word is predicted.

Main aim to predict target word öğrenci (student) by checking neighbor words. The input layer of CBOW consist of all context words that are encoded in one-hot vector, which sets the word as 1 and all other words as 0 in a vector. In the hidden layer, those input words are multiplied by weight matrices, sums are taken for each input and finally divided to total count to calculate average result. Aim in training is to maximize probability of target word, using the context words. How CBOW works is shown in Figure 2.2.

Skip-gram is the reverse of CBOW; aims to predict context words by taking main word into account. Above example with skip-gram predicts the neighboring words of öğrenci (student). Input layer of skip-gram has main word vector encoded in one-hot form. Then this vector is multiplied with wights and projected to output layer of each dimension in hidden layer. Aim in training is to minimize the summed prediction error for context words. How skip-gram works is presented in Figure 2.3.

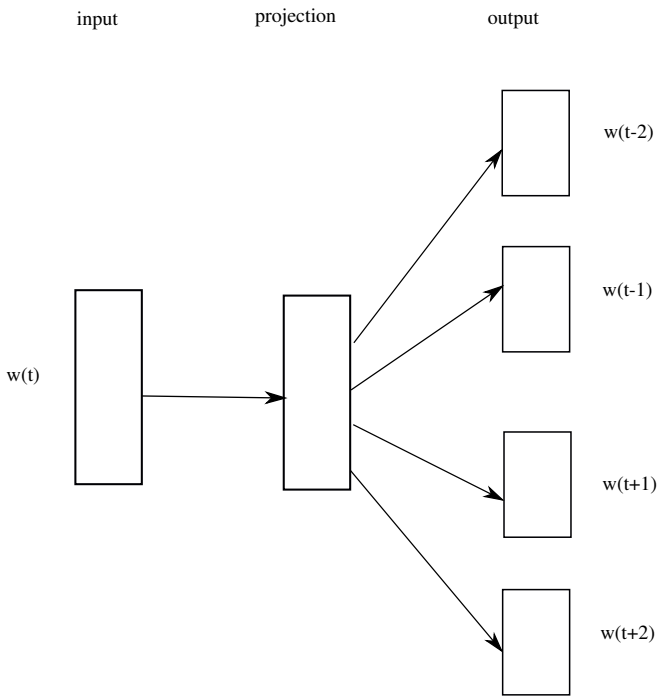

Figure 2.3: How skip-gram works is explained. Input value is the word at position t. As output values, the context values of word at position t-2, t-1, t+1, t+2 are predicted.

Main consideration of Word2vec is that during training, which words co-occur in the same window. Each word has a distributed weight representation across dimensions. As a result, each word is represented by its weights distributed across all of the elements in vector. As words tend to be together in context window, distance between those words decreases in vectoral space. Hence, the more times words grouped together, the smaller distance they have in resulting vectoral space. In each step of training with Word2vec, words are getting closer to the words they co-occur with. However it is an expensive operation to separate a word that does not co-occur with all other words in vocabulary.

To face the problem described above, Mikolov et. al. offer two methods named as hierarchical softmax and negative sampling in [17] to optimize Word2vec.

On the one hand, negative sampling has the idea that word vectors get closer to neighboring words, whereas they get separated against not all non-neighboring words but some subset of those words selected using unigram distribution of words. The probability of a word to be selected in negative sample actually increases with frequency of a word. Words that are more frequent are more prone to be selected as negative

subsample. This gives a benefit that stop words such as the, and, a etc. that give less information are more likely to be eliminated.

On the other hand, according to hierarchical softmax method, each word get closer to neighboring words and get separated from non-neighboring words by subset of words that are chosen from binary tree structure. All leaves of tree are words and the path from root to leaves used for estimating probabilities. Using tree structure improves to select non-neighboring words not from corpus size but from log2 (corpus size). In proposed work, Mikolov et. al. used Binary Huffman Tree structure as it assigns short codes to frequent words, which improves training time. Proposed work by Mikolov et. al. offers to use skip-gram with negative sampling that outperforms against other architecture types [17].

In order to group words in vectoral space to compute similarities, cosine similarities of vectors are calculated. Completely same texts have similarity score 1 and different texts have cosine similarity of their vectors.

There is an extension of Word2vec named Doc2vec, also called as Paragraph2vec. Doc2vec associates paragraphs with labels. While Word2vec associates words with other words in sentence, Doc2vec associates words with labels. Hence, while word ordering is important with Word2vec, it is not needed in Doc2vec. Doc2vec is mainly used for the purpose of finding similarities between paragraphs or documents and in some specific tasks such as sentiment analysis of groups of words.

2.3 Neural Networks

Neural networks are groups of algorithms that imitate human brain as recognizing patterns, categorize inputs, make interpretations etc. To achieve this, they take inputs, make some calculations over inputs and produce output. Just like the brain works such that taking the input, iterating data and transforming through neurons; neural networks iterate data and transform them through nodes.

Deep Learning is one of the hot topics in Machine Learning. Deep Learning is an artificial neural network that consists of more than one hidden layers.

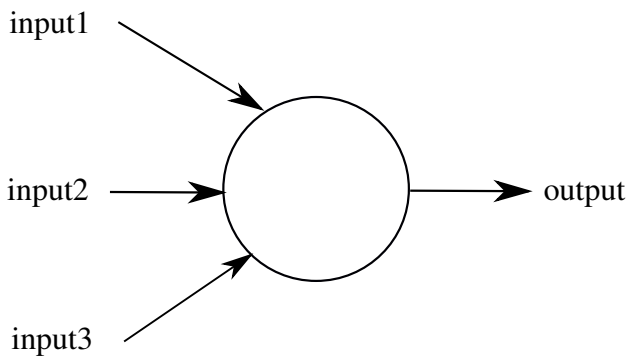

Figure 2.4: Single perceptron is shown. x1, x2 and x3 are inputs that enter into node and result output wither 0 or 1.

A single layer perceptron is the first built neural network in history, that classifies data into two classes. It takes several binary inputs and produces a single binary output. Using each input value, a weighted sum is calculated by multiplying weights with each input. If the result is below some threshold, the output of perceptron is 0, otherwise 1. Threshold value is a real number, that is given to perceptron as a parameter. Figure 2.4 shows an example perceptron. All neural networks base on perceptrons, but in a modified and more improved manner.

In the next stage, there comes Multi-Layer Perceptrons (MLP) so called feed forward neural networks. There are different types of neural networks and feed forward neural networks are one of the most widely used type among neural networks. MLP contains one or more hidden layers besides input and output layers. Each node in every layer are fully connected to all nodes in next layer. Amount of neurons in input layer is determined by attribute amount of dataset. Number of neurons in output layer is the number of total classes that classification will be done. The number of hidden layers determine the deepness of network; multiple hidden layers make neural network deep. An example MLP with one hidden layer is shown below Figure 2.5.

In hidden layer, some parameters like weights and biases are initialized with input using activation functions. There are different activation functions like threshold function, sigmoid function, hyperbolic tangent function, Rectified Linear Activation Function (RELU) and maxout function. According to the activation function calculation, an output for each node is produced.

Neural network learns in such a way that it computes error rate during training to

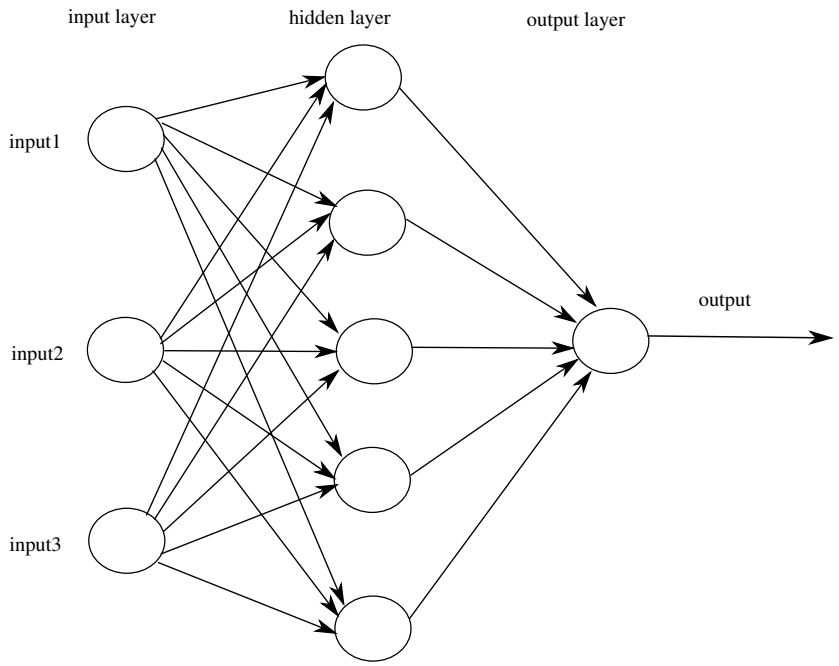

Figure 2.5: Multi-layer perceptron is shown. Input layer takes words as each node and sends them to hidden layer bu equally distribution. After calculations in hidden layer, output value is created.

compare the predicted value with real value. This error rate is calculated using a cost function. Examples of widely used cost functions are mean squared error function, cross entropy function and negative log likelihood function. According to results, neural network changes weights, biases and other parameters to make more accurate predictions and reduce error rate. To achieve this, a back-propagation algorithm is used, that finds best parameters after executing some number of epochs that means iteration number over training data. Back-propagation is especially a feedback mechanism, according to error rates, it offers better weights and biases to decrease error rate on the next iteration. Some widely used back-propagation algorithms are stochastic gradient descent, mini-batch gradient descent and full-batch gradient descent. In the very first iteration, the network knows nothing and gives very bad predictions. From that iteration, an error rate is calculated and those error values are propagated backward to modify weights. Hence on the next iterations, network gives more accurate results.

The result of hidden layer send to the output layer. In output layer, normalization of the results are done to make categorization for predictions. It is a common practice to

use softmax function for normalization. Softmax arranges the data so that the sum of all output will always be equal to 1.

Neural networks are efficient and powerful architectures. However there are some drawbacks. Over-fitting is one of the problems, when neural network memorizes training examples but does not learn to generalize results. So when new data comes, network can not make an accurate predictions. When over-fitting occurs although training error decreases, validation error starts to increase. There are some techniques to prevent from over-fitting such as early stopping, regularization and dropout. Firstly with early-stopping technique, on each iteration of training, validation error is checked. Whenever validation error starts to increase, training is stopped. Another method to combat with over-fitting is regularization. Regularization adds some value to error function. This value is given as parameter and set accordingly how much prevention needed. Finally, Srivastava et. al. offers a method to prevent from over-fitting that is called as dropout regularization [25]. Using dropout, some number of nodes can be omitted during training. Given a parameter value, the neurons that have lower value than the parameter are directly eliminated. This prevents network to be dependent to each single neuron.

An important note about feed forward neural network is, data is always transformed to next layer. There are no loops or transformation to backward layers. There is another type of network, where data can be transformed backwards. Such networks are so called Recurrent Neural Networks (RNN). These networks use sequential information, previous nodes can be feeded with its preceding node. It is possible to say that they have a memory and on each step, they can remember from previous calculations. As a result, the sequence of input data becomes important. RNNs are useful in NLP tasks. Another different type of neural network is Convolutional Neural Networks (CNN), that contains at least one convolutional layer. A convolution is a sliding window over data that takes inputs to activation functions and sub-samples the result. CNNs take advantage of 2-D inputs, hence they are widely used in image processing and speech recognition tasks.

