NTNU
Norwegian University of
Science and Technology

A Metadata Approach to Predicting Twitter User Geolocation

Sigurd Sandve

Master of Science in Computer Science
Submission date: June 2016
Supervisor: Kjetil Nørvåg, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Abstract

As the volume of available data created from social media grows, creating value out of this information becomes an interesting challenge. This work presents an approach to identify a users country of origin from just looking at a single tweet. The approach focuses on using only metadata, while disregarding the actual content of the tweet. This work also looks at the scalability of the approach using Naive Bayes algorithm. The experiments concludes that using metadata to predict a user geolocation is a viable approach that scales to a stable accuracy with increasing data size.

Sammendrag

Ettersom volumet av tilgjengelige data som genereres fra sosiale medier vokser, blir det å skape verdier ut av denne informasjonen en interessant utfordring. Dette arbeidet presenterer et system til å identifisere en brukers hjemlandet fra bare å se på en enkelt tweet. Systemet fokuserer på å bare bruke metadata, men ser bort fra det faktiske innholdet i tweeten. Dette arbeidet ser også på skalerbarhet av tilnærming ved hjelp av Naive Bayes algoritme. Forsøkene konkluderer at bruk av metadata for å forutsi en brukers geolocation er en funksjonell tilnærming som skalerer til en stabil nøyaktighet med økende datastørrelse.

Acknowledgements

I would first of all like to thank my supervisor, Kjetil Nørvåg, for valuable guidance and support. Furthermore, I would like to thank my friends at Fiol; William Peer Berg, Håkon Åmdål and Mads Løkken Paulsen for valuable help and encouragement. I would also like to thank my girlfriend Kristina Albertsen for supporting me through the entire process.

Contents

Chapter 1

Introduction

This chapter contains the motivation and relevance for this thesis together with an introduction to the problem and research question.

1.1 Motivation

As a growing number of people worldwide connect to the internet, more and more people find their way onto social media services such as Twitter, Facebook, Youtube and Snapchat. The amount of data we collectively produce each day is enormous, and the potential value in this data is huge. Using this data is called social media mining, where the aim is to take the data and turn it into information. The information is then analyzed and with the result that information is turned into knowledge. Turning data into knowledge is the ultimate goal of the data produced in social media.

Sometimes we wish we could know as much as possible of someones footprint. In the case of emergencies, online advertising or localized search results, we want to be able to know the geographical user location to perform such tasks efficiently. A journalist sorting tweets related to an event might want to be able to differentiate which tweets are local to the event and which tweets are not. When doing opinion mining for a topic it could

be necessary to sort by country, state or city. In the world of advertising, it is valuable to be able to adjust to a users true location to be able to serve relevant content, or look for potential customers.

Looking into user geolocation also touches relevant topics of today like privacy and internet surveillance. The data left by our interactions on the internet can be used to track and monitor us. Studying how we can be tracked online can help people who wish to remain anonymous, as knowledge about surveillance can help us to take the necessary steps to hide our identity.

The main motivation behind this thesis is to create a system with the ability to predict a single tweets origin country by using the metadata from that tweet. Much work done with tweet geolocation builds a model where many tweets from a user is analyzed to be able to predict the location. However this approach has two main weaknesses: A user with very few tweets is almost impossible to accurately classify. Additionally, language models are not the perfect fit for a platform such as social media. First of all, the tweet itself does not contain a lot of information, being only 140 characters long. Secondly, social media moves at a swift pace, what people post and tweet about changes rapidly. When a big event happens, like the Olympics, Twitter will have tweets from the whole world talking about the Olympics. Only a few of those users are actually located at the true location of the Olympics. Similar observations can be done for many other events that have a temporal feature to them. Language models rely on methods that build on past knowledge that can quickly become irrelevant as what we tweet about changes together with headline news. In this paper, the metadata surrounding the tweet will be the focus of analysis to see if location can be inferred

without looking at what the user actually tweeted about. Then it should be possible to take a new tweet and by looking at the metadata we can predict the most likely country of that tweet, without having to gather many tweets from the same user.

1.2 Problem Definition

In this thesis, the problem of correctly classifying a tweets origin country from the tweets metadata will be attempted. Building on previous research within the field, a system will be built that can predict a tweets origin country using a machine learning algorithm

1.3 Contributions

The result of the work and research done in this thesis is a prediction model capable of classifying single tweets using tweet metadata. The approach will also show scalability as it maintains its performance when it is tested by a larger dataset. It is then evaluated and compared with other state of the art classifiers. While this method achieves a lower accuracy it has utility in that it can predict for single tweets compared to having to gather several tweets from a user before predicting accurately.

1.4 Research Questions

The following research questions will be answered in this thesis:

RQ1 What different features and techniques are commonly used when predicting a users geolocation?

RQ2 Is it possible to perform user geolocation prediction using only the associated metadata?

RQ3 How does the performance of the machine learning algorithm perform with increasing data size?

1.5 Report Outline

Following is a short list explaining the focus of each of the following chapters.

Chapter 2 introduces the landscape of this thesis, as well as theory and relevant technologies used to solve the research questions.

Chapter 3 looks at related work in geolocation of twitter users as well as work in similar fields.

Chapter 4 gives a detailed explanation of the two datasets used to perform analysis.

Chapter 5 contains the feature analysis together with an explanation of how the predictor system was setup.

Chapter 6 contains the results of the experiments and discuss their implication. Following future avenues of work is looked at.

Chapter 7 gives the conclusion to the thesis.

Chapter 2

Preliminaries

This chapter firstly contains a brief introduction of social media and twitter. Then it presents the technologies used together with general information about machine learning with Naive Bayes Classifier (NBC).

2.1 Social Media

Social media is a term for a huge field of different tools and applications that allows users to create, share and exchange information. In general, social media services normally have the following features: It is a Web 2.0 application that allows users to create a profile and connect with other users by developing their social network and it allows for sharing of user made content between users through posts or messages (Wikipedia, 2016).
[TODO se på ref]

Some of the biggest social media sites today are Facebook, WhatsApp, QQ, Instagram and Twitter. These applications have had a huge impact on today's society as they are part of the communication between billions of people. Privacy rights advocates warn users about how the information on social media can be gathered. Information is captured without the user's knowledge or consent through tracking and third party applications.

Social media mining is the process of using social media data

gathered to perform a variety of task such as modeling, analyzing or mine patterns. It is a huge research field encompassing areas such as community structures in social networks, sentiment analysis, spam detection and much more.

2.2 Twitter

Twitter is an online social networking service that enables users to send and read short 140-character messages, commonly referred to as "tweets". Twitter is ranked as the 10th most popular website in the world by Alexa rank as of January 2016 (Alexa, 2016). Registered users can both read and post tweets, while unregistered users can only read them. It is possible to use twitter through the website interface, mobile device applications or even through SMS. The service has around 320 million active users posting around 400 million tweets per day (@Twitter, 2016).

Twitter allows for several ways to access tweets. First, we have the Twitter Search API, where we can query tweets that have occured. The data available is limited by Twitter's rate limits. An individual user can only retrieve the last 3200 tweets, regardless of query criteria. The Search API is further limited by the amount of requests you can make in a certain time window, about 180 requests in a 15 minute period. Obviously this method is not a therefore not a viable option for gathering millions of tweets.

Secondly there is the Streaming API. Unlike Twitter's Search API where we are polling from tweets that have already happened, the Twitter's Streaming API is a push of data as tweets happen in near real-time. Using a set of criteria, like a keyword, username or location, tweets that match gets pushed to the user who sets up the query. However the Streaming API only allows for a 1% retrieval of all the tweets produced on twitter at a given

time. Once the number of tweets matching the given parameters would return more than 1%, Twitter begins to sample the data returned to the user. The exact method Twitter uses to sample this data is currently unknown. To overcome this 1% limitation, there is Twitter Firehose, which is very similar to the Streaming API. The difference is that the Twitter Firehose guarantees delivery of 100% of the tweets that match the user-specified criteria, but using this service costs a lot of money. Firehose also requires a lot of resources to handle all the incoming data. The Twitter Streaming API is sufficient for light analytics or statistical analysis, while Twitter Firehose should be used when it is vital that every tweet is gathered. For example as a security measure when for example professional sports teams play in a big arena or by a local police jurisdiction in monitoring an area.

So what is really the difference between the Streaming API and Firehose? Morstatter et al. (2013) investigates this closely by comparing Firehouse and the Streaming API using common statistical metrics as well as metrics that allow us to compare topics, networks and locations of tweets. What they found is that as the number of tweets matching the set of parameters increase, the coverage is reduced. One way to mitigate this is to set more restrictive parameters. This way they could extract more precise data from the Streaming API. Furthermore they discovered that the difference in the top n hashtags in a dataset is well estimated when n is large but is often misleading when n is small. They found strong indications that the Streaming API has some bias in the way it provides data to the user. When comparing the top n hashtags in sampled datasets to the Streaming API the sampled data exhibited very high positive correlations with the Firehose dataset while the Streaming API performed very poorly. This is a strong indication of a bias when selecting data for the

Streaming API. The larger the coverage of the Streaming API, the more accurate the topical analysis.

When Morstatter et al. (2013) looked at geotagged tweets from both sources they found that the Streaming API almost returns the complete set of the geotagged tweets. They attributed this to the geographic boundary box. The number of geotagged tweets is very small, around 1%. They conclude that researchers that use geographic boundary boxes should be confident they work with an almost complete set of Twitter data when sampling the data this way.

