Finding the Happiest U.S. Cities Using Large Scale Sentiment Analysis on Twitter Data

1. Introduction

Social media data have been rapidly growing and widely explored in the past decade. As people share their posts and express their feelings, social media data is becoming a huge archive of human behavior. Many insights have been successfully extracted from social media data, both in the fields of industry [11] and academy [12]. For example, 96% of the earthquakes of Japan Meteorological Agency seismic intensity scale 3 or more can be detected by analysing Twitter data [13]. The U.S. Geological Survey detected 2014 earthquake in Napa 29 seconds after it happened using Twitter data [14].

Twitter is a popular social media providing microblogging service. Users disseminate information on Twitter by posting messages with a maximum of 140 characters. With 330 million monthly active users, a total number of 500 million tweets are posted per day [15].

Out of the work done using Twitter data, sentiment analysis is a field that has attracted much attention. By effectively extracting polarities (usually either positive or negative) from user-generated data, sentiment analysis can be valuable in many cases, such as collecting overall feedbacks for a specific product. For example, sentiment analysis has been proven to be helpful in building up recommender systems [16].

Happiness represents the mental or emotional state of well-being which can be defined by positive and pleasant emotions [10]. It is often of great public interest to identify the happiest cities in the world or in a certain country. To make a reasonable ranking, some happiness indices have been proposed to measure factors that are statistically associated with doing well and feeling well. For example, the National Geographic Gallup Special/Blue Zones Index conducts interviews on nearly 250,000 people in 190 metropolitan areas from U.S, to figure out the 25 happiest U.S. cities [9]. However, conducting such surveys is expensive, since it involves costs for preparing and conducting interviews, aggregating results, and performing statistical analysis. Since measuring happiness index of a city rely on people's feeling about various aspects of their lives, and social medias such as Twitter are commonly used by people to express their feelings, it would be interesting if we can measure happiness by analysing sentiment polarities of Twitter data. By extracting insights on happiness from Tweets, costs of conducting social surveys might be reduced. On the other hand, since Twitter data is real-time, we can learn about in-the-moment state of happiness, which is too difficult to assess by conducting interviews frequently. The cost difference is that of paying for Microsoft Azure's PaaS offering and paying the salaries of a team of social scientists.

In this project, we build up applications in Apache Spark to evaluate the happiness of U.S. cities by performing large scale sentiment analysis on Twitter data. We first train different machine learning models in Spark using a large labeled dataset, and benchmark their performance with VADER [25], a popular and effective sentiment analysis model proposed in 2014 and specifically designed for social media data. We then apply our trained models on another large Twitter dataset with geolocation information, which we collected ourselves using Twitter Streaming API. By grouping and aggregating

sentiment analysis results of millions of tweets by cities, we can measure and compare the happiness of different U.S. cities, and produce a ranking.

The rest of this report goes as follows. We are going to briefly review the previous work done on sentiment analysis using Twitter data in Section 2. Then, we will present the datasets we used in Section 3, as well as our methodology and workflow in Section 4. The experimental results are presented and discussed in Section 5 and 6. We then explain the problems we encountered and the lessons we learnt in Section 7. Finally, we are going to detail our personal contribution in Section 8, and summarize our project in Section 9.

2. Previous Work

Many work has been done on sentiment analysis on Twitter data or other text data [1]. Most of the traditional machine learning models used for Twitter sentiment analysis (e.g. Naive Bayes, Maximum Entropy, Support Vector Machines, etc.) use the bag-of-words model to extract features from Twitter data. There are also novel methods using deep neural networks such as Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) to perform sentiment analysis on text data. [5] uses a CNN model to model sentences, and trained the model respectively on 4 text datasets including a Twitter dataset. [19] uses LSTM model to obtain state-of-the-art results. [4] tries to compare the performance of several LSTM and Bi-LSTM models for sentiment analysis. For this project, we focus on the former type of models, since tuning and training a neural network on a large-scale dataset is too costly. Even the forward propagation to predict sentiment polarity can take much longer than traditional models.

Some works focus on scaling out sentiment analysis of Twitter data. In [17], a real-time system is built to analyse sentiment of Twitter data during 2012 U.S. President Election. The system is able to analyse the sentiment and deliver analysis results of all Twitter traffic related to U.S. President Election in real time, by building up a scalable streaming application running on IBM InfoSphere Streams platform. There are also works focusing on large scale sentiment analysis with Spark. [8] trains a sentiment analysis model on Twitter data using Spark, and shows that the efficiency improves significantly as the number of computational nodes increases.