2.4 Mallet

Mallet is open sourced Java library for machine learning. It supports wide variety of algorithms in machine learning for classification, clustering, NLP etc. tasks. Mallet also implements some sequence tagging algorithms like HMM and CRF for named entity extraction. Detailed information about Mallet can be found at [15].

The way Mallet works is about conversion of data. Mallet converts the text input to feature vectors or feature sequences using pipes, which is the structure in Mallet to transform texts into numerical representations. Pipes are flexible structures; many tasks such as string tokenization, stop-word removal and conversion of words to vectors are done with them. Different types of features can be created using wide variety of pipes. Pipes implemented in this study are presented in Proposed Work section.

In this study, Mallet is used for sequence tagging task. CRF algorithm is used with different trainers such as Label Likelihood trainer, Value Gradient trainer and Stochastic Gradient trainer. Label Likelihood trainer calculates label frequency in the training set. Then it calculates probability of word label by checking occurrences of words with that label in training set. Finally it multiplies both results and picks the highest one as mentioned in [2]. Value Gradient trainer works in a way that uses objective functions such as label likelihood, batch label likelihood and entropy regularization. Performance is dependent of the selected objective function. Stochastic Gradient trainer uses Stochastic Gradient Descent as optimization algorithm, that selects training instance randomly and updates parameters on each training example for prediction as stated in [22].

For each trainer, a model is constructed and its efficiency is tested with new data.

2.5 Deeplearning4J

Deeplearning4j is another open sourced library for deep learning tasks. It contains implementations of Word2vec, Doc2vec, deep belief network, CNN and RNN. It relies on Java, besides it also supports Clojure and Scala API. Detailed information about Deeplearning4j is available in [4].

In this thesis work, Word2vec implementation of Deeplearning4j is used in order to apply K-NN classifier for POI extraction.

2.6 Tensorflow

Tensorflow is an open source software library for numerical computation using data flow graphs. Data, formed as multidimensional arrays, are stored in the edges of the graph. These arrays are also called as tensors. Links between edges represent operations. Flow in graph is actually the total operations executed within a session. The aim is to turn machine learning problems into functions on tensors.

Tensorflow is developed by Google to be used in areas such as deep neural networks and machine learning. Tensorflow mainly provides an API with Python, however APIs for some other languages like C++ and Java are being constructed. Tensorflow can run in multiple CPUs and multiple GPUs parallel, which makes it powerful. Detailed information about Tensorflow can be reached from [1].

In this thesis work, Tensorflow is used to create deep neural network for POI extraction task. The accuracy of constructed neural network model is tested against new data.

CHAPTER 3

LITERATURE REVIEW

In the literature, there are several studies on location extraction, and a limited number of research on POI extraction. In this chapter, we summarize the studies that are most related to our work.

Location is more generalized form of a POI. Cities, provinces, streets, other geographical entities and POIs can be categorized as locations. It is possible to infer that every POI is a location but not every location is a POI. We examine literature review in two categories: location extraction and POI extraction.

3.1 Location Extraction

In literature, there are some studies about named entity recognition like the work of Gimpel et. al. [5]. In that work, a POS tagset for Twitter is constructed along with some features. CRF model is developed for tagging and word features such as suffix length and capitalization along with domain specific features were used. In evaluations, they compared their system with StanforNER and showed that their system had better accuracy when all features added.

As location is one of NER kinds, location extraction is also a NER task. One of the most recent papers that focus on location extraction is the work of Sagcan and Karagoz [23]. In that study, the aim is to recognize the location of events from Twitter, that are included in unstructured informal texts. What they extract are locations, not only POIs but also toponyms such as districts and streets. An important feature

of this work is; gazetteer data is not used for training the CRF model. Whereas, two different sets of Twitter data retrieved from previous studies were used. Then, morphological analysis and morphological disambiguation is applied on the words in the dataset. Moreover, a normalization step is done to eliminate multiple characters and English-Turkish character incompatibility. Proposed work constructs a hybrid system, where both rule based and machine learning based approaches were used. As rule based approach, regular expressions were used to detect patterns of locations and normalization of data. As machine learning approach, they used CRF to learn a model from unstructured text of Twitter. In CRF, some features such as POS tags and conjunction windows were used. They used Mallet for sequence tagging tool and compared the results of different features of CRFs using different training methods. Moreover in the experiments, they analyzed different training methods effects on prediction accuracy. According to that experiment, Label Likelihood trainer gets best result with 62% in terms of f-measure metric. Also they showed effects of different sized training datasets and toponym amounts they contain. Best result is 62% in terms of f-measure, from the dataset that has more toponyms in training set and less toponyms in test set. Besides, they compared their work with two studies in the literature and showed that their proposed work has better results compared to other studies.

Malmasi and Dras worked on location extraction from microblogs and social media [14]. Their approach is different than the other works described as they make unsupervised learning using syntactic parse trees to detect locations from text. Data retrieved from social media is noisy as it may contain unnecessary words, misspellings or grammatically wrong sentences. Moreover, it is hard to prepare annotated training data for a large set. Because of those reasons, they prefer unsupervised learning approach using parse trees. Also their work is based on gazetteer data; they retrieve location names from GeoNames, which is a database containing locations. In their study first of all, they process data by eliminating non-English tweets, removing @ and # symbols and URLs. Then, they apply syntactic parsing using Stanford CoreNLP tool and generate POS tags. After parsing, they extract noun phrases as location names are mostly described as nouns. Extracted noun phrases may contain multiple nouns within. Thus retrieved noun phrases need to be processed further. So

they recursively divide the parse trees to eliminate prepositions etc. Finally, n-gram based location matching methodology is applied as current location nouns may still contain more than one location. In this work, they extract different types of entities such as addresses, POIs, locations, distances and directions. For address matching, regular expressions are generated to capture addresses from text. Regular expressions also used for POI matching to catch candidate POI phrases. For location extraction, they used GeoNames database and retrieved results show whether the data is location or not. Finally they detect directions and distances about a location and these data are also counted as locations in that study. Proposed system detects location from not only texts but also hashtags and mentions. To achieve this, word segmentation method and word frequency applied to detect boundaries of a word. For evaluation, they used f-measure metric and get the result of 0.79 on the test set. Moreover adding hashtag segmentation improved their results by 0.05.

Inkpen et. al. worked on location extraction from Twitter by detecting and disambiguating locations in tweets [7]. Moreover, results of detected toponyms classified into cities, states and countries. In order to detect locations, they used CRF with sequential tagging. Proposed system consists of two main stages. Firstly, CRF with different features used for location detection. Secondly, more than one place may contain same name and this problem is resolved by making classifications on different levels as location disambiguation. Features to detect locations are bag of words, part of speech, left/right that takes the adjacent words of target word into account and gazetteer, that checks whether target word appears in gazetteer or not. After those steps, experiments are conducted using precision, recall and f-measure metrics for evaluation. In those experiments, different combinations of features taken together and results are observed. In city classification, using only bag of words features have significant effect on results. In province classification, it is observed that no pairs of features have important effect. Finally in country classification, using all features together or using bag of words, gazetteer and left/right feature combinations improve results.

There are also some experimental studies for Location extraction in the literature such as the work of Lingad et. al. [13]. That study focuses on Location extraction from microblogs such as texts on Twitter and news posted during disasters. In this work,

some Named Entity Recognition tools were compared with different models. These tools along with applied models are StanfordNER with CRF model, OpenNLP with Maximum Entropy model, TwitterNLP with LabeledLDA model and Yahoo Placemaker. On that study two different cases were tested; on the one hand they removed hashtags and on the other hand they used hashtags without # character. According to the results, the best model when hashtags removed is Stanford NER 4-class model with f-measure of 0.69. Moreover when set is retrained, f-measure becomes 0.87. When hashtags used without # character, StanforNER again gives the best result with 0.57 f-measue, which is lower than the removed hashtags version. However when model is retrained, f-measure becomes 0.90.

3.2 POI Extraction

Described works so far focus on location extraction, which may contain geographical entities, streets, cities and POIs. It is important to note that not all of locations are POIs; but all POIs are considered as locations. In the literature, there are several studies that specifically focus on POI extraction like the work of Li and Sun [12]. In that work, the aim is to extract POI location from Twitter and predict the temporal information about the tweet. What is meant by temporal information is that whether the user has visited, is currently at or will plan to visit that POI in the future. They have a different approach to POI extraction problem by focusing on temporal awareness. In that work, proposed solution consists of two main components: a POI inventory that contains Foursquare check-in tweets and a time aware POI tagger that predicts locations and their information about timeline. POI inventory contains POI candidate names taken from Foursquare in informal format as those information retrieved directly from users. In POI tagger part, they used CRF for tagging and CRF++ as CRF tool. Lexical, grammatical, geographical and Beginning Inside Last Outside Unit (BILOU) features were used as CRF features. Lexical features generated from the surface of the words and surrounding words of a word. As grammatical features, POS tag, Brown clustering along with the time trending scores were used. They manually created a dictionary for time words in English containing scores. A single sentence may contain different temporal information; in those cases they selected

closest one. As geographical feature, they calculated distribution of tweets using their coordinates and checked for spatial randomness by distribution on a map. As the last feature, they used BILOU Schema to detect whether word is at the beginning, inside, last word or outside in a multiple word POI. In experiments, they used different features and evaluated the results. Lexical features along with grammatical features get the best result without considering BILOU features. When they add BILOU features, results get even better in terms of f-measure. They also compared their system with 3 different methods; random annotation, k-nearest neighbor and StanfordNer. In results of those experiments, they showed that their system scored better performance than the other methods on precision, recall and f-measure values.

Rae et. al. used gazetteer based approach for POI extraction problem [21]. The aim of this work is to automatically identify POI mentions in text and localization of those locations. They used different social media platforms such as Wikipedia, Foursquare and Gowalla data for training purposes and compared the results of accuracy rates of finding POIs using different training methods. Their approach is different by bootstrapping training data as they increased their data amount by using the data as seed queries for search engines. Besides extraction of location names, they used coordinates of tweets for POI localization from Flickr. In order to ensure localization, retrieved names must match with coordinates. To that aim, they used the approach in [24]. Coordinate system consists of one kilometer cell grid and each cell is associated with geo-tagged Flickr images taken inside those cell coordinates. As a result, cell is represented by images with tags. Using the retrieved data, they applied CRF for sequential tagging and used CRFSuite project tool for implementation. For features of CRF they used geographical, grammatical, lexical and state transition features. Lexical features are calculated from the surface text of token. As grammatical feature, POS tagging is applied. As geographical feature, they feed inputs of location names to Yahoo Placemaker service and get results of candidate POI names. Finally in state transition feature, they consider previous and next state of all features generated. In experiments part, they showed that systems trained with manual annotated or Wikipedia data get better results from ones trained with Placemaker. Moreover, they compared the results of localization and concluded that they get better results using localization with one kilometer grid cells.