2.2.1 The Tweet

The twitter stream is a structured JSON stream of tweets accessible using the REST architectural style. Each tweet is returned containing a rich JSON tree that provides a lot of information in addition to the contents of the tweet. When looking at the JSON of a tweet it usually contains about 50-60 lines of actual JSON. A single tweet has information about the tweet itself, such as when it was created, how many times it has been retweeted, the unique ID, the text, the geographical information etc. Furthermore the JSON data contains a lot of information about the user who sent the tweet, about 35 lines of JSON. A detailed explanation of the structure is shown figure **??**. Some of the JSON attributes are designed for the twitter platform, and are not very helpful for most types of analysis. These fields should be removed before further analysis to avoid unnecessary overhead when dealing with the tweets.

A tweet contains a lot more than the 140 character limit when downloaded from the stream as a JSON payload provided by Twit-

ter. While the tweet itself is normally sized around 200 bytes, the additional data contained in the JSON file makes the size of a single tweet object come in at 2400 bytes, or about 12 times as large. This is because the twitter stream has many JSON fields for both the tweet and the user behind the tweet contained in the JSON of the tweet. Some of these are used inside the Twitter environment by applications, while some of the other fields can be very useful for data mining purposes.

2.3 Classification

Classification refers to the task of assigning labels to objects. Categorization would be another term for this task. In spam detection, we use classification to assign an incoming email to one of two labels, "spam" or "not spam". To do this, we could have a learning algorithm train on a training data, then use this knowledge to correctly assign "spam" or "not spam" to new incoming emails.

Training data is a set of already classified data. This is often done manually beforehand. In the case of spam emails, a dataset containing lots of email where they are all labeled either "spam" or "not spam" would be used to train an algorithm to recognize unique features of the two different labels, and use these when facing new emails.

When building the classifier system we use test and training sets to help build a good system with high performance. The classifier trains on the training data, and then uses that to attempt to classify the test data, another set where the correct answer is known, but hidden from the classifier. Normally a percentage split or k-fold cross-validation is used on the training

data to separate the data into test and training.

In percentage split we simply split the training data into two heaps, for example 40% training and 60% testing. This is a very simple approach but has a small drawback because the test and training set is not independent. The k-fold cross validation tries to address this by splitting the test data several times ensuring that all items are used for testing exactly once and the same number of times in training. Splitting training data into ten equal parts would see us use one part for test and the rest for training. Repeat this ten times, each time with a new part for testing. The results are then finally averaged to get the final result.

2.4 Naive Bayes

NBC is a basic technique to construct a classifier that can achieve strong results. It is one of the most efficient and effective inductive generative learning algorithms used in machine learning. It builds a vector of features where it assumes all features are independent. Naive Bayes classifiers can be trained very effectively through supervised learning, often using the maximum likelihood approach. Assuming that all the features are independent is in most cases a gross simplification, but despite this it has been shown that the NBC still work quite well in many complex real-world situations. In the paper "The Optimality of Naive Bayes" Zhang (2004) explains in more detail how the Naive Bayes approach can achieve such strong results despite its over-simplified assumptions.

In the Naive Bayes classifier the instance to be classified can be presented by a vector x with n features in 2.1

$$\mathbf{x} = (x_1, ..., x_n) \tag{2.1}$$

$$p(C_k|x_1,...,x_n) \tag{2.2}$$

$$p(C_k|\mathbf{x}) = \frac{(p(C_k) \bullet p(\mathbf{x}|C_k)}{p(\mathbf{x})} \tag{2.3}$$

$$C_{MAP} = \underset{c \in C}{\operatorname{argmax}}\, p(C_k|\mathbf{x}) \tag{2.4}$$

$$C_{MAP} = \underset{c \in C}{\operatorname{argmax}} \frac{(p(C_k) \bullet p(\mathbf{x}|C_k)}{p(\mathbf{x})} \tag{2.5}$$

$$C_{MAP} = \underset{c \in C}{\operatorname{argmax}}\, (p(C_k) \bullet p(\mathbf{x}|C_k) \tag{2.6}$$

$$C_{NB} = \underset{c \in C}{\operatorname{argmax}}\, (p(C_k) \prod_{x \in X} p(x_i|C_k) \tag{2.7}$$

This translates to 2.2 where k is the number of possible classes. If the number of features n grow large, it becomes problematic to use probability tables, therefore the equation is rewritten to Bayes' theorem, as seen in 2.3. This equation is further simplified when we treat the denominator as a constant, and remove it from our calculations. This is used in the classifier by obtaining the maximum a posteriori probability estimate (MAP), or C_{MAP}. This is shown in 2.6. We can remove the denominator as the class that maximizes the equation in 2.5 would also maximize it in 2.6 because the denominator is the same for all the different classes we compare. For Naive Bayes this simplifies to 2.7 when assuming independence between features.

Knowing what features to include, and which to omit is no simple task when it comes to using the Naive Bayes Method. There comes a point when to many features reduces the performance of the algorithm due to noise. Poor features with little value are counted equal to strong features, and if to many

weak features are added it will actually reduce the performance of NBC. When this happens it is necessary to experiment with the features to see what will yield the best result.

Naive Bayes was chosen because of its simplicity, but also because it has been shown to work on very large datasets. In "Scalable sentiment classification for Big Data analysis using Naïve Bayes Classifier", Liu et al. (2013) showed that NBC is able to scale up and analyze the sentiment of millions of movie reviews with a stable accuracy. Exactly which algorithms perform well with large datasets is not well documented, but researching this is outside the scope of this paper.

2.5 Apache Spark

Apache Spark (Spark) is an open source general cluster computing system currently being developed by Apache Software Foundation. It was originally developed in 2009 at UC Berkeley's AMPLab, where it was open sourced in 2010 as an Apache project. Spark aims to give a comprehensive and unified framework that fulfills big data processing requirements with a variety of data types. Machine learning is supported in Spark and it is an important piece of the puzzle when it comes to mining and analyzing Big Data.

Spark requires a cluster manager, which is usually a backend GUI or a command-line software, and a distributed storage system. Spark supports a wide variety of distributed storage systems, including Hadoop Distributed File System (HDFS), Cassandra, Amazon S3, Kudu, Openstack Swift or you can implement a custom solution. Spark also supports pseudo distributed local mode, where distributed storage is not required and local

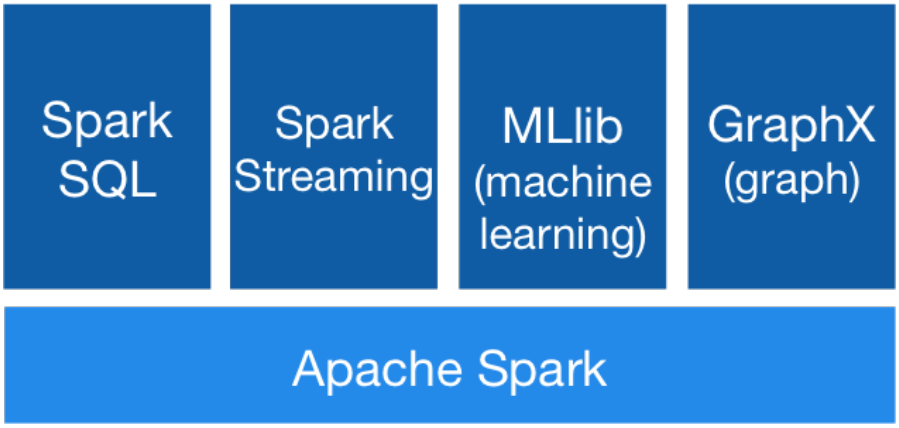

Figure 2.1: The Spark stack

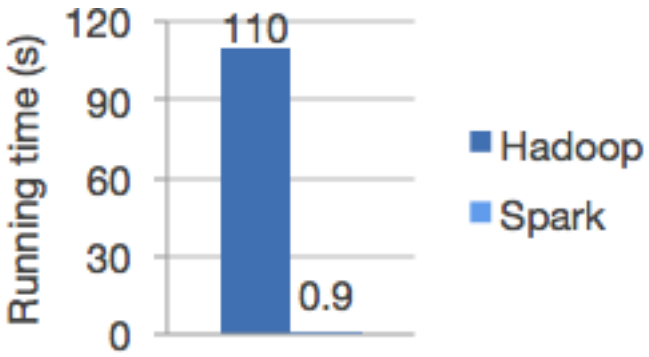

Figure 2.2: Comparing runtime when using data in memory

storage is used instead. This is useful for testing and development purposes. Spark can therefore be run on a single machine with one executor per core.

Spark boasts great speed compared to Hadoop MapReduce, running up to a 100x faster in memory, or 10x faster on disk. It is also easy to use as it supports development in Java, Scala, Python and R. Spark supports over 80 high-level operators that make it easy to build parallel applications using interactive shells.

Chapter 3

Related Work

This chapter will present some of the related work analyzed within the field of user geolocation prediction and other closely related areas.

3.1 Background

There has been much research aiming to predict a users geolocation. Accurate geolocation is for example the key driver for location specific services such as localized search, where Wang et al. (2005) looked to find a query's dominant location by using Location Indicative Words (LIW) in the search text. Using LIW is a common approach used to find a users geolocation on Twitter. In Australia, they use Twitter to monitor for potential wildfires in near real time, helping to prevent spreading and saving lives (Power et al.).

Identifying a Twitter user's real location is non-trivial, mainly because of the lack of reliable geographical information. In the location field users can declare their location by writing whatever they want, and using this field trivially has been shown to be ineffective (Hecht et al., 2011). Furthermore Hecht et al. (2011) also showed that the country or even state could in most cases be determined easily with decent accuracy from just the contents of a users tweet. However they noted that their model could

benefit by inclusion of user metadata. This approach was used mostly for English tweets, but others have used language models on all languages with some success, however fine tuning the approach for specific languages are needed, especially those using different letters (Han et al., 2014).