Two major types of datasets are used to train machine learning models for Twitter sentiment analysis. The first type of dataset has labels annotated by human. Popular labeled Twitter sentiment datasets include SemEval [19][4], Sanders Twitter sentiment corpus [6], and STS-Gold [7]. Due to time cost of adding human annotation, these datasets are all similar in size, with thousands of tweets in each of their training datasets. Such small datasets tend to result in overfitting if more complex function class and more expressive models are desirable. In addition, as the topics and linguistic characteristics of Twitter data change over time, the model trained on an outdated dataset might be less accurate, although the labels could be accurate when they were annotated by human. The second type of dataset belongs to the domain of distant supervision. Considering the large variety of topics on Twitter, it is challenging to collect a large amount of data and label them by human. Therefore, some content are extracted from Twitter messages to serve as noisy labels [2][3][5][8]. [3] uses marks given by reviewers as sentiment labels for movie reviews. [2] extract certain emoticons from Twitter posts, and use it as the noisy sentiment labels (either positive or negative based on the type of emoticon) for the posts. [5] trained their CNN model on the same dataset as [2]. [8] considers both hashtags and emoticons to generate noisy labels. Our project will use the second type of datasets, which is in the scale of hundreds of megabytes.

3. Datasets

Two datasets are used in our experiment. The first dataset is the Sentiment140 dataset [2], which is used to train our machine learning models for sentiment analysis. The second dataset is a scraped dataset containing millions of tweets that we collected on our own using Twitter streaming API. To measure happiness of U.S. cities, we perform sentiment analysis on this scraped dataset thanks to the model trained using the Sentiment140 dataset.

3.1 Training Dataset: Sentiment140 Dataset

The Sentiment140 dataset is created by Stanford to perform sentiment analysis on Twitter data. Instead of labeling sentiments by human, the dataset uses a novel approach to annotate tweets data. It is assumed that any tweets with positive emoticons have positive sentiment polarities, and tweets with negative emoticons have negative sentiment polarities. Though the labels can be more noisy than those annotated by human, this approach for labeling data is much more efficient and can provide millions of tweets for training a sentiment analysis model. In comparison, human-labeled datasets usually contain thousands of tweets. As a result, more expressive models can be trained using this dataset. The dataset contains 160,000 tweets in English language, each labeled to be either positive or negative. The dataset also contains a small test set, which is annotated by human to get an accurate estimate of generalization performance of trained models.

3.2 Dataset for Sentiment Analysis and Happiness Measurement

After training machine learning models using Sentiment140 dataset, we apply the models in a large scale to a scraped Twitter dataset to measure the happiness of cities in United States.

The dataset we are going to use to measure happiness is another set of tweets that we collected ourselves. We use the Twitter API and the tweepy library (a Python wrapper library) to request a stream of tweets from the United States, using the geotag feature and the ability to request tweets within specific latitudes and longitudes. The API returns all data relevant to the tweets, whereas we are only interested in the text and geolocation of the tweets. As the json-formatted data returned by Twitter has no field guarantees (any field can be empty), we divided the fields into imperative fields and non-imperative fields. The imperative fields are those without which it is impossible to do basic sentiment analysis, including text and location. If an imperative field is missing, the tweet is discarded. Otherwise the tweet is retained and any missing fields are substituted with a “NULL” value.

As we focus on cities in the U.S., only tweets with geolocations within the U.S. are collected. In addition, in order to be consistent with Sentiment140 dataset (which is in English), we chose to only collect tweets which are in English language. Such filtering in geolocation and language can be achieved easily using Twitter streaming API.

In order to collect a significant amount of data, we collected data for two hours a day for a week, writing the data to a csv file. The data is collected from 18 April 2018 to 24 April 2018. Though only a small proportion of tweets from Twitter streaming API have valid geolocation information, we finished collecting 3 Gigabytes of Twitter data containing 6.1 million tweets in our scraped dataset.

