Content-based geolocation prediction of Canadian Twitter users and their tweets

by

Ali Mert Metin

Thesis submitted to the
Faculty of Graduate and Post Doctoral studies
In partial fulfillment of the requirements
For the M.Sc. degree in
Computer Science

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AI Artificial Intelligence

BOW Bag-of-words

CA Census Agglomerations

CMA Census Metropolitan Areas

CNB Complement Naive Bayes

CNN or ConvNET Convolutional Neural Network

DNN Deep Neural Networks

DP Deep Learning

GNB Gaussian Naive Bayes

ML Machine Learning

MLP Multilayer Perceptron

MNB Multinomial Naive Bayes

NB Naive Bayes

NLP Natural Language Processing

POS Part-of-speech

RBF Radial basis function kernel

RELU Rectifier

SVM Support-vector machine

SVN Support-vector networks

W-NUT Workshop on Noisy User-generated Text

Abstract

Last decade witnessed the rise of online social networks, especially Twitter. Today, Twitter is a giant social platform with over 250 million users —who produce massive amounts of data everyday. This creates many research opportunities, specifically for Natural Language Processing (NLP) in which text is utilized to extract information that could be used in many applications.

One problem NLP *might* help solving is *geolocation inference* or *geolocation detection* from online social networks. Detecting the location of Twitter users based on the text of their tweets is useful since not many users publicly declare their locations or geotag their tweets. Location information is crucial for a variety of applications such as event detection, disease and illness tracking and user profiling. These tasks are not trivial, because online content is *often* noisy; it includes misspellings, incomplete words or phrases, idiomatic expressions, abbreviations, acronyms, and Twitter-specific literature.

In this work, we attempted to detect the location of Canadian users —and tweets sent from Canada —at metropolitan areas and province level; this was not done before, to the best of our knowledge. In order to do this, we collected two different datasets, and applied a variety of machine learning, including deep learning methods. Besides, we also attempted to geolocate users based on their social graph (i.e., user's friends and followers) as a novel approach.

ACKNOWLEDGMENTS

Even though this thesis is the result of my own efforts, it would not be possible without the contribution of many people. I would like to acknowledge and express my deepest gratitude to the following people for their time and effort:

- My supervisors, Dr.Diana Inkpen and Dr.Xiaodan Zhu, for providing me this opportunity to work on this thesis, for their support during my years in University of Ottawa. I had the very good fortune to have studied under their supervision.
- Dr. Kenton White for giving me great suggestions for this work.
- My wife, Irem Er, for her love, encouragement, support and making me believe that I could finish it.
- My family, especially my mom, for believing in me and supporting me.
- My lovely cat Müdür.
- Guido van Rossum for creating Python which I believe one of the greatest programming languages ever.
- All the University of Ottawa administration staff, especially, Annik Dion who helped me a lot during the course registration process and the director of Computer Science, Paola Flocchini, for giving me great advice during my journey at the faculty.

Chapter 1

Introduction

'If you think it's simple, then you have misunderstood the problem.'

— Bjarne Stroustrup, Creator of C++

1.1 Introduction and motivation

Online social networks have gained a lot of popularity over the last decade. They include every-day-use social networks such as Facebook, Twitter and Google Plus, location-based services such as Foursquare, Swarm and Yelp, photo-sharing websites such as Instagram, 500px and Flickr and some career-oriented social networks such as LinkedIn. Even though there are differences between these platforms, the common feature of them is the social network; that is, the network in which people with common interests meet and share text, photos, and videos.

Twitter stands out among all the online social networks because of its unique features. One of them is the short and noisy messages (a.k.a tweets); they are 280-character limited status updates —and can only be sent by Twitter users. The character limit encourages users to send more tweets. Moreover, Twitter has a different perspective about social networks than other online social networks such as Facebook: the *unmutual friendship*. In most platforms, *friendship* requires mutual following of both parties, but not in Twitter. In Twitter, users may follow others without requiring them to follow back.

Twitter, —like many other social platforms —is a virtual social network allowing interaction with the others users with the same interests and probably the most significant part of the virtual social network is the *location*. In Twitter, users can enter a real-world location in their profile. This is *usually* the long-term residential address of the user. Twitter users are aware of what is going on around them with their locations; they notice and find interesting things near them. Furthermore, users can also attach a location (i.e., geotag) to their tweets when sending out from mobile devices such as smartphones or

tablets. Additionally, users can mention location names in their tweets (i.e., locations where they are traveling).

Knowing the physical location information of Twitter users has multiple benefits: (i) we can find out about events such as activities, protests, emergency situations; (ii) offer location-based services to users (e.g., store locations, proximity-based marketing, and travel information); and, most importantly, we can develop many applications on top of location-based systems such as trend detection (Benhardus and Kalita, 2013), disease and illness tracking (Yoon et al., 2018) (Orabi et al., 2018), and user profiling (Ikeda et al., 2013).

Currently, there are two ways of sharing location information by users: (i) using the *user declared location* field in user profile —which is used for home location (i.e., long-term residential address) of user; (ii) geotagging a tweet —introduced in 2009 —when sending out. On the other hand, the research reveals that very few of users adopt tweet geotagging; user declared locations are far more popular though. As shown in Cheng et al. (2010), only 0.42% of the tweets are geotagged in their dataset and 26% of the users declared a city-level location (e.g., Toronto) in their profile. Besides, the same study shows that user-entered locations are *often* incomplete or inaccurate.

Finding the location of users —or objects —is not a new problem —and the terms *geolocation estimation* or *geolocation prediction* were used for this problem over the decade. There have been many approaches proposed for the problem in different platforms such as Wikipedia (Wing and Baldridge, 2011) (Wing and Baldridge, 2014) (Roller et al., 2012), Facebook (Backstrom et al., 2010), a variety of web pages Amitay et al. (2004) (Fink et al., 2009) and mostly for Twitter (Cheng et al., 2010) (Hecht et al., 2011). Besides, sub-problems such as location recognition and disambiguation are also well studied. Many natural language processing and machine learning techniques have been developed such as geolocating users based on the word-usage (i.e., finding location by local words), dialects, or slang. Other methods focused on the social network (i.e., geolocating users based on their friends). On the other hand, the characteristics of online text, especially for tweets, raise many challenges, due to their very informal nature, similar to SMS and online chat. Users *often* send tweets without completing words and phrases, use acronyms, make a lot of spelling mistakes —and often develop their own linguistic style —which is partly due to the character limit. Because of these challenges —and the need of the knowing Twitter users' location, Twitter location prediction gained a lot of attention; there was a lot of research done all over the world. In 2016, even a shared task was organized by Workshop on Noisy User-generated Text (W-NUT). [1]

There have been a lot efforts to geolocate Twitter users —and their tweets —for different parts of the world; mostly for North America and Europe. The first one is more challenging since the land coverage is bigger and the amount of places is a lot higher than Europe. The task is even more challenging when the entire world is covered. Although there are many geolocation studies targeting different parts of the world, there is no single study focusing on geolocating only Canadian users and tweets, to the best of our knowledge —which is the primary motivation of this study.

[1] W-NUT organizes workshops focusing on Natural Language Processing applied to noisy user-generated text such as tweets.

Location detection is a crucial task for several reasons: (i) it helps to find out social, economical and political trends; (ii) it makes it easy to track emergency situations (e.g. natural disasters) (iii) helps businesses to target audiences.

1.2 Goals

The primary goal of this work is to predict the home location (i.e., long-term residential location) of Canadian Twitter users, as well as to find out tweet locations by using natural language progressing techniques —and machine learning and deep learning (i.e., a class of machine learning algorithms) methods. We are also looking for the possible ways of geolocating users by their social connections.

1.3 Contributions

This thesis has several contributions:

1. We geolocate users and their tweets in metropolitan areas and provinces by applying various machine learning and deep learning methods
2. We introduce two datasets: `Twitter-CAD-Users` and `Twitter-CAD-Tweets`. The first dataset contains users with their tweets, and social network (i.e., friends and followers) as well as some metadata information (e.g., users' tweets, friends, and followers count as well as their gender). The second dataset contains only geotagged tweets. Both datasets are made available to other researchers. [2] [3]
3. We share several statistics about Canadian Twitter users (e.g., user-gender ratio) and tweets. We believe that this gives some insights about Twitter usage by Canadian users.

1.4 Outline of the thesis

This thesis is organized as follows:

- **Chapter 1 Introduction**
  In this chapter, the motivation and the goals are introduced.

[2] There are many efforts in Canada for providing open datasets such as Open Data. However, no dataset that consists of only Canadian Twitter users along with their tweets has been shared publicly prior to our studies.

[3] The dataset is available on Githib at https://github.com/mertmetin/Canadian-Twitter-users-and-tweets-datasets

- **Chapter 2 Background Information**
  This chapter discusses Twitter basics, geolocation, types of Twitter-related locations, as well as the techniques used for location prediction in this work.

- **Chapter 3 Related Work**
  This chapter introduces the past work done for geolocation prediction, mostly for Twitter.

- **Chapter 4 Data**
  In this chapter, we give an overview about existing datasets for geolocation prediction and describe the two datasets collected for this research. We also share statistics about the Canadian Twitter users.

- **Chapter 5 Methodology and Experiments**
  In this chapter, all the experiments done for this research are presented.

- **Chapter 6 Conclusion and Future Work**
  In this chapter, we share several ideas for extending this work —which might improve the results.

Chapter 2

Background Information

‘Courage is not having the strength to go on; it is going on when you don’t have the strength.‘
— Theodore Roosevelt, 26th. U.S. President.

2.1 An overview of Twitter

Twitter is an American social networking (or micro-blogging) platform that allows users to send short texts (a.k.a tweets) along with images and videos. The texts are quite limited in terms of character limit; the maximum allowed is 280 characters[1] (as of November 7, 2017; previously it was 140). Both registered and non-registered users can use Twitter, although the latter ones cannot send messages or share a message (i.e., retweet). Users can connect to Twitter by using variety of client programs, including the web interface (twitter.com), mobile apps (Twitter for Android and Twitter for iOS) and Twitter via SMS. The company’s headquarters are in San Francisco and there are more than twenty offices around the world.

[1] 140-character limit still applies to languages like Chinese, Japanese and Korean

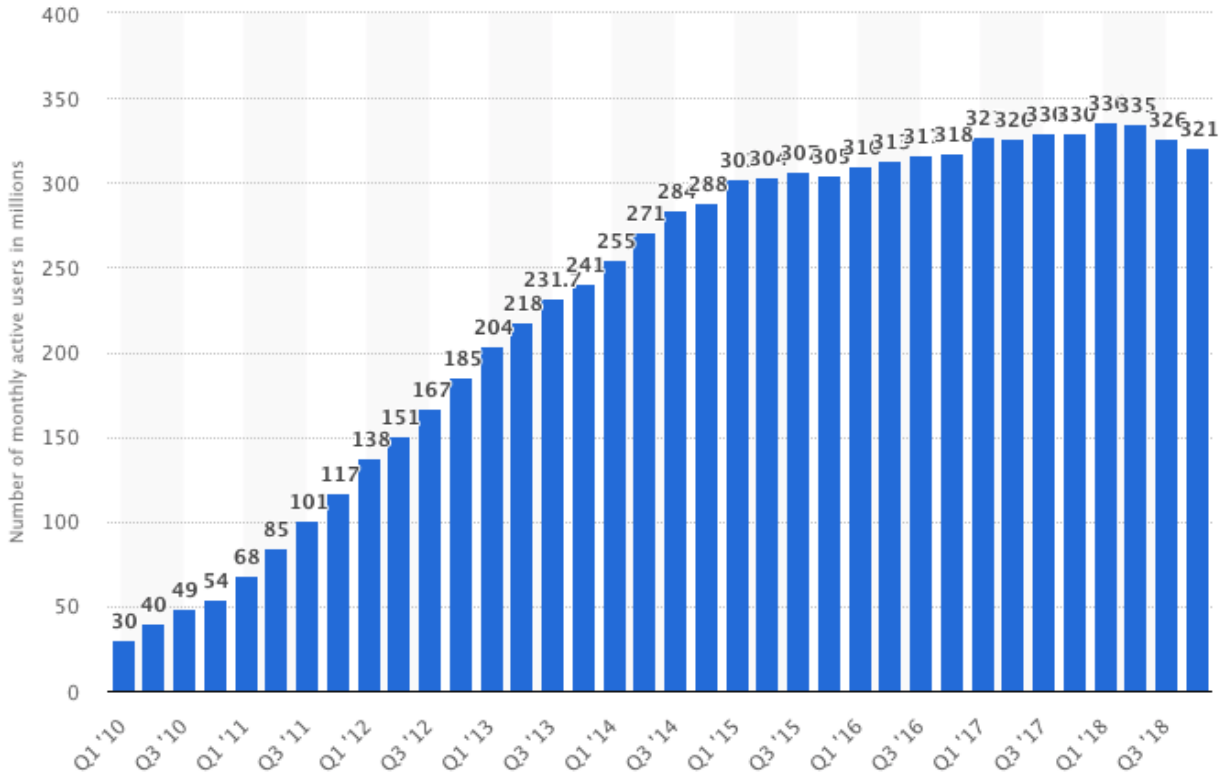

Figure 2.1: Twitter's growth year by year Statista

The platform was created in March 2006 by Jack Dorsey and his friends Noah Glass, Biz Stone, and Evan Williams. It was released in the same year in July and it had a blazing growth since then. According to Statista[2], Twitter had 138 million users as of 2012 and almost 350 million tweets sent. Today, it has more than 320 million users.

2.1.1 Fundamentals of Twitter

Twitter shares a lot of commonalities with the other online social networks such as Facebook, Instagram, Pinterest, and MySpace, but there are several Twitter-specific features. In this section, we briefly describe them:

[2]Statista is an online statistics portal.

Tweets

A *tweet* is a chunk of user-generated post sent by a user on the platform. It is limited to 280 characters and may describe anything the user wishes, including his mood, ideas about a particular topic, or comments for an event. Tweets can be composed of only text, other media (e.g., image, video) or combination of both them.

There are also other tools that can be used as part of tweets. Those are:

- **Mentions**: @ followed by a *username* or *mentions* is used to write or reply to a user. When this is done, the interacted user will see a notification that he or she was *mentioned* by a user.
- **Hashtags**: # followed by a *word*(e.g., #uOttawa) used to create topics in Twitter. People who use the same hashtag in their tweets can join to the topic. Besides, the tweets associated with the hashtag can be seen easily in the platform.
- **Retweeting**: A tweet can be shared. This action is called *retweeting*. *Retweet* or *retweet with comment* will reference to the original tweet.

Social network

In addition to sending tweets, a user might subscribe to other users' tweets and see them in his own timeline by *following* them. If user w_i follows user w_j, user w_i is called *follower* and w_j is called *followee*. However, this relationship is unidirectional, unlike in many other platforms: it does not mean that w_j has to follow w_i. If both sides follow each other, this is regarded as *friendship* or *Twitter friendship*.

User accounts and profiles

All registered users have an account in Twitter. Each account has a unique username (e.g., uOttawa). Accounts can have a number of basic settings such as email, language, timezone, as well as some security settings (e.g., login verification) or privacy settings.

Each account has a profile. The user profile is the place where users define who they are. It offers a wide variety of customization, including picking a user profile image and banner, adding name, bio, location, birth date, and website. Additionally, users have the option to change the theme color of their profile.

There are two types of account visibility for users: protected and unprotected. If an account is protected, then the user's tweets can be seen by only those who follow him; tweets of a unprotected account are shared with the world. By default, all the account are *unprotected*.

Some accounts are *verified*, meaning that they have a public interest. Usually, verified accounts are owned by famous people such as artists, politicians, sportspeople, and entertainers. The main goal of verification is to indicate that the account is owned by the

claimed person in the profile. A verified account has a blue badge in the profile. Currently, Twitter put its account verification program on hold due the controversy around their policy.

2.1.2 The purpose of using Twitter

Twitter is used for a variety of purposes. To begin with, **regular people** use Twitter to communicate quickly, effectively and writing with variety of levels of privacy. Because of its popularity, Twitter became a medium of sharing information directly with a person or a group of people - mainly to spread the information widely.

Companies also use Twitter for a number of reasons. The main goal of the companies in Twitter is building awareness of their brand and providing customer support and service and eventually boosting sales. Brand awareness is achieved in the platform in many ways such as tweeting about company's events, sharing special deals and promotions exclusively through the platform, and organizing social campaigns.

Celebrities, models, and sportspeople are also in Twitter primarily to connect with their fans.

Twitter is also heavily used by **politicians**. They use it to communicate and interact with their constituents and run their campaigns. In the 2016 U.S. presidential election, both nominees - Donald J. Trump and Hilary Clinton used Twitter a lot during their campaigns.

2.2 Geolocation

In general, geolocation is the process of identifying geographic location of a source such as a radar, mobile phone or a Internet-connected host (e.g., computer, server) on the Earth.

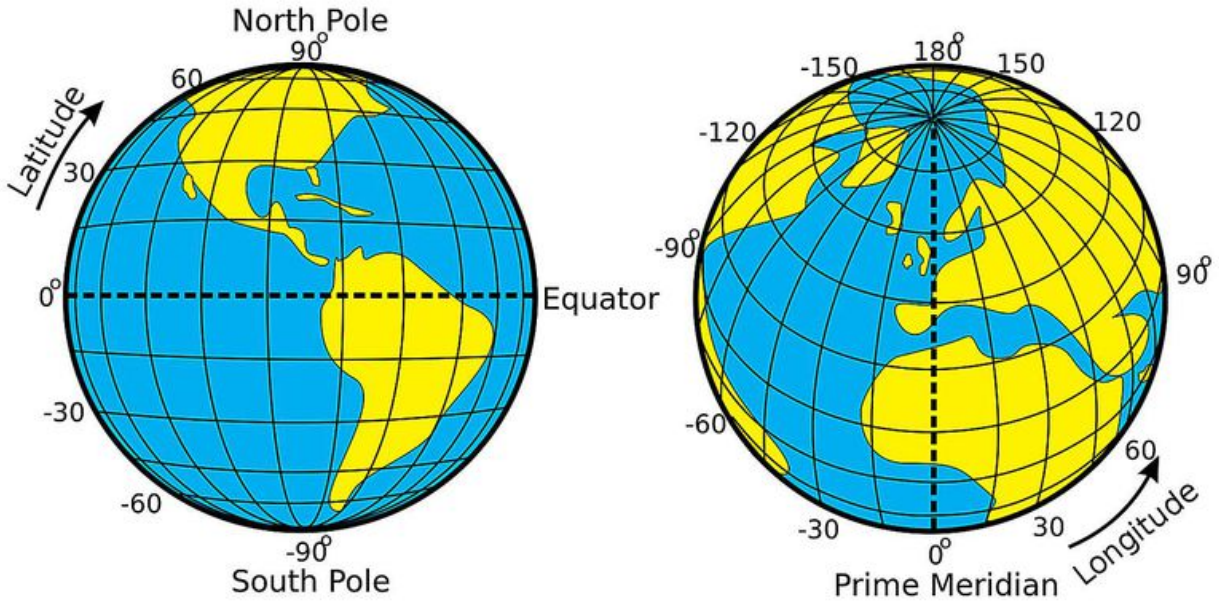

Figure 2.2: Latitude and Longitude on Earth

The location is usually defined by latitude and longitude coordinates. Latitude (or *parallel*) is defined lines parallel to each other that run East-West. It measures the North-South position between the poles. The North Pole is at 90 degrees North and the South Pole is 90 degrees South. The equator is at 0 degree. On the other hand, longitude lines run between North-South and measure the East-West position. The prime meridian has the value of zero and it is in Greenwich, U.K.

Depending on the target, different geolocation estimation methods are used such as:

- **Internet-connected hosts**: An Internet-connected host's (e.g., computer, server) location is found. The most common way is the IP geolocation where host's IP information is associated with a location.
- **Social media users**: Machine Learning —and sometimes Natural Language Processing —methods are used to find a location (e.g., user's home location) based on text (e.g., tweet, Facebook post), image (e.g., location picture) or user's metadata information (e.g., timezone).