In "You are where you tweet: a content-based approach to geo-locating twitter users" Cheng et al. (2010) collected tweets of users and used LIW with k-means algorithm to place users on a city level accuracy. They found that after collecting 100s of tweets of a single user they could accurately place most users (51%) within 100 miles of their actual location. The method went beyond simply using Named Entity Recognition (NER), as it tried to exploit words known to be primarily used in particular regions as well as mentioning locations in the tweets. This approach relied only on the content of the tweet, and did not utilize IP information or external knowledge bases. Using IP addresses to get the location of a user has been shown to achieve around 90% accuracy (Padmanabhan and Subramanian, 2001), however such methods are not applicable to Twitter and other social media sites where such information is not available through the public API's. While maybe Twitter itself has access to this information, any third party looking to use data from Twitter will have to use other sources to identify the geolocation of a user or tweet, for example the the content of the tweet or metadata information.

However the main disadvantage with using NER and LIW is when trying to classify user with few or only one tweet. Tweets are only 140 characters long, and many tweets will simply not contain any information that will help us determine the geolocation. Also another problem is that language and dialects are

constantly evolving, and slangwords or even foreign languages and emojis are difficult challenges for language models to tackle and properly reflect. Knowing that *melbo* is short for Melbourne or that *barteby* is a nickname for Trondheim is not trivial to solve. Language models also suffer from the temporal nature of tweets. Work done by Priedhorsky et al. (2014) found that after four months they had an additional 6% error to their predictions. They speculated that this was mostly due to n-grams relating to current events quickly becoming irrelevant.

Some have therefore sought to integrate other sources of information like temporal features. Li et al. (2011) created a point of interest tag of a tweet based on the content of the tweet as well as the time of posting. Others have tried to include friendship relations. Backstrom et al. (2010) argue that geography and social relationships are inextricably intertwined. The people we interact with the most are often people that live near us, and this he argue is also reflected in the online world. However building such a network on Twitter might prove difficult as the API used in this paper only gives us a limited amount of tweets and not the full stream, building such a complex social graph of a Twitter user would maybe be impossible. And while on Facebook a user is expected to connect and interact with their friends and family, this is not necessarily the case on Twitter. Many users follow and interact with people due to their interests in various topics such as politics, news or sports.

Another approach to exploit the geographical properties of words is to use a uniform grid overlaid on the earth. A set of pseudo-documents is created within a given grid cell by concatenating the documents within it. Then a location for a test document is chosen by its most similar pseudo-document. This

approach have been improved by Roller et al. (2012) to use adaptive grids where each grid cell contains approximately the same amount of documents. Urban grid cells would have a lot more documents than rural cells which would make correctly classifying documents to these cells almost impossible. The adaptive grid outperformed the uniform grid on large Twitter corpus.

Roller et al. (2012) also experimented with using non-geotagged tweets where the location of a user could be inferred from geotagged tweets in its training data, but was unable to see if this improved accuracy. Han et al. (2014) investigated the utility of using non-geotagged tweets and found that accuracy improved when incorporating non-geotagged data into their test and training set. User geolocation predictions' ultimate goal is to reliably predict the locations of users where the location is unknown. Han et al. (2014) argue that by using non-geotagged tweets from users where there is also geotagged tweets would better represent the overall datastream from Twitter, as non-geotagged data would be able to be sent from a wider range of devices. They did not look at the difference between users that have both non-geotagged and geotagged and users who have only non-geotagged, because it is difficult to known the true location for users that only have non-geotagged data.

3.1.1 Summary

Most of the work done on predicting user geolocation on Twitter is based on analyzing the content of user tweets to find geographical information located in the language used. Very few have used the included metadata, and those who did only use it as support and have not done a thorough analysis of the embedded features found in the tweet JSON file. Also the method used generally gather many tweets from a user before trying to pre-

dict their location. However in many cases this approach is not good enough if you only have one or very few tweets of a user.

Chapter 4

Meet the Data

In this chapter we will look closer at the two datasets used for geolocation prediction in this paper, Dataset 1 and Dataset 2.

4.1 Dataset 1

The first twitter dataset is simply called Dataset 1, the smaller of the two datasets used. All initial testing was performed on Dataset 1 before moving on to Dataset 2. In 4.1 we can see Dataset 1 contains 2.613.661 tweets. It is gathered from Twitter using the Streaming API over 5 days from October 03.2015 to October 08.2015. It was set up to gather all tweets matching the criteria of having a geolocation within a bounding box consisting of Central Asia and Western Asia, as can be seen in 4.6. In 4.6 and C.1 the tweets are represented by the red squares. Each square is the bounding box of a single tweet. Notice how the tweets bounding boxes have varying sizes, the smallest bounding boxes covering no more than a few streets in a city, while the biggest bounding boxes are seen covering entire countries.

To create the visualization shown in 4.6 we needed to extract the geolocation contained inside each tweet. To do this a custom python filter was made to extract the coordinates as seen in B.4. However the output of this filtering is not compatible with google.maps. A combination of Python code and scripting

in notepad++ was needed to change the format. The JavaScript code used to generate the map can be seen in B.2

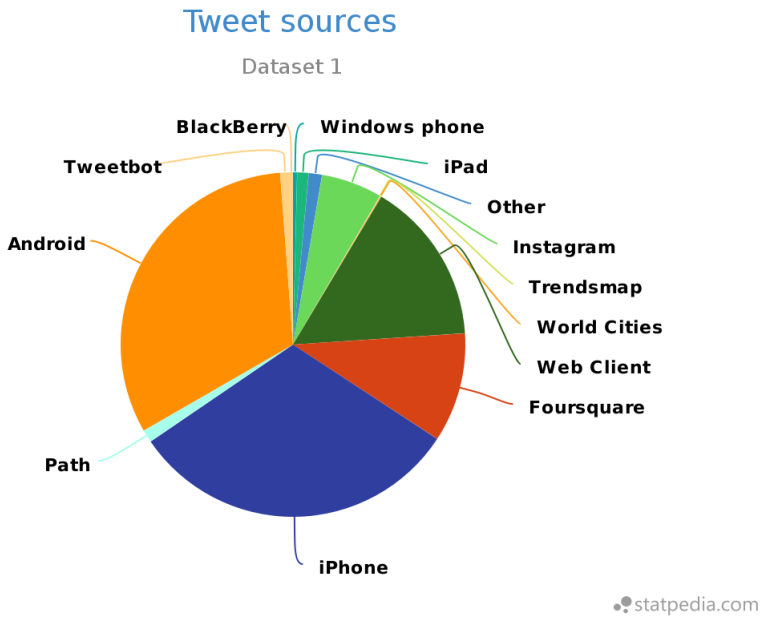

Figure 4.1: Distribution of source applications in Dataset 1

Table	Size	Number of tweets
Dataset 1	433.302 KB	2.613.661
Dataset 2	3.917.847 KB	23.304.498

Table 4.1: Table showing the two datasets

The tweets that have the largest bounding boxes which envelops entire countries have been filtered out to avoid cluttering the final representation of the tweets. Furthermore, the figure 4.6 is only a sample of the full set. It was built by extracting the bounding box from 23000 tweets, where about 800 got filtered out. Showing all the tweets is not impossible, but it would be very computationally expensive to represent all 2.6 million tweet with the JavaScript code used to make it. In any case, figure 4.6 serves as a nice way to visualize the dataset. We can also observe that not all countries have tweets in them even though they are within the bounding box set up in the Streaming API to gather tweets. The exact reason for this is unknown, but one could speculate it is because of filtering or internet censorship.

Only China and Iran have banned Twitter in their countries, but it is known that a lot of people manage to use twitter in China Bamman and Smith (2012), but whether you can send exact GPS locations in these countries is unknown.

```
1 "PK", "Sat Oct 03 15:48:13 +0000 2015", "<a href=\"http://twitter.com\"
     rel=\"nofollow\">Twitter Web Client</a>", 650336561041686528, "und", "Karachi",
     "en",
2 "TH", "Sat Oct 03 15:48:13 +0000 2015", "<a href=\"http://twitter.com\"
     rel=\"nofollow\">Twitter Web Client</a>", 650336561549066240, "en", "Bangkok",
     "en",
3 "SA", "Sat Oct 03 15:48:13 +0000 2015", "<a
     href=\"http://twitter.com/download/iphone\" rel=\"nofollow\">Twitter for
     iPhone</a>", 650336561750478848, "ar", null, "en",
4 "KW", "Sat Oct 03 15:48:13 +0000 2015", "<a
     href=\"http://twitter.com/download/android\" rel=\"nofollow\">Twitter for
     Android</a>", 650336561347760128, "en", null, "en",
5 "UA", "Sat Oct 03 15:48:13 +0000 2015", "<a href=\"http://instagram.com\"
     rel=\"nofollow\">Instagram</a>", 650336562098520065, "en", "Kyiv", "en",
```

Listing 4.1: Result of the filtering

At first the tweets contained a lot of unnecessary information. An example of just a single tweet can be seen in B.1. After filtering the raw JSON stream, the tweets were left with just a few fields that were to be used for analysis. This can be seen in 4.1. First there is the country code field, which is extracted from the place(country_code) in the JSON file. This field is automatically filled in by Twitter based on the bounding box of the tweet. The country code is later used as the ground truth for training the algorithm. Next is the creation date of the tweet, down to the second. The next field after that is the source field, followed by the automatically detected tweet language, then timezone and lastly the application language.