4. Methodology and Workflow

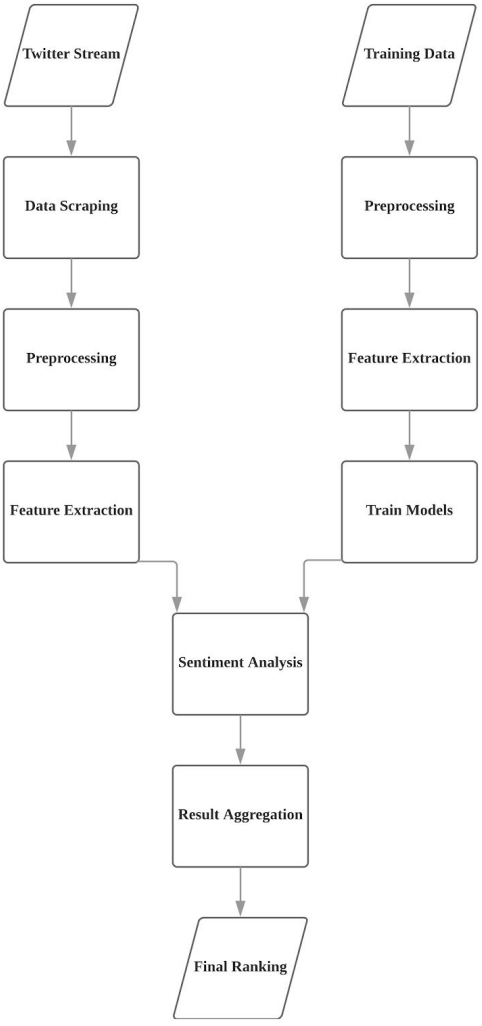

Figure 1 *Workflow diagram*

In this section, we introduce the methodology and experimental workflow of our project. As shown in the workflow in Figure 1, both of the two datasets (training dataset and scraped dataset) will go through a data preprocessing stage and feature extraction stage. The features extracted from the Sentiment140 training dataset will then be used to train various machine learning models, which will then be applied for analysing sentiment on the preprocessed scraped dataset with features extracted. The results of the analysis will finally be aggregated to measure the happiness of U.S. cities.

4.1 Data Preprocessing

Raw tweet strings have many issues that can bring difficulties to later tokenization and feature extraction steps. For example, it is common to see URLs and "@username" strings in tweets, which can hardly be helpful to tell whether a tweet has positive or negative sentiment. Therefore, we process the raw tweets in both our training dataset and scraped dataset using the following steps:

1. Convert all English letters to lowercase.
2. Remove URL address.
3. Remove "@username".
4. Convert contraction words to full form, e.g. "can't" is converted to "can not".
5. Remove numbers, punctuations, and special non-word characters.
6. Reduce the number of overly repeating characters to 2, e.g. "ooooops" is converted to "oops".

We make heavy use of regular expressions in step 2, 3, 5, 6 to preprocess the Twitter data. By preprocessing the tweets using these steps, we get tweets consisting of only lowercase words with no contractions.

4.2 Feature Extraction

The features we will use later to train machine learning models is term frequency-inverse document frequency (tf-idf). To compute tf-idf, we first extract n-grams from each tweet. A n-gram is a set of coorcuring words within a window of size n. The choices of n in our experiments are 1 (unigram), 2 (bigram), and 3 (trigram).

After getting the terms by extracting n-grams from tweet text, we can calculate the tf-idf features of all these terms. The first step of calculating tf-idf requires us to compute the term frequencies of each term. The Spark ML API offers two methods for counting term frequencies, which are HashingTF and CountVectorizer. HashingTF uses a hash trick to hash a term into a certain index and count its frequency. Hashing helps to improve the performance of counting, in the sense of running time and memory usage. However, collisions in hashing may exist and hence the frequency results may not be exact. In comparison, CountVectorizer simply count term frequency in an accurate manner. We then compute the tf-idf features from the term frequencies by using the IDF API offered by Spark ML. The final tf-idf features reweights the term frequency features and reduce the importance of common terms, by taking into account the term frequencies in other documents in the corpus (i.e. other tweets in the dataset).

In our project we consider and experiment with three combinations of feature extraction techniques:

1. Unigram → HashingTF → IDF;
2. Unigram → CountVectorizer → IDF;
3. Unigram + Bigram + Trigram → CountVectorizer → IDF.

Note that in the third combination above, we limit the number of unigrams/bigrams/trigrams to be 5000 respectively (i.e., 5000 most frequently appearing unigrams/bigrams/trigrams). By limiting the total number of features, we can guarantee the performance of feature extraction and model training.

4.3 Training Models

We train and evaluate the performance of 4 commonly used machine learning models in classification tasks, which are logistic regression, linear support vector classifier, multinomial Naive Bayes, and gradient-boosted tree.

Logistic Regression

Let x be the input features of a single sample, which in our case are the tf-idf features of a single tweet. Logistic regression applies a linear transformation and then a logistic function to classify samples:

$$f(x) = \frac{1}{1+e^{-w^T x}}$$

In our case, the computed $f(x)$ can be interpreted as the probability for the sentiment of x to be positive. A logistic regression model can be trained by minimizing the sum of the loss of each training sample:

$$L(w; x, y) = log(1 + e^{-yw^T x})$$

where y is the ground truth label of the training sample x. Despite its simplicity, logistic regression is a basic and popular method for classification.

Linear Support Vector Classifier

A linear support vector classifier tries to find a hyperplane to separate samples in high-dimensional feature space. To find a good hyperplane, we use hinge loss to train our linear SVC model:

$$L(w; x, y) = max\{0, 1 - yw^T x\}$$

Linear support vector classifier is a widely used and a standard method for large-scale classification tasks.