Mummidi and Krumm worked POI extraction problem with a different approach, using Volunteered Graphic Information (VGI) [18]. Users of Windows Live Maps can comment on some locations through pushpins over the map and data used on that work retrieved from those pushpins. Each pushpin contains a title with location name, a data part where users may comment on the locations and coordinate information. After retrieving pushpin data, candidate POI names were extracted from text parts of data using n-grams; which is a phrase that contains n words. To that aim, monograms, bigrams and trigrams are used. Maximum size of an n-gram is three and bigger sized n-grams were not used because of performance reasons. After creating candidate POI names, clustering is done over n-grams in order to get groups of close pushpins. For this, a dendrogram with hierarchical agglomerative clustering technique is constructed. Clustering huge amount data using dendrogram results in bad performance, thus they divided regions into subregions and used a dendrogram for each subregion. Clustering results with groups of n-grams. Over each distinct n-gram - cluster pair, they applied Term Frequency - Inverse Document Frequency (TF-IDF) metric to retrieve useful data. Moreover they set a lower bound for number of pushpins for each cluster in order to eliminate anomalies. Finally they set a parameter for dendrogram called term purity, to identify when the cluster is divided into another subcluster. Using those techniques, candidate POI names are retrieved. They applied experiments to measure the correctness retrieved data and categorized POIs as existing POI, that is found in Yellow Pages database and found POI, that does not exist in Yellow Pages database and found by their system. According to the results, existing POI has accuracy of 92.2%, while found POI has accuracy of 76.8%.

Zhang et. al. worked on automatically POI extraction from internet news in China [27]. In order to extract POI, they used lexical analysis of words. The system composed of 4 main stages; text processing, recognition of full entity name, POI extraction modeling and result optimization. In text processing part, they erase noisy data and apply lexical analysis using ICTCLAS2010 tool, which is a popular Chinese lexical analyzer. As a result of this analysis, time, organization and location names are identified. Those results may give each word a tag, but cannot identify multiple word POI name as a whole. To handle this, rule based method used to detect full name of a POI and it constructs candidate POI names. Using those candidate names, POI

extraction model is constructed to calculate coherence by measuring term frequency and distance of word to location, organization and person. To optimize results, temporal inference applied and old POI events get lower priority. Moreover, any data sent out from China eliminated. Experiments are performed using different search engines and precision - recall metrics used for comparison. It is observed that proposed optimization metrics improved both precision and recall. Location optimization gives the best performance with 93.4% precision and 73.5% recall.

CHAPTER 4

METHODOLOGY

In this section, the POI extraction methodology of this thesis work is presented. We have devised and applied various machine learning techniques to POI extraction problem.

The steps of the methodology and the techniques used can be summarized as follows. Firstly, Twitter data containing Foursquare check-ins are used. From that data, redundant parts are removed. In order to work on that data for POI extraction with CRF and neural network, it needs to be in suitable format. Finally the data that consist of sentences are tokenized into words after preparing different features.

After organizing data in needed format, POI extraction step has been applied. Firstly, CRF is used with different features. These features consist of sentiments in tweets, population and poi densities around the location where the tweet is posted. The effects of those features are observed. Besides these features, Mallet's built-in features also used and its effects over previously constructed features are observed. Secondly, neural word embeddings are created from the data using Word2vec and K-NN classification is applied over embeddings. Results are compared whether they are POI or not a POI. Finally, a deep neural network is constructed that extracts POIs from tweets.

4.1 Data Retrieval & Preprocessing

Twitter is a free formatted text sharing platform limited to 140 characters. In order to extract POIs from tweets, they need to have suitable format to contain candidate POIs. Foursquare check-in tweets fulfill that need as they mostly contain location information some of which are POIs. In Twitter, Foursquare check-in tweets are represented in two different ways as displayed in Table 4.1. The first type of tweets start with "I am at" and concatenated with location name. As a result, no clue or sentiment information can be extracted from tweets of that type, thus, they are not useful for this work. However for the second type, user comments along with the location names can be extracted. As proposed system uses sentiment information as a feature in the CRF part, it is convenient to retrieve Foursquare check-in tweets of second type. In the rest of proposed work, that type of data is retrieved and processed.

Table 4.1: Foursquare check-in tweet types

Type 1	I am at Oskar Pastanesi in Ankara (I am at Oskar Patisserie in Ankara)
Type 2	Tiyatro candır (@ Akun Sahnesi in Ankara, Turkiye) (I love theatre (@ Akun Scene in Ankara, Turkiye))

Moreover for POI extraction with CRF, in order to prepare POI density and population density features, coordinates of tweets are required. Hence, all retrieved tweets must contain coordinates and the ones that do not have coordinate information are eliminated.

An important note about the data is that, it contains only unique tweets and re-tweets are filtered out since re-tweet of location tweet does not mean that location is visited by the re-tweeter.

Retrieved data contains unstructured and unordered words, so they need to be further processed for POI extraction problem. As the first process, retrieved data is formed to contain a tweet in a single line. This is necessary as some retrieved tweets spread to multiple lines which makes data disordered for the further parts of this work.

Secondly, parts of tweets that are not related with POIs removed, such as mentions that are used for tagging people and http links in tweets. All mentions in tweets

have the same format: "w/ @mentionedperson". Using that format, people may post information about whom they visited the POI with. There may be also some http links of POI in a tweet. As those parts provide no information about a POI, they are eliminated.

Following examples in Table 4.2 show how mentions in a tweet are removed and Table 4.3 shows how links are removed and display final forms of tweets after elimination.

Table 4.2: Elimination of mentions in a tweet

Before Elimination	Kınamız var (@ Hekimevi in Ankara w/ @tutkuguner06) (We have redemption (@ Hekimevi in Ankara w/ @tutkuguner06))
After Elimination	Kınamız var (@ Hekimevi in Ankara) (We have redemption (@ Hekimevi in Ankara))

Table 4.3: Elimination of links in a tweet

Before Elimination	Bu bir aşk...Mekan https://t.co/vjvQkKSGdl olsun yeter. (@ Başkent Üniversitesi Spor Salonu in Ankara) (This is love...Place https://t.co/vjvQkKSGdl should be. (@ Başkent Üniversitesi Spor Salonu in Ankara))
After Elimination	Bu bir aşk...Mekan olsun yeter. (@ Başkent Üniversitesi Spor Salonu in Ankara) (This is love...Place should be. (@ Başkent Üniversitesi Spor Salonu in Ankara))

Then, unnecessary single punctuation characters are removed. As Twitter users are allowed to post texts without any format, punctuation characters may become unnecessary. The aim is to remove such characters that possibly affect the meaning for labeling. However if there are multiple characters, they are not removed since most of the time, multiple characters come together to form emojis. Emojis give important clues about the sentiment of the sentence which makes it important for sentiment analysis part. Hence, with this process, while single punctuations appended to words are removed; emojis appended to words are separated and are not removed.

Finally, on each sentence, some words may be separated with more than one blank single characters. Such lines are formed to have a single empty character between each word. On CRF application, data are retrieved word by word using this format.

After all of the features are added to data, they need to be tokenized before applying CRF. As in the remaining parts of this work, such as CRF with Mallet, Word2vec with Deeplearning4J and neural network with Tensorflow, data is required to be in the format that contain one word and features of this word on a single line. Therefore, data is prepared to have suitable format for the rest of the work. A sample tokenized tweet is shown in Table 4.4.

Table 4.4: An example tweet with tokenized form

Original Tweet	Bi orada bi burada Kahveci Hacıbaba in Çankaya Ankara (Here and there Kahveci Hacıbaba in Çankaya Ankara)
Tokenized Tweet	Bi orada bi burada Kahveci Hacıbaba in Çankaya Ankara

For labeling the data, Beginning Inside Outside (BIO) notation is used. BIO notation is one of a segment representations used in NER tasks as mentioned in [9]. In this notation, B stands for beginning word of the multiple words, I means that the word is inside the group of words and O stands for the word is not in any category. In this thesis work, the words that do not contain POI information are marked as "NOPOI". If only a single word is POI, it is marked as "FPOI". If multiple words together form a POI, the first word is labeled as "FPOI" and all remaining words are labeled as "IPOI".

An example for labeled tweet using BIO notation is presented in Table 4.5.

In CRF application, constructed features added to the format presented above. However in Word2vec and neural network applications, no features are added and the training and test sets have same format with the above example.

Table 4.5: An example labeled tweet using BIO notation

Original Tweet	Bi orada bi burada Kahveci Hacıbaba in Çankaya Ankara (Here and there Kahveci Hacıbaba in Çankaya Ankara)
Labeled Tweet	Bi NOPOI orada NOPOI bi NOPOI burada NOPOI Kahveci FPOI Hacıbaba IPOI in NOPOI Çankaya NOPOI Ankara NOPOI

4.2 Feature Construction for CRF

In this work, CRF is used for sequential tagging and the effect of various features on the performance of CRF have been investigated. The features studied in this thesis work are presented in Table 4.6.

Table 4.6: Explanations of features constructed in this thesis work

Sentiment Feature	This feature is constructed from sentiment analysis results of user comments in a tweet
Population Density Feature	This feature is the population around the point where the tweet is posted
Nearby POI amount Feature	This feature is the number of POIs around the point where the tweet is posted

Besides these, Mallet also provides some features that can be added for training CRF. In this work, different features are combined and applied together for POI extraction. Explanation of each feature is displayed in Table 4.7.

4.2.1 Sentiment Feature

The first feature prepared for CRF is sentiment feature. Data retrieved from Twitter has a form that contains user comments; thus collected tweets possibly contain sentiment. To make sentiment analysis on that data, sentiment scores are calculated using

Table 4.7: Feature Set Definitions

Regular Expressions	This feature tags words by making regular expression match.
Token Char Suffix	Specified number of suffix characters from each word extracted and added as a feature.
Offset Conjunction	These are two dimensional arrays that takes specified number of words before and after of each word and adds each word separately as a feature.
N-gram	N-grams created by specified lengths.
Feature Window	This feature constructs a window that adds all features of each token in a window, slides over all tokens and adds results as a feature.

the sentiment dictionary that contains Turkish words and sentiment scores, which is given in [26].

Turkish is a morphologically complex and agglutinative language such that the words are formed by adding suffixes on a root word. To derive new words rather than prefixes, suffixes are used generally. The sentiment score dictionary supports this format; there are roots and derived words from those roots placed separately. If a word is root, it has '*' character at the end of it. As a result, it is possible to distinguish from roots to derived words while reading those values. In order to determine the sentiment values of the words in a given tweet, the following steps are applied:

- If there is a direct match with the word of tweet, directly take the score of it.
- If there is no direct match, look for roots which may be subset of the word of tweet. There may be more than one root that can match with the word. In such a case, compare all roots and take the one that has longest length. This helps to catch the most relevant root to word.
- If there is still no match after those comparisons, give 0 as the sentiment value for the word.

Besides those comparisons, there is also another important thing to consider while making sentiment analysis, that is, some POI names may contain sentiments. An example for such case is shown below.

Bakım zamanı (@ Galea Güzellik Salonu in Ankara)
(Beauty time (@ Galea Beauty Saloon in Ankara))

Although this tweet does not contain any sentiments and can be categorized as neutral, the Güzellik (Beauty) word in POI name gives a positive sentiment. Because of this, the words that are labeled as POI are excluded while calculating sentiment score; only tweets are included.

Twitter users may post not only texts but also emojis in their tweets. Emojis are important for our work as they are one obvious way to express opinion. In order to calculate scores, emoji score dictionary prepared in [19] is used and scores presented there are normalized for our work.