In figure 4.1, 4.2 and 4.3 different feature distributions of Dataset 1 can be seen. Exact numbers can be found in the appendix, D.1 and D.3 The source of the tweets is dominated by Android and iPhone, with Foursquare and Instagram right behind. The first two are simply people using the twitter application on ei-

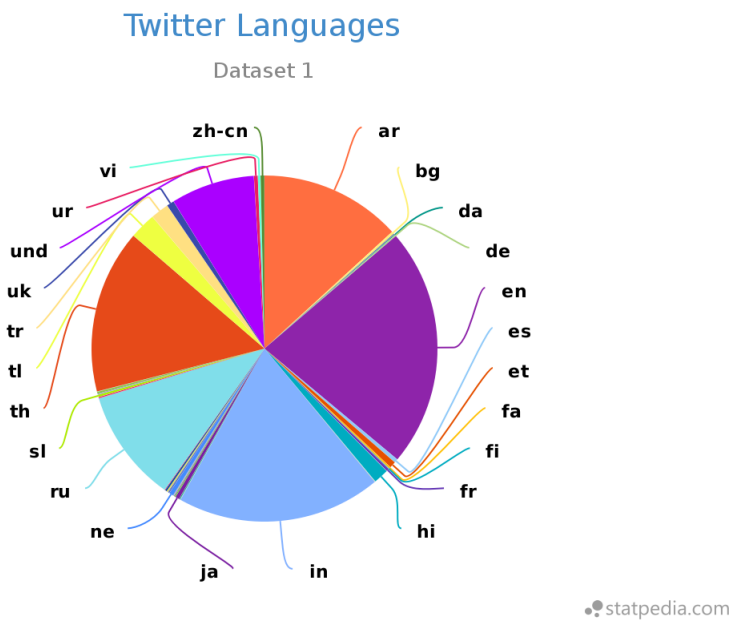

Figure 4.2: Machine detection of tweet language

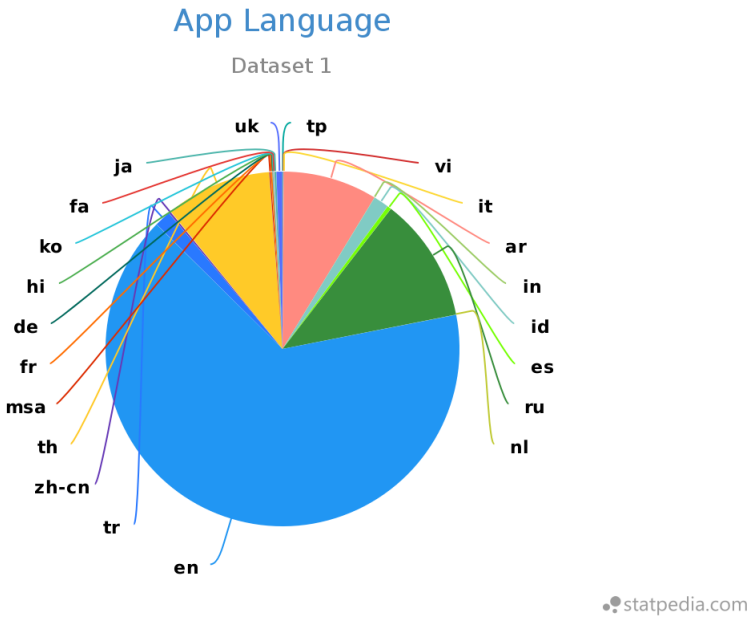

Figure 4.3: Interface language of users sending tweets

ther an Android or an iPhone. Instagram and Foursquare are two mobile applications. Instagram is used to share pictures, and modern phones automatically includes GPS coordinates in the Exchangeable Image File (EXIF) along with device information and timestamp. When connecting Instagram with Twitter, this information is then automatically included unless you tell

the application not to. Foursquare is a very popular location search application, that provides users with recommendations when traveling new places based on what they preferred in the past. When people "check-in" their location is shared on Twitter by tagging the place they went to, even if the user does not normally share their location through tweets.

In 4.2 the different languages in the tweets, detected automatically by Twitter, are shown. Dataset 1 have 6 major language features. The biggest features are Thai (th), Russian (ru), Indonesian (in), English (en), Arabic (ar) and undefined (und). For a complete list of all the language abbreviations see ISO-639. Undefined are all the tweets where Twitters algorithm fails to detect a language. A tweet containing a link, or just emojis would be a typical tweet that is difficult to assign a language.

4.2 Dataset 2

Dataset 2 is a lot larger than Dataset 1, containing over 23 million tweets. It is also arguably more interesting than Dataset 1, because it contains tweets from the entire world instead of from a limited part of the world. It is a combination of 4 different bounding boxes catching tweets over Eastern Asia, North and South America, Europa and Africa, and Western and Central Asia.

In 4.4 and 4.2 some of the different features are visualized. For completed data-tables on them see D.2 and D.4. The largest language in Dataset 2 is by far English with about 36% of all tweets. The other major languages with over 5% coverage are Spanish (es), Indonesian (in), Japanese (ja) and Portuguese (pt). The number of undefined tweets is about the same at 7%. The source distribution is pretty similar to Dataset 1.

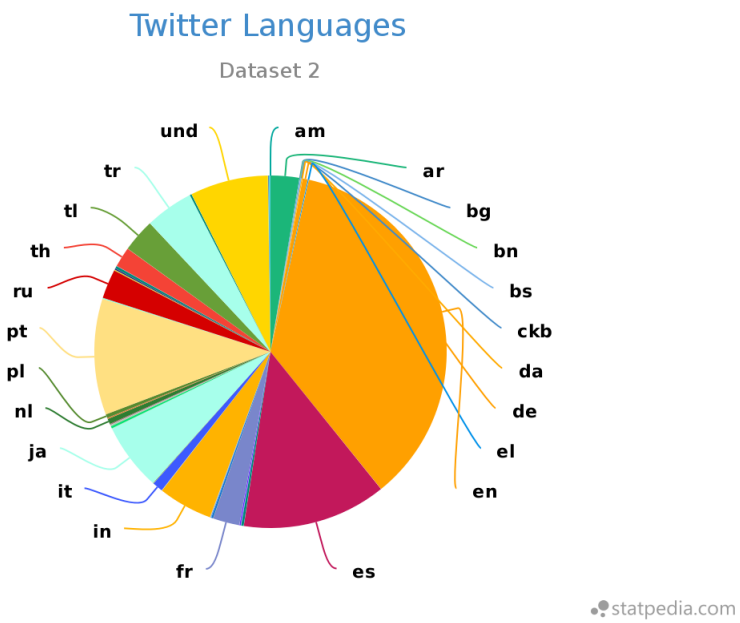

Figure 4.4: Machine detection of tweet language in Dataset 1

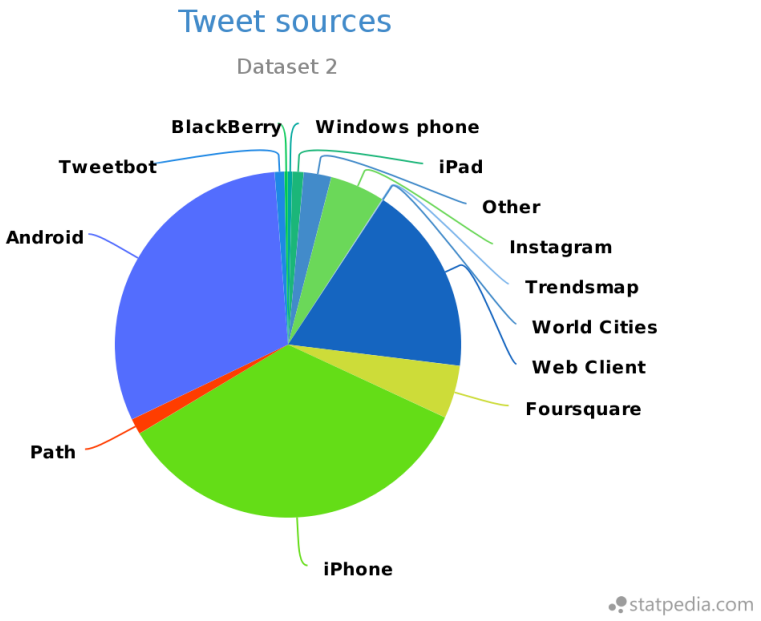

Figure 4.5: Distribution of source applications in Dataset 1

4.2.1 Filtering the Datasets

Before we can begin any form of analyzing we need to filter the dataset so that it becomes easier to work with. Since a tweet is plain JSON, we can use python with the JSON library to easily select which JSON-fields we want to extract from each individual tweet. This allows us to create a simple way to filter out the

fields we are interested in quickly, and it also drastically reduces the size of the dataset. After this first filtering our Dataset 1 is now only 0.41 GB, down from 7GB

However to use the data in our statistical analysis we need to represent it differently, we therefore run our now filtered data through another filter to convert it into a suitable format for our machine learning algorithm. Here we have to make decisions in how we filter the different features of each tweet and how best to represent them. In B.7 we can see how the filter will have to look through a giant table to find the correct match for the fifth element. Once it finds a match it will convert the timezone into a series of 0's and a single 1 representing the timezone as a feature. Once filtered the result can be viewed in B.8. The first integer represents the class, which is the country. It has also been filtered in a similar way to the code in B.7.

So now a single tweet has been reduced to a very compact string of integers that can be utilized in the ApacheSpark Machinelearing NBC library. We create seperate files for every feature and combinations of features for later use. The dataset containing the filtered time-zone data is 0.14GB, which is under 2% of the original size.