Multinomial Naive Bayes

A Naive Bayes model makes predictions by calculating MAP (maximum a posterior) estimation:

$$\hat{y} = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}} \; p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)$$

Where K is the total number of classes (2 in our case, either positive or negative), and C_k is the k-th class. x_i is the i-th dimension of the input feature. In the Naive Bayes model we use, we assume $p(x_i \mid C_k)$ follows multinomial distribution.

Despite its strong assumption that all input features are independent, Naive Bayes has been found to have surprisingly good performance on text classification tasks.

Gradient-Boosted Tree

Gradient-boosted tree (GBT) is a popular and powerful classification method using ensembles of decision trees. To train a GBT model, we iteratively train a sequence of decision trees using a loss function. In our model, the maximum number of iterations is set to be 10, and our choice of loss function is logistic loss:

$$L(\theta) = \sum_i [y_i \ln(1 + e^{-\hat{y}_i}) + (1 - y_i) \ln(1 + e^{\hat{y}_i})]$$

After training the 4 models mentioned above using Sentiment140 dataset, we would like to benchmark their performance. The benchmark model we use is the VADER model [25], which is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. VADER offered state-of-the-art performance when it was proposed in GIT in 2014. VADER is currently still the default sentiment analysis method used in the popular natural language processing package NLTK. Because of its strong performance in sentiment analysis on social media text, we benchmark our models with VADER. The VADER implementation we use is from NLTK.

4.4 Sentiment Analysis and Result Aggregation

We apply our models on our scraped dataset, to get a sentiment polarity result (either positive or negative) of each tweet. We then group the tweets by their geolocations into different groups of U.S. cities. The percentage of positive tweets of each city is calculated, and is used as a score to measure happiness. We then sort the cities in descending order of the scores. In this way we get a score and a rank for each city. Notice that in order to reduce the effect of skewed samples in very small cities, we ignore cities with less than 500 tweets appeared in our scraped dataset.

5. Experimental Results

5.1 Experimental Configuration

We conduct all our workflow except data collection in Apache Spark. We implement all Spark applications in Python, and conduct all our experiments in Jupyter Notebook. For the configuration of Spark, we customize the following parameters:

- `spark.executor.memory = 6g`
- `spark.driver.memory = 6g`
- `spark.master = local[6]`

All other configurations are set as default in Spark local standalone mode. In our experiments, we found that all our models are effective to train, and analyzing sentiment on 6.1 million tweets only takes a few minutes. Therefore, running Spark with 6 threads on a single node with suitably large driver memory and executor memory is sufficient for our usage.

5.2 Effect of Features on Model Performance

The Sentiment140 dataset is divided into training and test set by 98:2. We measure the effect of different features on the performance of logistic regression, by training logistic regression models using different features. Namely, we extract features from the same set of training data using three different combinations of techniques, and train three logistic regression (LR) models respectively. Then we compare the performance of these LR models on the same set of test data. The experimental results of different features are as the following:

	Unigram+HashingTF+IDF	Unigram+CV+IDF	N-Gram+CV+IDF
Accuracy	0.7904	0.7971	0.8115
Area under ROC	0.8615	0.8662	0.8879
Running time	33.3s	39.9s	191s

All three combinations of feature extraction techniques were introduced in Section 4.2. Be noted that the running time includes time for extracting features, training the model, and predicting on validation set. As shown in the table, by extracting features using n-gram, CountVectorizer, and IDF, we achieve the best performance on the trained linear regression model, although it takes longer time to run comparing two the other two feature extraction measures. Therefore, we choose this feature extraction measure for all our subsequent experiments in subsection 5.4 and 5.5.

5.3 Effect of Spark Parallelism on Model Performance

Though the parallelism does not affect the predictive accuracy and happiness measurement, it is also of our interest to measure the effects of parallelism (number of threads) on the running time of our Spark applications. Similar to subsection 5.2, we measure the running time of different feature extraction measures, by varying the available number of threads. The results are as the following:

	Unigram+HashingTF+IDF+LR	Unigram+CV+IDF+LR	N-Gram+CV+IDF+LR
1 thread	89s	104s	442s
2 threads	57s	60s	260s
6 threads	33s	39s	191s

It is clear from the table that as the number of threads increase, the performance of the applications also improve, though not linearly. As training machine learning models involves close and frequent coordination between driver and executors and certain overheads cannot be saved by increasing parallelism, such non-linear relationship is expected. We use 6 threads for all subsequent experiment in subsections 5.4 and 5.5.

5.4 Comparing Model Performance on Testing Set

The Sentiment140 dataset is divided into training and testing set by 98:2. We train all our 4 models introduced in Section 4 on the same training set, and test their performance on the same testing set. The performance of 4 models on testing set is as follows:

	LR	LSVC	NB	GBT
Accuracy	0.8115	0.8080	0.7846	0.6668
Area under AUC	0.8879	0.8840	0.5529	0.7429
Running time	191s	743s	161s	1h 59min

Due to its simplicity, LR gives the best performance among the four models. LSVC also gives similarly good performance, though running significantly slower than LR. Naive Bayes runs the fastest among the 4 models, but performs slightly worse than LR and LSVC. Surprisingly, despite its power in many machine learning tasks, GBT does not perform well in our experiment, and takes significantly longer time to run.