Finally, adding scores for each word and emojis, a total score is calculated for each tweet. Using that score, sentiment categorization done with three groups; positive, negative and neutral. The results of total analysis for each sentence is labeled next to the word so that Mallet can use it as a feature on CRF. Example of sentiment labeling is presented in Table 4.8.

Table 4.8: An example sentiment feature added tweet

Original Tweet	beytepeden kopamamak ☺ hacettepe üniversitesi in ankara türkiye
Tweet with Sentiment Feature	beytepeden POSITIVE NOPOI kopamamak POSITIVE NOPOI ☺ POSITIVE NOPOI hacettepe POSITIVE FPOI üniversitesi POSITIVE IPOI in POSITIVE NOPOI ankara POSITIVE NOPOI türkiye POSITIVE NOPOI

Above tweet is calculated as a positive sentiment group and positive label is added next to each word of this tweet.

4.2.2 POI Density Feature

The second feature constructed in this work is the number of POIs nearby of the tweet's location. A POI may contain some number of different POIs nearby; this feature focuses on that number. To achieve this, Google Maps API is used. Data retrieved from Twitter contain coordinates where the tweet was posted. Using those coordinates as main point, POI density is calculated and added as a feature.

Google Maps requires users to select the POI types they want to get and retrieve nearby POIs of selected types according to those selections. In our work, we select types given in Table 4.9 and fetch the POIs within 500m radius of the main location as 500m is within the boundaries of walking distance [1].

Table 4.9: Retrieved nearby POI types from Google Places API

airport	amusement park	aquarium	art gallery	bakery
bank	beauty salon	bar	cafe	casino
city hall	food	gym	hospital	library
lodging	mosque	theater	museum	night club
park	restaurant	school	shopping mall	stadium
train station	university	zoo		

Google Maps API returns data in JSON format and extracted data should be parsed to get the relevant information. Using radar search function, API retrieves nearby POI count number up to 200 [2]. For the places that have more than 200 nearby POIs around, the result is always 200. Results are categorized into three groups. To make the categorization, standard deviation is applied to all results. According to the results, the ones that have less than 65 POIs are labeled as "Low". If nearby POI amount is between 65 and 195, that POI is labeled as "Medium" and if it has more than 195 POIs, specified POI is labeled as "High". Those features are added next to each word so that Mallet can use them as a feature in CRF.

After categorizing each tweet's nearby POI density, results are added next to each

[1] http://humantransit.org/2011/04/basics-walking-distance-to-transit.html
[2] Detailed information at: https://developers.google.com/places/webservice/search#RadarSearchRequests

word as a feature. Example tweet that contains this feature is presented in Table 4.10.

Table 4.10: An example number of POIs nearby feature added tweet

Original Tweet	alışverişş ankamall in yenimahalle ankara (shopping ankamall in yenimahalle ankara)
Tweet with number of Nearby POI Feature	alışverişş MEDIUM NOPOI ankamall MEDIUM FPOI in MEDIUM NOPOI yenimahalle MEDIUM NOPOI ankara MEDIUM NOPOI

Example tweet in Table 4.10 is sent from a location that the nearby POI amount is categorized in the second group and labeled as medium.

4.2.3 Population Density Feature

Another feature developed and used in this work is population density around a location. This feature focuses on the district of a location and aims to show effects of human density of a district over POI extraction.

There are two main steps to build this feature. Firstly, in order to retrieve population of a neighborhood, the name of neighborhood and district pair should be retrieved. Retrieved tweets from Twitter4J do not contain district/neighborhood names; there are only coordinates of tweets. Using those coordinates as a feed to Google Maps API's geocoding function, output is the address information in JSON format.

The returned address result contains high amount of information about the address; most of which are redundant for our work. Parsing this detailed address information, district and neighborhood names can be retrieved. From district/neighborhood information, it is possible to get population of those neighborhoods from TUIK database [3].

As the second part of creating population feature, the amount of people who live around a location is retrieved from TUIK. TUIK is the largest official database of Turkish government that keeps statistics of different areas. TUIK also keeps how

[3] TUIK database can be reached at: http://www.tuik.gov.tr/

many people lives in a neighborhood for each district of each city in Turkey. Population statistics of Ankara in 2016 with all districts are queried, which is the latest at that time. For each tweet, the number of people who lives in the neighborhood that the tweet posted is retrieved.

This approach has a positive side effect for our work. Although tweets from Twitter4J queried to sent from Ankara, there are some outlier tweets that are sent from the neighbor cities of Ankara. As the retrieved address from Google Maps API using the coordinates does not match any address in TUIK database, those tweets are eliminated. As a result, retrieved data is guaranteed to be sent from Ankara and contain valid district and neighborhood names. As an example, the following tweet sent from outside of Ankara is retrieved as an outlier data from Twitter4J, which is eliminated.

Bekleriz... (@ Bayramefendi Osmanlı Kahvecisi - @osmanlikahve in Kırıkkale)
(We are waiting... (@ Bayramefendi Osmanlı Kahvecisi - @osmanlikahve in Kırıkkale))

To categorize the results, a threshold is applied for each group. To this aim, two different methods are applied and those methods are compared in the experiments part. Firstly, the mean value of all neighborhoods is calculated for each district. The density values, which are lower than half of the mean are tagged as "1", values between half and one half are tagged as "2" and values more than one half of mean are tagged as "3". Secondly, standard deviation applied for each neighborhood and its all districts. Labeling is the same as the mean calculation. These labeling values represent population densities and as density increases, the category value also increase. These tags are provided to Mallet as features for CRF.

After categorizing each tweet, results of tagging are appended to each word. This is the same format that we applied on previous features. Example labeled tweet with population feature is presented in Table 4.11.

Table 4.11: An example population around a location feature added tweet

Original Tweet	biyokimya konferansi odtü kültür ve kongre merkezi in ankara (biochemistry conference odtü kültür ve kongre merkezi in ankara)
Tweet with number of Nearby POI Feature	biyokimya 3 NOPOI konferansi 3 NOPOI odtü 3 FPOI kültür 3 IPOI ve 3 IPOI kongre 3 IPOI merkezi 3 IPOI in 3 NOPOI ankara 3 NOPOI

Example tweet in Table 4.11 is sent from a location where nearby population is categorized in the highest group and labeled as "3". For each word of the tweet, the result is appended as a feature.

4.2.4 Mallet Features

Besides the features we provide, Mallet offers various other features that can be used for CRF. In this work, some of these features are used in order to investigate their effect on POI extraction performance. All of the features provided by Mallet added through pipes to be used in this thesis.

Regular expression feature is one of the features that Mallet provides. It tags words by making regular expression match. In this work, some patterns are predefined and make Mallet to tag words matching these patterns as "LOCATION". Important to note that this tag does not mean that entity is a POI or location; but it is a candidate POI. In this work, we tag words containing otel, park, bar etc. as location. Examples for these patterns are shown below.

pipes.add(new RegexMatches(" LOCATION", Pattern.compile(".[Oo]tel.*")));*
pipes.add(new RegexMatches(" LOCATION", Pattern.compile(".[Pp]ark.*")));*
pipes.add(new RegexMatches(" LOCATION", Pattern.compile(".[Bb]ar.*")));*

Example tweet after adding this pattern feature is shown in Table 4.12.

Table 4.12: An example regular expression feature added tweet

Original Tweet	keyif kahveli cafe in etlik ankara (pleasure kahveli cafe in etlik ankara)
Tweet with Regular Expression Feature	keyif NOPOI kahveli FPOI cafe LOCATION IPOI in NOPOI etlik NOPOI ankara NOPOI (pleasure NOPOI kahveli FPOI cafe LOCATION IPOI in NOPOI etlik NOPOI ankara NOPOI)

Mallet provides char prefix and suffix of words as a feature such that it extracts specified number of characters from word and adds those words as a feature. As Turkish tweets are used in this work, suffix feature is convenient due to the Turkish language structure that requires adding suffixes to words to derive new words rather than prefixes. Three suffix characters from each word extracted and added as a feature. The example of tagging tweet with suffix feature is shown in Table 4.13.

Table 4.13: An example suffix feature added tweet

Original Tweet	keyif kahveli cafe in etlik ankara (pleasure kahveli cafe in etlik ankara)
Tweet with Suffix Feature	keyif C1=f C2=if C3=yif NOPOI kahveli C1=i C2=li C3=eli FPOI cafe C1=e C2=fe C3=afe IPOI in C1=n NOPOI etlik C1=k C2=ik C3=lik NOPOI ankara C3=ara C2=ra C1=a NOPOI

Offset conjunction window is another feature used in this study. Offset conjunctions

are two dimensional arrays that takes specified number of words before and after of each word. Then it adds each word separately as a feature. Example tweet in Table 4.14 contains four conjunctions for each word. These are the next word, the previous word, the next two words and the previous two words. If there are no word exist in specified position, some default tags like <START0> or <END0> are added by Mallet.

Table 4.14: An example offset conjunction feature added tweet

Original Tweet	keyif kahveli cafe in etlik ankara (pleasure kahveli cafe in etlik ankara)
Tweet with Offset Conjunction Feature	keyif <START1>@-2_&_<START0>@-1 <START0>@-1 kahveli@1_&_cafe@2 kahveli@1 NOPOI kahveli <START0>@-2_&_keyif@-1 keyif@-1 cafe@1 cafe@1_&_in@2 FPOI cafe in@1_&_etlik@2 in@1 keyif@-2_&_kahveli@-1 kahveli@-1 IPOI in kahveli@-2_&_cafe@-1 cafe@-1 etlik@1_&_ankara@2 etlik@1 NOPOI etlik ankara@1 ankara@1_&_<END0>@2 in@-1 cafe@-2_&_in@-1 NOPOI ankara <END0>@1_&_<END1>@2 <END0>@1 in@-2_&_etlik@-1 etlik@-1 NOPOI

Mallet also provide n-gram feature that creates n-grams having specified lengths for each token. In this work, this feature is used to take two to seven other characters of each word and add each n-gram created as a feature. Next example in Table 4.15 is a word having this feature.

Table 4.15: An example n-gram feature added tweet

Original Tweet	keyif kahveli cafe in etlik ankara (pleasure kahveli cafe in etlik ankara)
Tweet with n-gram Feature	keyif NGRAM_ke NGRAM_ey NGRAM_yi NGRAM_if NGRAM_key NGRAM_eyi NGRAM_yif NGRAM_keyi NGRAM_eyif NGRAM_keyif NOPOI kahveli NGRAM_ka NGRAM_ah NGRAM_hv NGRAM_ve NGRAM_el NGRAM_li NGRAM_kah NGRAM_ahv NGRAM_hve NGRAM_vel NGRAM_eli NGRAM_kahv NGRAM_ahve NGRAM_hveli NGRAM_kahve NGRAM_ahvel NGRAM_hveli NGRAM_kahvel NGRAM_ahveli NGRAM_kahveli FPOI cafe NGRAM_ca NGRAM_af NGRAM_fe NGRAM_caf NGRAM_afe NGRAM_cafe IPOI in NGRAM_in NOPOI etlik NGRAM_et NGRAM_tl NGRAM_li NGRAM_ik NGRAM_etl NGRAM_tli NGRAM_lik NGRAM_etli NGRAM_tlik NGRAM_etlik NOPOI ankara NGRAM_an NGRAM_nk NGRAM_ka NGRAM_ar NGRAM_ra NGRAM_ank NGRAM_nka NGRAM_kar NGRAM_ara NGRAM_anka NGRAM_nkar NGRAM_kara NGRAM_ankar NGRAM_nkara NGRAM_ankara NOPOI

As the last feature of Mallet in this work, feature window is used. This is basically a window that adds all features of each token in a window, slides over all tokens and adds results as a feature. In this work, windows have size of 2x2; which takes two features before and two features after in a window. This feature is similar to offset conjunctions window in a way that takes the sequence of words of specific word.