In this work, we focus on geolocation estimation of social media users, especially Twitter users, by using Machine Learning and Deep Learning methods combining with NLP techniques. There are several reasons why we chose Twitter for this task. In the next section, we explain them in details.

2.3 Why Twitter for geolocation estimation?

Twitter has ranked as the second-largest social network and it collects a very large volume of data in short periods of time. The data includes (i) short texts (tweets) sent by Twitter users, (ii) the social network between the users, as well as (iii) user profile information. The availability of these three types of data makes it possible to use Twitter for geolocation estimation.

2.3.1 Tweets

Tweets have a powerful role in the context of geolocation estimation for a number of reasons. First and most importantly, tweets can include location information in different forms. The location information might appear as *location names* in hashtags or as pure words or in other media (e.g., video, image).

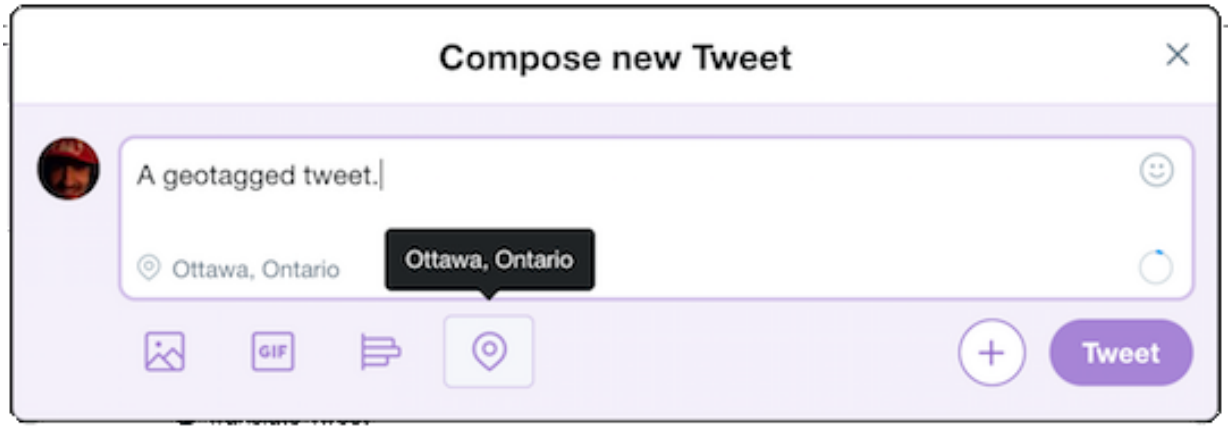

Figure 2.3: A geotagged tweet with Ottawa, Ontario

Furthermore, users can attach a location to their tweets. This action is called *tweet geotagging* or simply *geotagging*. Geotagging is disabled by default, so users have to opt-in if they want to send a tweet with a location. The location can be a city, state/province, country, a place name (e.g., restaurant, gym), or the combination of them.

Only original tweets can be geotagged. Retweets are not considered original, so they do not contain the location information even if the original tweet is geotagged. Nonetheless, if the retweet is shared by creating another tweet by the user, it can be geotagged.

2.3.2 Twitter social network

Twitter friendship does not imply the real-life friendship. Friends on Twitter do not have to be real-life friends necessarily. Celebrities on Twitter do not follow back their followers. However, research such as (Backstrom et al., 2010) shows that friends in real-life tend to be friends in online social platforms as well. Therefore, one's social network might give insights about his/her location.

2.3.3 User profiles

Twitter user profiles have a few fields that can be utilized to infer a location. The most important of them is the location field. The location field refers to the *home location* of the user and it can define the location at different granularity level, such as city, province, and country. Moreover, it is observed that some users enter location information to their bio and name fields.

2.4 Types of Twitter-related locations

There are three types of locations inferred from Twitter, that are *home location, tweet location*, and *mentioned location*. We give an overview of those locations below and discussing how to build the ground truth for each task:

2.4.1 Home location detection

Predicting the home location of Twitter users is the most common task of all three. Home location identification of a user is the finding the long-term residential location of a user. The prediction of home locations makes possible various applications such as *location based advertisement, health monitoring, opinion polling estimation*, and even *natural disaster capturing*.

The home location of a user can be represented in different granularity. These are:

- **Administrative divisions** such as countries, states/provinces, cities, neighbourhoods, and towns.
- **Geographical grids**: A grid system is set up by dividing the Earth into cells of equal or different sizes and home locations are represented by grids.
- **Geographical coordinates**: home locations are represented by latitude and longitude pairs.

The data for home locations is *usually* collected from the *user declared location* field in user profiles. However, since this a free-form field, it might contain noisy information. Besides, some users do not prefer to enter any information to this field for privacy reasons. Moreover, some studies also collect geotagged tweets and use them for home locations.

Common techniques are:

- Consider the most common city in geotagged tweets as the home location.
- Find the first valid geotagged tweet and convert it into a division, grid, or latitude/longitude pair.
- Calculate the geometric median of geotagged tweets and consider it as home location.

2.4.2 Tweet location prediction

Finding a tweet's location is also quite common. Tweet location refers to the location where a tweet is sent out from. By predicting tweet locations, researchers aim to discover a users' mobility over time. In most cases, tweet locations are solely based on geotags of tweets instead of a combination of user declared location and tweet geotags like in the home location. Also, a tweet location can be any administrative division (country, state/province) or a point of interest (e.g., restaurant, bar).

2.4.3 Mentioned location prediction

Users mention *location names* when they compose tweets. Detecting the mentioned locations in tweets might provide better analysis of tweets, and enable applications like location recommendation or disaster capturing. Essentially, there are two sub-tasks for mentioned location detection:

- **Mentioned location prediction** which involves finding the mentioned locations in the tweets' content.
- **Mentioned location disambiguation** which is process of disambiguation of geo/geo (*Toronto, Ontario vs Toronto, Ohio*) and geo/non-geo[3] entries.

[3]Some locations have the same name with non-geopgraphical entities such as *of* as English preposition vs *Of, Turkey*

2.5 Natural language processing and machine learning

Geolocation of Twitter users requires the use of Machine Learning (including Deep Learning) methods combined with NLP techniques. In this section, we briefly describe all the learning methods and techniques used in this study.

2.5.1 Natural Language Processing

Natural Language Processing (NLP) is a branch of Computer Science and Artificial Intelligence interested in developing methods and techniques to understand natural language (i.e., human language). These methods and techniques are used for many tasks such as sentiment analysis, machine translation, automatic summarization, and speech recognition.

In this work, we used several NLP techniques:

- **Tokenization**: Tokenization is the initial task for the processing human language (text) data. It is the process of separating text into token such as words, punctuation, and numbers. A tool that does this job is called a *tokenizer*. There are many tokenizers available freely such as the NLTK's tokenizer [4] and Stanford's NLP tokenizer[5]. There are some other tokenizers, such as NLTK's tweet tokenizer for specific types data (tweets in this case).

- **Part-of-speech Tagging**: Part-of-speech (POS) tagging is the task of assigning the correct part-of-speech tag (e.g., *VB* for verbs or *JJ* for adjectives) to each token. The tags come from a predefined set, and this set might change depending on the language. However, there are even different tags for the same language (e.g., English).

- **Bag-of-words**: Bag-of-words (BOW) is very popular document representation model. In this model, each text document is represented by its words —and each word in corpus is represented by a unique id assigned to it. The words then stored in *vectors*. Thus, each text document is a vector of numbers. The values could be binary (1 for words that appear in a document and zero for the rest) or based on term frequency, or on terms frequency divided by the inverse document frequency (td-idf).

- **Word Embeddings**: Word embedding is also quite popular document representation model. In word embedding models, words with similar meaning (e.g., good, nice, beautiful) have similar representations. Word embeddings are claimed to better represent the semantics of the words Kusner et al. (2015).

Tokenization and part-of-speech tagging are *usually* the initial steps for the purposes of preparing text documents for models (e.g., bag-of-words, word embeddings). Then,

[4] https://www.nltk.org/api/nltk.tokenize.html
[5] https://nlp.stanford.edu/software/tokenizer.shtml

documents are modelled (i.e., they are converted into vectors of numbers) since the machine learning algorithms cannot work directly with with the text, but with numbers, especially vectors of numbers.

2.5.2 Machine Learning

Machine Learning (ML) is an application of Artificial Intelligence (AI) which allows computers to learn from data and make future predictions without being explicitly programmed.

Machine learning algorithms are generally divided into three types:

- **Supervised learning**: Supervised learning is the learning from labeled data. After the learning is completed (i.e., training), the algorithm makes predictions —or assigns a label —to given data example based on the patterns learnt through the training process.
- **Unsupervised learning**: Unsupervised learning is the learning from unlabeled data (i.e., learning the without guidance of labels). Unsupervised learning finds patterns in data without being explicitly taught and reacts to new data based on the experience accumulated duting the training process.
- **Reinforcement learning**: Reinforcement learning is different than both supervised and unsupervised learning. The system learns from the results of its actions, instead of learning from data. The algorithm selects its actions based on past decisions (experiences) and also by new choices (exploration).

We used supervised learning since our data is labeled (e.g., each user and tweet are labeled with a location). Next, we explain the machine learning algorithms (SVM and Naive Bayes)[6] used in this research.

Support Vector Machine

Support-vector machine (SVM) or Support-vector networks (SVN) (Boser et al., 1992) are supervised learning models used for classification and regression in machine learning. An SVM model consists of data points in the vector space divided by hyperplane(s). Hyperplanes are dividers which maximizes the margin between two classes. Support vectors are the data points that lie closest to hyperplanes, which in turn aid to build them. In other words, support vectors are used to set the boundaries between classes.

[6]We preferred to use SVM and Naive Bayes for baseline tests since both algorithms are supported extensively by sci-kit learn (i.e. providing many variants of the same algorithm such as NuSVC and LinearSVC for SVM)

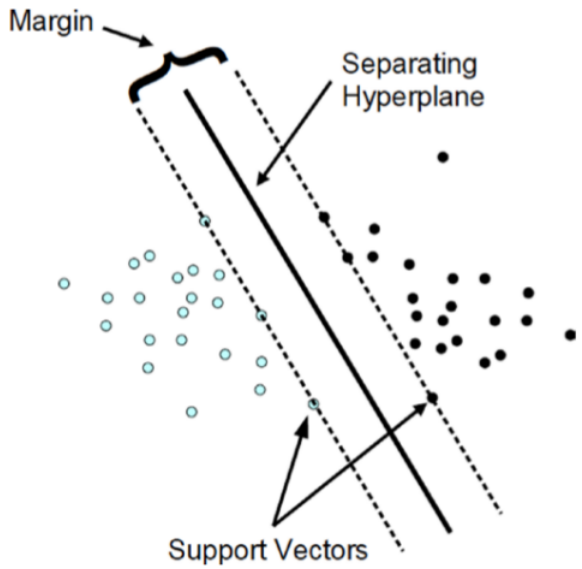

Figure 2.4: An SVM model with a single hyperplane and multiple support vectors.

The similarity between two data points is decided by the kernel function (also called similarity function). SVM kernels can be *linear* and *non-linear* (also Radial basis function kernel (RBF)). The linear kernel is straightforward; given two data points, the similarity is the length of the projection of one data point to another. On the other hand, the similarity of two data points is reduced by the radius of *sigma* in the RBF kernel.

Naive Bayes

Naive Bayes (NB) (John and Langley, 1995) is a collection of classifiers based on Bayes' theorem. Bayes' theorem is a simple probabalistic theorem that calculated the probability of an event based on knowledge affecting it. Here is an example: if dialectical words terms give clues about one's location, then, by using the Bayes theorem, words are used to decide if a person is in a certain location or not.

There are many Naive Bayes classifiers. Some of those are:

- Multinomial Naive Bayes
- Gaussian Naive Bayes
- Complement Naive Bayes

Multinomial Naive Bayes (MNB) is a popular Naive Bayes variant based on multinomial distribution which is the generalization of the binomial distribution. In the binomial distribution, there are only two possible outcomes —or classes on any individual trial (e.g., class prediction) whereas the number of possible outcomes on any given trial is greater

than two in multinomial distribution —makes it possible to use it for data like ours. Furthermore, in MNB, each document is assigned a class label with a rule called maximum a posteriori (MAP) that assigns a class label to each word w. Each word has an importance score for the final outcome of the trial. More specifically, each word w is assigned a probability for each class label. The algorithm calculates a prior probability $Pr(c)$ for each class c by checking the frequency of each class label in the training set.

Gaussian Naive Bayes (GNB) is another Naive Bayes variant based on normal distribution (also Gaussian distribution). The normal distribution is very common continuous probability distribution, that is, a distribution of vectors in a bell-shaped curve. In other words, vectors are seen as distributed around a central value.

Finally, **Complement Naive Bayes (CNB)** predicts probabilities for a class c based on the complement of c (i.e., all classes except c) rather than using the probabilities in the training data for class c.

2.5.3 Deep Learning

Deep Learning (DP) or **Deep Neural Networks (DNN)** is a subset of machine learning, focusing on algorithms that simulate the biological neural networks in the brain and copying their behavior. As above, learning can be supervised, unsupervised, or based on reinforcement.

Many neural network architectures, such as deep neural networks, have been applied to many tasks such as machine translation, image recognition, or bioinformatics tasks, and gained huge popularity due to their superior performance compared to standard machine learning algorithms.

In this work, we applied two neural network architectures: the multilayer perceptron and the convolutional neural network:

Multilayer Perceptron

Multilayer Perceptron (MLP) is a type of feedforward artificial neural network[7]. A simple MLP architecture consists of at least three layers: input layer, hidden layer, and output layer where each layer is connected to each other. However, more complex MLP architectures can have many hidden layers. Each layer in MLP is responsible for a certain task. First, the input layer provides data to the adjacent layer: the hidden layer. In the hidden layer, computation is performed (e.g., classification). Lastly, the output layer provides the computation result.

Convolutional Neural Network

Convolutional Neural Network (CNN or ConvNET) is another type of deep learning algorithm; they are used mostly for analyzing visual objects such as images, but they

[7] https://en.wikipedia.org/wiki/Feedforward$_n$eural$_n$etwork

achieved state-of-art results for text data over the years —and were applied to many natural language processing tasks since then.

Similar to MLP, convolutional neural networks also have multiple layers such as input layer, hidden layers, and output layer. The hidden layers are *usually* convolutional layers, Rectifier (RELU) layers, pooling layers, or fully-connected layers. A convolutional layer is responsible for applying *convolution* [8] operation to data; RELU (i.e., activation function) produces output for each node and supplies it to next one; pooling layer reduces the dimension of data; and finally a fully-connected layer is a bridge between layers. It can be used to connect neurons in one layer to another layer.

2.6 Evaluation metrics

In this section, we review the common evaluation techniques used by the literature for geolocation estimation of Twitter users and tweets. There are two types of evaluation metrics, namely *distance-based metrics* and *token-based metrics*. Distance-based methods use geographical coordinates to compare locations, whereas token-based metrics use administrative divisions.

2.6.1 Distance-based metrics

Distance-based metrics are used for both home and tweet location prediction in which a user and tweet location is predicted, respectively. Depending on the level (user or tweet), let s be a user or a tweet. Let S be the set of all users or tweets for prediction. The goal is to find out a location $l(s)$ for each s. The predicted location must be in or close to the actual location. After that, the actual location and the predicted location are converted into latitude and longitude pairs, and the difference between them is calculated. The result is called *error distance*.

Because the evaluations are made on set of users or tweets, the mean and median distances of error distances can be found. This is called **Mean Error Distance** and **Median Error Distance**, respectively.

Different from Mean and Median Error distance, the **distance based accuracy** or Acc@d is also used. d represents the error distance. In the related work (Rahimi et al., 2017) (Cheng et al., 2010) (Eisenstein et al., 2010), d is taken as 100 miles or 161km.

[8] http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

2.6.2 Token-based metrics

Token-based metrics is an alternative to distance-based metrics. They are based on administrative divisions such as country, province or a point of interests.

The most used metric for all three geolocation prediction problems (home, tweet, or mentioned location) is *accuracy*. An accurate prediction occurs when the predicted location is in the ground-truth location(s). Consider l(s) and l*(s) as the predicted and the ground truth locations respectively for a user, tweet or mentioned location, then **Accuracy** is defined as the ratio of correct predictions in S where S is the set of users, tweets or mentioned locations:

$$Acc = \frac{|s \in S : l(s) = l*(s)|}{|S|}$$

In some systems, there might be several predicted locations instead of a single one, and they are ranked by the most accurate to the least. The straightforward approach is to treat the top-ranked location as the predicted location. On the other hand, this approach does not consider the other predicted locations in the output, and they might be useful for some applications. For this reason, **Ranking-based accuracy** was introduced. The ranking is considered *accurate*, if the actual location is in top-n L_k(s) results. Therefore, ranking-based systems defined as:

$$Acc = \frac{|s \in S : l*(s) \in L_k(s)|}{|S|}$$

It is possible that geolocation systems might fail to predict a location often because of the inadequate information. To demonstrate, home and tweet level prediction fail if inadequate information is provided (Compton et al., 2014) (Schulz et al., 2013). The same problem also occurs in mentioned location disambiguation when the system cannot match the given location with an entry. In this case, *Precision*, *Recall*, and F_1 are used as metrics.

Precision is defined as correct predictions over all predictions:

$$Precision = \frac{|s \in S : l(s) = l*(s)|}{|s \in S : l(s) \neq \emptyset|}$$

and **Recall** is quite similar to **Accuracy**:

$$Recall = \frac{|s \in S : l*(s) \in L_k(s)|}{|S|}$$

Finally, F_1 is the harmonic mean of Precision and Recall:

$$F_1 = \frac{2x(PrecisionxRecall)}{Precision + Recall}$$

2.7 Libraries

This research would not be possible without several Python libraries, especially machine learning and deep learning libraries such as Scikit-learn, Keras, Tensorflow and Gensim; which are why we chose Python for the development of models in the first place.

- **Sci-kit learn** is a free and open source machine learning library for Python and used extensively for the research in this area. It offers many tools and APIs for a variety of tasks, including classification. We used Sci-kit learn for preprocessing as well as many classifiers as baselines, such SVM and Naive Bayes.
- **Keras** is a deep learning library —and it can run many other neural network libraries including **Tensorflow** which is also used in this research. We experimented with different neural network architectures from Keras; it provided us simple-to-use APIs to build them.
- **Tensorflow** by Google is another free and open source library for deep learning. It was recently open-sourced (in November 2015) and got many researchers' attention since then. Tensorflow can be used for many neural network tasks, but it was also ported to many other existing libraries such as Keras.
- **Gensim** is used for many purposes, mostly for the unsupervised topic modeling and natural language processing. We used Gensim as a file format converter for the word embeddings files; this requires reading files in one format and saving them in another, so we can use them in our models.
- **Tweepy** is a Python wrapper for Twitter API. We used it for data collection.

2.8 Summary

In this chapter, we provided background information for this research. First of all, we presented one of the most popular social media platform, Twitter —and explained some Twitter-specific terms such as *tweets, retweets* and the *social network* structure of the platform. Then, we discussed *geolocations* and possible ways of geolocation estimation including machine learning combined with natural language processing techniques, which is the basis for this research. Furthermore, we introduced the natural language processing techniques and machine learning —and deep learning models used in this research. Finally, we presented the common evaluation metrics used for Twitter geolocation estimation.

In the next chapter, we will summarize the research done so far for geolocation estimation from Twitter and other online social networks.

Chapter 3

Related Work

> 'The mystery of the beginning of all things is insoluble by us; and I for one must be content to remain an agnostic.'
>
> — Charles Darwin, Father of Theory of Evolution.