5.5 Comparing Model Performance on Human-Labeled Dataset

In 5.4, we only measured the performance of our models on a testing set sampled from Sentiment140 dataset, which is labeled under distant supervision. To further evaluate the performance of our models, we benchmark the performance of our models with the VADER model on a human-labeled dataset containing 177 positive tweets and 188 negative tweets. The comparison of their prediction accuracies is shown in the following table:

	LR	LSVC	NB	GBT	VADER
Accuracy	0.8106	0.8162	0.7967	0.6657	0.7855

All of our LR, NB, and GBT models outperform VADER. Considering the small size of this human-labeled dataset, such difference is not significant. However, it is still clear from the table that the performance of our LR, LSVC, and NB models trained in Spark using Sentiment140 dataset is comparable to the performance of VADER.

5.6 Ranking U.S. Cities by Aggregating Tweets Sentiments

As described in subsection 4.4, we measure the happiness of U.S. cities by calculating the percentage of positive tweets in each city, and ignoring cities with less than 500 tweets. We choose to use our logistic regression model trained on Sentiment140 dataset to measure the happiness of U.S. cities, due to its good prediction accuracy and fast speed. The ranking result of our logistic regression model is shown in Figure 2. As a comparison, the ranking result of the 25 happiest U.S. cities according to National Geographic in 2015 is shown in Figure 3.

```
West Hollywood, CA
      Bellevue, WA
       Boulder, CO
  Santa Monica, CA
   Miami Beach, FL
      Stamford, CT
 Beverly Hills, CA
  Coral Gables, FL
       Fremont, CA
West Palm Beach, FL
     Manhattan, NY
Salt Lake City, UT
    Greenville, SC
      Franklin, TN
     Arlington, VA
       Seattle, WA
  Springfield, IL
     Carlsbad, CA
  Sioux Falls, SD
      Burbank, CA
     McKinney, TX
        Plano, TX
     Pasadena, CA
    Cambridge, MA
San Francisco, CA
```

Figure 2 *Ranking result of LR model survey*

Boulder, CO
Santa Cruz-Watsonville, CA
Charlottesville, VA
Fort Collins, CO
San Luis Obispo-Paso Robles-Arroyo Grande, CA
San Jose-Sunnyvale-Santa Clara, CA
Provo-Orem, UT
Bridgeport-Stamford, CT
Barnstable Town, MA
Anchorage, AK
Naples-Immokalee-Marco Island, FL
Santa Maria-Santa Barbara, CA
Salinas, CA
North Port-Sarasota-Bradenton, FL
Urban Honolulu, HI
Ann Arbor, MI
San Francisco-Oakland-Hayward, CA
Colorado Springs, CO
Manchester-Nashua, NH
Oxnard-Thousand Oaks-Ventura, CA
Washington-Arlington-Alexandria, DC-VA-MD-WV
Minneapolis-St. Paul-Bloomington, MN-WI
San Diego-Carlsbad, CA
Portland-South Portland, ME
Austin-Round Rock, TX

Figure 3 *Ranking result of National Geographic's survey*

6. Discussion of Results

To interpret our ranking of happiest U.S. cities, one direct and apparent way is to compare it with traditional rankings such as the 25 happiest cities ranked by National Geographic in 2015. It is fairly clear from subsection 5.6 that our sentiment analysis does not yield results similar to those of National Geographic. Only three cities, Boulder, San Francisco, and Carlsbad are similar between the two rankings. There are several factors that we have discovered, which impact the results of our analysis. We detail them in the following subsections.

6.1 Difference of Demographics

The first factor is the difference of demographics between Twitter users and a city's population. It is apparent that the people using Twitter are not an exact representation of the population of a city. According to recent statistics collected about Twitter demographics [26], 37% of Twitter users are between 18 and 29 years of age. However, this age group is only representative of 8% of the U.S. population [27]. This presents a bias on our results, because they are more influenced by the sentiment of the younger generation and underrepresents the older population of a city. More generally, we can say that some categories of the population are under-represented among the Twitter users, while other categories are over-represented. This fact explains in part why our dataset scraped from Twitter cannot perfectly replace interviews of a carefully chosen sample of the population.