However this feature does not group words and does not put default label if word at specified position does not exist. Example tweet tagged with window feature is demonstrated in Table 4.16.

Table 4.16: An example window feature added tweet

Original Tweet	keyif kahveli cafe in etlik ankara (pleasure kahveli cafe in etlik ankara)
Tweet with Window Feature	keyif cafe/+2 kahveli/+1 NOPOI kahveli cafe/+1 keyif/-1 in/+2 FPOI cafe etlik/+2 keyif/-2 in/+1 kahveli/-1 IPOI in ankara/+2 kahveli/-2 cafe/-1 etlik/+1 NOPOI etlik ankara/+1 in/-1 cafe/-2 NOPOI ankara in/-2 etlik/-1 NOPOI

4.3 POI Extraction with CRF

CRF is a probabilistic model for labeling a sequence of words. It has been used in several NLP applications, such as NER. It is used for encoding known relationships between observations (features) and construct consistent interpretations. Once the features are prepared for CRF, they are added into pipes, which are structures in Mallet for adding features to tokens. In order to traverse input data, InstanceList is created and pipes are given as input. Then an iterator is defined and added as parameter to InstanceList.

```
pipe = new SerialPipes(pipes);
InstanceList instances = new InstanceList(pipe);
instances.addThruPipe(new LineGroupIterator(in, Pattern.compile(pattern), true));
```

After this step with presented code above, data is ready for training. In this study, three different training methods of Mallet are compared. These are Value Gradient Trainer, Label Likelihood Trainer and Stochastic Gradient Trainer.

```
CRF crf = new CRF(trainingData.getDataAlphabet(), testingData.getTargetAlphabet());
crf.addFullyConnectedStatesForLabels(); crf.addStartState();
```

Before training with all the models, a CRF object is created taking as training and testing data alphabets. For labels, fully connected states are added. As CRF creates a finite state machine, a starting state is added. All of the training methods described below use same finite state machine.

Firstly, Label Likelihood Trainer is added with the code below.

CRFTrainerByLabelLikelihood trainer = new CRFTrainerByLabelLikelihood(crf);

Other training method; Stochastic Gradient Trainer is added below.

CRFTrainerByStochasticGradient crft = new CRFTrainerByStochasticGradient(crf, trainingData);

Last training method of Value Gradient is presented below.

CRFTrainerByValueGradients trainer = new CRFTrainerByValueGradients(crf);

After training CRF model with all of those methods separately, its performance is measured on test set. With CRF, we constructed models with and without features of sentiment values, POI density and population density and investigated the effect of these features for POI detection.

4.4 POI Extraction by Word Embeddings with K-Nearest Neighbor Classifier

In addition to CRF, for POI extraction, we have devised and elaborated on some other approaches as well. Word embeddings are one of these approaches. Word embedding is a technique in NLP, where words of texts are mapped into vectors of numbers. Words are represented in n-dimensional space as embeddings.

In this work, word embeddings are used along with K-Nearest Neighbor (K-NN) classifier. Firstly, word embeddings are created using Word2vec algorithm of Google [4]. Word2vec is a model to produce word embeddings, basically converts words into vectors. In our work, Word2vec is used to convert the tokens/words of tweets into vectors. Details on Word2vec are presented in Chapter 2. From constructed vectors,

[4] More details at: https://code.google.com/archive/p/word2vec/

K-NN classifier is applied using cosine similarity to retrieve nearest words. K-NN algorithm works in a way that finds closest points in vectoral space by calculating euclidean distances. The number of closest points is represented by K value, which is provided as a parameter. After applying K-NN to words in tweet, three nearest words are retrieved and they are checked whether they are labeled as POI or not. According to the result, a guess is done on main word.

In order to create embeddings, processed tweets without features were used. DeepLearning4J has a tokenizer for tokenizing the sentences into words and a sentence iterator for iteration over sentences as shown below.

```
SentenceIterator iter = new BasicLineIterator(new File(dataPath));
TokenizerFactory t = new DefaultTokenizerFactory();
t.setTokenPreProcessor(new CommonPreprocessor());
```

Each word is given as input to Word2vec algorithm by iterating over the input using the above code. As a result of Word2vec, an embedding model is created. This process is showed below:

```
Word2Vec vec = new Word2Vec.Builder()
    .minLearningRate(rate).minWordFrequency(freq)
    .iterations(iter).layerSize(size) .seed(seed)
    .windowSize(window).iterate(iter).tokenizerFactory(t).build();
```

There are some parameters for Word2vec that are set before construction of model. These parameters are explained below.

- Minimum learning rate parameter is the floor value of the learning rate, which is the step size of each update of coefficients of model. This value should be small enough to keep network effective.
- The minimum number of times that a word appears in text is set by minWordFrequency parameter. If the number is below the parameter, than the word is not learned. In our work, that parameter is set to 1 as our aim is to create embeddings for every word in text.
- Number of times network updates its coefficients for one batch is set by iterations

parameter.

- Layer size is the parameter for number of dimensions in vector space.
- Window size is simply the size of sliding window over words.
- As sentences directly fed into Word2vec and neural word embeddings constructed over words, TokenizerFactory is used for tokenization sentences into words.

After setting up Word2vec model and feeding it words, vector representations of words are constructed. Using these vectors, closest different number of words for each word retrieved using cosine similarities. If at least specified number of the retrieved words ever tagged as "FPOI" or "IPOI", then that word is set accordingly. Otherwise the word is tagged as "NOPOI". Results of this work is discussed in experiments part.

4.5 POI Extraction with Deep Neural Network

As another method, deep neural network is investigated as a solution for POI extraction problem. The neural network takes pre-trained word embeddings as input. Training and testing files have the same format as the files used in CRF. An example is displayed in Table 4.17.

Table 4.17: An example format of tweet used as input to Neural Network

Original Tweet	Kardeşim evleniyor Altınköy Açık Hava Müzesi in Ankara (My sibling is getting married Altınköy Açık Hava Müzesi in ankara)
Tweet format used as input for Neural Network	Kardeşim NOPOI evleniyor NOPOI Altınköy FPOI Açık IPOI Hava IPOI Müzesi IPOI in NOPOI Ankara NOPOI

Words are discrete entities. In order to be used in neural network, they need to be transformed to vectors i.e. word embeddings. Embedding layer achieves this; it maps words in dictionary to embeddings. This layer takes words as input and returns word vectors, whose size are specified as embedding size in configurations part.

```
embedding = tf.get_variable('Embedding', [len(self.wv), self.config.embedSize])
window = tf.nn.embedding_lookup(embedding, self.inputPlaceholder)
```

Using above code, firstly an embedding variable is created, having the length of word vectors and embedding size. Then embedding_lookup function adds specified number of embedding vectors to each window and further process with embeddings are done with those windows.

The results of embedding layer then passed to hidden layer where the calculations done for POI extraction problem. Below code shows the model. The input of hidden layer is the output of the embedding layer; which are embedding tensors having length of embedding size multiplied by window size. In hidden layer, weights and biases are applied, window embedding is multiplied by weights defined and biases are added to result. According to those calculations, predictions are done.

Number of nodes in hidden layer are subject to change and this parameter is adjusted in configurations part. There are positive and negative effects of increasing the hidden layer size. On the one hand, more complex calculations can be done with the more number of layers in hidden layer. On the other hand, more computations give the system performance burden. Also over-fitting of data may occur with this case. In experiments part, the results of neural network with different hidden layer sizes are examined.

```
with tf.variable_scope('Layer1', initializer=xavier_weight_init()) as scope:
    W = tf.get_variable(
        'W', [self.config.windowSize * self.config.embedSize,
            self.config.hiddenSize])
        b1 = tf.get_variable('b1', [self.config.hiddenSize])
        h = tf.nn.tanh(tf.matmul(window, W) + b1)
```

In hidden layer, an activation function must be arranged. Activation function is used

to convert the input of the hidden layer to output. For this, tanh function is chosen, that calculates hyperbolic tangent of each element. Analyzing the upper code, weights are prepared in W and biases are arranged in b1 variable. Embeddings and wights are multiplied and biases are added to the result. Activation function applied and set to variable h.

The result of first hidden layer contains the hidden size, that is set while configuration of network. The result passed to second hidden layer. Second hidden layer has input size as hidden size from previous layer and has output size of label size, that is the number of classes prediction are done. In our work this size is three; whose values are POI, IPOI and NOPOI.

```
with tf.variable_scope('Layer2', initializer=xavier_weight_init()) as scope:
    U = tf.get_variable(
        'U', [self.config.hiddenSize, self.config.labelSize]
            self.config.labelSize])
        y = tf.matmul(h, U) + b2
```

The output of second layer as shown above, added to a variable y. This value need to be linearly transformed for prediction. To that aim, softmax function is applied as shown the code below. Softmax transforms raw values into categorization results.

```
self.predictions = tf.nn.softmax(y)
```

Network does predictions using forward propagation, from two hidden layers, calculations are done and sent to next layer. From final result, softmax function is calculated and results are predictions. Figure 4.1. shows the neural network architecture created in this thesis fork for POI extraction.

During training, loss function namely cross entropy is calculated to calculate error values on training data. In order to minimize that error value and training loss, an optimizer namely AdamOptimizer is created as shown below. In experiments part, different types of optimizers are applied. Applied optimizers and their short definitions are displayed in Table 4.18. Implementation details are shown below.