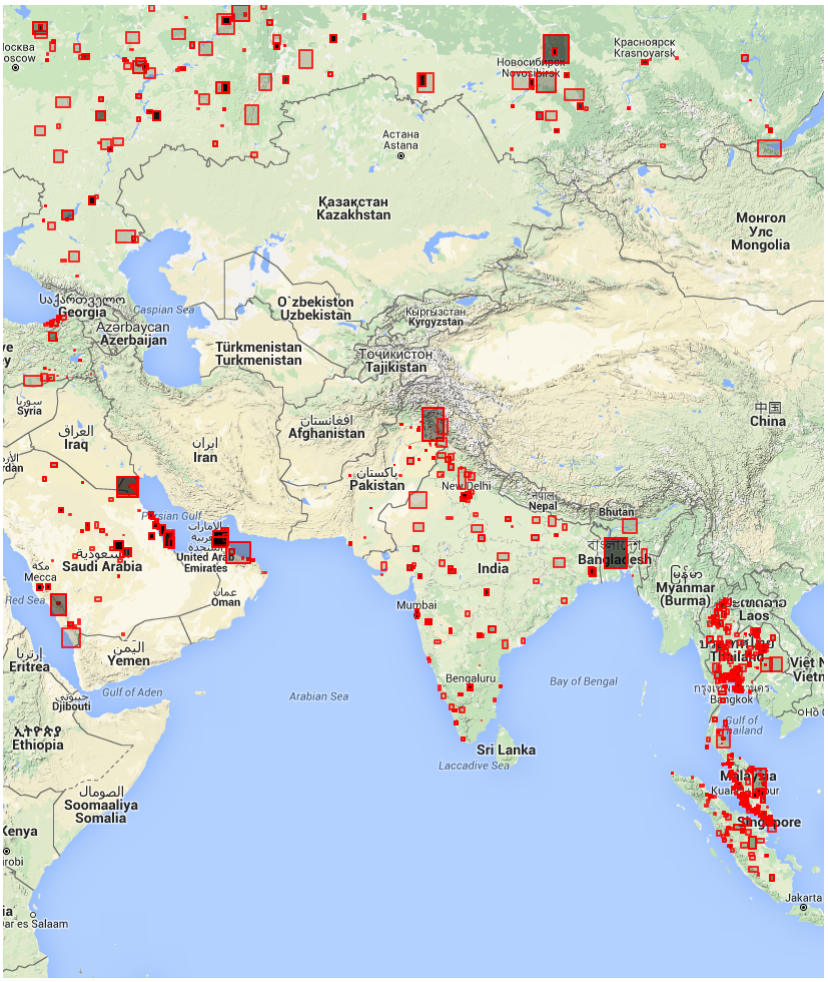

Figure 4.6: View of geo-tweets in Dataset 1

Chapter 5

Building the prediction system

This chapter will show the evaluation of the features as well as introducing the prediction system built to find user geolocation.

5.1 Feature Selection

Selecting which features that would be interesting required some additional analysis. In B.3 we can see the code used to filter. There are about 50 possible features that can be extracted from a tweet, but most of these do not have any significant information (profile_text_color for example). Features that have geographical information tied to them are of highest priority. Two types of language features as well as the two types of time zone features "time zone" and "UTC-offset" have geographical information bound in them. However the two language features are slightly different, while the time zone and UTC-offset are similar, except for UTC-offset having slightly less information as two similar timed time-zones would give the same UTC-offset. The language directly connects the user to an area because languages follow country borders for the most part, except for English. The time-zone feature also connects the user to a country in many cases, as there exists many time zones for each offset.

Secondly temporal features could be interesting, as tweeting frequency are known to follow a daily pattern, and this could

help to guess where the tweet is from statistically. For example Portuguese is tweeted in both Brazil and Portugal, but since the countries are in different time zones the countries would have different peak hours. This feature could help us if the time zone field is empty.

Many social media services allow the user to write their location. Such data is known to be unstructured and ad hoc. A study from 2012 showed that 34% of users did not provide real location information Hecht et al. (2011). And even though many users provided a location, they used slang and abbreviations, making the task to correctly identify a users location through such a field non-trivial. Some users might also want to hide their true location, and might even purposefully provide wrong information where possible. Therefore the location field provided by the user was not used, as many have already tested it's usefulness for geolocation on Twitter.

Other features that potentially could have information are a users friends, who they are following, and who follows them. This information could help build a bigger picture around a user, but it is not available directly from the Twitter stream. Therefore these types of features are not possible with the restrictions of this experiment. Other features that change between users but are just random and not considered relevant are features like friends count, amount of followers etc. In table 5.1 the features are evaluated based on predicted temporal information.

5.1.1 Evaluating the Features

Creation time would follow the day and night cycle, as there are certain times during the day where users tweet more. Users tweet

Feature	Predicted Iinformation
Creation time	Low
Source	Low
Tweet language	High
User language	Medium
Time zone	High
UTC offset	Medium

Table 5.1: Table showing the features extracted

more during the day than night, especially during morning and evening. Sorting this feature by the hour seems most natural. However the biggest country in a given time-zone would dominate this feature and would not help solve that many cases that other features could tackle better. *The source* feature is the device used to create the tweet. There are many options for a user when tweeting, as they could tweet from their phone (android/iPhone/windows) or from their PC or using third-party applications. This is all represented in the source field. This field is expected to be dominated by a few actors and therefore not really going to help much in determining the location of the user, however lesser known applications might be popular in only a few countries, which in turn can help with the prediction.

The *tweet language* is a machine detected field that Twitter provides for us. Twitter uses an unknown algorithm to detect the language the user wrote in the tweet. This field is very helpful as most languages follow country boundaries. English will still be a big problem though as most tweets are in English. *User language* is the language in the source application used by the user. If someone were to use the Twitter application with Norwegian as the interface language, this would show up in this field. As with tweet language this field will be very useful when it gives a language other than English. Most users will however use an application in the default language, and not all languages are even supported. Therefore it is expected this feature will have even

more English hits than the tweet language. *Time zone* provides the time zone used on the device sending the tweet. Most users have a default time zone provided through their internet connection, which will give a strong indication to where the user is located when sending the tweet. *UTC offset* gives us some of the information already given in time zone, without the ability to differentiate between countries in the same time zone. Potentially useful, but not needed if time zone is extracted.

5.1.2 Choosing the Features

The features were picked for their uniqueness and potential. UTC offset and time zone were similar so time zone got picked. Both language features were interesting, so they were also picked. Source was arguably a very low potential feature, but since it shows no overlap with other features it was included. The creation time feature was not included because its potential information was low, and it somewhat overlaps with the time zone feature. In table 5.2 the features are listed together with their dimension numbers.

Feature	Dimensions
Source	13
Tweet language	70
User language	70
Time zone	24

Table 5.2: Table showing the features chosen and the number of dimensions of each feature

5.1.3 Representing the Features

The time zone field is somewhat similar to the user interface language as it contains a lot of time zones. In Dataset 2 there was over 420 different time zones found, however some of them are duplicates with slightly different ways of representing the time zone, "Kuwait" or "Asia/Kuwait". If used correctly, this field

should be very useful to determine the users origin country. We considered two different approaches, either each time zone is treated as a unique feature, or the world is separated into 24 parts and have the different time zones grouped together based on UTC offset. The first method is undoubtedly the best, but would result in a very large feature.

When choosing how to represent the features there were two considerations: Potential value and size of feature. In hindsight UTC offset could have been used instead, but once the setup for time-zone was done there was no need to go back and extract the other. Both the language features were used used in a similar manner where each unique language was a separate feature. The source feature the 12 biggest features got its own feature while the rest was gather in a 'other' field. See table D.1 to table D.5 for an overview of these features.

5.1.4 Analyzing the Dataset

After the dataset had been filtered and was ready to be analyzed we loaded the dataset into apache spark. The NBC was chosen because of its simplicity and the fact that it had been shown to work well on very large datasets before (Liu et al., 2013). NBC was included in the source code in Spark, and only small modifications was needed to make it run with our dataset. Once running, the dataset is converted into RDD's and the driver program of Spark runs the NBC. Spark was used so that it would be easy to perform the analysis on even larger sets of data in the future if needed. Since Naive Bayes is relatively fast to run, the analysis took no more than a couple of hours to run on the biggest set with all the features included.

Chapter 6

Experiments and results

In this chapter the different experiments will be presented as well as their results. Then the results are evaluated and discussed.

6.1 Experiment Setup

The tests were run on a single laptop using Apache Spark MLlib, with a training and test data split randomly 60/40 regardless of data size. Each run on a specific setup was repeated 5 times to avoid outlier results. Since the data was written sequentially it was important that the training/test data was randomly split to avoid heavy bias.

6.1.1 Testing: Scalability

Initial testing was on the Dataset 1 to see the effectiveness of the different features when run on variable data sizes. The data was split in different number of tweets: 30000, 0.5M, 1M, 1.5M 2.0M, and 2.6 M were all tested. This tested the scalability of the NBC on the dataset, as well as possible overfitting problems with the chosen testing regime. Figure 6.1 shows the result of this test. The tweet language feature performs the best, while the source feature the worst, just barely improving the baseline. The features perform best at the lowest data size, but the result are stable once the set grows larger.

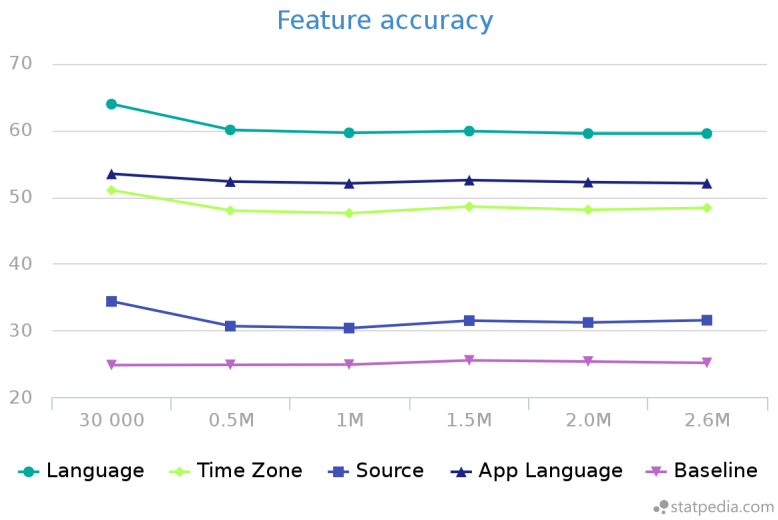

Figure 6.1: Single Features compared vs baseline

6.1.2 Testing: Combined Features

Next the combination of features needed to be tested, again on different number of tweets, to see how accuracy would scale. This test was also performed on Dataset 1. In 6.2 the results of this combination can be seen. The best combination was using all four features together, achieving a stable 76-75% accuracy. The features again perform slightly better when the dataset is small, and stabilizes as the dataset size increases.