In this chapter, the past studies for geolocation detection based on online social networks are presented, mostly focusing on Twitter though. Other geolocation estimation systems such as *IP-based geolocation detection*[1] or *visual-based* systems are outside the scope of the thesis, and, therefore, excluded from this chapter.

3.1 Readily available geolocation information

Users are able to add location information in most online social networks. This usually comes in two forms: (i) location in user profile and (ii) location per post. The first one addresses the long-term location of the user (i.e., home location) and it is *usually* a city, a province, a country, or a combination of all three. The second one points the location where the post is being sent from and it *usually* gives the exact coordinates of the location. In most cases, both types of location information is exposed to third-party applications through the APIs with the user consent.

3.1.1 Location from user profile

Many users share their location in their profiles publicly. This is *usually* a free-form field with a values such as 'San Fransisco', 'SF', or 'Los Angeles, California'. These entries are often converted into a more structured location through libraries or APIs. These locations are meant to be the *home location* of users. They do not represent the location at the

[1] https://tools.keycdn.com/geo

time of the message posting. Therefore, the home location and the post location *might* be different if the user is travelling. On the other hand, the user might enter non-location entries to this field.

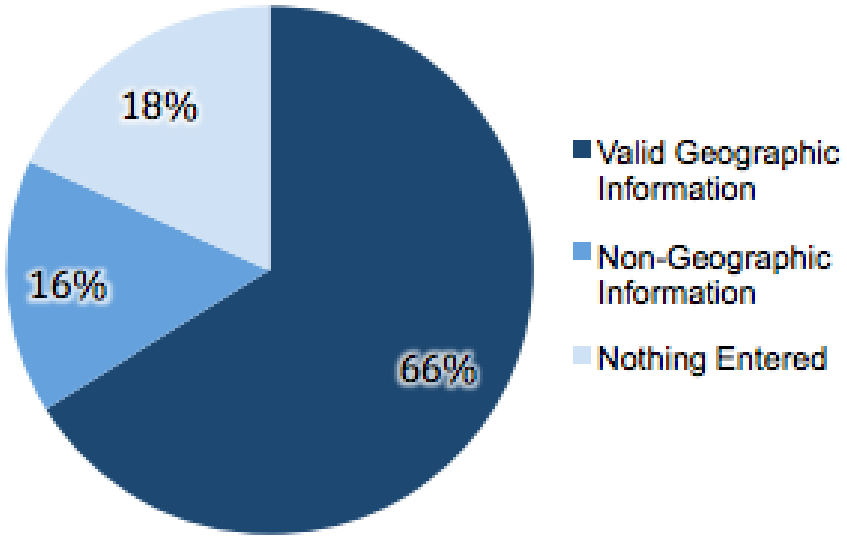

Figure 3.1: Distribution of manually entered location field data in Hecht's study

The research shows that location field is quite popular indeed, especially for Twitter. Hecht et al. (2011) studied the dynamics of the location field in Twitter user profiles for ten-thousand users and reported several findings. First of all, they concluded that almost one-third of the users did not input a valid geographic location —as seen in the figure 3.1. Furthermore, 16% of the users entered non-geographic information and the rest did not enter anything at all. On the other hand, 66% of the users entered a valid geographical location but at different granular (i.e., city, province/state, or country).

Hecht et al. (2011) also studied the geographic and non-geographic information entered in the location field. They found that non-geographic information is highly dependent of trends. By the time they were conducting their research, the trend was Justin Bieber —a popular Canadian singer and songwriter —and many used the location field to express their feelings about him (e.g., `Justin Biebers heart` or `Bieberacademy`). Some used the field to express their desire to keep their location private by entering sentences like `NON YA BISNESS!!` or `OUTTA SPACE`. Other non-geographic information entered in the location field can be seen in the table 3.1.

Information Type	Number of users
Popular Culture Reference	195
Privacy-Oriented	18
Insulting or Threatening to Reader	69
Non-Earth Location	75
Negative Emotion Towards Current Location	48
Sexual in Nature	49

Table 3.1: Number of users and types of non-geographic information entered into the location field in Twitter.

Hecht et al. (2011) also analyzed valid geographic information; this is the information that matches one to many locations. Approximately, 64% of the users entered their location at city level. The other most popular granularity was state or province. There were also users who entered their location at neighborhood level —which was quite rare but held important identities such as *Orange County* and *San Francisco Bay Area*.

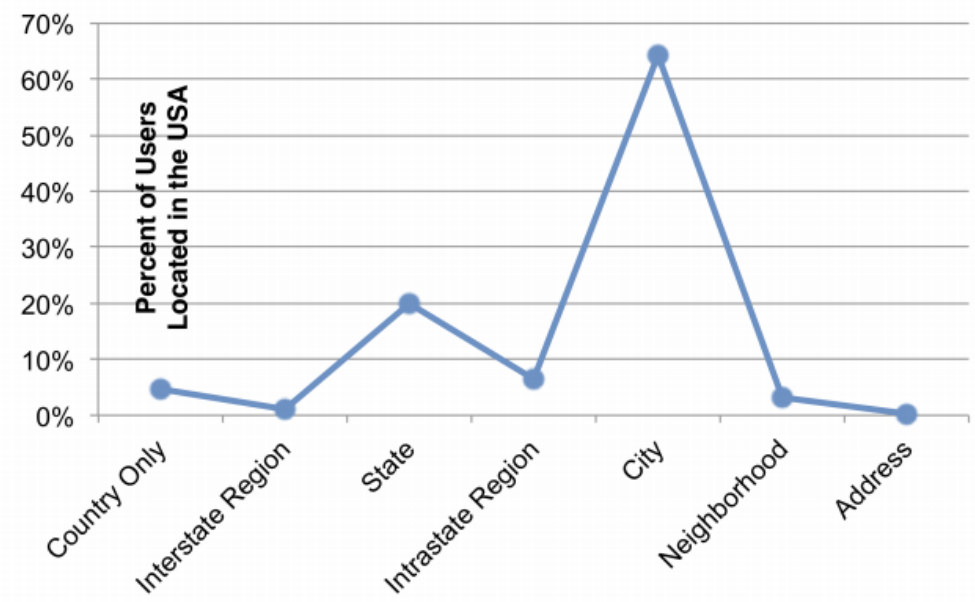

Figure 3.2: Scale of geographic information entered by users.

One problem with readily available location information in Twitter —and in other online social networks is *location ambiguity*. As mentioned before, most valid geographic information is available at city level and often there is no any other information to narrow down this information. However, there are many cities all over the world with the same name (e.g., Ottawa, Illinois vs. Ottawa, Kansas vs. Ottawa, Ontario, Canada). Therefore, the true location of users with only city-level information might not known.

3.1.2 Location per posting

Twitter users have the option to tag their tweets with a location, since 2009. This option is disabled by default so users have to opt-in if they want to send a tweet with a location. The location can be a city, state/province, country, a place name (e.g., restaurant, gym), or the combination of them.

Only original tweets can be geotagged. Retweets are not considered original, so they do not contain the location information even if the original tweet is geotagged. If the retweet is shared by creating another tweet by the user, it can be geotagged.

Studies confirm that geotagging tweets is not quite popular yet. Sloan et al. (2013) demonstrated that only 0.85% of the tweets were geotagged among 113 million tweets. Likewise, Prasetyo et al. (2016) showed that only 2.6% tweets were geotagged among roughly 40 million tweets.

Even though the percentage of the geotagged tweets is small, it still means that a lot of tweets have location information. Moreover, the geotagged tweets solve the location ambiguity problem mentioned in the previous section, as Twitter does not allow users to select a *city-only* location in tweet geotagging. All the geotagged tweets come with city and/or province/state, or a country.

3.2 Geolocation based on social network structure

The connection between the proximity and the friendship has been studied by sociologists over many years. The geography and the social context people are in mainly determines the people to contact with. Many studies concluded that there is no direct relationship between the distance and likelihood of friendship. Instead, when the distance between people increases, they are less likely to be friends. This phenomenon is easy to explain: the probability of meeting with one another decreases when the distance increases. Surprisingly, several studies verified that this seems to be a valid point for social networks too. Often, people tend to be *friends* with each other in online social networks if they are geographically close to each other. Therefore, users' social network might give strong clues about their location.

Many recent studies have focused on the proximity and friendship on social media. Liben-Nowell et al. (2005) studied several aspects of geographical basis of friendship by using the US LiveJournal users with their network and cities. Their research had several outcomes. Most importantly, they found out that likelihood relationship inversely proportional to distance. On the other hand, they show that one third of friendships are geography independent. Likewise, the analysis of Gilbert et al. (2008) about MySpace users also agrees with Liben-Nowell et al. (2005).

Backstrom et al. (2010) from Facebook worked on the problem of geolocation detection by using only the social graph of Facebook users. Their algorithm is quite straightforward. They calculated the distance between each pair of friends as an initial step. Next, they measured the probability of friendship at a certain distance. This was calculated as the

ratio between the *number of friends* divided by the *number of friends at that distance*. They concluded that the probability of friendship is inversely proportional to distance —as seen in the figure 3.3 —, especially at medium to long ranges.

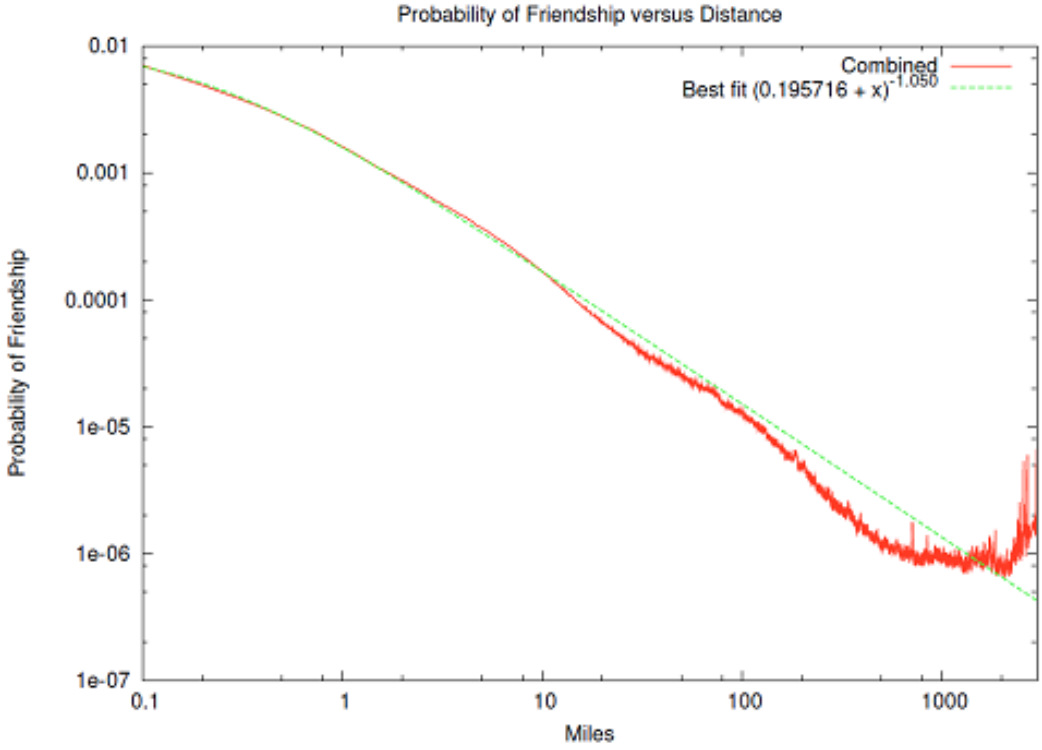

Figure 3.3: Probability of friendship versus distance in Backstrom et al. (2010)'s study.

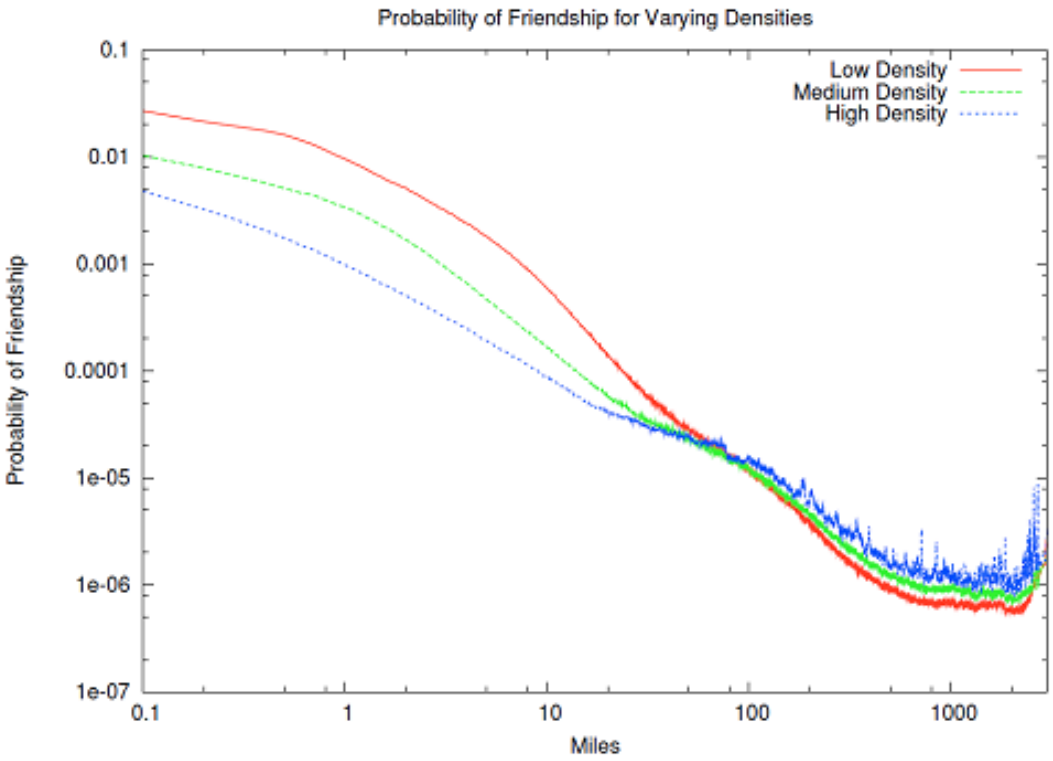

Figure 3.4: Probability of friendship for varying densities in Backstrom et al. (2010)'s study.

More interestingly, Backstrom et al. (2010) categorized users based on the population density of the region they live in. The result is quite interesting since they found that the probability of friendship differs for each user group. The users living in dense areas are more likely to have friends living far away from them, whereas the users living in rural areas to have friends in short distances. A similar outcome was also confirmed by Gilbert et al. (2008). They studied the behavioral differences of MySpace users in both urban and rural areas by categorizing users' relationships based on the communication frequency. They showed that urban users' social network is more geographically distributed than rural users' social network.

The primary question this line of research intended to answer is '`Can the location of a user be known if the location of his/her friends are given¿`. Overall, the results are quite promising, as they guessed the physical location of 69.1% of the users with sixteen or more located friends within the radius of twenty-five miles.

This paper was criticized by another paper that followed the same methodology (relying on users' social network structure alone) for geolocation estimation. Rout et al. (2013)'s main criticism was Backstrom et al. (2010)'s assumption that all the users have the same distribution of friends globally in terms of the distance. Rout et al. (2013)'s dataset included only UK Twitter users. On the other hand, Backstrom et al. (2010) created their distribution model based on the US users. The US is quite larger than the UK in size and

distribution of friends could be quite different in the two countries. The cities in the UK are usually a lot closer to each other than US cities. Therefore, one can easily assume that proximity of friends in the UK would be a lot less than in the US. Another problem they point out is that Backstrom et al. (2010) does not account for the density of people in a region. They hypothesize that sparsely populated regions give more ideas about users' location than the densely populated ones. This is because users tend to have friends in large settlements, since many people live in there. Hence, the probability of having a friend in a large city is quite high. Therefore, friends in smaller settlements give more ideas about location.

Rout et al. (2013) used both simple and complex features in their work; probably the simplest feature is the *friendship count* —which we also used in our work. *Friendship count* calculates the *number of friends* per city —and selects the top result as the location of the users. Nonetheless, they faced an issue with this approach: some users did not have any friends living in the same city with them. In this case, they assigned them to the nearest city and achieved 55.18% in @ACC161. Furthermore, they also used complex features such as *city population bins*, *triads* and *reciprocated friendships*. First, they divided cities by population and placed them into *bins* (a city with population of 150 Twitter users can be placed into *bin 15*). Then, the probability of a user living in a certain location was measured by the number of followers and the number of bins the location was placed. Second, a *triad* means users will have common friends in their network (i.e., the user a has friends b and c, and b and c are also friends.) Finally, *Reciprocated friendships* uses the bidirectional relationship (i.e., both parties follow each other). Combining these three features, they achieved the highest result in @ACC161 —which is 69.03%.

There are several problems with the geolocation detection by social network structure or *you are where your friends are*. First and foremost, immediate shift of a location may be quite slow to materialize in users' network. To demonstrate, a person moves from his home city to another city for a new job will need to have some time to make new friends there in order to balance the friends in the previous location. Furthermore, some users' social network is quite limited (i.e., they have very few friends) and this makes it quite difficult to estimate an approximate location. Because of these problems, researchers looked for alternative ways (e.g., content based geolocation detection) to predict a location or combined network-based geolocation systems with the other methods (i.e., hybrid methods). Those are explained in the next sections.

3.3 Content-based geolocation detection

A lot of research has been conducted to study the geographical scope of the online content over the last decade. Researchers analyzed the textual content of a variety of web contents including blogs, websites, logs, and web users. These efforts can be classified into two groups, based on the methods applied: *content analysis with external knowledge bases* and *content analysis with probabilistic language models*.

3.3.1 Content analysis with external knowledge bases

Most common —and probably the least complex—technique used for geolocation estimation is using external knowledge bases. These are usually *gazetteers* which are used as a tool to identify locations in textual data.

Gazetteers are geographical dictionaries; they are used together with a map. They contain a variety of information related to a geographical entity such as a country, a state/province, a region, or a city. This information includes geographical and physical features, coordinates, social statistics and demographics. Each place in a gazetteer ca be associated with several names (e.g., 'Ontario', 'ON', 'ONT') that all refer to the same place. Many online gazetteers are available online freely. Some of those are listed in the table 3.2.

Gazetteer	**Total countries**	**Total states**	**Total cities**
GATE	465	1215	1989
GeoNames	756	129	163285

Table 3.2: Popular online gazetteers and their features

The common technique for content analysis with gazetteers is quite simple: looking up tokens of a textual document (e.g., web content, tweet etc.) in a gazetteer. If a matching is found, associate the token —or the entire document —with the location.

There are many studies for geolocation estimation with the gazetteer approach. Some of those studies were conducted by Amitay et al. (2004) and Fink et al. (2009). The common issue was location disambiguation which comes in two forms: geo/geo and geo/non-geo. Geo/geo ambiguity is the case when two distinct places have the same name. Many major cities around the world share the same names. Some of those are London, Paris, Moscow, and Berlin. According to Wikipedia, there are 24 cities named Paris, 3 cities named London and Berlin, and only one other city with the name Moscow in the USA. The number of neighborhoods and municipalities with those names is large. Geo/non-geo ambiguity, on the other hand, is the case when the place name has another non-geographical meaning. Common examples are Reading (England), Mobile (Alabama), Of (Turkey) and To (Myanmar). Studies solved the location disambiguation in different ways.