6.2 Biases Induced by Tourists and Students

Along these same lines, it is also clear that many of the top results of our analysis can fall into several broad categories, including college towns and tourist destinations. College towns, such as Stamford and Cambridge, appear high on our list compared to the traditional studies. We believe that this is because the student population is both more active on Twitter and more positive than the standard population. Another category of city that appears high in our rankings is tourist cities. Indeed, among the top 10 cities of our ranking, we have identified 6 tourist cities, for instance West Hollywood, Miami Beach, or Beverly Hills. As we include all tweets from a city in its ranking, there is no reliable way to distinguish tourists from citizens. Therefore, large tourist cities will tend to have a large number of positive tweets from tourists that drowns out the local population's sentiment. This is one area where the predefined method of conducting in person surveys is extraordinarily beneficial, as it eliminates the tourist crowd and leaves only residents.

To further understand why tourist cities and college towns are ranked high in our approach, we extracted the most frequently used words of three representative cities, which are West Hollywood, CA (a typical tourist city), Boulder, CO (a traditionally highly-ranked city), and Cambridge, MA (a typical college town). The common stop words such as "the" are excluded, and most frequently tweeted 20 words of each city are as the following:

- **West Hollywood, CA**: face, heart, hollywood, west, like, love, red, get, time, one, joy, tears, good, see, know, thank, tonight, california, new, night
- **Boulder, CO**: boulder, face, job, hiring, heart, like, one, see, get, love, day, work, tears, good, joy, people, today, know, great, eyes
- **Cambridge, MA**: face, cambridge, like, one, job, get, hiring, see, today, good, think, know, people, great, heart, time, day, massachusetts, love, boston

There are many common words such as "like", "love", "good", "face", "get", "know", and so on, which appear frequently in the tweets of all three cities. Words like "like", "love", and "good" clearly have positive sentiment, and in some sense this can also justify why these cities are ranked high. Notice that in Boulder and Cambridge, words like "hiring", "job" appear frequently, where in West Hollywood it seems that tourists do not worry much about work during travelling. Overall the frequent terms of Boulder and Cambridge are similar. Words like "new" and "thank" are unique in the frequent words of West Hollywood, which may indicate that tourists may show different behavior from local residents and hence affect the ranking of tourist cities.

6.3 Time Frame

Another factor affecting our results is the time frame. Although we have now scraped for one week as opposed to two days prior to the presentation, the results are still skewed, and this can be because certain cities are more active/happy at certain times of the year. For instance, beach cities such as

Miami Beach will see more activity and positive tweets during the spring and summer months (from April to August) compared to colder cities. We are also comparing these results to a 2015 survey, which may not have the most up-to-date information.

6.4 Sample Size

Last, the sample size of tweets from different cities varies quite a bit. While this is proportional to the population of the cities, it provides a statistical barrier to our analysis, as the larger cities have a much higher precision of their sentiment, whereas with the smaller cities, several tweets can impact the overall sentiment significantly.

7. Problems Encountered and Lessons Learnt

7.1 Azure Usage

We planned to use Azure for our project, in particular to train our sentiment analysis model using Spark. Therefore, we decided to use HDInsight, a service that deploys big data processing tools in the Azure cloud, and to deploy a Spark cluster on it. Something interesting to mention is the fact that we are charged on a hourly basis for a HDInsight cluster, 24 hours a day, until the cluster is deleted. In fact, Azure charges us for the cluster even when we are not using it. As the default cluster configuration implied paying 5$ per hour, due to the credit limitation, we decided to decrease a bit the size of the nodes. Doing that, the cost was reduced to 2$ per hour. Apart from that, we let all configurations as set by default.

With HDInsight was created a storage account, where we uploaded our dataset and the useful files to train our model. Jupyter notebook is available on Azure, enabling us to run our code. So, our Spark cluster and storage solution were set up, but the first time we ran our code on Azure, we realized that it was slower than on our local machines. It was unclear why, but we suspected the nodes' size or the number of workers to be not large enough (we had 4 of them). Hence, we tried to rescale the cluster by increasing the number of worker nodes, but after a long loading time, we got an error message. We tried again several times, but we got the same result every time: after loading during approximately 20 minutes, Azure returned an error message.

As the code was running rapidly enough on our local machines, we decided to abandon Azure and use our local machines. This was a bit disappointing, because it took quite a long time to set everything up in order to run our code on Azure. Moreover, everything was working, it was simply not fast enough. Nevertheless, despite the fact that we eventually did not use Azure, it was interesting to discover this platform and learn how to set up a Spark cluster.

7.2 Data Preprocessing

There were also several important lessons learned about preprocessing. Chief among these is that cleaning data is both imperative and difficult. At first our Twitter scraper was very rudimentary, simply forcing the Twitter json returned by tweepy into an expected format, and if it did not fit that format, discarding the tweet. However, this led to many tweets being discarded, as twitter has very few guarantees on the data returned from the Twitter API. Therefore, many tweets had one or two minor fields missing and were therefore discarded as trash, when in reality the majority of the tweet's data was still usable.