6.1.3 Testing: The Large Dataset

Confident that the features scaled with the data size, the features needed to be tested on Dataset 2. Trials were run on the combination of all features, and the single features alone. Since it took a lot longer to run the algorithm on 23.3 million tweets these simulations were run only three times. For a more detailed rundown of these tests see table D.5. Results can be viewed in 6.3 where the Dataset 2 results are compared to Dataset 1 results. The results were very similar to Dataset 1, even performing slightly better since the baseline was lower. Differences can

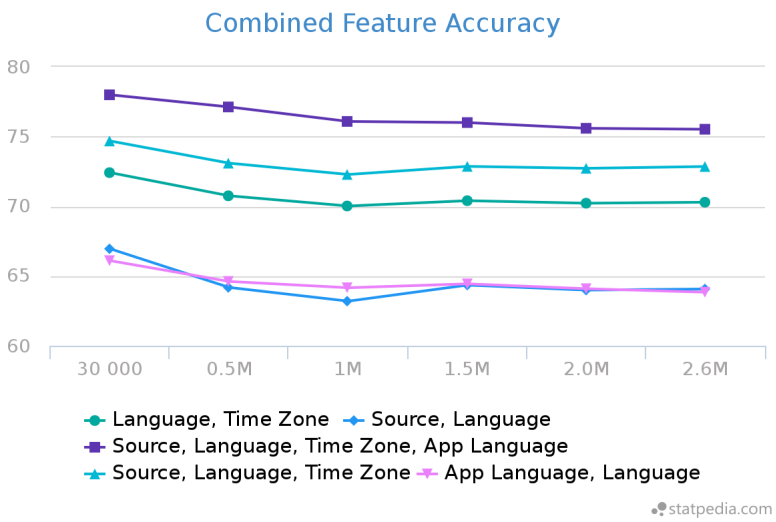

Figure 6.2: Combined feature accuracy in Dataset 1

be seen in the source feature where Dataset 2 performed worse, while the application language feature performed better in Dataset 2.

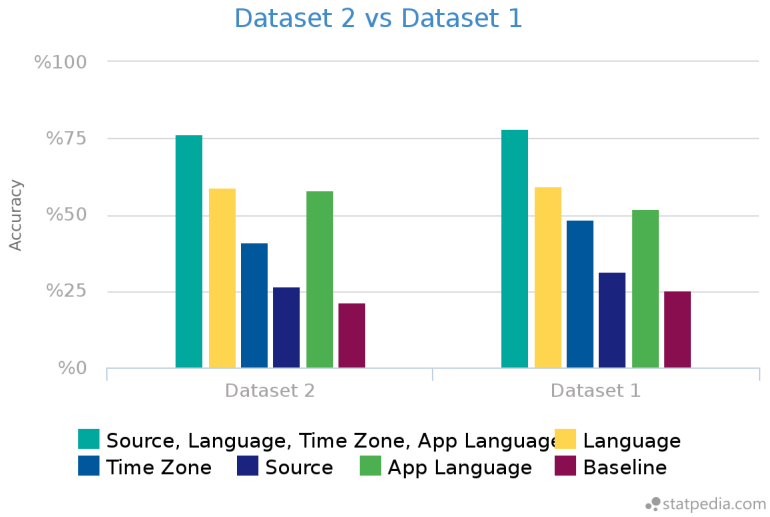

Figure 6.3: Table showing Dataset 2 results on 23.3 million tweets compared to Dataset 1 results

6.2 Evaluation and discussion

The results demonstrates the power of metadata, achieving high accuracy on a large dataset with a relatively easy machine learning algorithm. Our results compared with more advanced methods (table 6.1 show that there is still more that can be done to

achieve higher accuracy. It was difficult to compare the results to a lot of other papers because they either only focused on geolocating users in USA or they did not mention their country accuracy in the results.

6.2.1 Scalability

Why does the feature accuracy score the highest on a low number of data then fall slightly before stabilizing? There could be several factors influencing this. Firstly, the 30000 test sample is split into 60/40% training/testing. Liu et al. (2013) showed that when datasets become small the training data are not big enough for the model to learn enough knowledge about each class. In their experiment they saw the same initial rise in accuracy before stabilizing as the dataset grew large. Another factor could be overfitting. When the dataset grow large, smaller classes never gets selected as a possible candidate because of the bigger classes dominating the smaller.

6.2.2 Improvements & Future Work

What could have been done better to improve accuracy? As previously mentioned, some of the features are not implemented to their full potential. The *time zone* feature is implemented in a grid fashion, dividing the earth into 24 different grids. Using this feature more precisely letting each time zone be represented should see an increase in accuracy from this feature.

Twitter returns a lot of undefined results for their language detection field. It might be that the Twitter algorithm is conservative in its estimates wanting a high precision while sacrificing recall. Looking into improving this feature would help a lot to

achieve higher accuracy.

Doing a proper feature analysis of the remaining untested metadata could help increase accuracy. The *creation time*, feature was not tested in the experiments. Another feature that could be looked at is the location field, where the user can write anything.

Author	Country Accuracy	Baseline Accuracy
Dataset 2	76,59%	21,40%
Han et al. (2014)	93,30%	60,00%
Hecht et al. (2011)	72,71%	25,00%

Table 6.1: Table comparing result of different papers geolocation user country

Another thing that should be explored is the effect of eliminating more than one tweet per user in the datasets. This would remove some of the bias in the current results due to the possibility of many tweets from one user affecting the results. At the same time, it would be possible to compare the metafields of each user and look for differences within the same user. If the user has varying metadata then it could be a traveling user. These users could then be removed to potentially get more precise results.

Han et al. (2014) found that splitting the data to create language-partitioned models for geolocation have shown improvements. Creating a separate classifier for each language would then create even stronger results, which is desirable. This approach is the *one-against-many* scheme where we begin with say "Class A" and "all else". The result from "all else" is returned to the algorithm for a new classification into "Class B", and so on.

It would be interesting to look at discriminative models compared to generative models. Discriminative models might perform better than the generative model, because the number of

Author	Time Zone	Language	Baseline
Dataset 2	41,02%	59,03%	21,40%
Han et al. (2014)	56.5%	77.2%	60,00%

Table 6.2: Table showing the result of testing on Dataset 2

features are still relatively few compared to the amount of training data available. Ng and Jordan (2001) showed that while initially generative models such as naive Bayes perform better, as the number of training examples increases the discriminative logistic regression algorithms perform better.

Another case is the inclusion of non-geotagged data. The analysis done here is only using geotagged tweets. This has some obvious flaws, because this leaves out a range of devices that can tweet, but does not have a GPS. Is the data between geotagged and non-geotagged similar? The goal of this technology is to classify tweets that do not have a geolocation. It would be important to analyze for this. Han et al. (2014) did some experiments on the differences between the two datasets but the results were inconclusive as to whether there was a significant difference, so this would be an important issue to look further into for accurate geolocation prediction.

Chapter 7

Summary and Conclusions

7.1 Summary and Conclusions

In this paper the task of finding a Twitter user's geolocation from one tweet has been explored. The users geolocation have been predicted by extracting metadata features from the tweets gathered from Twitter using the Streaming API. The generative model of NBC was used on a dataset consisting of 23.3 million tweets spanning the entire world. The classifier's accuracy stabilized as the dataset grew in size. It eventually predicted a country accuracy of 76.57%. Compared to more advanced classifiers the metadata approach achieves a lower accuracy, but the approach is certainly viable. It can predict tweets without having to gather several tweets from a user, which is something most language based approaches will struggle to get high accuracy. There are also many avenues to take to improve on the current implementation to achieve a better accuracy.

Glossary

EXIF Exchangeable Image File. 26

HDFS Hadoop Distributed File System. 14

LIW Location Indicative Words. 17, 18

MAP maximum a posteriori probability estimate. 13

NBC Naive Bayes Classifier. 7, 12, 14, 29, 35, 37, 43

NER Named Entity Recognition. 18

Spark Apache Spark. 14

Appendix A

Spark

At its core, every Spark applications consists of a driver program. The driver program launches various parallel operations on a cluster. Furthermore the driver program consists of the applications main function. The driver program can be just a shell, or it can be an application that invokes Spark. To access Spark, the driver program uses the SparkContext object, which represents a connection to a computing cluster. SparkContext builds RDDs, which are immutable distributed objects that support two types of operations: transformations and actions. Transformations simply construct a new RDD from a previous one, for example if we filter data. This makes sense since RDDs are immutable, meaning they cannot be changed once created. Actions compute a result based on an RDD and either return the result to the driver program or save it to an external storage system. Returning the first element in an RDD is an example of an action.

These two operations are different because of an important concept in how Spark computes RDDs. Defining a new RDD can be done at any time, but Spark only computes them in a lazy fashion, such that they are only computed once they are used in an action. This approach saves disk space and computing time. Consider if we create a new RDD object consisting of the text in a book. Then we perform an action on this book where we filter

out certain words. If Spark where to load and store all the lines in the book as soon as we created it, it would waste a lot of storage space, given that our next move after creating the object is to filter out words. Spark waits until it sees the whole chain of transformations so that it can compute just the data needed for its result.