Amitay et al. (2004) developed a system (Web-a-Where) for associating geography with the web pages by using the gazetteer approach. Essentially, the system had three main phases. Those are *spotting*, *disambiguation*, and *focus determination*. In the spotting, the entire web page is scanned to find the occurrences of names in the gazetteer. Then, disambiguation process decides if the occurrence is an actual location. Each occurrence is scanned and assigned a meaning —location or not. The confidence of the assignment is also calculated with several heuristics. To demonstrate, if a geographical entity (e.g., 'Chicago') is followed by another entity (e.g., 'Illinois'), this gives the confidence score between 0.95 to 1. In the last phase, focus determination, the knowledge from each occurrence is aggregated and a final score is calculated to estimate the geographical location of the web page. In the

best scenario, they achieved 80% accuracy to geotag individual place name occurrences and 91% accuracy for the correct focus of the web pages. Similarly, Fink et al. (2009) geolocated bloggers from their posts. Like Amitay et al. (2004), they also followed three phases —spotting, disambiguation, and focus determination. However, focus determination determined the author's location instead of blog post's location. The disambiguation process involved three steps: (i) disambiguate any name that is a geographical entity; (ii) look for the cases where an ambiguous name was selected as disambiguated name (if Maryland disambiguated as the U.S state then text 'Laurel, Maryland' disambiguated as it is since the text has 'Laurel'); (iii) disambiguate any remaining occurrences. They collected approximately 1,000 blogs by different authors whose *home location* is United States and successfully geolocated 61% of them.

The gazetteer approach was also used by Mahmud et al. (2012) for geolocation estimation of Twitter users. They used three different classifiers —each trained from different sources in users' tweets, namely *places names*, (all the city and state names) *hashtags* (all the words starting with #) and words (all the non stop words). The best recall performance was achieved with the *place names*, —which is 0.54%.

There are two important key elements of success in geolocation estimation with gazetteers. First, spelling correction in preprocessing step is crucial as the online documents contains a lot of misspellings —and location names are often misspelled (e.g., *' San Francesco'* or *'San Fransico'* for *'San Francisco'*). Careful correction of misspellings is required for the gazetteer look up. More importantly, selection of right gazetteer plays an important role. One must be sure that the gazetteer of choice includes all the locations that the dataset includes.

3.3.2 Content analysis with probabilistic language models

The relationship between the geography and language —and the lexical variations has been well studied by linguists, sociolinguists and dialectologists since at least the early part of the twentieth century. The work of Wolfram and Schilling (2015) points out many lexical differences in the US for the American English. Similarly, the 2003 Harvard Dialect survey done by Bert Vaux and Scott Golder shows the lexical differences for carbonated drink (*soda, pop, cola*) across the US.

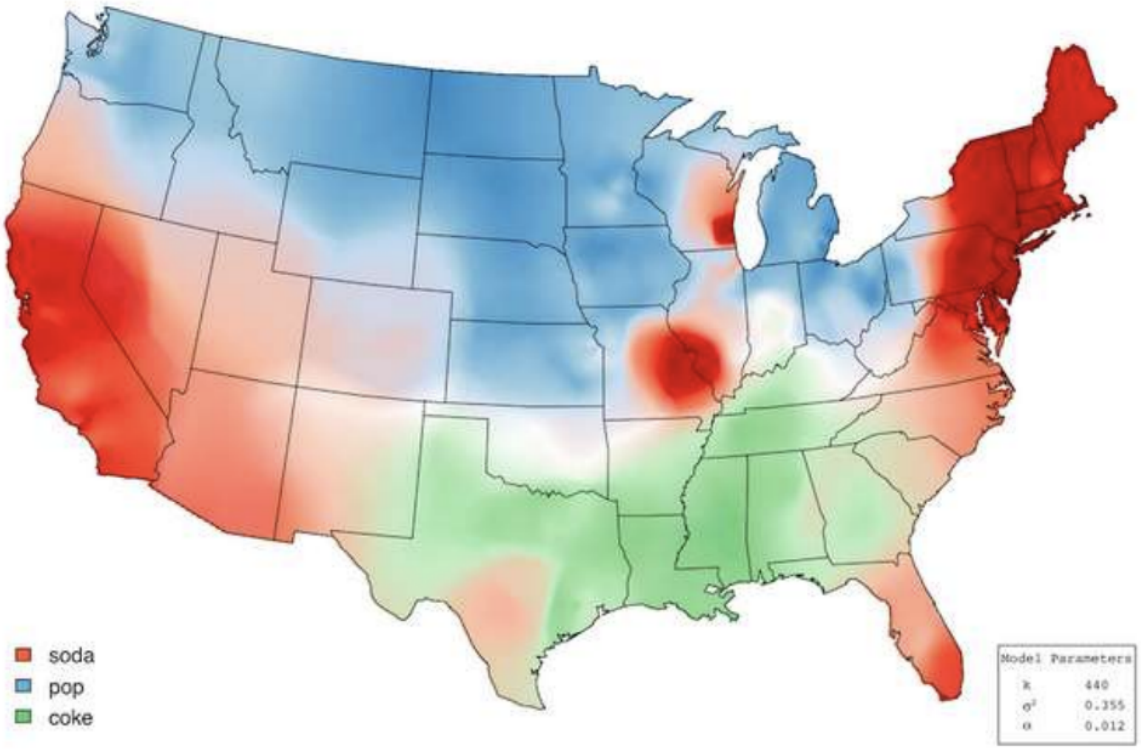

Figure 3.5: Heat map for the distribution of soda, pop and coke in the U.S.

The lexical differences are also observable in Twitter. Users in different regions have different word choices. To demonstrate, users in New York would talk about New York Rangers, whereas users in Chicago would talk about Chicago Blackhawks in Twitter. Similarly, some words are used excessively in some regions (e.g., *howdy* for Texas), but not in others.

Over the years, content-based methods using probabilistic models were proposed. Essentially, these studies are divided into two categories: *geographical topic models* and *word-based models.*

Geographical Topic Models

Topic modelling is a statistical tool to extract 'abstract' topics from documents. The most common model used for topic modelling is *Latent Dirichlet allocation* or *LDA*; it is utilized

for classifying documents by topics. Over the early period of geolocation inference of Twitter users —and tweets—topics models were extended to analyze text with geolocation information. These models are called *geographical topic models* and used to discover topics by locations.

The work of Eisenstein et al. (2010) is the pioneer of geospatial topic modelling. They studied the lexical variations across the geographical areas in the US with tweets alone, which in turn enables the prediction of user's location. More specifically, their study has multiple capabilities, including: (i) evaluation of lexical differences by topic and geography; (ii) dividing geographic regions into linguistic communities; (iii) estimating geolocation of users. In their model, they generate the base or *'pure'* topics such as sports or entertainment, by using LDA. Next, regional variants of those topics are generated. This process is called *cascading* topic model. The cascading topic model generates text from a chain of variables. The chain starts with *priors* and base topics which then might be *corrupted*. To demonstrate, if the base topic is **Football**, the corrupted version —or subtopic or a lexical variant —would be Football-Boston. The corrupted topics are generated with the latent variable r that represents the geographical region of the user. Some of the example topics and subtopics can seen in the figure 3.6.

	"basketball"	**"popular music"**	**"daily life"**	**"emoticons"**	**"chit chat"**
	PISTONS KOBE LAKERS game DUKE NBA CAVS STUCKEY JETS KNICKS	album music beats artist video #LAKERS ITUNES tour produced vol	tonight shop weekend getting going chilling ready discount waiting iam	:) haha :d :(;) :p xd :/ hahaha hahah	lol smh jk yea wyd coo ima wassup somethin jp
Boston	CELTICS victory BOSTON CHARLOTTE	playing daughter PEARL alive war comp	BOSTON	;p gna loveee	*ese* exam suttin sippin
N. California	THUNDER KINGS GIANTS pimp trees clap	SIMON dl mountain seee	6am OAKLAND	*pues* hella koo SAN fckn	hella flirt hut iono OAKLAND
New York	NETS KNICKS	BRONX	iam cab	oww	wasssup nm
Los Angeles	#KOBE #LAKERS AUSTIN	#LAKERS load HOLLYWOOD imm MICKEY TUPAC	omw tacos hr HOLLYWOOD	af *papi* raining th bomb coo HOLLYWOOD	wyd coo af *nada* tacos messin fasho bomb
Lake Erie	CAVS CLEVELAND OHIO BUCKS od COLUMBUS	premiere prod joint TORONTO onto designer CANADA village burr	stink CHIPOTLE tipsy	;d blvd BIEBER hve OHIO	foul WIZ salty excuses lames officer lastnight

Figure 3.6: Base topics and their geographical variants based on four different regions (Boston, N.California, Los Angeles, Lake Eric) in the Eisenstein's study.

Eisenstein et al. (2010) achieved 58% for the region-level accuracy (4 regions in total) and 24% for state-level (49 states) accuracy which seems very low. The paper was also criticized because of the dataset as it does not cover the country (US) evenly. Most users

are in the western or eastern border and very few in the southern part of US.

In their next work, Eisenstein et al. (2011) used previously mentioned topic modelling again. Additionally, they developed Sparse Additive GEnerative mode (SAGE) which provides sparsity and simplicity in model inference. Both papers (Eisenstein et al., 2010) and (Eisenstein et al., 2011) were criticized by Zheng et al. (2018) because of the problematic preprocessing stages, that is concatenating all the tweets of a user and creating one long text, then using the first location found in the long tweet as the ground truth location of the tweet. This is questionable since users mention many locations in their tweets —not only one. Therefore, first location found by the model might not represent the location of the long tweet.

Word-based models

Word-based models are similar to geographical topic models. Instead of using topic modelling, word-based models focus on *location indicative* words. More specifically, they work in two steps:

1. Finding location-indicative words (local words).
2. Calculating word usage to predict a user's or a tweet's location.

Identifying the local words is the first step for the word-based methods. As the initial task, the *stop words* such as *'a', 'an', 'the'* are removed since they are used across all locations. Then, local words are identified by using unsupervised methods. Some of those methods were proposed by Van Laere et al. (2014) and Hecht et al. (2011).

In the next step, local words are used in order to find out a location. This step is often modelled in probabilistic manner and there are different approaches. One of the approaches was introduced by Cheng et al. (2010). Probability of a user u, location l and his/her tweets $S(u)$ is calculated by the formula below:

$$P(l|u) \propto \sum_{w \in S(u)} P(l|w)P(w) \tag{3.1}$$

Here w represents the local word and $P(w)$ is the probability of it in the entire corpus. With this method, Cheng et al. (2010) achieved 51% accuracy in ACC@161. Hecht et al. (2011) followed the same approach and utilized local terms in their model and achieved 72.71% and 30.28% for country and state level, respectively.

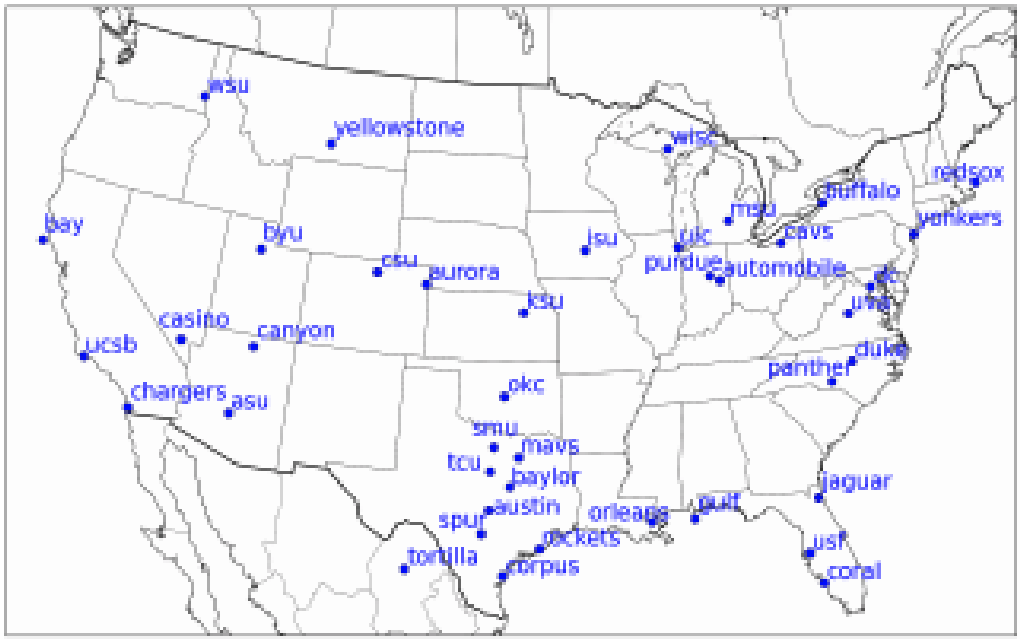

Figure 3.7: Local words by region in Cheng et al. (2010)'s dataset.

Rahimi et al. (2017) used dialectical terms dictionary for the local words. Unlike Cheng et al. (2010), they used a static dictionary (DARE or Dictionary of American Regional English) instead of identifying local words with an unsupervised approach. Their Multilayer Perceptron model with three layers (input, hidden, and output) achieved 40% in ACC@161 on the GeoText dataset.

Content-based methods achieve better results than the other methods mentioned in this section. One problem about them is the *tweet sparsity* in datasets, especially for the content-based methods relying on the *local words*. Some words do not appear enough across different locations and there are excessive number of little word distributions due to the spareness. Another common problem is the insufficient number of location-indicative words in the tweets' content. Some users do not mention location names enough in their tweets —which makes it difficult to predict a location in both user and tweet level.

3.4 Hybrid methods

Geolocation estimation of Twitter users is almost a decade-old problem. The initial approaches were *content-based methods* (i.e., relying on the user's tweets alone). Later, many researchers investigated the possibility of utilizing user's social network to infer his/her location. More recent research focused on combining these two methods (*content-based and network based*) as well as some other metadata information of users (e.g., *timezone*) to guess a location. These methods are called *hybrid methods* since they combine different sources of information for geolocation inference.

In 2016, the Workshop on Noisy User-generated Text (W-NUT) organized a shared task for the geolocation estimation. Given a training dataset, the goal was to build models for detecting the location of tweets and users, then to apply them on the provided test set. Many researchers joined in the shared task and many of them utilized *hybrid methods* for the task.

Miura et al. (2016) proposed a model for the classification of users and tweets at city-level by using neural networks. They used a variety of information including tweet content and user metadata, such as user-declared location, timezone, and user description. For their tweet-level prediction model, they created the word embeddings out of words in the tweet content, location, as well as user description. The user-level prediction model was quite similar. Tweet contents of a user were merged into together —similar to Eisenstein et al. (2010) —and formed a long text which captures all the tweets of a particular user. The other three metadata information was also used for this model. They achieved 40.91% as the median distance error of 69.50 km in tweet-level and 47.55% as the median distance error of 16.13 km in user-level.

Similar to Miura et al. (2016), Jayasinghe et al. (2016) proposed an ensemble approach for the same shared task by incorporating text, network (i.e., social graph) and metadata —which consist of five approaches:

1. **Metadata Location-to-Gazetteer**: Looks up user declared location text in GeoNames. If the text is a valid coordinate, then map it into a candidate city using GeoNames.

2. **Metadata Timezone**: Utilizes timezone information to find out candidate cities within the time zone boundary.

3. **Metadata UTC (Coordinated Universal Time)**: Converts the UTC that is assigned by Twitter into time zone which in turn used to find out candidate cities like in the previous approach.

4. **Metadata URL IP-Lookup**: Checks tweet content if any URL is shared, if it is, URLs are mapped into IPs and then countries. This approach comes from the assumption that people are likely to share the URL in their home country and it excludes websites such as Facebook and Google.

5. **Metadata Location-Based Service (LBS) Links**: Looks for the URLs from location based service such as FourSquare, Swarm etc. and extract the URLs to infer the location eventually.

Apart from the metadata approaches, they also applied label propagation methods for the social network of users. Based on Twitter interactions, they created a relationship map between the source and target users and the source user's location was inferred from his friends. Finally, they employed tweet content to cluster them geographically. At the end, they gathered results from each approach and combined the best one's. Overall, they achieved 43% accuracy in tweet-level geolocation and 52% accuracy in user-level.

There have been also attempts for applying *hybrid approaches* in other social media platforms. Liu et al. (2014) investigated the ways of suggesting geolocation for Flickr images and profiles. They find out geolocation of Flickr users by analyzing image upload patterns, user's geotagging behaviors and calculating the gap of upload times of two photos. For image geotagging, they utilize the image tags and user profile location. In another study, Ding et al. (2000) introduced an algorithm to find out the geographical scope of various web resources (e.g., web page) by using the textual content of the resources and URLs in them.

Hybrid methods outperform both content-based and social network based methods for the geolocation detection of social media users and web resources (e.g., web pages, blogs, etc.). They are much more capable of analyzing web contents to infer one's location. On the other hand, there are only a few data sets publicly available that can be used to create hybrid systems. (i.e., a few data sets contain various type of information for the target web resource). Moreover, hybrid systems require more processing power since they deal with more data. Traditional work environments might fail or be very time consuming for the hybrid systems. In this case, GPU-powered systems might be helpful since they provide more powerful processing power to deal with the excessive amount of data.

3.5 Summary

In this section, we reviewed some geolocation inference techniques by methods. Those are:

- Readily available geolocation information
- Geolocation relying on social network structure.
- Content-based geolocation.
- Hybrid methods (combines two or more different methods.)

Readily geolocation information is available in many social network platforms, mostly as part of the profile information. This is probably the easiest method to access a user's or post's location, if the information is available in the user profile publicly and accessible through APIs.

Geolocation detection by social network structure is probably the most unique technique used for geolocation inference of social media users since it is domain specific (i.e., only applicable to social networks). In its simplest form, this technique captures a user's location based on his friends' location. To demonstrate, if a user has more friends in *New York City* than *Manhattan*, then the user is probably located in NYC.

Content-based methods are common, presumably because they are applicable to many different contexts (e.g., blogs, websites, social media platforms) and textual content has a lot of clues about location. In this method, textual content (e.g., a user's tweets, a blog post or website content) is utilized to infer a location. The location might refer to the

user's home location, a location mentioned in a content, or the content's original location. Content-based methods can be divided into two categories based on the methodology: *content-analysis with external knowledge bases* and *probabilistic models*. The first one uses external knowledge bases such as gazetteers to match the locations found in a content with the gazetteer entries. In the second one, probabilistic models are used (e.g., word distribution index by location) to infer a location.

Finally, *hybrid methods* are also common for geolocation detection, especially for the Twitter users. This is because they achieve better results for different granularity levels than the individual methods. The success of hybrid method comes from the variety of information used in the models. This information includes social graph, content (e.g., tweet), and metadata information such as timezone.

Each method has some downsides. To begin with, *readily available geolocation information* has multiple problems. Considering the location field in Twitter user profile, some users do not prefer to enter any information at all and some enter non-location text since it is a free-form field. *Privacy* is the common reason for this; some users are too concerned with their location being exposed and used by third-party applications.

Methods relying on content and social graph have also issues, particularly due to the data sparsity. Some content-based systems based on tweets reported that some tweets either include very little location information or not at all. Similarly, social graph-based methods also suffer from data sparsity since some users do not have any friends in social networks or they do not have any friends within their location. Finally, hybrid methods have similar problems like content and social graph-based methods since they utilize both the content and the social network structure of users.

Although each method used different techniques to find out a user's or tweet's location, they are often using the same NLP techniques, such as tokenization, part-of-speech tagging, and bag-of-words or word embedding representations.

Chapter 4

Dataset

‘Success is not final, failure is not fatal: it is the courage to continue that counts.‘

— Winston Churchill, Former British Prime Minister.

4.1 Overview of available datasets

Geolocation detection in Twitter is a mature problem; therfore, many datasets were introduced over the decade. They are different in size, scope, as well as the method used for collecting them. In this section, we outline the most popular datasets used for the location estimation problem in Twitter.

GeoText (Eisenstein et al., 2010) is a small dataset that was collected in the first week of March 2010. It includes 377,616 tweets from 9,475 users who sent at least 20 tweets. All the tweets are tagged with a location (latitude and longitude pairs) and sent from a mobile client (Twitter for iOS or Twitter for Android). A document in this dataset is the concatenation of all tweets of a user. Furthermore, the dataset covers 48 US states and 1 district (District of Columbia). *GeoText* was criticized by some research as it has a data sparsity problem. It mostly covers the east part of the US but there are very few users in both southern and western US.