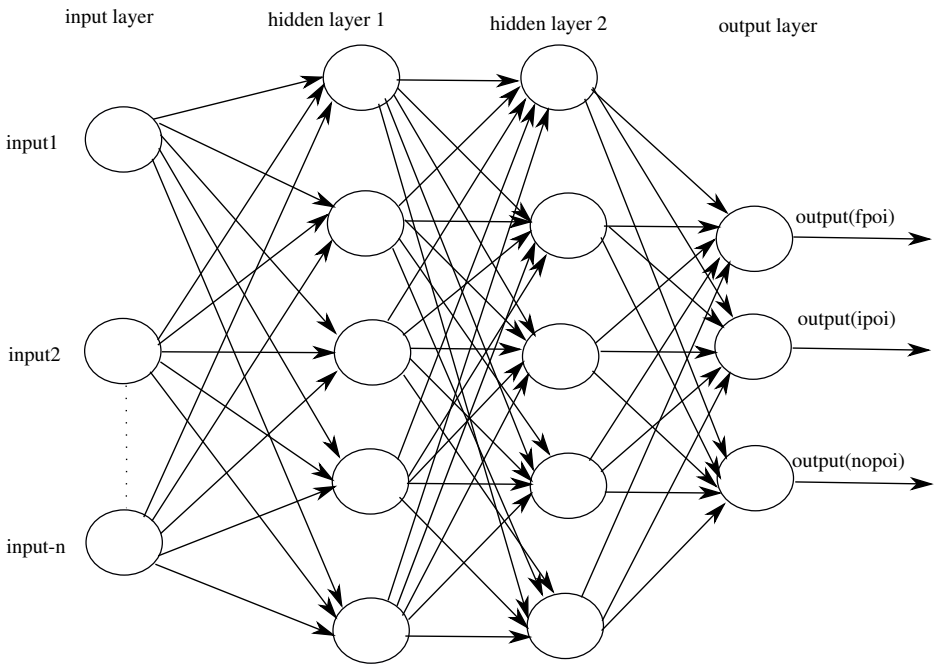

Figure 4.1: Multi-layer neural network is shown. Input layer takes words as each node and sends them to hidden layers. After calculations are done in hidden layers, output value is created.

optimizer = tf.train.AdamOptimizer(self.config.lr)
globalStep = tf.Variable(0, name='globalStep', trainable=False)
trainOp = optimizer.minimize(loss, globalStep=globalSStep)

Table 4.18: Different types of optimizers applied

Gradient Descent Optimizer	Gradient descent algorithm applied. Other optimizers are variants of this optimizer. Learning rate is same for all updates.
Adam Optimizer	Applies Adam algorithm proposed by Kingma and Ba in [8] Different learning rate values are hold for each parameter. Does not keep track of previous weight changes. Only keeps average of previous gradients.
Adagrad Optimizer	Adaptive learning algorithm that adapts different learning rates to parameters as proposed in [3]. Small update for frequently changing parameters, large update for infrequently changing parameters.

An epoch is one iteration of all training data for training. The amount of epoch for

training is important; if network is trained with small number of epoch, the network would have poor and ineffective results. On the other hand, if the network is trained too much times with high number of epoch, there would be a problem of overfitting; neural network starts to memorize rather than learning. In order to prevent from overfitting, a dropout value is used.

After the model is trained, it is tested on testing data. The results of neural network with different parameters are compared in experiments section detailed.

CHAPTER 5

EXPERIMENTS

5.1 Dataset Information

We collected a set of tweets collected through Twitter4J. The initial collection included 2900 tweets, however after data cleaning and removing outliers, there remained 2000 tweets. In the experiments we used this collection of 2000 tweets. For all experiments, we used 10-fold cross validation.

All of the tweets retrieved are Foursquare check-in tweets. That is, all of tweets contain candidate POI mentions posted as a tweet via Foursquare. Retrieved tweets are posted from Ankara and the dataset is collected on different days between October and December of 2016. Data is manually annotated and annotation is done by two human assessors living in Ankara. Each assessor annotated 1000 tweet and verified other assessors' data annotations.

5.2 Experiments

In this study, there are three main groups of experiments; experiments with CRF, Word2vec and neural networks. For each experiment group, there are detailed experiments to detect the method that gives best results in terms of accuracy. All of the experiments are conducted by using the same data set.

5.2.1 CRF Experiments

In order to perform CRF, Mallet is used [1]. Mallet is a software tool to perform various machine learning tasks. As the first experiment, different trainer methods are compared and the best one is selected for the rest of the analysis. Then, using the best trainer method, different features provided by Mallet are compared in different combinations and the best combination is selected.

Finally, the features developed in this study that are sentiment, population and POI density features are used. In the first part, using the best combination of Mallet features; the features we develop were compared to pick the best one in terms of accuracy. Then features developed in this study were used without any Mallet features in order to explore the effect of Mallet features combined with other features. Also, how sentiment, population and poi amount features affect the system accuracy observed with experiments.

5.2.1.1 CRF Experiments Using Different Trainer Methods

In the first part of CRF experiments, we used different methods for training the CRF model. These are Label Likelihood Gradient, Value Gradient and Stochastic Gradient Trainer methods. Detailed information about those trainer can be gained at Section 2.4. The data used in this experiment contains no explicit features; on each line a word and the label of the word is placed. The example data used in this experiment is shown in Table 5.1.

Table 5.1: An example input tweet used for CRF in Mallet

Original Tweet	bana müsade hotel samm in çankaya Ankara (allow me hotel samm in çankaya Ankara)
Tweet format used for CRF	bana NOPOI müsade NOPOI hotel FPOI samm IPOI in NOPOI çankaya NOPOI Ankara NOPOI

[1] Mallet can be reached at: http://mallet.cs.umass.edu/

For each of those trainers, 10-fold cross validation is applied and accuracy results are presented in Table 5.2.

Table 5.2: CRF Training with Different Methods

Trainer	**Accuracy**
Label Likelihood Gradient	**0.936510**
Value Gradient	0.924799
Stochastic Gradient	0.931713

From the results, it can be inferred that Label Likelihood trainer gives the best accuracy and Stochastic Gradient trainer results are very close to it. Therefore on the further experiments of CRF, Label Likelihood trainer is used.

5.2.1.2 CRF Experiments Using Different Mallet Features

As already mentioned, Mallet provides different features. In this experiment, those features with different combinations were used to observe their effects. Data format is the same as in the previous experiment, and Label Likelihood trainer is used.

As Feature Set 1, only Offset Conjunction feature is used having with length 4, which that contains previous word, previous 2 words, next word and next 2 words. In Feature Set 2, only regular expressions are added to detect some patterns. These words are bar (bar), otel (hotel), avm (shopping mall), park (park), müze (museum), pastane (patisserie), hastane (hospital), restoran (restaurant). Then, both of the regular expressions and offset conjuctions used together as Feature Set 3 to detect how those features perform together. For each token i.e. word, suffix feature is added in Feature Set 4. This feature contains last three characters of each word. As Feature Set 5, the best result of first three features are combined with suffix feature and observed how combination changes accuracy. N-gram feature is added in Feature Set 6. For each word, n-grams between 2nd and 7th characters are constructed. Then Feature Set 7 is constructed to only contain window feature to keep sequences of words. In this work, previous 2 and next 2 words are used in window of each word. In Feature 8, n-gram and window features are used together to observe how they affect using simultaneously. In Feature 9, the best resulted feature selected among Feature 6, 7 and 8 and

used together with regular expression feature. Finally in the last feature set, all of the features are added together. Definitions are summarized in Table 5.3.

Table 5.3: Feature Set Definitions

Feature Set	**Definition**
Feature Set 1	Offset Conjunction
Feature Set 2	Regular Expressions
Feature Set 3	Feature Set 1 + Feature Set 2
Feature Set 4	Token Char Suffix
Feature Set 5	BestOf(Feature Set 1, Feature Set 2, Feature Set 3) + Feature Set 4
Feature Set 6	N-gram
Feature Set 7	Feature Window
Feature Set 8	Feature Set 6 + Feature Set 7
Feature Set 9	Feature Set 2 + BestOf(Feature Set 6, Feature Set 7, Feature Set 8)
Feature Set 10	All features combined

Table 5.4 shows the accuracy results of using different combinations of Mallet feature sets.

Analyzing the results, offset conjunctions improves the accuracy but regular expressions has the most significant improvements above all. However when regular expressions and offset conjunctions are used together, accuracy decreases. Suffix feature decreases the accuracy and when it is used together regular expressions, accuracy also decreases compared to only regular expression feature. N-grams and window features decrease accuracy when they are added alone. Moreover when they are used together, accuracy increases compared to only n-gram but decreases compared to only window. Although regular expressions increase accuracy, when it is used with windows, accuracy decreases. Finally, when all of the features are added together, total accuracy decreases compared to none of features added version.

When all of feature sets compared, Feature Set 2, which is the regular expression feature has the best accuracy. Therefore in the next part of the experiment, Feature Set 2 is used.

Table 5.4: Feature Set Accuracy Results

Feature Set	Accuracy
No Features	0.936091
Feature Set 1	0.937030
Feature Set 2	**0.938982**
Feature Set 3	0.936600
Feature Set 4	0.929413
Feature Set 5	0.932465
Feature Set 6	0.923618
Feature Set 7	0.935049
Feature Set 8	0.926363
Feature Set 9	0.934707
Feature Set 10	0.933617

5.2.1.3 CRF Experiments Using Externally Constructed Features

In the previous analysis results, CRF is applied by using different Mallet features with Label Likelihood Trainer. Feature Set 2, which is the regular expression feature, has the best accuracy.

As the next experiment, some external features are constructed and used with CRF. First feature is the sentiment feature. Sentiment information retrieved from tweet is calculated. Results are categorized into three groups that are "positive", "neutral" and "negative". Categorization is done according to the total sentiment scores for each tweet; the results categorized total score is greater than 0 as positive, less than 0 as negative and equal to 0 as neutral.

Another feature constructed for this study is the POI density feature. The amount of nearby POIs of a location where the tweet is posted is calculated. Amount of nearby POIs are categorized as "low", "medium" and "high". To make the categorization, standard deviation is applied to the results that vary between 0 and 200. According to the results, the values that are lower than 65 are labeled as low, between 65 and 195 are labeled as medium and higher than 195 are labeled as high.

Finally population density feature is constructed, which aims to retrieve the amount of people living around the location where the tweet is posted. Results of population densities are categorized into three groups which are "1", "2" and "3". Labeled numbers stand for densities; the grater number implies the high density amount. Population density feature calculated using two different methods. As the first method, the mean value of all neighborhoods is calculated for each district. The density values, which are lower than half of the mean are tagged as "1", values between half and one half are tagged as "2" and values more than one half of mean are tagged as "3". As the second method, standard deviation value is calculated for each district. Some districts have higher standard deviation value from mean value. In such cases, values are tagged as "1" and "3". Otherwise, tagging is same as the first method. Results of two different methods are displayed in Table 5.5.

Table 5.5: Accuracy Results of Population Density Features Using Different Methods

Methods For Calculating Population Density	**Accuracy**
First Method: Mean Calculation	**0.936367**
Second Method: Standard Deviation Calculation	0.936193

According to the results, the first method that calculates mean to tag words for population density has better accuracy score. Hence, in the next experiments, this method is used for population density.

As the next experiment, CRF is applied using three proposed features that are sentiment, POI and population density features. This experiment consists of two parts. Firstly, these features are used without adding any Mallet features to compare the effects of each feature. Secondly, these features are used along with the best feature set of Mallet, regular expressions, to observe how those features perform together.

Table 5.6 shows the results of sentiment, POI and population density features without adding any Mallet features.

When used alone; population density feature decreases accuracy, however sentiment and POI density features increase accuracy. When features are combined in groups of two, all of the combinations increase accuracy and all groups results are very close to each other. Finally when all of the features are added together, the accuracy result

Table 5.6: Accuracy Results of Features Without Mallet Features

Used Features	**Accuracy**
No Explicit Features	0.936510
Sentiment Feature	0.937671
POI Density Feature	**0.939301**
Population Density Feature	0.936367
Sentiment + POI Density	0.937755
Population Density + POI Density	0.937805
Sentiment + Population Density	0.938535
Sentiment + Population Density + POI Density	0.937049

increases compared with the baseline. However accuracy is lower than all of groups of two features.

Table 5.7: Accuracy Results of Features Combined With Best Mallet Accuracy Feature

Used Features	**Accuracy**
No Explicit Features	**0.938982**
Sentiment Feature	0.937809
POI Density Feature	0.938477
Population Density Feature	0.937980
Sentiment + POI Density	0.938397
Population Density + POI Density	0.938479
Sentiment + Population Density	0.935505
Sentiment + Population Density + POI Density	0.937959

Table 5.7 shows the results of sentiment, POI and population density features combined with the feature of Mallet that gives best accuracy on previous part of experiment, regular expressions.