RDDs are actually not even stored once used, they are by default recomputed each time you run an action on them. If we want to reuse a RDD in multiple actions we can ask Spark to persist it using `RDD.persist()`. After computing the RDD the first time using persist, it will be stored in memory, partitioned across the machines in the cluster, and reused in future actions. Not persisting by default avoids storing data that is used only once to compute a result, which is an important concept when working with huge amounts of data. This is also a limitation, as it is up to the user to know when to use persist or when to not for maximum efficiency.

RDDs are created in two ways: By loading an external dataset or parallelizing a collection in the driver program. The simplest is to parallelize, however it is not very useful outside of prototyping and testing as this requires the entire dataset to be in memory on one machine. However it can be used in a distributed fashion if the files are available at the same path on all nodes in the cluster. Some network filesystems (NFS, AFS) appear as a regular filesysem. However putting the files on a shared filesystem is generally faster. HDFS, NFS or S3 are all examples of shared filesystems.

Another important feature is how we can work with RDD's of key/value pairs, a common data type required for many opera-

tions in Spark. RDD's containing key/value pairs are called pair RDD's. Pair RDD's allow for actions that can act in parallel on each key or regroup data across the network. "reduceByKey()" is a method that can aggregate data separately for each key, and a join() method that can merge two RDD's together by grouping elements with the same key. This way we can extract fields from an RDD and use them as keys in pair RDD operations, like a customer ID or an event time.

Appendix B

Listings

```
{
  "in_reply_to_status_id_str": null,
  "in_reply_to_status_id": null,
  "created_at": "Sat Oct 010 15:44:44 +0000 2230",
  "in_reply_to_user_id_str": "100000000",
  "source": "<a href=\"http://twitter.com\" rel=\"nofollow\">Twitter Web Client<\/a>",
  "retweet_count": 0,
  "retweeted": false,
  "geo": null,
  "filter_level": "low",
  "in_reply_to_screen_name": "Someuser",
  "is_quote_status": false,
  "id_str": "999999999999999999",
  "in_reply_to_user_id": 888999999,
  "favorite_count": 0,
  "id": 99999999999999999,
  "text": "This is a tweet",
  "place": {
    "country_code": "NO",
    "country": "Norway"
    "full_name": "Trondheim, Norway",
    "bounding_box": {
      "coordinates": [[[69.328873,
      27.708226],
      [69.328873,
      34.019989],
      [75.382124,
      34.019989],
      [75.382124,
      27.708226]]],
      "type": "Polygon"
    },
    "place_type": "Example",
    "name": "Trondheim",
    "attributes": {

    },
    "id": "0000000000300008",
    "url": "https://api.twitter.com/1.1/geo/id/0.json"
  },
  "lang": "en",
  "favorited": false,
  "coordinates": null,
```

```
  "truncated": false,
  "timestamp_ms": "1000000000000",
  "entities": {
    "urls": [],
    "hashtags": [{
      "indices": [17,
      37],
      "text": "Example"
    }],
    "symbols": []
  },
  "contributors": null,
  "user": {
    "utc_offset": 10000,
    "friends_count": 999,
    "profile_image_url_https":
        "https://pbs.twimg.com/profile_images/00000000000000000000000000000006/csXkkSCx_normal.jpg",
    "listed_count": 3,
    "profile_background_image_url":
        "http://abs.twimg.com/images/themes/theme18/bg.gif",
    "default_profile_image": false,
    "favourites_count": 3000,
    "description": "\u200f\u200f\u200f\u200"
    "created_at": "Wed Oct 01 06:53:57 +0000 2000",
    "is_translator": false,
    "profile_background_image_url_https":
        "https://abs.twimg.com/images/themes/theme18/bg.gif",
    "protected": false,
    "screen_name": "Example user",
    "id_str": "000003910",
    "profile_link_color": "038543",
    "id": 000003910,
    "geo_enabled": true,
    "profile_background_color": "ACDED6",
    "lang": "en",
    "profile_sidebar_border_color": "EEEEEE",
    "profile_text_color": "333333",
    "verified": false,
    "profile_image_url":
        "http://pbs.twimg.com/profile_images/000000000000/csXkkSCx_normal.jpg",
    "time_zone": "Oslo",
    "url": "http://someurl.com",
    "contributors_enabled": false,
    "profile_background_tile": false,
    "profile_banner_url": "https://pbs.twimg.com/profile_banners/190000010/000000",
    "statuses_count": 36000,
    "follow_request_sent": null,
    "followers_count": 300,
    "profile_use_background_image": true,
    "default_profile": false,
    "following": null,
    "name": "TeamTwitter#",
    "location": "Oslo",
    "profile_sidebar_fill_color": "F6F6F6",
    "notifications": null
  }
```

Listing B.1: Example tweet

```
    function initMap() {
        var map = new google.maps.Map(document.getElementById('map'), {
            zoom: 5,
            center: {lat: 27.708226, lng: 69.328873},
            mapTypeId: google.maps.MapTypeId.TERRAIN
        });

// Construct the polygon.
        var p = 0
        for (i = 0; i < (triangleCoords1.length-4); i += 4) { //Test to remove largest
            bounding boxes
            if (triangleCoords1[i+2].lat - triangleCoords1[i].lat > 2.5 ||
                triangleCoords1[i+2].lng
                    - triangleCoords1[i].lng > 2.5){
                var p = p +1
                continue;
            }   else {


            var bermudaTriangle1 = new google.maps.Polygon({
                path: [triangleCoords1[i], triangleCoords1[i+1], triangleCoords1[i+2],
                    triangleCoords1[i+3]],
                strokeColor: '#FF0000',
                strokeOpacity: 0.8,
                strokeWeight: 2,
                fillColor: '#000000',
                fillOpacity: 0.15
            });
            bermudaTriangle1.setMap(map);
              }
        }
                    var triangleCoords1 = [
            {lat: 27.708226, lng: 69.328873},
            {lat: 34.019989, lng: 69.328873},
            {lat: 34.019989, lng: 75.382124},
            {lat: 27.708226, lng: 75.382124},
            etc..
}
```

Listing B.2: JavaScript for Google Coordinates

```

with open('dataset1.txt', 'r') as f:
    try:
        for line in f: # read only the first tweet/line
            try: # handle JSON errors
                tweet = json.loads(line) # load it as Python dict
                myFile.write(json.dumps(tweet['place']['country_code']) + ', ')
                myFile.write(json.dumps(tweet['created_at']) + ', ')
                myFile.write(json.dumps(tweet['source']) + ', ')
                myFile.write(json.dumps(tweet['id']) + ', ')
                myFile.write(json.dumps(tweet['lang']) + ', ')
                myFile.write(json.dumps(tweet['user']['time_zone']) + ', ')
```

Listing B.3: Filtering our dataset

```
with open('dataset1.txt', 'r') as f:
```

```
    try:
        for line in f: # read only the first tweet/line
            try:
                tweet = json.loads(line) # load it as Python dict
                myFile.write(json.dumps(tweet['place']['bounding_box']) + '\n')
```

Listing B.4: Tweets converted into integers

```
{"type": "Polygon", "coordinates": [[[69.328873, 27.708226], [69.328873, 34.019989],
    [75.382124, 34.019989], [75.382124, 27.708226]]]}
{"type": "Polygon", "coordinates": [[[100.503859, 13.737873], [100.503859,
    13.763748], [100.517374, 13.763748], [100.517374, 13.737873]]]}
{"type": "Polygon", "coordinates": [[[49.884784, 26.462978], [49.884784, 26.631294],
    [50.039586, 26.631294], [50.039586, 26.462978]]]}
{"type": "Polygon", "coordinates": [[[48.061341, 29.267485], [48.061341, 29.309347],
    [48.088741, 29.309347], [48.088741, 29.267485]]]}
```

Listing B.5: Extracted coordinates

```
{lat: 27.708226, lng: 69.328873},
{lat: 34.019989, lng: 69.328873},
{lat: 34.019989, lng: 75.382124},
{lat: 27.708226, lng: 75.382124},
{lat: 13.737873, lng: 100.503859},
{lat: 13.763748, lng: 100.503859},
{lat: 13.763748, lng: 100.517374},
{lat: 13.737873, lng: 100.517374},
{lat: 26.462978, lng: 49.884784},
{lat: 26.631294, lng: 49.884784},
{lat: 26.631294, lng: 50.039586},
{lat: 26.462978, lng: 50.039586},
{lat: 29.267485, lng: 48.061341},
{lat: 29.309347, lng: 48.061341},
{lat: 29.309347, lng: 48.088741},
{lat: 29.267485, lng: 48.088741},
```

Listing B.6: Formated coordinates

```
with open('exampleFilter3.txt', 'r') as file1:
       try:
           myFile.write(options[dataset[5]])
       except KeyError:
           print dataset[5]
           continue
       myFile.write('\n')

options = {"Pacific/Midway": '1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0',
          "Midway Island": '1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0',
          "American Samoa": '1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0',
          etc..
```

Listing B.7: Second filtering timezones

```
194,0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
219,0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
106,0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
219,0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
```

```
226,0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
159,0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
159,0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
```