Twitter-US or *UTGEO2011* (Roller et al., 2012) is a medium-sized dataset collected between September 4th and November 29th 2011. The goal of the dataset was to solve the data sparsity problem in the *GeoText*. Also, Roller et al. (2012) applied several filters to avoid spammers and robots. They kept only the users following 5 to 1,000 people and users who did not tweet more 1,000 times in the last three months period. The dataset contains 449,694 users and 390 million tweets and covers the same area as *GeoText*.

Figure 4.1: Coverage area of GeoText, Twitter-US, and Twitter-WORLD, respectively

Twitter-WORLD (Han et al., 2012) is a large dataset which covers the entire globe. In total, it contains 1.39 million users and 12 million tweets. Although the dataset covers the world, it only contains English tweets. Similarly, Hecht et al. (2011) collected an enormous-sized dataset between April 18 to May 28, 2010 which covers the entire globe. In their original work, they collected 62 million tweets which roughly makes it 4% of the total number of tweets sent in the collection period. However, they filtered out non-English tweets and almost half of the tweets were removed.

Another quite popular dataset was shared by W-NUT (Workshop on Noisy Users-Generated Text) and used by many studies for the workshop. The dataset was divided into subsets for training, development, and test set. The training set includes 1 million users and almost 13 million tweets and both the development and the test set consist of 10 thousand users. Similar to *Twitter-WORLD* and the dataset of Hecht et al. (2011), this dataset also covers the entire globe, with 3,362 cities.

All the datasets mentioned so far have very limited data variables (i.e., they have only user and tweet information). For this reason, only content-based models can be developed on them. On the other hand, there have been also efforts to create datasets for developing other types of location estimation systems (i.e., network based).

Twitter-UK (Rout et al., 2013) is a medium-sized dataset and was collected in November 2011. As the name suggests, it only contains U.K Twitter users. The dataset was collected with the goal of geolocating users based on their social ties. For this reason, it contains users with their friendship distances —and tweets. In total, it contains over 200,000 users.

Dataset	**Data**	**Coverage**	**Total users**	**Total tweets**
GeoText	Tweets	US-only	9,475	377,616
Twitter-US	Tweets	US-only	449,694	390M
Twitter-WORLD	Tweets	World	1.39M	12M
W-NUT	Tweets	World	1M	over 13M
Hecht et al. (2011)	Tweets	World	Unknown	32M
Twitter-UK	Tweets and network	UK-only	Unknown	200,000

Table 4.1: Popular datasets and their features.

We did not use any of the datasets above —and many others not mentioned here, for several reasons. First and foremost, those datasets are quite outdated. They were collected around 2010 and 2011 and there have been fundamental changes in Twitter since that time. The most significant change is the character limit, as Twitter doubled the character limit per tweet for most languages, including English. According the Twitter-Data —the Twitter account that shares some insights about Twitter —there is a notable increase in users writing out full words and phrases. In addition to that, there have been more replies to tweets since doubling its character limit.

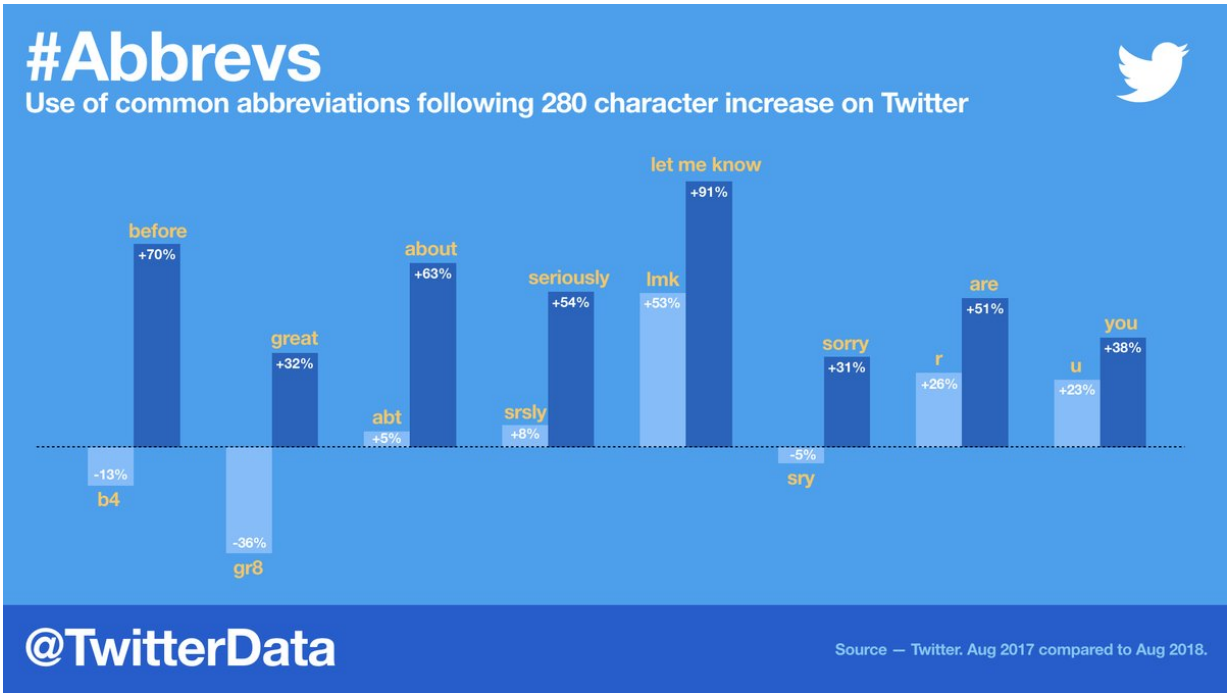

Figure 4.2: Use of common abbreviations following 280 character increase on Twitter

The datasets above include tweets written with 140-character limit, which forces many users to use abbreviations shown in the figure. This decreases the semantic understanding of the tweets —unless they are carefully preprocessed. However, the character limit expand creates an opportunity to understand tweets better, thus creating a brand-new dataset might be a good idea.

Another reason why we created our own dataset is because of the linguistic dynamics in Twitter. According to the research by the Centre for Corpus Research at the University of Birmingham, 'Twitter language' is quite dynamic, meaning that Twitter words are evolving over time. The research center analyzed almost 9 billion words from the 980 million tweets posted across the US between 2013 and 2014 and concluded that there are 54 newly detected words on Twitter that are used in everyday speech by many users. We think that same fashion might occur in Canada, since both countries speak English primarily. Therefore, if we train our models with the old datasets, it might fail to identify the new language used today in Twitter which might result in a failure to geolocate users.

Finally, although several existing datasets include Canadian users such as `Twitter-WORLD`

and W-NUT, they include users by city.On the other hand, we classify users by Census Metropolitan Areas (CMA) and Census Agglomerations (CA) which are the two census divisions (i.e., a type of geographical unit) in Canada; we use census divisions since they cover more people.

4.2 Twitter developer program, APIs, and challenges

The Twitter developer program provides many APIs, tools, and resources that enable developers to build products on top of Twitter data, create advertisements on the platform, as well as several publisher tools to integrate websites with Twitter. Specifically, the APIs are the integral part of the developer program; they are used for different purposes such as academic research, monitoring and advertisement.

There are two types of Twitter APIs: data and ads. The ads API helps businesses to create, manage, and monitor ad campaigns *programatically*. Furthermore, the data APIs are the way to access to variety of Twitter-related data (e.g., tweets, social connections). The data APIs are offered in three levels: standard (free), premium (pay as you go), and enterprise (paid subscription). The standard APIs —also the most popular —are very limited in data output and requests, whereas premium and enterprise APIs provides more data output and less strict rate limits.

The APIs are further classified into two types based on their design, access methods and data type:

- **RESTful APIs**: Those are the APIs based on the REST architecture. [1] They are used to access to readily-available data (e.g., all tweets of a user).
- **Streaming APIs**: Streaming APIs provides steady stream of public information based on the query (e.g., latest tweets from Toronto). When the request is made, the API provides continuous stream of updates from Twitter.

In this work, we used both APIs —Restful and Streaming. This is explained in the next section.

There are many challenges of Twitter APIs. First and foremost, the platform introduced a new developer requirements as of July 2018 [2]. An important part of it is the new developer account application process which requires anyone who wants to access Twitter's data to apply for a developer account. Developers are required to provide details about their intent-of-use of Twitter data for their application. Once they are applied, the Twitter-dev team checks it for any violations of the platform's policies and approve it otherwise. Furthermore, there are strict API limits (also rate limiting) for standard APIs. Each registered application can make certain number of requests per rate limit window. Rate

[1] https://en.wikipedia.org/wiki/Representational$_s tate_t ransfer$

[2] https://blog.twitter.com/developer/en$_u s/topics/tools/2018/new-developer-requirements-to-protect-our-platform.html$

limit windows are 15 minutes, 3 hours and 24 hours. 15 minutes rate limit window is used by GET endpoints, whereas the other two are used by POST endpoints. Maximum number of allocated requests per rate limit window (e.g., 15 minutes) depends on authentication type and method[3]. Basically, there are two types of authentication: user authentication and application authentication. User authentication (also 'Sign in using Twitter') requires users' consent for the application —and usually gives additional number of requests per rate limit window. On the other hand, the entire application is authenticated in the latter one —and rate limits are determined globally for the application and gives less request room.

4.3 Data collection

Data collection was an important part of this research. We collected two datasets for this work —and each targets different problems.

The first dataset is called `Twitter-CAD-Users` and was collected between December 2018 and March 2019. It includes only Canadian users —and their social network information. The goal is to estimate *home location* (i.e., long-term residential address) of the users with it. Our methodology to collect this dataset is quite straightforward:

1. We send location-related queries (e.g., Toronto Raptors) for each class (e.g., Toronto) to collect tweets by using Twitter's Streaming API. The result of the query is an array of tweets matching to the query.

2. From the tweets, we get the metadata information of the tweet authors (i.e., users). This gives details about the users including their user ID, name, and user-declared-location and many other information.

3. Finally, we check if the user's location —user-declared-location —is matching to one of our classes (i.e., CMAs and CAs). If it does, we keep the user. Therefore, the ground-truth for this dataset is user-declared-location in the Twitter profile.

During the dataset collection, we eliminated many users since their user-declared-location did not match any of our classes. Although many users entered some kind of information in the location field, many of them were not valid locations. Some users expressed their feelings and desires in the field, while others put non-related information such as hashtags for special topics. Some users preferred not to put any information at all. The reason for this —as we previously mentioned —might be privacy.

`Twitter-CAD-Users` also contains information about the social network —or social graph —of the users although it is very limited. We collected up to one hundred most recent friends and followers of the users —including their location. Originally, we desired to collect the complete social graph of users, but this seems quite difficult as Twitter APIs

[3]https://developer.twitter.com/en/docs/basics/rate-limits

impose very strict rate limits for social-graph information; that is, maximum 15 requests for 15 minutes request windows.

The second dataset —`Twitter-CAD-Tweets` —was collected during the second week of March 2019 for tweet-level location estimation. The dataset contains geotagged tweets —tweets attached with geolocation information —and tweet author information. In Twitter, location information for tweets falls into two categories:

- Tweets with 'Point' coordinate
- Tweets with 'Place' coordinate

Tweets with 'Point' coordinate come from GPS-enabled devices (e.g., smartphones, tablets, laptop computers) and they have the exact location of the tweet in latitude and longitude pairs (or GPS coordinates). The location information does not include any coxtextual information; that is the information GPS coordinates being referenced such as a city or a province. Other other hand, Tweets with 'Place' coordinate refers to name of places (e.g., coffee shops, restaurants). Place names only can be attached with Twitter mobile clients (Twitter for iOS and Twitter for Android).

We created `Twitter-CAD-Tweets` tweets with the point coordinates. The collection methodology is similar to *Twitter-CAD-Tweets*:

1. We gathered boundaries of census metropolitan areas and census agglomeration that we label with latitude and longitude pairs.
2. We provided the GPS coordinates to Twitter's Streaming API and fetched the tweets.

`Twitter-CAD-Tweets` contains over forty-one thousand tweets, so it is a minor dataset. This is partly because we collected it in a very short period of time (a week or so). However, the main reason is that many users do not prefer to geotag their tweets; this makes it very difficult to collect geotagged tweets.

One important point is that a tweet location is not necessarily a user's home location; it might be different. Some users in `Twitter-CAD-Tweets` have different user-declared-location than the location of most of their tweets. This can be explained in several ways: they were either traveling at the time they were tweeting, or they forgot to update their location field when they moved, or they use a different home location for no reason.

Both datasets had a very careful preprocessing step —which we explain in detail in the next chapter. Next, we talk about classes (i.e., census divisions) in our work.

For both datasets, we only collected the public tweets. Therefore, privacy is not a concern.

4.4 Census divisions

Canada consists of administrative divisions; an administrative division is a part of the country for the purpose of administration. Each administrative division has a certain degree of autonomy —and manage itself with a local government. In Canada, administrative divisions are defined by Statistics Canada —the government's agency that produces statistics for the country [4]. They are used for many purposes including census, which happens every five-years.

The administrative divisions of Canada exist in four different levels: *top-level* (i.e., provinces and territories), *second-level* (i.e., census divisions), *third-level* (i.e., census subdivisions) and finally *fourth-level* (i.e., dissemination areas).

Figure 4.3: Map of Canada with 10 provinces and 3 territories. Source: Wikipedia

[4] https://en.wikipedia.org/wiki/List$_of_census_metropolitan_areas_and_agglomerations_in_Canada$

Top-level divisions are sub-governments within the geolocation areas of Canada; they rule the certain part of the country under the constitution. There are two types of top-level divisions: provinces and territories. Provinces receives their power from the Constitution, and they use it for their own right, but the power is delegated to territories by Parliament of Canada. There are ten provinces (Alberta, British Columbia, Manitoba, New Brunswick, Newfoundland and Labrador, Nova Scotia, Ontario, Prince Edward Island, Quebec and Saskatchewan) and three territories (Northwest Territories, Yukon and Nunavut) in Canada. Moreover, the top-level divisions (i.e., provinces and territories) are composed of the second, third and fourth level divisions.

Second-level divisions are census divisions that are formed by provincial and territorial governments. They are basically multiple municipalities combined together for regional planing, as well as managing some services (e.g., bus service). The census divisions are further divided into *third-level* (i.e., census subdivisions) and *fourth-level* (i.e., dissemination areas). Dissemination areas are the smallest administration unit in the country.

In Canada, multiple administrative divisions also create geographic units (also specifically defined geographic units) such as census metropolitan areas (CMA) and census agglomerations (CA). Both consists of several census divisions. According to Statistics Canada, a *census metropolitan area* consists of one or multiple neighbouring municipalities (census subdivisions) to comprise a large urban area with a population of at least 100,000 where core (i.e. the center of the division also the most populated area) has at least half of it. Those municipalities around the core closely integrated to it. The *census agglomerations* are smaller version of CMAs: urban areas that have at least 10,000 people.

The borders of CMAs and CAs might extend the provincial or territorial borders. Out of 152 CMAs and CAs in Canada, there are only three CMAs that are inter provincial (i.e., contains municipalities in more than one province). Those are Ottawa-Gatineau (Ontario/Quebec), Hawkesbury-Grenville (Ontario/Quebec) and Lloydminster (Alberta/ Saskatchewan). Our dataset only includes the Ottawa-Gatineau CMA.

In Canada, the population is not evenly distributed; most people live in the CMAs near the southern border. The most populated CMAs are Toronto (City of Toronto, Mississauga, Brampton, Markham, Vaughan), Montreal (City of Montreal, Laval, Longueuil), Vancouver (City of Vancouver, Surrey, Burnaby) and Ottawa-Gatineau (City of Ottawa and Gatineau). According to Statistics Canada, 7 in 10 Canadians (69.9%) are living in a census metropolitan area [5] which encouraged us to classify by CMAs —and CAs instead of cities or municipalities, since CMAs and CAs cover a lot more people.

[5] https://www150.statcan.gc.ca/n1/pub/91-214-x/2015000/section01-eng.htm

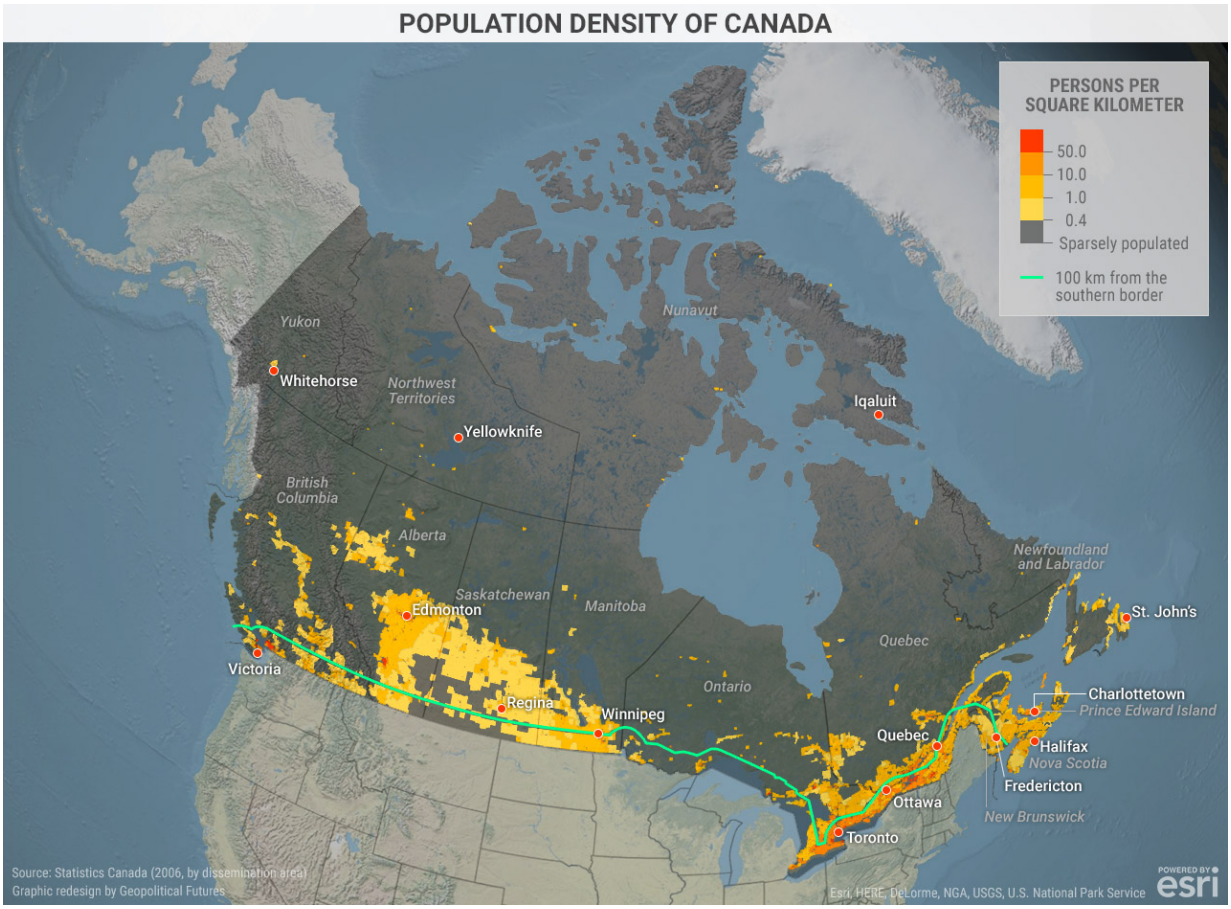

Figure 4.4: Population density of Canada

Our dataset includes users from CMAs and CAs with the population of at least 100,000[6]. According to this criterion, there are forty-one CMAs and CAs in the country. Out of the forty-one divisions, we have thirty-five CMAs and six CAs. The largest subdivision is Toronto; it has 5,928,040 million people and the smallest one is Red Deer in Alberta with a population of 100,418.