This experiment has interesting results such that when features are combined with the regular expression feature, accuracy results decrease in all combinations. The most significant decrease occurs when sentiment and population features are used together.

The second significant decrease occurs when only sentiment feature used. Other combinations of features have close results, but all of them decrease the baseline accuracy that only regular expression feature is used.

When compared previous two experiments that are using features with and without regular expression features, there are also some interesting results. When only sentiment feature and only population features used with the regex, total accuracy increases when compared the versions without using regex. In contrary, POI density features decrease accuracy when used with regex. When sentiment with POI density and population with POI density used with regex, results are very little improved compared to without regex used version. However on sentiment-population density feature, accuracy decreases when adding regex feature. Finally when all features added together with regex, accuracy is a little higher than the version without using regex.

5.2.1.4 CRF Experiments with Baseline Algorithms & Previous Work

Baseline algorithms used for training in this work are Naive Bayes and Decision Tree algorithms. Besides baseline algorithms, the work [23] also deals with CRF and implementation is done using Mallet. In that work as training method, Label Likelihood Trainer is used. However, features used to train CRF are different from the ones constructed in this thesis work. They have Offset Conjunctions, Suffix feature for the last three characters of each word, First token feature to label first word in sentence, dollar sign feature to mark dollar signed words, startsnumber feature to label words starts with number and capitalized feature to find out the words start with capitalized letters. They do not have the features like sentiment, POI and population density. We use the constructed model of this work with our data and compare the results.

Analyzing Table 5.8, our system gives much better results compared with baseline algorithms. Moreover, comparing with previous work, our system using different features gives better results in terms of accuracy.

Amounts of true positive, true negative, false positive and false negative instances for

Table 5.8: Baseline Algorithms Accuracy Results

Algorithm	**Accuracy**
Naive Bayes	0.758295
Decision Tree	0.715684
Previous Work [23]	0.935584
CRF with POI density feature	**0.939301**

POI density, population density and sentiment features for CRF along with previous work model applied in [23] are displayed in Table 5.9. True positive and true negative rates for all models are displayed in Table 5.10. The highest amount of true positive instances are obtained by sentiment feature and true negative instances are obtained by POI density feature. The highest amount of false positive instances are obtained by previous work model in [23] and false negative instances are obtained by population density feature. The highest true positive rate is obtained by CRF with sentiment feature with %83,1. The highest true negative rate is obtained by CRF with POI density feature with %97,0.

Table 5.9: Externally Developed Features True Positive - True Negative - False Positive - False Negative Counts

CRF Feature	**TP**	**TN**	**FP**	**FN**
POI Density Feature	2762	**11645**	357	574
Population Density Feature	2738	11624	391	**585**
Sentiment Feature	**2763**	11619	395	561
Previous work [23]	2754	11596	**425**	563

Table 5.10: Externally Developed Features True Positive - True Negative Rates

CRF Feature	**True Positive Rate**	**True Negative Rate**
POI Density Feature	0.827937	**0.970254**
Population Density Feature	0.823954	0.967457
Sentiment Feature	**0.831227**	0.967121
Previous work [23]	0.830268	0.964645

Among all experiments with CRF, the best accuracy is retrieved with POI density

feature.

According to the prediction results of CRF with POI density feature, 1147 of 2000 tweets contain POI and rest of 853 tweets do not contain POI. Moreover, over 1147 tweets, there are 640 unique POI names.

5.2.2 Word Embedding with K-NN Experiments

In this part of the experiment, Word2vec with different configurations are used and K-NN classification applied to the model. The aim of this experiment is to find best window size and layer size combination. All of the experiments are calculated using 10-fold cross validation.

Firstly various different combinations of window size and layer size for Word2vec algorithm. Three nearest neighbors of each word in vectoral space is retrieved. The results are shown in Table 5.11.

Table 5.11: Word2vec Accuracy Results

Window Size - Layer Size	**100**	**500**	**1000**	**2000**
3	0.751367	0.738886	0.775576	0.760525
4	0.761244	0.774658	0.766429	0.768554
5	**0.777531**	0.748741	0.762423	0.752393

According to the results, best combination for accuracy occurs when window size is 5 and layer size is 100. It is observed that increasing layer size does not have a significant effect on accuracy. Moreover, it is observed that increasing window size from 3 to 4 generally increases accuracy but increasing window size from 4 to 5 decreases accuracy, except for the layer size of 100. Using those values, another experiment is done. Previous accuracies calculated using 3 nearest neighbors. In the next experiment, we retrieve nearest 2, 4 and 5 neighbors and compare accuracies with nearest 3. Results are shown in Table 5.12.

According to results, although most results are close to each other, best combination is retrieved when 3 nearest words are retrieved.

Table 5.12: Word2vec with Different K-NN Accuracy Results

k value	**Accuracy**
2	0.773909
3	**0.777531**
4	0.735016
5	0.748525

However, the last experiment of Word2vec assumes that if only one of the closest word is labeled as POI, the main word is also a POI. In this part of experiment, we test this assumption with different parameters to find the optimal number of how many closest words should be POI to accept the main word as POI. Results of this experiment presented in Table 5.13.

Table 5.13: Word2vec with Different K-NN Matching Different Number of Words Accuracy Results

k-nn/ matching word number	**1**	**2**	**3**	**4**	**5**
k = 2	0.773909	0.7949209	-	-	-
k = 3	0.777531	0.799375	0.789293	-	-
k = 4	0.735016	**0.805766**	0.792324	0.788460	-
k = 5	0.748525	0.787217	0.795947	0.791355	0.788396

According to the results, it is possible to infer that increasing the number of closest word matching with POI names increase the accuracy in general. However for K-NN value of 4, matching POI number 3 has better accuracy than 4. Moreover for K-NN value of 5, matching POI number 4 has better accuracy than 5. One important observation is that accuracy result differs at most when jumping matching POI number from 1 to 2. When compared with different parameters, the best accuracy of 0.805766, is obtained when nearest 4 words retrieved and 2 of the retrieved closest words should be POI in order to say that main word is a POI.

So far, embeddings are created from the tweet list, that contain emojis. Emojis only contain sentiment; they are not POIs or part of a POI. In this experiment, it is tested that whether data list containing emojis affects the accuracy result for POI extraction.

The results of using data without emojis are presented in Table 5.14.

Table 5.14: Word2vec with Different K-NN Accuracy Results Without Emoji

k-nn/ matching word number	**1**	**2**	**3**	**4**	**5**
k = 2	0.731705	0.772723	-	-	-
k = 3	0.691345	0.746498	0.790242	-	-
k = 4	0.700216	0.775493	0.782350	0.788995	-
k = 5	0.713504	0.777467	**0.792211**	0.791002	0.787144

According to the results, in general accuracy decreases when remove emojis compared to the experiment where emojis are not removed. However, for K-NN value and matching POI is 4, without emoji experiments give better accuracy. Same applies for K-NN is 3 and matching POI count is 3. The best accuracy value is 0.792211 when K-NN is 5 and matching POI count is 3, contrary to the experiment where emojis are not removed. However this accuracy is lower than previous experiment.

5.2.3 Neural Network Experiments

In the final part of experiments, a deep neural network is constructed and different configurations applied. Like the different parameters used for configuring model in Word2vec experiments; also in that experiment, different window size and layer size values are used to observe the effects. 10-fold cross validation is used to obtain the accuracy results.

Table 5.15: Neural Network Accuracy Results

Window Size - Layer Size	**50**	**100**	**500**	**1000**
3	0.9175	0.9216	0.9184	0.9196
5	0.9194	0.9295	0.9290	0.9272
7	0.9294	0.9267	0.9289	**0.9298**

According to the results, the best accuracy is obtained when hidden layer size is 1000 and window size is 7. From the results, it can be inferred that increasing the size of window generally has a positive effect on accuracy for POI extraction for constructing

neural networks.

Using those parameter combinations, the next part of experiment is about the optimizer used for minimizing trainer errors. So far, calculated results are produced from deep neural network, that uses optimizer namely AdamOptimizer. That optimizer is compared with GradientDescentOptimizer and AdagradOptimizer. The learning rates for those organizers is set to 0.1; on the other hand the learning rate of AdamOptimizer is 0.001. The previous two optimizers have higher learning rate as they do not produce results on very small learning rates. Results of different optimizers obtained with 10-fold cross validation are presented in Table 5.16.

Table 5.16: Neural Network Accuracy Results for Different Optimizers

AdamOptimizer	0.9298
AdagradOptimizer	**0.9320**
GradientDescentOptimizer	0.8925

According to the results, AdagradOptimizer provides the best accuracy among other optimizers.

Among all experiments, the best accuracy is retrieved from CRF with POI density feature. Hence, that feature is compared with other CRF models such as sentiment model, population density model and previous work related to this study [23] using statistical significance test. Unpaired T test that compares means of two groups is applied for this purpose. Confidence interval for all tests is 95%. Significance result of POI density feature with population density feature is 0.6450, POI density feature with sentiment feature is 0.8232, POI density feature with previous work [23] is 0.4830. The p values obtained the difference is not statistically significant, however, this may be due to the limited size of the data set.

Time is another metric to compare different approaches for POI extraction problem which are CRF, word embeddings with K-NN classifier and neural networks. For all experiments, 10 fold cross validation is applied. One fold execution time in seconds for different trainer methods of CRF which are Stochastic Gradient Trainer, Value Gradient Trainer and Label Likelihood Trainer are displayed in Table 5.17. According to the results, shortest execution time is obtained from Value Gradient Trainer.

Table 5.17: Execution Times for Different Trainers of CRF

Trainer	**Execution Time in seconds**
Stochastic Gradient Trainer	173.2
Value Gradient Trainer	**6.3**
Label Likelihood Trainer	24.5

One fold execution time in seconds for different approaches for POI extraction problem are displayed in Table 5.18. Label Likelihood Trainer is used for CRF training as it has the best accuracy compared to other trainers. According to the results, shortest execution time is obtained from neural networks.

Table 5.18: Execution Times for Different Approaches for POI Extraction Problem

Method	**Execution Time in seconds**
CRF with Label Likelihood Trainer	24.5
Word Embeddings with K-NN	20.2
Neural Networks	**14.4**

5.2.4 Unsuccessfully Predicted Cases

According to the results of all the experiments for POI extraction problem, the best accuracy is obtained when CRF applied with POI density feature with the accuracy of 0.939493. In this section, outputs of that experiment are examined to observe and analyze the unsuccessfully predicted cases.

In Table 5.19, several examples for false positive results are presented. Besides, there are some cases where the POI extraction method detected POI in a tweet but the detected POI name is incorrect. Such cases are shown in Table 5.20.

Several examples for false negative detection are presented in Table 5.21.

False positive results occur generally in location names that consist of multiple words. Moreover if at least one of these words are labeled as POI in another tweet, this may lead to have wrong results.