Listing B.8: Tweets converted into integers

Appendix C

Figures

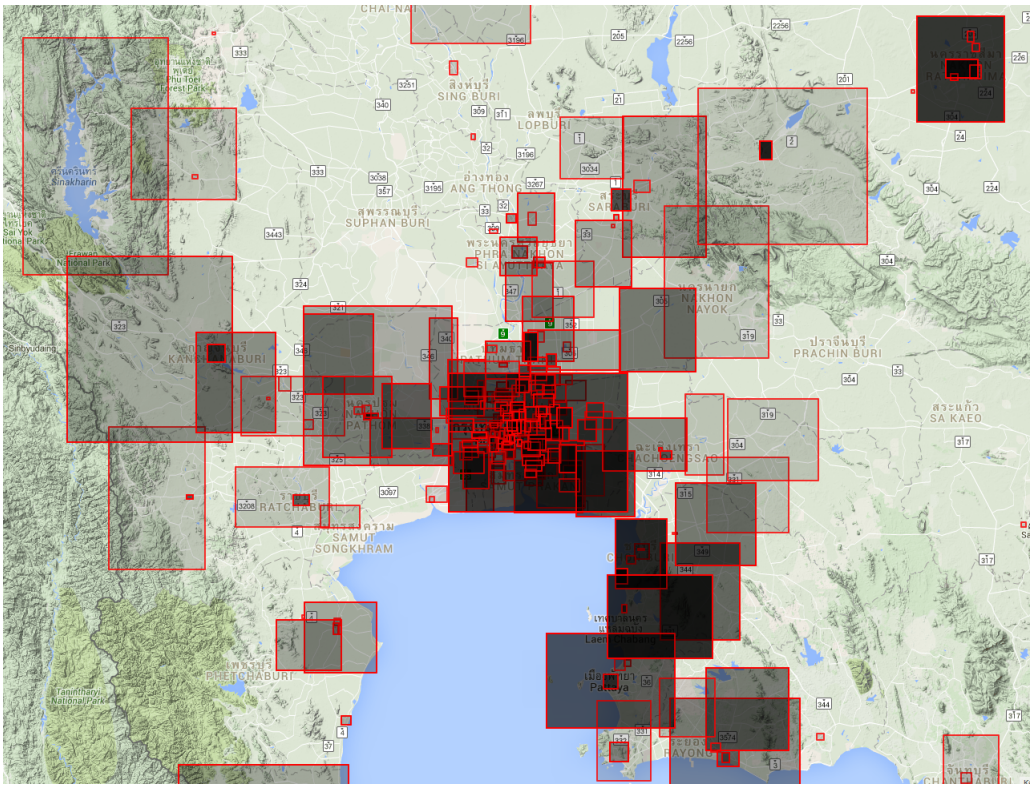

Figure C.1: View of geo-tweets around Bangkok

Appendix D

Dataset

Source	Count	Percentage
Windows phone	10415	0,40%
iPad	27660	1,06%
Other	32918	1,26%
Instagram	149282	5,71%
Trendsmap	1264	0,05%
World Cities	2727	0,10%
Web Client	402035	15,38%
Foursquare	268424	10,27%
iPhone	816986	31,26%
Path	31110	1,19%
Android	839799	32,13%
Tweetbot	23302	0,89%
BlackBerry	7739	0,30%

Table D.1: Table showing the different sources for Dataset 1

Source	Count	Percentage
Windows phone	94257	0,40%
iPad	245514	1,05%
Other	600781	2,58%
Instagram	1195528	5,13%
Trendsmap	9224	0,04%
World Cities	13512	0,06%
Web Client	4126813	17,71%
Foursquare	1153926	4,95%
iPhone	8037834	34,49%
Path	345079	1,48%
Android	7196245	30,88%
Tweetbot	208792	0,90%
BlackBerry	76907	0,33%

Table D.2: Table showing the different sources for Dataset 2

Language	Count	Percentage	Language	Count	Percentage
am	33	0,00%	lt	1134	0,04%
ar	345391	13,22%	ml	621	0,02%
bg	6052	0,23%	mr	1022	0,04%
bn	1329	0,05%	my	30	0,00%
bs	1084	0,04%	ne	10565	0,40%
ckb	41	0,00%	nl	3473	0,13%
da	1955	0,07%	no	1293	0,05%
de	3014	0,12%	or	245	0,01%
el	44	0,00%	pa	126	0,00%
en	583797	22,34%	pl	2834	0,11%
es	9112	0,35%	ps	83	0,00%
et	15751	0,60%	pt	4085	0,16%
fa	4086	0,16%	ro	1688	0,06%
fi	1815	0,07%	ru	275492	10,54%
fr	6632	0,25%	sd	152	0,01%
gu	386	0,01%	si	412	0,02%
hi	37193	1,42%	sk	2775	0,11%
hr	461	0,02%	sl	6020	0,23%
hu	963	0,04%	sr	864	0,03%
hy	172	0,01%	sv	1539	0,06%
in	498687	19,09%	ta	4743	0,18%
is	467	0,02%	te	600	0,02%
it	3234	0,12%	th	400070	15,31%
iw	69	0,00%	tl	64362	2,46%
ja	9647	0,37%	tr	44762	1,71%
ka	122	0,00%	uk	18305	0,70%
ko	2556	0,10%	und	204564	7,83%
km	73	0,00%	ur	9465	0,36%
kn	114	0,00%	vi	6605	0,25%
lo	69	0,00%	zh-cn	9425	0,36%
lv	1247	0,05 %			

Table D.3: Table showing different languages used in tweets in Dataset 1

Language	Count	Percentage	Language	Count	Percentage
am	67	0,00%	lt	12106	0,05%
ar	615880	2,64%	ml	666	0,00%
bg	17789	0,08%	mr	1036	0,00%
bn	1609	0,01%	my	32	0,00%
bs	24283	0,10%	ne	11127	0,05%
ckb	86	0,00%	nl	118218	0,51%
da	21741	0,09%	no	19792	0,08%
de	113228	0,49%	or	251	0,00%
el	21538	0,09%	pa	138	0,00%
en	8323481	35,72%	pl	83236	0,36%
es	3082357	13,23%	ps	93	0,00%
et	53675	0,23%	pt	2480193	10,64%
fa	8167	0,04%	ro	18517	0,08%
fi	29266	0,13%	ru	626846	2,69%
fr	570052	2,45%	sd	159	0,00%
gu	400	0,00%	si	484	0,00%
hi	46272	0,20%	sk	18546	0,08%
hr	12703	0,05%	sl	28561	0,12%
hu	12763	0,05%	sr	2977	0,01%
hy	189	0,00%	sv	53716	0,23%
in	1161208	4,98%	ta	5061	0,02%
is	6810	0,03%	te	601	0,00%
it	225304	0,97%	th	429410	1,84%
iw	16683	0,07%	tl	709352	3,04%
ja	1445817	6,20%	tr	1033540	4,43%
ka	132	0,00%	uk	39657	0,17%
ko	49445	0,21%	und	1668149	7,16%
km	145	0,00%	ur	10688	0,05%
kn	116	0,00%	vi	13524	0,06%
lo	153	0,00%	zh-cn	30585	0,13%
lv	25981	0,11%			

Table D.4: Table showing different languages used in tweets in Dataset 2

Feature	Run 1	Run 2	Run 3	Average Accuracy
All Features	76,57%	76,57%	76,60%	76,59%
Language	59,03%	59,04%	59,03%	59,03%
Time Zone	41,02%	41,01%	41,02%	41,02%
Source	26,82%	26,83%	26,83%	26,03%
App Language	58,83%	58,03%	58,03%	58,03%
Baseline	21,40%	21.40%	21.41%	21.40%

Table D.5: Table showing the results of testing on Dataset 2

Bibliography

Alexa (2016). Alexa top 500 global sites http://www.alexa.com/topsites. [Online; accessed 18-March-2016].

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*, WWW '10, pages 61–70, New York, NY, USA. ACM.

Bamman, D. and Smith, N. A. (2012). Censorship and deletion practices in chinese social media. *First Monday*.

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*, CIKM '10, pages 759–768, New York, NY, USA. ACM.

Han, B., Cook, P., and Baldwin, T. (2014). Text-based twitter user geolocation prediction. *J. Artif. Int. Res.*, 49(1):451–500.

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*, CHI '11, pages 237–246, New York, NY, USA. ACM.

Li, W., Serdyukov, P., de Vries, A. P., Eickhoff, C., and Larson, M. (2011). The where in the tweet. In *Proceedings of the 20th*

ACM International Conference on Information and Knowledge Management, CIKM '11, pages 2473–2476, New York, NY, USA. ACM.

Liu, B., Blasch, E., Chen, Y., Shen, D., and Chen, G. (2013). Scalable sentiment classification for big data analysis using naive bayes classifier. In *Big Data, 2013 IEEE International Conference on*, pages 99–104.

Morstatter, F., Pfeffer, J., Liu, H., and Carley, K. M. (2013). Is the sample good enough? comparing data from twitter's streaming API with twitter's firehose. *CoRR*, abs/1306.5204.

Ng, A. Y. and Jordan, M. I. (2001). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes.

Padmanabhan, V. N. and Subramanian, L. (2001). An investigation of geographic mapping techniques for internet hosts. *SIGCOMM Comput. Commun. Rev.*, 31(4):173–185.

Power, R., Robinson, B., Colton, J., and Cameron, M. A case study for monitoring fires with twitter.

Priedhorsky, R., Culotta, A., and Del Valle, S. Y. (2014). Inferring the origin locations of tweets with quantitative confidence. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '14, pages 1523–1536, New York, NY, USA. ACM.

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*, EMNLP-CoNLL '12, pages 1500–1510, Stroudsburg, PA, USA. Association for Computational Linguistics.

@Twitter (2016). Company | about https://about.twitter.com/company. [Online; accessed 15-March-2016].

Wang, L., Wang, C., Xie, X., Forman, J., Lu, Y., Ma, W.-Y., and Li, Y. (2005). Detecting dominant locations from search queries. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, pages 424–431, New York, NY, USA. ACM.

Wikipedia (2016). Web 2.0 — wikipedia the free encyclopedia https://en.wikipedia.org/w/index.php?title=Web_2.0&oldid=717329425. [Online; accessed 6-June-2016].

Zhang, H. (2004). The optimality of naive bayes. In Barr, V. and Markov, Z., editors, *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference (FLAIRS 2004)*. AAAI Press.