4.5 User and tweet statistics

4.5.1 Sex

Twitter does not ask for sex information as part of profile. For this reason, researchers used alternative ways for detecting the sex of the users. These methods include hybrid methods (Vicente et al., 2019) in which multiple sources of information are used, such as profile picture, screen name, name, description, and the tweets. Some of the other methods

[6]We did not want to include all the CMAs/CAs in the country which are over 150. This could have decreased the accuracy drastically in the study.

focus on writing styles differences (Argamon et al., 2003) between males and females, as well on the topics detected from the tweets (Burger et al., 2011).

Probably the easiest way to identify sex of Twitter users is using some kind of *gender APIs* which guesses the sex from the name of the user. This method might not be quite as accurate as the sophisticated methods mentioned earlier, since it solely relies on name of users, which is a free form field on Twitter. Some user often put there emojis, personal thoughts, etc. However, the gender APIs still seem to be working okay.

Our gender identification method is based on three steps:

1. Extract all the users with real human names.
2. Use the gender-api.com to predict the sex.
3. Manually assign, if the sex could not be assigned by the API.

In the first step, we used a Python library *probablepeople* to get all the users with a real human name and ignored all the other types of accounts (e.g., organization, advertisement) since they do not have sex. In the next step, we used an API (gender-api.com) to fetch the sex based on person's name solely. However, we faced several problems in both steps. First of all, we realized that some users do not use a name; instead they enter a variety of different things to the free-form name field (e.g., emoticons, special characters). Besides, some users change their real name by removing characters from it, changing some characters (e.g., @ for *a*) or using a nickname. For this reason, we failed to detect the account type (e.g., person) and assigning a gender. Another problem was people's inability to assign a sex for non-English names. The library failed to detect many non-English (e.g., Arabic, Spanish, Turkish) person names. For these users, we manually assigned a sex based on their profile picture. The results is shown in the figure 4.5.

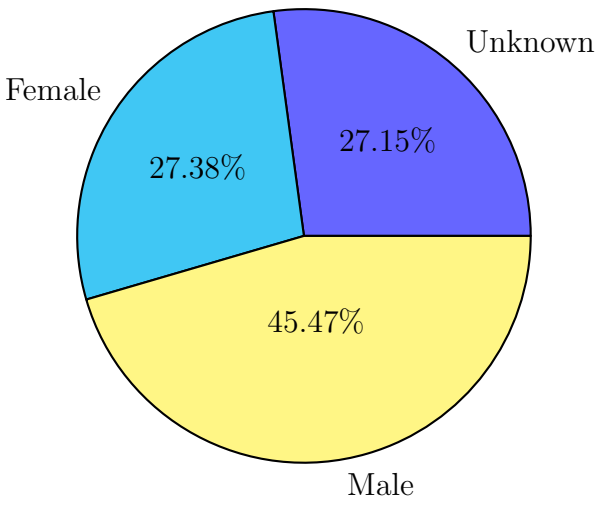

Figure 4.5: Sex of users in Twitter-CAD-Users

The majority of the users in the dataset are male with the 45.47%. Female users are 27.38%. For the remainder of 27.15% of the users we were not able to assign a sex because of their account types[7].

4.5.2 Tweet source

Twitter has multiple clients to send statuses (i.e., tweets). Those are Twitter Web Client (twitter.com), Twitter for iPhone and Twitter for Android. There are also third-party applications to send tweets such as Instagram.

For Twitter-CAD-Tweets datasets (i.e., the dataset containing geotagged tweets), we also collected Twitter clients (i.e., tweet sources) for each tweet, in order to see which Twitter client is preferred more. The result is in the figure 4.6.

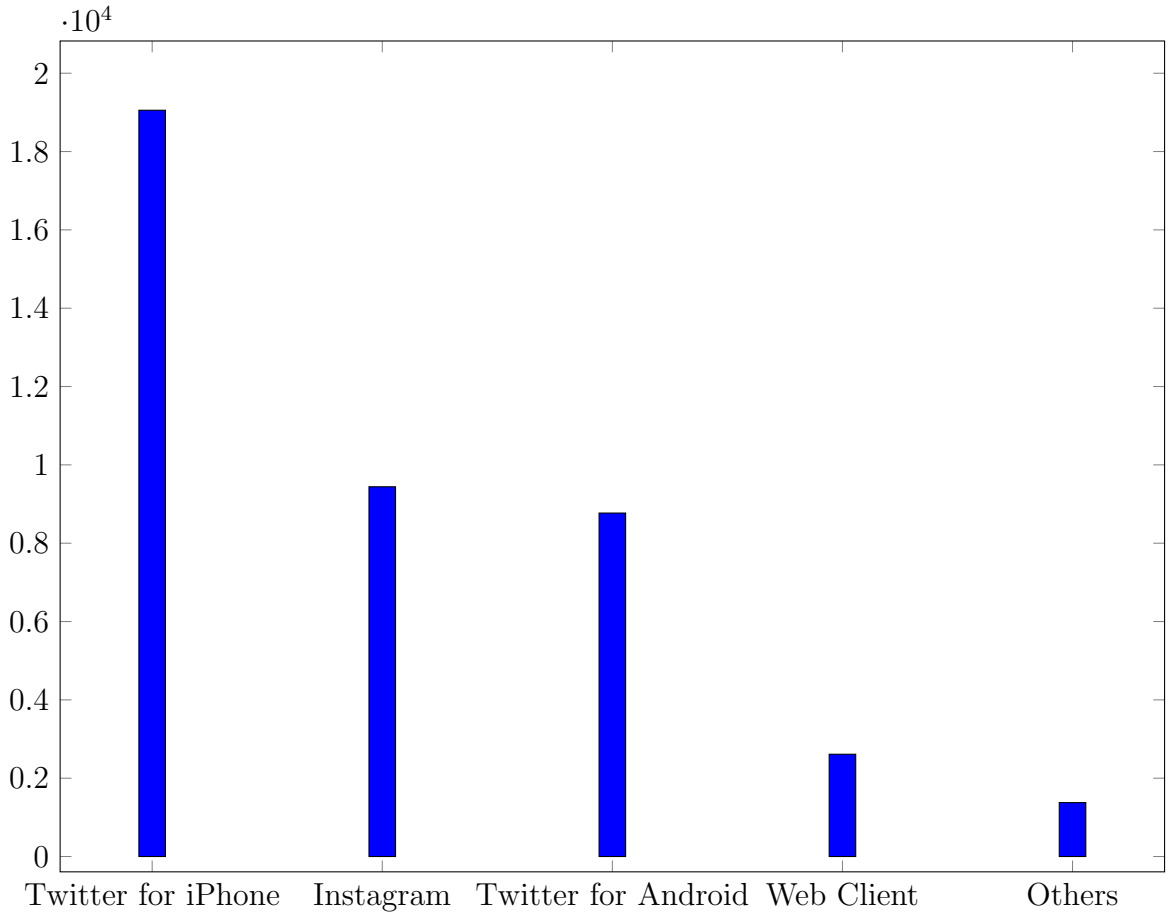

Figure 4.6: Top Twitter clients used for sending tweets.

In the dataset, the most used client is `Twitter for iPhone` (19,055) by far; then

[7]We are aware that gender/sex is a complex issue: it could be more than binary. In our dataset, the unknown category could include these non-binary cases.

surprisingly a third-party client, `Instagram` (9,441), is in the second place and `Twitter for Android` is in third place (8,770). Only 2,612 of the tweets sent from `Twitter Web Client`. This proves that most geotagged tweets are from mobile devices (i.e., smartphones or tablets). There are a number of other third-party clients in `Others`. Some of these are: Foursquare (166), Tweetbot for ios (151), `Hootsuite Inc.` (117) and Untappd (87).

4.5.3 Verified accounts

Twitter verified accounts are given a blue badge by Twitter; showing that the account has an public interest. These accounts are *usually* owned by famous people (i.e., news reporters, entertainers, sportspeople) or large organizations. In the Twitter-CAD-Users dataset, we have the information of verified accounts (i.e., if the account is verified or not). Out of 420 users, we have only 24 verified accounts.

Most verified accounts in the dataset were owned by reporters working for the Canadian Broadcasting Corporation (CBC) and the CTV Television Network. Also, there were accounts owned by government organizations (e.g., Tourism Kelowna). There were also a few individuals; they were mostly famous bloggers.

4.5.4 Number of tweets, followers, and followees

We collected the number of tweets, followers and followees each user has for the `Twitter -CAD-Users` dataset. The average number of tweets sent is `16,540`. The user with the maximum amount of tweets has `456,493` tweets.

Surprisingly, users seem to have more followers than friends. The average number of friends is `1,412` and the average number of followers is `2,979`. The users with most followers has `186,101` followers in total (it is `Montreal.AI` —a Montreal-based Artificial Intelligence company). The account with most friends is `CISN Country 103.9` —an Edmonton-based radio station —and it has `55,922` friends in total.

4.6 Summary

This chapter started with the overview of the available datasets for the geolocation estimation problem in Twitter —and why we could not use them for this research. Later, we explain the Twitter Developer Program and APIs used to collect two different datasets: `Twitter-CAD-Users` and `Twitter-CAD-Tweets`. Moreover, we explained the census divisions (i.e., census metropolitan areas and census agglomerations) —which are the ground truth locations for both datasets. Finally, we shared several statistics about the datasets.

In the next chapter, we will discuss a series of experiments performed on both datasets for metro and province level geolocation estimation. The experiments include some basic baseline classifiers, as well as more advanced ones.

Chapter 5

Methodology and Experiments

‘If one day, my words are against science,
choose science.‘
— Mustafa Kemal Ataturk, Founder of
Modern Turkey.

5.1 Development and testing environments

In this work, the experiments were deployed to two different environments; they are called development and testing. The development environment (also the local environment) was used for multiple purposes. As the name suggests, all the models were developed in this machine, but other tasks such as data collection, preprocessing and initial tests were also performed in the same machine. All the development was conducted using Python 2.7 and 3.6 [1]. The development environment has very a simple setup: macOS High Sierra (10.13.6) with Intel Core i5 and 8GB of memory.

Due to the limited memory and processing power of the local environment —and high volume of data, we needed a more powerful environment. We purchased machines —virtual machines —from Digital Ocean —which is an American cloud computing company. The test environment specifications (e.g., memory, number of CPUs) varied based on the experiments. Some experiments such as some baseline tests required less processing power and memory. We had a very simple setup for them such as using 8 or 16 GB of memory and a couple of virtual CPUs. [2] On the other hand, we were required much more powerful machines for more complex experiments (e.g., deep learning methods) and purchased quite powerful machines (e.g., 128 GB memory and 24 vCPU). For the testing environment, we used CentOS 7.6 x64 for all the setups. [3]

[1] Some libraries depended on Python 3 extensions and this is why we used both Python 2.7 and 3.6

[2] vCPU stands for virtual central processing unit. Each vCPU is considered as a single CPU core. Therefore, 1 vCPU is seen as a single core whereas 2 vCPUs equal to two cores in the central processing unit.

[3] CentOS is very stable Linux distribution: https://www.centos.org/

5.2 Experiment setups

Experiment setups differed depending on the types of experiments. Some experiments required some specific tasks (e.g. creating bag-of-words, word embeddings, or using existing word embeddings). We explain them in the related section. Prior to experiments, the documents (i.e., tweets) were preprocessed. This process is explained in detail in the next section. Furthermore, the definitions of document are different, depending on the datasets used. In Twitter-CAD-Users dataset —for which we aim to predict the home location of a user —a document is all the tweets of a user. However, for Twitter-CAD-Tweets, in which we aim to predict the location of the tweets, each tweet is considered as a single document. For all the experiments, the datasets were divided into the training and test set, with 80% for training and 20% for test. [4]

5.3 Text preprocessing

Preprocessing is a data preparation step necessary for our task since the language used by Twitter users is very informal. Many users create their own words, spelling shortcuts, punctuation, misspellings, new words, genre specific terminology and abbreviations. Tweets contains noise which is not useful for our task. The noise include URLs, HTML characters, emoticons, hash tags and retweet tags. The following steps discuss text prepossessing.

5.3.1 Removing non-English tweets

Both datasets include tweets that are not written in English. Many tweets are written in French, Spanish and Arabic. These tweets are mostly from the eastern census metropolitan areas such as the Quebec City, Montreal, and Ottawa-Gatineau.

There are two different approaches to detect non-English tweets: **word-based** and **sentence-based** methods. Word-based methods work quite straightforward; they use a dictionary of a specified language (e.g., English) and checks if a given word is in the dictionary or not. There are several issues with word-based methods. First of all, they are not quite scalable for large datasets; it is too time-consuming to perform word-by-word checking. Moreover, the word-based methods often fail to detect some named entities such as cities or provinces; they require the use of a separate gazetteer to detect them. Initially, we used a word-based method (PyEnchant), but it suffered from the issues mentioned above.

Sentence-based methods are probabilistic; they measure the probability of a given sentence s in language L. For our datasets, we tried two sentence-based libraries: *langdetect* and *googletrans*. The library of *googletrans* is a Python wrapper that can access Google

[4] We used scikit-learn's *train_test_split* function to split the datasets into train and test subsets. Besides, we used a random state value (e.g., 41) to select the values randomly.

Translate API. It worked well, but we had issues with strict API limits. For this reason, we used *langdetect* which also worked well and it helped to remove *most* non-English tweets from the two datasets.

5.3.2 Tweet repetition

Tweet repetition is an phenomenon that happen when the same tweet is copied and pasted over and over again by the same user. This is quite normal for users like news agencies and advertisement accounts, and it is against the Twitter policy.

We removed all the repetitive tweets of the same user. This was important since removing repetitive tweets reduced the number of tweets overall in our data sets, which in turn reduced the amount of time we spent for training and testing.

5.3.3 Tweet cleaning

Tweets have a lot of noisy data such as emoticons, HTML/special characters, URLS, numbers, email addresses etc. Our cleaning process involves removing all of those entities since they have no value for our task.

5.3.4 Spell checking

Misspelled words are quite common in tweets and correcting them is crucial for our task.

We use a probabilistic spell corrector described here.[5] The spelling correction consists of two steps. The first step is detecting misspelled word. To do that, we use PyEnchant's English dictionary to check if the word is a valid English word. If not, we go to the second step which performs the correction. In most cases, the spelling corrector shows multiple results for a misspelled word and we chose the first choice provided. [6]

5.3.5 Stop words

Stop words are the most common words seen in a text document that often less useful for many tasks. There are different stop word lists provided by different libraries. To demonstrate, NLTK's English stop word list includes only 179 stop words (mostly the pronouns) whereas sklearn has 318 words in total. In this work, we used sklearn's stop words list which includes more words than NLTK's. [7]

[5] http://norvig.com/spell-correct.html

[6] We did not try experiments with the different word choices (i.e., instead of selecting the first word suggestion as the correction, selecting the second one.)

[7] We removed the stop words after filter out the non-English tweets.

5.4 Baseline experiments

We ran several user-level and tweet-level baseline experiments on our data sets.

5.4.1 Experiments with the sklearn's DummyClassifier

DummyClassifier is the sklearn's supervised classifier that makes prediction by using very simple rules. It implements several classifiers that serve as the baselines, so we can compare their results with the results of other classifiers.

DummyClassifier has five different strategies:

- **Stratified**: It generates predictions by respecting the training sets class distribution.
- **Most frequent**: Always predicts the most frequent label in the training set.
- **Prior**: Always predicts the class that maximizes the class prior (like most frequent) and `predict proba` (i.e., the function that returns probability estimates for the test vectors X) returns the class prior.
- **Uniform**: Generates predictions uniformly at random.
- **Constant**: Always predicts a constant label that is provided by the user. This is useful for metrics that evaluate a non-majority class.

All the classification strategies use different classifiers, which do not learn from anything about the underlying relationships between the features and the target labels, but use the class distribution of training data to classify the instance. This means we did not use any text data (i.e., tweets) except for tweet or users' home locations.

Among the five different classifiers, we only used three of them: *most frequent*, *stratified*, and *uniform* since *prior* is quite similar to *most frequent* and *constant* always predicts a constant label (e.g., Vancouver) so it is not very useful for our work. However, the *constant* strategy might be handy for understanding the class distributions in a data set.

The results for the two data sets are presented in the tables 5.1 and 5.2.

	Metro			**Province**		
Strategy	**Mean**	**Median**	**ACC@161**	**Mean**	**Median**	**ACC@161**
Most Frequent	1073	467	34.6	940	827	45.2
Stratified	1513	838	19.8	1381	894	28.3
Uniform	1657	1218	13.0	1667	1472	14.4

Table 5.1: DummyClassifier Mean, Median error distances and ACC@161 accuracy for tweet-level location estimation

	Metro			Province		
Strategy	**Mean**	**Median**	**ACC@161**	**Mean**	**Median**	**ACC@161**
Most Frequent	975	416	40.4	873	450	50.0
Stratified	1558	859	22.6	1502	955	25.0
Uniform	1660	1415	13.0	1605	1418	15.4

Table 5.2: DummyClassifier Mean, Median error distances and ACC@161 accuracy for user-level location estimation

For both data sets, the highest accuracy was achieved by always predicting the most frequent class (i.e., metro or province) in the training set; that is *Toronto* in metro-level and *Ontario* in province-level which covers several metros such as Toronto, Hamilton and Kitchener- Cambridge-Waterloo —and a few more.

The second best result was achieved with the *stratified* strategy in both datasets. *Stratified* predicts based on the training set class distribution —and our class distribution is directly proportional to the class weights which are determined by the metro population— more populated metros have more instances in both datasets. For this reason, the strategy always predicts the more populated metros (e.g., Toronto, Montreal and Vancouver) and provinces (e.g., Ontario) and fails to predict other classes, which finally achieves a poor result.

Lastly, the *uniform* strategy produced the poorest scores in both data sets. *Uniform* makes random guesses without regarding the class distribution in the data sets, which ended up with the worst results.

5.4.2 Bag-of-words model

Machine Learning algorithms cannot work with raw text data (e.g., tweets); text data must be into converted into features. A bag-of-words or BoW in short, is a popular way of converting raw text data into feature vectors. More specially, a bag-of-words representation does two things:

1. Creating a vocabulary of words.
2. Using a scoring function to measure the presence of words.

The bag-of-words representation focuses on words. It collects all occurring words in the corpus and puts them into a *bag* —and the order of the words is discarded. The bag-of-words representation is used both in classification and regression tasks. The idea is that two documents are similar if they have similar words. Although the method is quite simple, it can be often become complex when deciding which words to include in the vocabulary and choosing the scoring function.

In a bag-of-words model, *every word is a feature*. If a word occurs in a text, then it is represented as 1 in the vector; otherwise 0. Thus, there would be many zeros for big datasets —like ours, since only a small portion of words in the *bag* can occur in a single document (e.g., a tweet). Keeping so many 0's in the memory is not feasible especially for regular computers. Fortunately, sci-kit learns to solve this problem by only keeping 1's and their positions in the vectors, and ignoring the 0's. Therefore, even the big datasets are scalable for a bag-of-words model.

Scoring functions

A scoring function decides the way of converting word occurrences to features. The simplest method is the binary method as explained earlier: marking a word as 1 if it occurs in the text and 0 otherwise.

There are some other scoring functions, as mentioned in Chapter 2. They convert documents into matrix of:

- Token counts
- Token occurrences
- TF-IDF features.

For the first one, text documents are converted into vectors of word counts (i.e., count vectorizer). It counts the number of times a token exists in a document. However, this representation creates a very sparse matrix because only a few words in the vocabulary exist in each document —and sparse matrix representations *often* cause problems for algorithms like Gaussian naive Bayes. Fortunately, there are ways to convert a sparse matrix representation into a dense one.

Calculating token occurrences (i.e., hashing vectorizer) is a memory-efficient solution; it applies the hashing trick [8] to strings in order to encode them as numerical indexes. One disadvantage of hashing vectorizer is that once the strings are vectorized, the feature names (i.e., words) can not be retrieved again. This is because the vectorizer does not store the strings in memory.

