Location Classification on Tweets

Erdi Çallı e.calli@student.ru.nl

Abstract

Previous research on Twitter user location classification is mostly focusing on multi-class prediction. Approaching at the same problem from another perspective, in some cases, it may be enough to answer if someone belongs to a location or not. In this perspective, we have compared some models for predicting tweet locations using Binary and Multi-Class Classification. We have seen that if the problem at hand is reducible to Binary Classification, it is possible to make much better predictions.

1. Introduction

Proliferation of microblogging platforms, such as Twitter, enabled companies to learn valuable information about their products, customers and businesses. Twitter, introduced the option to include location information on tweets(geotagging) in 2009[1]. Since then, many applications(i.e Foursquare) and users started embedding this information to enrich the tweet content. Using geotagged tweets to train classification algorithms, it is proven to be possible to infer tweet locations[CCL10].

Using sentimental analysis[PP10], a company can learn if its customers are happy with their products or a political figure can get feedback from his/her voters. The possibilities are endless. Incorporating this information with geolocation may be valuable in lots of situations for politicians, businesses and other parties.

Unfortunately geolocation is not shared in most of the tweets and most of the tweets tagged with location are sent from apps or bots. Also geotags are containing different granularities such as city, neighbourhood or state. Because of those problems most of the data supplied by twitter is filtered out.

In previous studies, this problem has been solved up to a precision as a multi-class classification problem with different algorithms and assumptions. We are introducing binary classification to find the tweets belonging to a specific location.

In this study, different Supervised Learning methods implemented in Scikit Learn[PVG+11] are tested with different configurations. We have also introduced binary classification on location prediction.[2]

2. Related Work

Using text mining methods and location tags in tweets, [CCL10] redefined this problem. [CCL10] used only the text data and introduced a method that identifies the words with

1. https://blog.twitter.com/2009/location-location-location
2. Source code available at http://www.github.com/xarion/waddle

strong local geo-scope. This study achieved up to 51% accuracy to locate users within a 100 mile radius.

[HHSC11] introduced the *CALGARI* classifying algorithm specific to this problem, focusing on country and state prediction. They got a 72% accuracy for classifying 4 countries and 30% accuracy on classifying 18 states.

[EOSX10] created a geographic topic model to make use of the textual differences in topics based on regions, they got a median distance error of 494 kms.

[WB11] tried to solve this problem using geotagged Wikipedia articles and tweet texts, focusing on predicting the geolocation. Their work has obtained a median error of 479 km's for tweets.

[MND12] introduced hierarchical classification methods combined with statistical methods. They have achieved 58%, 66% and 78% accuracy on city, state and timezone levels, respectively. Also their findings show that users may be able to partially hide their locations, making their tweets untraceable. Same researchers revisited the problem in [MND14], trying to classify the home location of users, achieving similar results and also introducing movement and location prediction.

[SKB12] approached this problem introducing the friend information, constructing a friend location graph. Their study was able to achieve 77% to 84.3% accuracy, relative to available number of friends.

[GHG14] studied the problems with the baselines accepted by recent studies and explained why location and language classification is not an easy problem and how some of the assumptions are incorrect, such as accepting the device location as user's location.

3. Method

3.1 Data Collection

Using Twitter's sample status streaming api[3], we have collected data. This api returns approximately 1% of the public tweets from all around the world. We have filtered this data to reduce the problem space to only location classification. Our filter selects only the geotagged English tweets, sent within the United States. Running this collector for 1 month, we have collected a dataset consisting of 33112 individual tweets, each from a unique user. We have also collected a sample of 40 tweets from those users, if their timelines are public, to enrich the dataset.

Upon investigating this dataset, we have realised that there are a lot of low quality, automated tweets, coming from 886 different tweet sources(devices or applications that a tweet has been sent from). To remove these sources, we have created a whitelist of most used 20 sources that selects applications which real users use to tweet from. This reduced the amount of data by around 50%, to 17040 individual geotagged tweets.