Table 5.19: False positive results for CRF with POI density feature

Tweet	**Location names falsely detected as POI**
aydın in the house in naci çakır ankara (aydın in the house in naci çakır ankara)	the house
bir sokakta 2 adet düğün olur mu yaa seyit gazi sokak in seyran bağları ankara (2 weddings in a street seyit gazi street in seyran bağları ankara)	seyit gazi sokak
backline bilkent holding in ankara (backline bilkent holding in ankara)	bilkent holding
bugün cumartesi tunalı (today saturday tunalı)	cumartesi tunalı
birine bakıp çıkcaz okyanus plaza in ankara türkiye (we check for someone okyanus plaza in ankara türkiye)	okyanus plaza

The most of those false negative results occur when there is only the name of location without any descriptive words. In addition, different formats of words affect the results such that the system can detect cafe as POI but café as NOPOI.

Finally there are some cases where guessed POI name is partially correct. That is, POI name is detected but has missing or additional words that are not POI. It is observed that there is a tendency to group POI names to have three words. Hence, when some POI names consist of less than three words, some words may added to make it three. Some POI names may contain more than three words; in such cases, last three words may labeled as POI.

Table 5.20: False POI name results for CRF with POI density feature

Tweet	Falsely detected POI name	Correct POI name
veli toplantısı bahçelievler deneme anadolu lisesi in ankara (parents meeting bahçelievler deneme high school in ankara)	deneme anadolu lisesi	bahçelievler deneme anadolu lisesi
felatun bey ile rakım efendi küçük tiyatro in ankara türkiye (felatun bey ile rakım efendi küçük theater in ankara türkiye)	efendi küçük tiyatro	küçük tiyatro
futsal prof dr yaşar sevim cebeci hentbol salonu in ankara (futsal prof dr yaşar sevim cebeci handball saloon in ankara)	cebeci hentbol salonu	prof dr yaşar sevim cebeci hentbol salonu
sahneee akalın şato balo salonu (stage akalın castle prom saloon)	şato balo salonu	akalın şato balo salonu
salim baba kokoreç köfte in ankara çankaya	baba kokoreç köfte	salim baba kokoreç köfte

Table 5.21: False negative results for CRF with POI density feature

Tweet	Missed POI name
mrb yengeç kardeş deli yengeç in ankara (hello crabby brother deli yengeç in ankara)	deli yengeç
waffle corner in ankara (waffle corner in ankara)	waffle corner
zakkum if performance hall slang ile gece ifte biter in ankara türkiye (zakkum if performance hall with slang night ends at if in ankara türkiye)	if performance hall
eda blank café in ankara (eda blank café in ankara)	blank café
yıldırım budak çığ gösteri merkezi in ankara (yıldırım budak çığ performance center in ankara)	çığ gösteri merkezi (çığ performance center)
yaşasın yemek yemek düveroğlu in ankara türkiye (love eating düveroğlu in ankara türkiye)	düveroğlu
işe devam atatürk koşu yolu in ankara (bussiness goes on atatürk running park in ankara)	atatürk koşu yolu (atatürk running park)
çiftlik kebabı müthiş pishmish in ankara (perfect çiftlik kebab pishmish in ankara)	pishmish
sonunda coffeemania in ankara (finally coffeemania in ankara)	coffeemania
mağazayi komple alabilirim zara in çankaya ankara (I wish to buy everything zara in çankaya ankara)	zara

5.2.5 Successfully Predicted Cases

In this section, successfully predicted cases are mentioned. Table 5.22 shows such examples from the results of CRF with POI density feature.

Table 5.22: True positive results for CRF with POI density feature

Tweet	Extracted POI name
yeni türkü ♡nazım hikmet kültür merkezi in yenimahalle türkiye (yeni türkü ♡nazım hikmet cultural central in yenimahalle)	nazım hikmet kültür merkezi (nazım hikmet cultural central)
mersinli leo ankara da liva pastanesi in ankara (leo from mersin in ankara liva patisserie in ankara)	liva pastanesi (liva patisserie)
sabah sabah köfteci yusuf in ankara (köfteci yusuf in the morning in ankara)	köfteci yusuf
kitaplar ♡ben arkadaş kitabevi in ankara (books ♡I arkadaş bookstore in ankara)	arkadaş kitabevi (arkadaş bookstore)
sırt karın boyun macfit tunalı in ankara (back stomach neck macfit tunalı in ankara)	macfit tunalı
ve ankara'ya vardık dikmen vadisi ankara (arrived to ankara dikmen valley ankara)	dikmen vadisi (dikmen valley)
ağlarım kuğulu park in ankara (I cry kuğulu park in ankara)	kuğulu park
teyze oldum bilgi hastanesi in ankara (I become aunt bilgi hospital in ankara)	bilgi hastanesi (bilgi hospital)
sonunda kurtuluyorum esenboğa havalimanı in ankara (I finally get rid of it esenboğa airport in ankara)	esenboğa havalimanı (esenboğa airport)
geldik yine hacettepe üniversitesi in ankara (arrived again hacettepe university in ankara)	hacettepe üniversitesi (hacettepe university)
sevgiyle anıtkabir in çankaya ankara (with love anıtkabir in çankaya ankara)	anıtkabir

It is important to disambiguate locations from POIs as not all location names are POIs. Some geographic entities, street names and district names can be considered as locations which are not POIs. Moreover, there are some cases where users post their home as location, which are not also POIs. Table 5.23 shows some examples where location names correctly tagged as NOPOI.

Table 5.23: True negative results for CRF with POI density feature

Tweet	**Location name**
seher karanfil sokak in ankara (seher karanfil street in ankara)	karanfil sokak (karanfil street)
ablamda kutlu mahallesi (at sister kutlu district)	kutlu mahallesi (kutlu district)
yolu düşen herkesi bekleriz polatlı in ankara (we host everyone who arrive polatlı in ankara)	polatlı
ersan home in sincan ankara	ersan home
iyi geceler yozgat malikanesi in ankara (good night yozgat malikanesi in ankara)	yozgat malikanesi (yozgat mansion)
güzel eryaman evleri in ankara (nice eryaman house in ankara türkiye)	eryaman evleri (eryaman house)

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this thesis work, the aim is to extract POI mentions from informal microblog messages. Our observations and experimental analysis show that not all locations are POIs and location extraction solutions do not work well enough for POI extraction problem. Another important issue to deal with is the POI extraction from microblog messages, which are short and informal. In this work, we have investigated the effect of three specific feature on POI extraction from tweets within CRF-based solution. In addition, we developed two neural network based solutions for this problem.

In order to make accurate predictions using CRF, features should be able to differ POIs from other named entities. This makes the construction of useful features crucial for CRF tasks. In the proposed system, user sentiments analyzed from tweets are used as the first feature. Additionally, POI density around a geo-tag of a tweet is calculated and added as the second feature. Finally, population density around the location where the tweet was posted is constructed and added as a feature, as well. In neural approaches, POI extraction task is solved based on using word embeddings, such that textual data is represented in vector space so that neural networks can process on it. In this approach, the first solution we develop is the use of K-NN classification applied on word embeddings and nearest labeled elements retrieved. In the second proposed solution, a deep feed-forward neural network is constructed for POI extraction.

The proposed CRF application has the best accuracy of 93,9% when only POI density used without using any built-in Mallet features. When Mallet features are included, best accuracy is 93,8% when only regular expressions are used. Baseline algorithms

which are Naive Bayes and Decision Tree have accuracies of 75,8% and 71,5% respectively. Moreover, CRF based model in [23] has accuracy of 93,5%. Our system outperforms both baseline algorithms and previous work in terms of POI extraction accuracy. Comparing deep and shallow neural networks, it is possible to observe that deep networks have better accuracy for POI extraction task. Deep feed-forward neural network has accuracy of 93,2% whereas the best accuracy result obtained with word embeddings and K-NN is 80,5% under nearest four words and restricting at least two words of them to be POI to label the target word as POI.

There are several research directions on this research for future work. Recurrent Neural Networks (RNN) have good performance on NLP tasks such as text classification, as stated in [11]. In this work, our deep neural model includes a feed-forward neural network. Analyzing the performance of RNN for POI extraction is a promising future task. Moreover, within CRF, new features can be constructed to improve system accuracy.

REFERENCES

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] S. Bird, E. Klein, and E. Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[3] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[4] A. Gibson, C. Nicholson, J. Patterson, M. Warrick, A. D. Black, V. Kokorin, S. Audet, and S. Eraly. Deeplearning4j: Distributed, open-source deep learning for Java and Scala on Hadoop and Spark. 5 2016.

[5] K. Gimpel, N. Schneider, B. O'Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan, and N. A. Smith. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 42–47. Association for Computational Linguistics, 2011.

[6] Y. Goldberg and O. Levy. word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

[7] D. Inkpen, J. Liu, A. Farzindar, F. Kazemi, and D. Ghazi. Detecting and disambiguating locations mentioned in twitter messages. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 321–332. Springer, 2015.

[8] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] M. Konkol and M. Konopík. Segment representations in named entity recognition. In *International Conference on Text, Speech, and Dialogue*, pages 61–70. Springer, 2015.

[10] J. Lafferty, A. McCallum, F. Pereira, et al. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, volume 1, pages 282–289, 2001.

[11] S. Lai, L. Xu, K. Liu, and J. Zhao. Recurrent convolutional neural networks for text classification. In *AAAI*, volume 333, pages 2267–2273, 2015.

[12] C. Li and A. Sun. Fine-grained location extraction from tweets with temporal awareness. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 43–52. ACM, 2014.

[13] J. Lingad, S. Karimi, and J. Yin. Location extraction from disaster-related microblogs. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 1017–1020. ACM, 2013.

[14] S. Malmasi and M. Dras. Location mention detection in tweets and microblogs. In *International Conference of the Pacific Association for Computational Linguistics*, pages 123–134. Springer, 2015.

[15] A. K. McCallum. Mallet: A machine learning for language toolkit. http://www.cs.umass.edu/ mccallum/mallet, 2002.

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

[17] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[18] L. N. Mummidi and J. Krumm. Discovering points of interest from users' map annotations. *GeoJournal*, 72(3-4):215–227, 2008.

[19] P. K. Novak, J. Smailović, B. Sluban, and I. Mozetič. Sentiment of emojis. *PloS one*, 10(12):e0144296, 2015.

[20] N. Ponomareva, P. Rosso, F. Pla, and A. Molina. Conditional random fields vs. hidden markov models in a biomedical named entity recognition task. In *Proc. of Int. Conf. Recent Advances in Natural Language Processing, RANLP*, pages 479–483, 2007.

[21] A. Rae, V. Murdock, A. Popescu, and H. Bouchard. Mining the web for points of interest. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 711–720. ACM, 2012.

[22] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[23] M. Sagcan and P. Karagoz. Toponym recognition in social media for estimating the location of events. In *Data Mining Workshop (ICDMW), 2015 IEEE International Conference on*, pages 33–39. IEEE, 2015.

[24] P. Serdyukov, V. Murdock, and R. Van Zwol. Placing flickr photos on a map. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 484–491. ACM, 2009.

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

[26] A. G. Vural, B. B. Cambazoglu, P. Senkul, and Z. O. Tokgoz. A framework for sentiment analysis in turkish: Application to polarity detection of movie reviews in turkish. In *Computer and Information Sciences III*, pages 437–445. Springer, 2013.

[27] H.-P. Zhang, Q. Mo, and H.-y. Huang. Structured poi data extraction from internet news. In *Universal Communication Symposium (IUCS), 2010 4th International*, pages 116–122. IEEE, 2010.