Finally, TF-IDF features are selected based on the frequency of a word in a document —and how recurrent that word in the whole corpus. For our bag-of-words models, we applied all three scoring functions to perform our experiments.

5.4.3 SVM with bag-of-words

Scikit-learn has a complete SVM module —that offers many different SVM implementations such as *SVC* (uses RBF kernel by default), *NuSVC* (allows you to decide number of

[8] https://en.wikipedia.org/wiki/Feature_hashing

support vectors) and *LinearSVC* (uses Linear kernel). The *SVC* and *NuSVC* implementation is based on *libsvm* —a C++ implementation of SVM algorithm whereas *LinearSVC* is based on *liblinear*.

Each implementation can be tuned with a number of parameters. Some of those parameters are shared across all the implementations, but some are only implementation-specific (i.e., only available for a specific implementation). To demonstrate, only NuSVC uses a parameter to control the number of support vectors —which plays an important role in the accuracy obtained. Furthermore, LinearSVC is the most flexible option by far out of the three implementations; it offers the choices to select penalty and loss functions.

There are three important SVM parameters which might impact the overall accuracy of the models. Those parameters are:

- **Kernel**: Kernel functions determine the type of hyperplanes used for the SVM. In other words, they are the similarity functions used to calculate the similarity of two samples. Scikit's SVM module supports three types of kernel functions: *linear*, *rbf*, and *poly*. *Linear* draws a linear hyperplane to separate the support-vectors, but *rbf* and *poly* use a non-linear hyperplane.
- **Gamma**: Gamma is used for non-linear kernel functions (i.e., rbf and poly). The value of gamma determines how exactly to fit the training dataset.
- **C**: C is a penalty parameter; it is used to control between decision boundary and classifying the training points correctly.

We perform our experiments with all the available SVM implementations —and with the rbf and linear kernel, where applicable. For the non-linear kernel functions, we picked *gamma* as *auto* which uses 1 / number_of_features and the C parameter was selected as 1, which is the general trend. The results for both datasets are shown in the tables 5.3 and 5.4.

	Count Vectorizer					
	Metro			**Province**		
	Mean	**Median**	**ACC@161**	**Mean**	**Median**	**ACC@161**
SVC (rbf)	1073	467	34.6	940	827	45.2
SVC (linear)	890	323	46.3	773	0	55.3
NuSVC (rbf)	1139	467	34.8	1139	467	34.8
NuSVC (linear)	1165	467	36.6	1215	827	37.0
LinearSVC	1023	370	43.0	852	74	52.8
	Hashing Vectorizer					
	Metro			**Province**		
	Mean	**Median**	**ACC@161**	**Mean**	**Median**	**ACC@161**
SVC (rbf)	1073	467	34.6	940	827	45.2
SVC (linear)	889	323	46.3	**770**	**0**	**55.3**
NuSVC (rbf)	1251	467	30.4	1298	827	33.3
NuSVC (linear)	1172	467	36.3	1171	827	40.3
LinearSVC	1016	331	43.5	835	74	53.4
	TF-IDF Vectorizer					
	Metro			**Province**		
	Mean	**Median**	**ACC@161**	**Mean**	**Median**	**ACC@161**
SVC (rbf)	1073	467	34.6	940	827	45.2
SVC (linear)	**909**	**323**	**46.4**	782	0	55.2
NuSVC (rbf)	1251	467	30.4	1298	827	33.3
NuSVC (linear)	1107	467	35.3	1184	827	38.3
LinearSVC	1029	365	43.2	855	74.4	52.8

Table 5.3: SVM Mean, Median error distances and ACC@161 accuracy for tweet-level location estimation

	Count Vectorizer					
	Metro			Province		
	Mean	Median	ACC@161	Mean	Median	ACC@161
SVC(rbf)	975	416	40.4	873	450	50.0
SVC (linear)	975	416	40.4	848	74.4	51.1
NuSVC (rbf)	1698	714	27.3	1466	1165	25.0
NuSVC (linear)	**658**	**0**	**59.5**	460	0	72.6
LinearSVC	732	217	48.8	632	0	63.0
	Hashing Vectorizer					
	Metro			Province		
	Mean	Median	ACC@161	Mean	Median	ACC@161
SVC (rbf)	975	416	40.4	873	450	50.0
SVC (linear)	943	344	41.6	848	74.4	51.1
NuSVC (rbf)	1698	714	27.3	1523	1454	22.6
NuSVC (linear)	616	0	55.9	**385**	**0**	**79.8**
LinearSVC	727	176	50.0	632	0	63.0
	TF-IDF Vectorizer					
	Metro			Province		
	Mean	Median	ACC@161	Mean	Median	ACC@161
SVC (rbf)	975	416	40.4	873	450	50.0
SVC (linear)	929	323	41.6	780	74.4	53.5
NuSVC (rbf)	1200	641	26.1	1736	1454	21.4
NuSVC (linear)	565	0	55.9	433	0	73.8
LinearSVC	727	176	50.0	395	0	75.0

Table 5.4: SVM Mean, Median error distances and ACC@161 accuracy for user-level location estimation

The best results were achieved with different SVM implementations; however one thing is common across all the experiments: the *linear* kernel outperforms *rbf* by a large margin in all the results. For tweets, the best results were achieved with SVC with linear kernel; LinearSVC was the closest classifier in terms of accuracy. Although NuSVC performed poorly in the tweet-level location estimation, it achieved the best results at the user level. The difference between the linear and rbf kernel with NuSVC in user-level location estimation is more than half.

5.4.4 Naive Bayes with bag-of-words

Similar to SVM, scikit learn offers many Naive Bayes variants based on applying Bayes' theorem such as Multinomial Naive Bayes, Complement Naive Bayes, and Gaussian Naive Bayes. Unlike SVM, Naive Bayes classifiers are less flexible (i.e., they provide less parameters to tune). Besides, some Naive Bayes variants such as Gaussian Naive Bayes do not work with sparse representations (i.e., bag-of-words representations). We had to convert

the vectors into dense ones. To do that, we used a very simple `DenseTransformer`. [9]. The results are shown in the tables 5.5 and 5.6.

	Count Vectorizer					
	Metro			**Province**		
	Mean	**Median**	**ACC@161**	**Mean**	**Median**	**ACC@161**
Multinomial NB	1014	467	39.0	860	74	50.0
Complement NB	995	323	44.8	866	74	53.0
Gaussian NB	1308	598	29.9	1318	955	33.0
	Hashing Vectorizer					
	Metro			**Province**		
	Mean	**Median**	**ACC@161**	**Mean**	**Median**	**ACC@161**
Multinomial NB	1014	467	39.0	860	74	50.0
Complement NB	**910**	**323**	**45.7**	**807**	**74**	**53.0**
	TF-IDF Vectorizer					
	Metro			**Province**		
	Mean	**Median**	**ACC@161**	**Mean**	**Median**	**ACC@161**
Multinomial NB	1000	365	39.9	842	74	51.1
Complement NB	1031	323	43.9	903	74	51.5
Gaussian NB	1309	598	29.9	1320	955	33.3

Table 5.5: Naive Bayes Mean, Median error distances and ACC@161 accuracy for tweet-level location estimation

[9] http://zacstewart.com/2014/08/05/pipelines-of-featureunions-of-pipelines.html

	Count Vectorizer					
	Metro			Province		
	Mean	**Median**	**ACC@161**	**Mean**	**Median**	**ACC@161**
Multinomial NB	975	416	40.4	873	450	50.0
Complement NB	943	344	41.6	848	74	51.1
Gaussian NB	947	419	35.7	832	37	53.5
	Hashing Vectorizer					
	Metro			Province		
	Mean	**Median**	**ACC@161**	**Mean**	**Median**	**ACC@161**
Multinomial NB	975	416	40.4	873	450	50.0
Complement NB	943	344	41.6	873	450	50.0
Gaussian NB	865	323	38.0	**822**	**0**	**54.7**
	TF-IDF Vectorizer					
	Metro			Province		
	Mean	**Median**	**ACC@161**	**Mean**	**Median**	**ACC@161**
Multinomial NB	975	416	40.4	873	450	50.0
Complement NB	**856**	**323**	**46.4**	780	74	53.5
Gaussian NB	963	419	35.7	849	37	53.5

Table 5.6: Naive Bayes Mean, Median error distances and ACC@161 accuracy for user-level location estimation.

For both datasets, Complement Naive Bayes outperformed the other two; it achieved the best results in the metro-level problem. On the other hand, although Gaussian NB performed poorly in general, it achieved the best result in the user-level location estimation at the province granularity.

5.5 Limitations of bag-of-words

The bag-of-words model is very simple to use and often serve as a baseline. It has several limitations, namely:

- **Careful preprocessing** is crucial for the vocabulary; otherwise the vocabulary might be massive which causes to sparsity (i.e., many 0's in the vector representations).
- **Sparsity** in the vectors makes modeling more difficult since only a few information is offered. (e.g., 1's in the vector.)
- There is no way to understand the **meaning** of the words. The bag-of-words ignores the order of the words —and this causes inability to understand the semantics of them. Understanding the meaning can often significantly help the classification tasks.

Because of the problems in the bag-of-words model, researchers proposed more complicated representations such as word embeddings which provide semantic representations for words.

5.6 Deep Learning Methods

Recently, deep learning methods have been applied to a variety of fields in computer science —natural language processing is one of them. Lately, NLP research increasingly focuses on using deep learning architectures instead of using standard machine learning algorithms such as SVM and Naive Bayes.

In this work, we experimented with different deep learning architectures such as convolutional neural networks and multilayer perceptron with sparse and high dimensional vectors (i.e., bag-of-words) as well as dense and low dimensional vectors (i.e., word embeddings).

5.6.1 Applying bag-of-words to multilayer perceptron and finding hyper parameters

In general, deep learning techniques need to learn a large number of parameters —named hyper parameters —and finding the best configuration for models is a challenging task. Poorly chosen values for these parameters might cause poor results; it can even lead to worse performance than for simpler machine learning models.

We tried two different methods to find the best values for the hyper parameters:

- Exhaustive Grid Search
- Randomized Search

Exhaustive grid search is an approach to build and evaluate a model for each combination of algorithm parameters, whereas **randomized search** samples algorithm parameters from a random distribution (i.e., uniform) for a number of iterations.

For both datasets, we tried a number of parameters. [10] Those parameters are:

- **Activation**: Activation function (i.e., the function in a artificial neuron that delivers an output) for the hidden layer.
- **Alpha**: Penalty function.
- **Hidden layer sizes**: Number of hidden layers and number of neurons in each layer.

[10] https://scikit-learn.org/stable/modules/generated/sklearn.neural$_network.MLPClassifier.html$

- **Solver**: Solver refers to weight optimization. There are three available solvers: *lbfgs* —an optimizer in the family of quasi-Newton methods —sgd —stochastic gradient descent —and adam —another stochastic gradient-based optimizer introduced by Kingma and Ba (2014).
- **Learning rate**: Learning rate schedule for weight updates.
- **Maximum iteration**: Maximum number of iterations.

The values used for each parameter are shown in the table 5.7.

Parameter	Available values	Used values
Activation	identity, logistic, tanh, relu	relu, tanh
Alpha	Float number	0.0001, 0.05
Hidden layer sizes	An integer tuple size	(100,), (50,50,50), (50,100,50)
Solver	sgd, adam, lbfgs	sgd, adam, lbfgs
Learning rate	constant, invscaling, adaptive	constant, adaptive
Maximum iteration	An integer value	200, 500, 1000

Table 5.7: List of hyper parameters and corresponding values used with the Multilayer Perceptron.

As seen in the table 5.7, we experimented with only a few available values for each parameter; this is because we run out of memory when all the values are used. For each parameter, we look at the common practices in Scikit-learn documentation and used them.

We could run both an exhaustive grid search and randomized for only Twitter-CAD-Tweets —at the province level —since it is smaller than Twitter-CAD-Users and in turn spends less memory. Regarding this parameter set, both exhaustive grid search and randomized search produced different results for the dataset. The results are shown in the table 5.8.

Parameter	Exhaustive Grid Search	Randomized Search
Activation	**relu**	**tanh**
Alpha	0.0001	0.0001
Hidden layer sizes	(100,)	(100,)
Solver	**sgd**	**lbfgs**
Learning rate	adaptive	adaptive
Maximum iteration	**1000**	**200**

Table 5.8: Hyper parameters produced by two different methods for Twitter-CAD-Tweets dataset.

In both methods, using a single hidden layer outperformed both 50 and 100 hidden layers. Also, we got the same alpha, and learning rate values from both methods. On the other hand, activation function, solver, maximum iteration parameters were different. Out of the two results, we created two configurations; they are `config1` for exhaustive grid search and `config2` for the randomized search. Then, we run the configurations for the datasets for both the metro and province level. The results are in the table 5.9 and the table 5.10.

	Metro			**Province**		
	Mean	**Median**	**ACC@161**	**Mean**	**Median**	**ACC@161**
config1	1095	467	40.4	924	74	**50.1**
config2	1061	450	**41.8**	958	728	48.8

Table 5.9: MLP Mean, Median error distances and ACC@161 accuracy for tweet-level location estimation.

	Metro			**Province**		
	Mean	**Median**	**ACC@161**	**Mean**	**Median**	**ACC@161**
config1	975	416	40.4	873	450	50.0
config2	625	0.0	**57.1**	515	0	**67.8**

Table 5.10: MLP Mean, Median error distances and ACC@161 accuracy for user-level location estimation.

`Config2` (i.e., randomized search) achieved better results mostly, however it was still failed to outperform NuSVC's results; NuSVC with linear kernel achieved better results with a small margin.

5.6.2 Word embeddings

The main problem with bag-of-words representation is the lack of semantic understanding of text. This is because word order in text is discarded —the one-hot representation also discards semantic information. This problem is solved with *word embeddings*.

Word embeddings are a group of techniques for feature extraction in natural language processing. Similar to bag-of-words, words and phrases are converted into vectors, especially vectors of numbers. On the other hand, vectors of real numbers show similarity between words and phrases (i.e., similar words have similar representation) in word embedding models; not the *token counts*, *occurrences* or tf-idf features like in bag-of-words. This has a huge advantage for a few reasons: (i) context and semantic understanding offers more semantics; (ii) word embedding vectors are dense and low dimensional —unlike sparse and high dimensional vectors of the bag-of-words model —and neural models work better with them; (iii) finally dense representations provide more *generalization*.

There are three types of word embedding algorithms. We overview them below.

5.6.3 Algorithms for word embeddings

Word embedding algorithms create vector representations from a text corpus in which the vocabulary size is finite. This process is called *learning*. Learning is done either with the aid of neural models for a task specific task (e.g., text classification) or using unsupervised learning. Mainly, there are three widely used word embedding algorithms: Word2Vec, GloVe and Embedding Layer.

Word2Vec

Word2Vec Mikolov et al. (2013) is an array of two-layer neural networks to create word embedding. Vectors are contained in a vector space —and the vector space can be as large as thousands of dimensions. Each word in the corpus is assigned a vector; vectors with similar meaning are represented closely in the vector space.

There are two different learning models within the context of Word2Vec:

- Continuous Bag-of-Words
- Continuous Skip-Gram Model

Continuous Bag-of-Words focuses on each word; producing a vector based on its meaning, whereas Continuous Skip-Gram Model is a bit more complex and prediction is performed based on the adjacent words.

GloVe

The Global Vectors for Word Representation (or GloVe) is an another unsupervised model for generating word embeddings. Unlike predictive models such as Word2Vec, GloVe follows a *count-based* model. Count-based model works in two steps. Initially, the algorithm builds a huge matrix to keep the co-occurrence information by counting the frequency of each word w in the context c. This matrix is huge; containing many pairs of words and context. In the second step, matrix reduction is performed in order to create a low-dimensional vector space. This is the task of finding lower-dimensional representations which covers the most-variance in the entire corpus.

Embedding Layer

Embedding Layer is different from Word2Vec and GloVe; word embeddings are produced as part of another task (i.e., document classification). Many neural network algorithms such as recurrent neural networks (RNN) have many layers —and embedding layer can be an initial layer to produce vectors so that other layer(s) can use them for the computation. Furthermore, there are two requirements to create the embeddings: (i) the corpus data must be preprocessed (i.e., cleaning); (ii) and size of the vector space (e.g., 100, 200, 300 dimensions) must be specified in advance.

5.6.4 Keras embedding layer

Keras provides an embedding layer —and it can be used as part of a neural network. The embedding layer requires further preprocessing for data: documents must be integer-encoded (i.e., all words must be assigned an integer value). We did this by using the Keras Tokenizer API. Moreover, the embedding layer can be used in many ways such as:

- Learning word embeddings and saving them for later use.
- Learning word embeddings as part of another task (i.e., document classification).
- Using pre-trained word embeddings such as Word2Vec or GloVe.

We computed word embeddings from our dataset and also used pretrained embeddings, with the CNN. In the coming sections, we explain those processes.

5.6.5 Learning word embedding

We learnt word embedding from our dataset — as part of our Multilayer perceptron model which is explained later in this chapter. There several parameters the Keras embedding layer requires for embedding learning. These are:

- **Input dimension**: Input dimension specifies the size of the vocabulary for documents.
- **Output dimension**: Output dimension determines the vector space in which words will be embedded.
- **Input length**: Length of the input (or document) sequences.

The learning part involved several preprocessing steps. Initially, we one-hot encoded all documents (i.e., converting all text documents into vectors of numbers) with the vocabulary size of 30. This is also the **input dimension**. Furthermore, the documents have different lengths —and deep learning models often need input vectors with the same size. For the next step, we pad all to documents to a fixed length (e.g., 8, 12 or 16). This specifies the **input length** for embedding layer. Finally, we set the output dimension as 8.

5.6.6 Applying word embeddings to convolutional neural networks

Convolutional neural networks (CNNs or ConvNets) are one of the most powerful deep learning architectures; the success of CNNs comes from image classification. Nonetheless, CNNs have been used for text classification for many years. Goldberg (2017) claims that

CNNs *might* outperform linear classifiers such as LinearSVM, especially when they are used with pre-trained word embeddings such as GloVe or Word2Vec.

The power of CNNs comes from its ability to select features, more specifically salient features (tokens or an array of tokens). Those features give many clues about class membership.

In an usual CNN design, there are three fundamental layers:

- **Word Embedding**: This is the layer in which pretrained word embeddings are provided —or embeddings are learnt from text data.
- **Convolutional Layer**: This is the feature extraction layer. It learns the salient features from the embeddings provided by previous layer.
- **Fully-connected Layer**: Fully-connected Layer has access to all the nodes in the previous layer; it is the layer also makes the final decision about the class membership.

In this work, we used a simple CNN architecture with multiple layers:

- **Embedding Layer**: This is the first hidden layer of the network. In this layer, we used different embeddings including embeddings from our dataset, GloVe and Word2Vec.
- **Dropout Layer**: Applies Dropout[11] to input —and reduces overfitting.
- **Flatten Layer**: Flattens the input array.
- **Dense Layer**: This is a standard layer —equivalent of fully-connected Layer —it connects all the nodes in the previous layer into the current one. In Dense layer, we used *softmax* activation function which help us to classify multiple categories.

Similar to the Multilayer Perceptron with bag-of-words, we searched for hyperparameters using —Exhaustive Grid Search and Randomized Search for both datasets in metro and province level. For both datasets, we used numerous parameters to find the optimal values. These parameters are:

- **Loss function**: It is the function to calculate *loss* associated to an event.
- **Optimizer**: It is similar to solver. Optimizer is used for weight optimization.
- **Batch size**: Number of training samples used in one iteration.
- **Epoch**: Determines how many times the input will be passed forward and backward in the neural network.

[11] https://en.wikipedia.org/wiki/Dropout_(neural_networks)

The list of available values —and the values we used are shown in the table 5.11.