One problem with the locations is the granularity. A location can appear as a state(New York, US), a city(New York, NY) or a neighbourhood(Manhattan, NY). A selection of 8 top locations, not intersecting each other, is selected from the ones containing the most tweets.

3. https://dev.twitter.com/streaming/reference/get/statuses/sample

3.2 Empirical Bayes

To better understand the data, we have created a posterior with empirical prior[CL97] and used MAP estimation[DeG] to find the best classification result. Tokenised each tweet using the *twokenize* module[GSO+11], to convert them to a *bag of words*, calculate the probability of each "token given location" and the probability of each location to find their probabilities.

$$\mathbf{T} = Tokens$$
$$L = Location$$
$$p(L|\mathbf{T}) = \frac{p(\mathbf{T}|L)p(L)}{p(T)}$$

With likelihood $p(\mathbf{T}|L)$ and prior $p(\mathbf{L})$.

For the prior, instead of assuming a distribution, we have generated an empirical prior and applied MAP estimate to the posterior;

$$\hat{L} = \arg\max_{L}\{p(\mathbf{T}|L)P(L)\}$$

where;

$$p(\mathbf{T}|L) = \prod_{i} p(t_i|L)$$

Empirical prior $p(L)$ is referring to the probability of a location L in the training data.

This experiment have been tested for 2 different configurations. In the first configuration, the collection of sample tweets from a user and the geotagged tweet belonging to same user are accepted as one document, assuming the location of the geotagged tweet. In the second configuration, each sample tweet of an individual are accepted as separate documents with same assumption.

3.2.1 Implementation

First run of the experiment with the whole dataset resulted with an 0.02 and 0.04 accuracy rates for the first and second configurations, respectively. After this experiment, we have seen that second configuration is better than the first one. The rest of this experiment is ran with the second configuration.

One problem with this experiment was, if a token(t_x) didn't appear in the training dataset, tweets containing this token got a probability of 0. To solve this problem, we have ignored those tokens. Second run of this classifier resulted with an accuracy rate of 0.05.

3.2.2 Source Filtering

Upon investigating the dataset, we have realised that there are a lot of low quality, automated tweets, coming from 886 different tweet sources(devices or applications that a tweet has been sent from). To remove these sources, we have created a whitelist of most used 20 sources that selects applications which real users use to tweet from. This reduced the amount of data by around half, to 17040 individual geotagged tweets. Third run of the experiment for those 20 sources improved the accuracy to 0.09.

3.2.3 Location Filtering

Investigating the locations the tweets has been sent, we have seen that there are 3306 different locations(location full name in this case) that tweets have been geotagged with. Most of them having only 1 tweet. We have created another whitelist for locations, only including the top 10. This has reduced the dataset to 4751 individual geotagged tweets. Taking a closer look to those 10 locations, we have seen that there are 1 neighbourhood, 2 states and 7 cities. To have finer class definitions, we have removed those 2 states from location whitelist. Running the experiment for the last time, we have acquired an accuracy of 0.11. Which doesn't seem better than making random classifications for 8 locations. After this experiment we have decided to try other algorithms.

3.3 Scikit Learn - Multinomial Naive Bayes

This experiment uses the Multinomial Naive Bayes model to predict classes. This model captures information on documents as an ordered sequence of tokens[MN+98], using a Multinomial Distribution. We have used the method implemented in Scikit Learn[PVG+11](sklearn, v0.17).

Also, in this experiment, we have redefined the data configurations. For the first data configuration, all tweets of a user are tagged with the original tweet's geotag location. For the second configuration, we have ignored the user history and only used the geotagged tweets.

These implementations also had a different structure. Instead of token counts used in the first experiment, we have used tf-idf scores. Also sparse matrix implementation of Numpy is used to represent those frequencies by default. We have stopped using the *twokenize* module and switched to the internal tokenizer of sklearn.