This requires a much finer detail process, requiring us to rewrite the entire scraping program to account for these errors. If certain "imperative" fields, such as text, location, or time are missing, the tweet is discarded. However, any other missing fields were filled as null values, so that they can still

be stored and used later, but will not cause errors in the future with invalid indexing. While this may seem like a minor issue, this was a major issue for us as few tweets are geotagged, so we needed every tweet we could get.

7.3 Lessons and Experience on Spark

Instead of using the fundamental RDD API provided by Spark, we decide to use the higher-level DataFrame API. Different stages of data preprocessing and model training can be easily done by creating new DataFrame columns from existing columns. In our process of building up Spark applications, it is observed that caching a dataframe can improve the performance significantly. To overcome lazy execution in certain occasions, we also find it a good trick to use `cache()` and `count()` as a combo to trigger the cache of a dataframe.

Our applications make heavy use of feature extraction and machine learning techniques. Spark provides a great number of high-level machine learning APIs on top of DataFrame API. Moreover, `Transformers` and `Estimators` from Spark APIs can be combined into a Spark ML Pipeline, which greatly simplifies building/tuning/saving/loading machine learning models.

8. Personal Contribution

Robert Stigler implemented the preprocessing phase of the sentiment analysis and ranker. This included twitter scraping as well as integrating this data into a dataframe that is then passed to the sentiment analysis function, as well as aggregating and sorting the results of the sentiment analysis into a city ranking.

Guillaume Freisz investigated the Azure platform and set it up in order to be able to train our sentiment analysis models on it. To do this, he built a Spark cluster and created a storage account. He used Jupyter Notebook available on Azure to run our code. Although the code was running, the execution was very slow. Rescaling the cluster failed several times, and we finally used our local machines to train our models.

Yang Ruizhi implemented Spark applications for all feature extraction pipelines and machine learning models presented in Section 4. He also conducted experiments presented in Section 5 and collected the experimental results.

Robert Stigler, Guillaume Freisz, and Yang Ruizhi contributed equally to writing this report, with each person writing different sections. The same applies for the preparation of the final presentation.

9. Project Summary

In this project, we build up Spark applications to evaluate the happiness of U.S. cities by performing sentiment analysis on Twitter data. We first collect a dataset containing 6.1 millions English tweets tweeted in the U.S. and their geolocation information. We then preprocess this scraped dataset and the Sentiment140 dataset, and extract tf-idf features of all tweets. 4 different models are trained in Spark, and their performance are compared with VADER. We observe that our LR, LSVC, and NB models achieve good performance comparable to VADER on a human-labeled dataset. We measure the happiness of U.S. cities by performing sentiment analysis on tweets of different U.S. cities and aggregating the results. All our preprocessing, model training, and happiness measurement are done in Spark, and can easily scale by running the applications on a Spark cluster. We have seen that our results are not a one-to-one matching with the survey's ranking. Several reasons can explain this

difference, including biases induced by tourists and college students, limited time frame and sample size of the data we collected, etc.

The contribution of our project is to combine distant supervision with large scale training using Spark. The fact that distant supervision allows us to create large size of up-to-date training data justifies the value and necessity of training machine learning models using Spark. In addition, it is also novel to extracting insights on happiness using Twitter data. By scraping data using Twitter API in real time, we can get in-the-moment measurement of sentiments of Twitter users, and hence get an up-to-date understanding of the happiness of various U.S. cities.

A possible extension for this project could be analyzing tweets in other english speaking cities, like in Singapore, Great-Britain, Canada, or Australia for example. The only problem is that it will be more difficult to test our results, because there is not really a survey comparing happiness between cities of these countries. In addition, we could analyse tweets classified by country, because several studies already exist at the country level. However, that implies managing to have a good sentiment analysis in lots of different languages.

It would also be interesting in future to extend our project so that we can consider different factors of happiness, and understand happiness better from Twitter data. For example, the actions of a local government can affect the happiness of a city. By combining sentiment analysis results corresponding to different topics such as politics, health, or nature, we could get a more fine-grain measurement of happiness using Twitter data, allowing us to dissect *why* and *in what way* one city is “happier” than another.

10. References

[1] Pawar, K. K., Shrishrimal, P. P., & Deshmukh, R. R. (2015). Twitter sentiment analysis: A review. International Journal of Scientific & Engineering Research, 6(4), 9.

[2] Go, A., Bhayani, R., & Huang, L. (2009). Twitter sentiment classification using distant supervision. CS224N Project Report, Stanford, 1(12).

[3] Pang, B., Lee, L., & Vaithyanathan, S. (2002, July). Thumbs up?: sentiment classification using machine learning techniques. In Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10(pp. 79-86). Association for Computational Linguistics.