Parameter	Available values	Used values [12]
Loss function	14 available loss functions [13]	categorical_crossentropy
Optimizer	7 available optimizers [14]	sgd, adam, adamax
Batch size	An integer value	5, 10, 20, 40, 60, 100
Epoch	An integer value	1, 5, 10, 20

Table 5.11: List of hyper parameters —and corresponding values used with CNN.

The results for CNN are showed in the tables 5.12 and 5.13. We show the results achieved with the optimal hyper parameters. Although Exhaustive Grid Search and Randomized Search produced different optimal values, respectively, both configurations produced the same Mean, Median and @ACC161 accuracy.

	Metro			Province		
	Mean	**Median**	**ACC@161**	**Mean**	**Median**	**ACC@161**
Embedding Layer	1073	467	**34.6**	1040	600	**46.0**
Glove	1073	467	**34.6**	930	827	45.7

Table 5.12: CNN Mean, Median error distances and ACC@161 accuracy for tweet-level location estimation.

	Metro			Province		
	Mean	**Median**	**ACC@161**	**Mean**	**Median**	**ACC@161**
Embedding Layer	969	416	**40.4**	873	450	**50.0**
Glove	975	416	39.2	894	827	45.2

Table 5.13: CNN Mean, Median error distances and ACC@161 accuracy for user-level location estimation.

Surprisingly, CNN produced poorer results than both Multilayer Perceptron with bag-of-words and most other baseline tests. There might be a number of reasons for that: (i) our preprocessing schema might not be optimal for CNN; (ii) there might be more suitable parameters for this task; (iii) our neural architecture might not be the best for this task.

5.7 Home location identification using social network

Geolocation detection by using Twitter social network is the most recent approach. Lately, many studies either use social network solely or use it with other geolocation methods (i.e., hybrid methods) such as context-based methods for location estimation. There are many ways of using one's social network to estimate a location. In this work, we followed a very simple method partly proposed by Rout et al. (2013); that is, using the count of number of friends for each city —metro and province in our case —as feature. Moreover, we also used the count of the number of followers [15] for each metro area as feature in a separate task, to see which feature set —count of friends or followers —achieves the best result. More specifically, our method includes several steps: (i) we fetch each user's friends —or followers —; (ii) then check if each friend or follower has a valid geographical location entered in his profile by using GeoText[16]; (iii) finally —depending on the granularity (i.e., metro or province) —we check if the user has entered any metros or provinces in the location field we classify. The results are in the tables 5.14 and 5.15.

Friends					
Metro			**Province**		
Mean	**Median**	**ACC@161**	**Mean**	**Median**	**ACC@161**
916	323	42.6	785	74	56.0

Table 5.14: User-level home location identification by using users' friends location.

Followers					
Metro			**Province**		
Mean	**Median**	**ACC@161**	**Mean**	**Median**	**ACC@161**
1008	467	42.1	901	827	46.9

Table 5.15: User-level home location identification by using users' followers location.

The results at metro level are quite close; friends feature set achieves slightly better than followers. On the other hand, there is huge difference at province level —almost a ten percent. The results prove that the social network —or social graph —of users might actually include clues about users' location. However, the social-graph based geolocation achieved poorer results compared to other methods (i.e., content based with bag-of-words representation).

[15] Some research including (Backstrom et al., 2010) and (Rout et al., 2013) hypothesize that people who follow other people in Twitter might be in the same city. This is because they want be aware of their friends. However, this is not mostly the case since many celebrities live in big city and they have followers all over the world.

[16] GeoText is a Python library used to extract locations in text.

5.8 Summary

In this chapter, we presented a number of experiments achieved with several classifiers (e.g., SVM, Naive Bayes, MLP and CNN) and their variations (e.g., LinearSVC, NuSVC etc.). We also used different feature sets (e.g., bag-of-words, word embeddings, friends and followers count).

For the user-level dataset (i.e., Twitter-CAD-Users), we achieved the best results with **NuSVC** with linear kernel at both metro and province level; that is, 59.5% and 79.8%, respectively, in ACC@161. At metro level, the second best result was achieved with a neural classifier —the Multilayer Perceptron —which was 57.1. At province level, the second-best result was achieved with LinearSVC, 75.0%.

For the tweets dataset (i.e., Twitter-CAD-Tweets), the best results were achieved with **SVC with linear kernel** in both metro and province level, 46.4% and 55.3%, respecively. The second-best result was achieved with Complement NB, 45.7 % in metro level and with SVC with linear kernel at province level.

Here we summarize what we learned from the experiments discussed in this chapter:

1. Linear kernels outperform non-linear kernels (e.g., rbf). In all the experiments, linear kernel produced better results than non-linear kernels.

2. Some classifiers produced good results for some tasks —and achieves poor results for the others. For example, NuSVC with a linear function did not work well for the tweet-level location estimation, but produced the best results at user-level, in both granular.

3. `Finding good values for the hyper parameters` is quite important; we achieved poor results with MLP at first, but changinf the hyper parameters helped us to achieve much better results.

4. Bag-of-words often outperformed the word embedding representation. CNN with word-embeddings produced the worst results.

Moreover, although deep learning architectures did not produce the best results in our research, they worked well in some other such as Rahimi et al. (2017) in which dialect terms are used as features with Multilayer perceptron rather than simple tweet content (i.e., bag-of-words).

There are many geolocation detection studies done for the United States and other parts of the world. However, we cannot do a direct comparison between our study and them since each region has its own characteristics. Considering the US and Canada, there are fundemental differences: (i) there are more cities and provinces in the US than Canada (ii) cities in the US are more densely populated than Canadian cities (iii) population in the US is more evenly distributed than in Canada. Each of these characteristics affects the geolocation accuracy.

In the next chapter, we conclude this thesis and discuss directions of future work.

Chapter 6

Conclusion and Future Work

> 'I have no special talent. I am only passionately curious.'
>
> — Albert Einstein, Theoretical physicist.

This thesis concludes with a summary and with a discussion about possible future work to continue the development of this work.

6.1 Conclusion

This thesis addressed the task of automatic classification of tweets in order to find the home location of Twitter users —and the location from which the tweets were sent out. We investigated multiple ways of doing this.

For this research, we collected two datasets that serve different purposes: `Twitter-CAD-Users` and `Twitter-CAD-Tweets`. `Twitter-CAD-Users` contains Canadian Twitter users with their tweets, their social graph (i.e., friends and followers) and several metadata information about them. On the other hand, `Twitter-CAD-Tweets` contains only geotagged tweets. The ground truth location for users in `Twitter-CAD-Users` is the *user declared location* in Twitter profile —like in many datasets from related work (Hecht et al., 2011), (Cheng et al., 2010), whereas the ground truth location for geotagged tweets is given by the latitude and longitude coordinates attached to the tweets. Both datasets contain users and tweets from census metropolitan areas and census agglomerations with a population at least a hundred thousand. According to this criterion, there are forty-one CMAs/CAs and nine provinces.

We conducted a variety of experiments by using both machine learning and deep learning methods. Starting from `Dummy Classifier` —we did initial baseline experiments with very simple rules such as `most frequent` in which the most frequent class (i.e., Toronto in city-level or Ontario in province-level) is selected or uniform —selecting all class labels uniformly. Next, we did experiments with bag-of-word models. Text data was converted

into numbers, more specifically vectors of numbers, and we tried different vectorizers such as `Count Vectorizer`, `Hashing Vectorizer` and `TF-IDF vectorizer`. We used bag-of-words representation of our datasets with SVM classifiers (SVC, LinearSVC and NuSVC) and Naive Bayes classifiers (Multinomial NB, Complement NB and Gaussian NB). Furthermore, we applied two deep learning architectures: Multilayer Perceptron and Convolutional Neural Network with both bag-of-word and word embedding representations. However, simple bag-of-words models outperformed word embedding models. Finally, we added a very trendy approach: geolocation prediction in Twitter with social graph. We counted each user's friends —and followers in a separate model —in the list of cities and provinces we are using for classification and assigned the city and province based on the top result. The `Friends` model achieved better accuracy than the `Followers` model.

6.2 Future work

There are many directions for future work. First and foremost, similar to (Rahimi et al., 2017), geolocation prediction based on dialectical terms can be applied for both datasets. However, a dialectical dictionary like DAREDES[1] is needed for Canada.

Another directions is to improve the social-graph based geolocation prediction. Similarly to (Rout et al., 2013), *triads* —and *reciprocated friendships* —mutual friendships (i.e., both parties follow each other) —can be taken into account. Also, mentions between the users can be used as clues for the location of users (i.e., users who live in the same cities mention each other in their tweets more). Furthermore, social-graph based geolocation can be integrated with the content-based methods (e.g., using tweets content).

Finally, different tweet processing techniques can be applied to see if they make any difference in bag-of-words and word embeddings models.

[1] http://dare.wisc.edu/

APPENDICES

Appendix A

Dataset characteristics and sample values

A.1 Tweets

eliminating ontario child advocate office mistake <url> via <user>

toronto police union head says handgun ban would impact <allcaps> cbc <allcaps> news <url>

toronto eyebrow microblading cancer survivor <allcaps> hrh <allcaps> beauty <url> via <user>

ottawa race weekend reminds much admire runners discipline practice good luck <url>

ottawa jr senators assistant coach something pretty classless towards bench guys l <url>

montreal bartender make cocktails one hell storyteller <url> via <user>

calgary doctor honoured improving health care access women <hashtag> cdnhealth <hashtag> <hashtag> abhealth <hashtag> <url> via <user>

vancouver moody weather makes head spin

winnipegers come wine words <user> looks like great night tickets <url>

halifax bishop alumnus scotty potter <number> maybe greys playing economy shoe shop <number> argyle <url>

victoria day get away <user> great time way busy today <hashtag> victoria day <hashtag> <url>

A.2 Users

User ID	196783947
Name	Aarti Pole
Screen name	aartipole
User declared location	Toronto, Ontario
Tweets count	14555
Followers count	4977
Friends count	2323
Metro	Toronto
Province	Ontario
Gender	Male
Verified	True

Table A.1: A sample user from Twitter-CAD-Users dataset.

A.3 Dataset characteristics

CMA/CA	**Number of users**	**Number of tweets**
Toronto	420	234.956
Montreal	64	139.969
Vancouver	40	101.798
Calgary	23	58.992
Ottawa-Gatineau	21	54.049
Edmonton	21	50.621
Quebec City	13	27.061
Winnipeg	13	33.835
Hamilton	12	33.135
Waterloo	8	20.825
London	8	20.556
St. Catharines Niagara	6	12.719
Halifax	6	15.958
Oshawa	6	14.959
Victoria	6	15.003
Windsor	5	13.360
Saskatoon	5	12.700
Regina	5	9.097
Sherbrooke	4	7.625
St. John's	4	7.625
Barrie	3	9.180

cont'd

CMA/CA	**Number of users**	**Number of tweets**
Kelowna	3	6.549
Abbotsford —Mission	3	7.177
Greater Sudbury	3	9.426
Kingston	3	8179
Saguenay	3	933
Trois-Rivires	3	489
Guelph	3	8.510
Moncton	2	5.336
Brantford	2	4.558
Saint John	2	5.723
Peterborough	2	4.745
Thunder Bay	2	6.381
Lethbridge	2	3.671
Nanaimo	2	4.266
Kamloops	2	4.112
Belleville	2	4.548
Chatham-Kent	2	2.579
Fredericton	2	4.339
Chilliwack	2	5.065
Red Deer	2	6.041

Table A.2: Characteristics of Twitter-CAD-Users dataset.

cont'd

CMA/CA	**Number of tweets**
Toronto	10.000
Montreal	6.650
Vancouver	4000
Calgary	2324
Ottawa-Gatineau	2158
Edmonton	2158
Quebec City	1328
Winnipeg	1294
Hamilton	1228
Waterloo	868
London	820
St. Catharines Niagara	664
Halifax	664
Oshawa	629
Victoria	609
Windsor	546
Saskatoon	489
Regina	391
Sherbrooke	351
St. John's	340
Barrie	327

CMA/CA	**Number of tweets**
Kelowna	322
Abbotsford —Mission	298
Greater Sudbury	272
Kingston	267
Saguenay	113
Trois-Rivires	258
Guelph	250
Moncton	239
Brantford	222
Saint John	209
Peterborough	181
Thunder Bay	200
Lethbridge	194
Nanaimo	172
Kamloops	170
Belleville	170
Chatham-Kent	169
Fredericton	167
Chilliwack	167
Red Deer	166

Table A.3: Characteristics of Twitter-CAD-Tweets dataset.

Appendix B

Source code

B.1 Scripts

All the source code of the experiments (i.e. scripts) shared via Github. [1] The experiments include:

- Dummy Classifier
- SVM and Naive Bayes Classifiers
- Multilayer Perceptron
- Convolutional Neural Network

B.2 Prerequisites

Python 3 is recommended to run all the experiments —we also recommend to create a separate Python 3 virtual environment. [2]. Besides, datasets must be downloaded before running all the scripts. [3].

Following libraries must be installed into your system —or virtual environment to execute the scripts:

- **Numpy** (numpy.org)
- **Scikit Learn** (scikit-learn.org)
- **GeoPy** (geopy.readthedocs.io)

[1] https://github.com/mertmetin/Geolocation-Prediction
[2] https://www.geeksforgeeks.org/python-virtual-environment/
[3] https://github.com/mertmetin/Canadian-Twitter-users-and-tweets-datasets

B.3 Executing scripts from the command line

The scripts can be executed with the following command. If the virtual environment is created, it must be enabled before executing them (`source bin/activate`):

```
python <script-name>.py
```

References

Amitay, E., Har'El, N., Sivan, R., and Soffer, A. (2004). Web-a-where: geotagging web content. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 273–280. ACM.

Argamon, S., Koppel, M., Fine, J., and Shimoni, A. R. (2003). Gender, genre, and writing style in formal written texts. *Text-The Hague Then Amsterdam Then Berlin-*, 23(3):321–346.

Backstrom, L., Sun, E., and Marlow, C. (2010). Find me if you can: improving geographical prediction with social and spatial proximity. In *Proceedings of the 19th international conference on World wide web*, pages 61–70. ACM.

Benhardus, J. and Kalita, J. (2013). Streaming trend detection in twitter. *International Journal of Web Based Communities*, 9(1):122–139.

Burger, J. D., Henderson, J., Kim, G., and Zarrella, G. (2011). Discriminating gender on twitter. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1301–1309. Association for Computational Linguistics.

Cheng, Z., Caverlee, J., and Lee, K. (2010). You are where you tweet: a content-based approach to geo-locating twitter users. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 759–768. ACM.

Compton, R., Jurgens, D., and Allen, D. (2014). Geotagging one hundred million twitter accounts with total variation minimization. In *Big Data (Big Data), 2014 IEEE International Conference on*, pages 393–401. IEEE.

Ding, J., Gravano, L., and Shivakumar, N. (2000). Computing geographical scopes of web resources.

Eisenstein, J., Ahmed, A., and Xing, E. P. (2011). Sparse additive generative models of text.

Eisenstein, J., O'Connor, B., Smith, N. A., and Xing, E. P. (2010). A latent variable model for geographic lexical variation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1277–1287. Association for Computational Linguistics.

Fink, C., Piatko, C. D., Mayfield, J., Finin, T., Martineau, J., et al. (2009). Geolocating blogs from their textual content. In *AAAI Spring Symposium: Social Semantic Web: Where Web 2.0 Meets Web 3.0*, pages 25–26.

Gilbert, E., Karahalios, K., and Sandvig, C. (2008). The network in the garden: an empirical analysis of social media in rural life. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1603–1612. ACM.

Goldberg, Y. (2017). Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1):1–309.

Han, B., Cook, P., and Baldwin, T. (2012). Geolocation prediction in social media data by finding location indicative words. *Proceedings of COLING 2012*, pages 1045–1062.

Hecht, B., Hong, L., Suh, B., and Chi, E. H. (2011). Tweets from justin bieber's heart: the dynamics of the location field in user profiles. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 237–246. ACM.

Ikeda, K., Hattori, G., Ono, C., Asoh, H., and Higashino, T. (2013). Twitter user profiling based on text and community mining for market analysis. *Knowledge-Based Systems*, 51:35–47.

Jayasinghe, G., Jin, B., Mchugh, J., Robinson, B., and Wan, S. (2016). Csiro data61 at the wnut geo shared task. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pages 218–226.

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

Kusner, M., Sun, Y., Kolkin, N., and Weinberger, K. (2015). From word embeddings to document distances. In *International Conference on Machine Learning*, pages 957–966.

Liben-Nowell, D., Novak, J., Kumar, R., Raghavan, P., and Tomkins, A. (2005). Geographic routing in social networks. *Proceedings of the National Academy of Sciences*, 102(33):11623–11628.

Liu, B., Yuan, Q., Cong, G., and Xu, D. (2014). Where your photo is taken: Geolocation prediction for social images. *Journal of the Association for Information Science and Technology*, 65(6):1232–1243.

Mahmud, J., Nichols, J., and Drews, C. (2012). Where is this tweet from? inferring home locations of twitter users. *ICWSM*, 12:511–514.

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

Miura, Y., Taniguchi, M., Taniguchi, T., and Ohkuma, T. (2016). A simple scalable neural networks based model for geolocation prediction in twitter. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pages 235–239.

Orabi, A. H., Buddhitha, P., Orabi, M. H., and Inkpen, D. (2018). Deep learning for depression detection of twitter users. In *Proceedings of the Fifth Workshop on Computational Linguistics and Clinical Psychology: From Keyboard to Clinic*, pages 88–97.

Prasetyo, P. K., Achananuparp, P., and Lim, E.-P. (2016). On analyzing geotagged tweets for location-based patterns. In *Proceedings of the 17th International Conference on Distributed Computing and Networking*, page 45. ACM.

Rahimi, A., Cohn, T., and Baldwin, T. (2017). A neural model for user geolocation and lexical dialectology. *arXiv preprint arXiv:1704.04008*.

Roller, S., Speriosu, M., Rallapalli, S., Wing, B., and Baldridge, J. (2012). Supervised text-based geolocation using language models on an adaptive grid. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1500–1510. Association for Computational Linguistics.

Rout, D., Bontcheva, K., Preoţiuc-Pietro, D., and Cohn, T. (2013). Where's@ wally?: a classification approach to geolocating users based on their social ties. In *Proceedings of the 24th ACM Conference on Hypertext and Social Media*, pages 11–20. ACM.

Schulz, A., Hadjakos, A., Paulheim, H., Nachtwey, J., and Mühlhäuser, M. (2013). A multi-indicator approach for geolocalization of tweets. In *Icwsm*, pages 573–582.

Sloan, L., Morgan, J., Housley, W., Williams, M., Edwards, A., Burnap, P., and Rana, O. (2013). Knowing the tweeters: Deriving sociologically relevant demographics from twitter. *Sociological research online*, 18(3):1–11.

Van Laere, O., Quinn, J., Schockaert, S., and Dhoedt, B. (2014). Spatially aware term selection for geotagging. *IEEE transactions on Knowledge and Data Engineering*, 26(1):221–234.

Vicente, M., Batista, F., and Carvalho, J. P. (2019). Gender detection of twitter users based on multiple information sources. In *Interactions Between Computational Intelligence and Mathematics Part 2*, pages 39–54. Springer.

Wing, B. and Baldridge, J. (2014). Hierarchical discriminative classification for text-based geolocation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 336–348.

Wing, B. P. and Baldridge, J. (2011). Simple supervised document geolocation with geodesic grids. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 955–964. Association for Computational Linguistics.

Wolfram, W. and Schilling, N. (2015). *American English: dialects and variation*, volume 25. John Wiley & Sons.

Yoon, J., Kim, J. W., and Jang, B. (2018). Ditex: Disease-related topic extraction system through internet-based sources. *PloS one*, 13(8):e0201933.

Zheng, X., Han, J., and Sun, A. (2018). A survey of location prediction on twitter. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1652–1671.