[4] Barnes, J., Klinger, R., & Walde, S. S. I. (2017). Assessing State-of-the-Art Sentiment Models on State-of-the-Art Sentiment Datasets. arXiv preprint arXiv:1709.04219.

[5] Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A convolutional neural network for modelling sentences. arXiv preprint arXiv:1404.2188.

[6] Sanders, N. J. (2011). Twitter Sentiment Corpus. Retrieved March 07, 2018, from http://www.sananalytics.com/lab/twitter-sentiment/

[7] Saif, H., Fernandez, M., He, Y., & Alani, H. (2013). Evaluation datasets for Twitter sentiment analysis: a survey and a new dataset, the STS-Gold.

[8] Nodarakis, N., Sioutas, S., Tsakalidis, A. K., & Tzimas, G. (2016, March). Large Scale Sentiment Analysis on Twitter with Spark. In EDBT/ICDT Workshops (pp. 1-8).

[9] Witters, D., & Buettner, D. (2017, October 21). These Are the Happiest Cities in the United States Retrieved March 07, 2018, from https://www.nationalgeographic.com/travel/destinations/north-america/ united-states/happiest-cities-united-states-2017/

[10] Wikipedia contributors. (2018, February 15). Happiness. In Wikipedia, The Free Encyclopedia. Retrieved 18:21, March 7, 2018, from https://en.wikipedia.org/w/index.php?title=Happiness&oldid=825812 227

[11] Twitter Blog. Growth in the Twitter data ecosystem. (2015, July 21). Retrieved March 8, 2018, from https://blog.twitter.com/official/en_us/a/2015/twitter-data-ecosystem.html

[12] Twitter Blog. (2015, July 30). Twitter data for research: from understanding relationships to spotting the Aurora Borealis. Retrieved March 07, 2018, from https://blog.twitter.com/official/en_us/a/2015 /twitter-data-research.html

[13] Sakaki, T., Okazaki, M., & Matsuo, Y. (2010, April). Earthquake shakes Twitter users: real-time event detection by social sensors. In Proceedings of the 19th international conference on World wide web(pp. 851-860). ACM.

[14] Twitter Blog. How the USGS uses Twitter data to track earthquakes. (2015, October 7). Retrieved March 8, 2018, from

https://blog.twitter.com/official/en_us/a/2015/usgs-twitter-data-earthquake-detection.html

[15] Aslam, S. (2018, January 1). Twitter by the Numbers: Stats, Demographics & Fun Facts. Retrieved March 8, 2018, from https://www.omnicoreagency.com/twitter-statistics/

[16] Tang, H., Tan, S., & Cheng, X. (2009). A survey on sentiment detection of reviews. Expert Systems with Applications, 36(7), 10760-10773.

[17] Wang, H., Can, D., Kazemzadeh, A., Bar, F., & Narayanan, S. (2012, July). A system for real-time twitter sentiment analysis of 2012 us presidential election cycle. In Proceedings of the ACL 2012 System Demonstrations (pp. 115-120). Association for Computational Linguistics.

[18] Hltcoe, J. (2013). Semeval-2013 task 2: Sentiment analysis in Twitter. Atlanta, Georgia, USA, 312.

[19] Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In Proceedings of the 2013 conference on empirical methods in natural language processing (pp. 1631-1642).

[20] Classification and regression. (n.d.). Retrieved March 09, 2018, from https://spark.apache.org/docs/latest/ml-classification-regression.html#multinomial-logistic-regression

[21] Kendall's Tau and Spearman's Rank Correlation Coefficient. (n.d.). Retrieved March 08, 2018, from http://www.statisticssolutions.com/kendalls-tau-and-spearmans-rank-correlation-coefficient/

[22] Nagel, T. (n.d.). Unfolding Maps: Unfolding is a library to create interactive maps and geovisualizations in Processing and Java. Retrieved March 09, 2018, from http://unfoldingmaps.org/

[23] Natural Language Toolkit. (n.d.). Retrieved March 09, 2018, from http://www.nltk.org/

[24] Sentiment140 For Academics. (Stanford University). Retrieved March 10, 2018, from http://help.sentiment140.com/for-students

[25] Gilbert, C. H. E. (2014). Vader: A parsimonious rule-based model for sentiment analysis of social media text. In Eighth International Conference on Weblogs and Social Media (ICWSM-14). Available at (20/04/16) http://comp. social. gatech. edu/papers/icwsm14. vader. hutto. Pdf.

[26] S. (2018, January 26). Twitter by the Numbers: Stats, Demographics & Fun Facts. Retrieved April 28, 2018, from https://www.omnicoreagency.com/twitter-statistics/

[27] Population Pyramids of the World from 1950 to 2100. (n.d.). Retrieved April 28, 2018, from https://www.populationpyramid.net/united-states-of-america/2017/