3.3.1 Multinomial Naive Bayes

For this experiment, MultinomialNB(Multinomial Naive Bayes) implementation is selected. This implementation has achieved a higher macro accuracy results of 0.19, 0.24 for first and second configurations, respectively. But when we investigated the micro accuracy we have seen that the classifier almost always predicted the location with highest amount of tweets.

3.3.2 Normalisation of Documents per Class

To prevent the problem in the last experiment, we have normalised the dataset by accepting equal amount of documents per class. Upon normalising the number of documents per class, we have achieved accuracy rates of 0.29, 0.42 for first and second data configurations.

3.3.3 Ngrams

To improve the results found in the last experiment, we have tried Ngrams in range (1,2). It did not result with an improvement in accuracy. The accuracy has decreased by 0.003 for both configurations.

3.3.4 Binary Classification

To be able to answer the question, which tweets are from given a location, we have changed our multi-class classification problem to a binary classification problem. Running tests to classify tweets from *Manhattan*, *NY* by including previous improvements and ngrams in range (1,2) we have achieved accuracy rates of 0.71, 0.62 for first and second data configurations, respectively.

3.4 Scikit Learn - Algorithm Comparison

This experiment compares the classification methods implemented in Scikit Learn(sklearn)(v0.17) library with different configurations. Sklearn implements various classification methods such as, Logistic Regression, Perceptron, Support Vector Classifiers and etc. We have compared the sklearn methods under supervised learning[4], applicable to classification and able to work with sparse matrices. We have also tried various parameters for those algorithms. Resulting set of algorithms and parameters was containing 22 combinations.

1. LogisticRegressionCV: a linear model for classification. Uses a logistic function to calculate the class probabilities are of a single trial. This version uses cross validation to find optimal values for parameters.
2. SGDClassifier: Stochastic Gradient Descent learning model supporting multiple loss functions and penalties.
   (a) SGDClassifier(loss="hinge"): Hinge is a commonly used convex loss function for classification.
   (b) SGDClassifier(loss="log"): Converts SGD to a probabilistic logistic regression classifier.
   (c) SGDClassifier(loss="modified_huber"): Hinge with a setting, $\gamma = 2$
   (d) SGDClassifier(loss="squared_hinge"): Quadratically penalised hinge.
   (e) SGDClassifier(loss="perceptron"): Linear loss function used with the Perceptron algorithm.
   (f) SGDClassifier(loss="squared_loss"): Least squares loss function.
   (g) SGDClassifier(loss="huber"): Similar to squared_loss but switches from squared to linear after a threshold.
   (h) SGDClassifier(loss="epsilon_insensitive"): Linear loss function, ignoring errors less than the threshold, epsilon.
   (i) SGDClassifier(loss="squared_epsilon_insensitive"): Squared version of epsilon_insensitive.
3. Perceptron: Same as SGDClassifier(loss="perceptron") with a constant learning rate and different default parameters.
4. PassiveAggressiveClassifier: Similar to Perceptron. Instead of using a learning rate, uses a regularisation term.
   (a) PassiveAggressiveClassifier(loss="hinge")
   (b) PassiveAggressiveClassifier(loss="squared_hinge")
5. SVC: Classifier model using Support Vector Machines. Creates a high or infinite dimensional one or many hyperplanes and clusters classes on that space.
   (a) SVC(kernel='poly'): Polynomial kernel for SVM.
   (b) SVC(kernel='sigmoid'): Sigmoid kernel for SVM.
6. LinearSVC: SVM with linear kernel.
7. KNeighborsClassifier: K-Nearest Neighbours classifier.
   (a) KNeighborsClassifier(n_neighbors=8, weights="uniform"): Uses uniform weights to classify neighbours.

4. http://scikit-learn.org/stable/supervised_learning.html

   (b) KNeighborsClassifier(n_neighbors=8, weights="distance"): Inverse of distance weights. Closer neighbours have a greater influence on classification.

8. MultinomialNB: Multinomial Naive Bayes, assumes a Multinomial Distribution of features.
9. BernoulliNB: Bernoulli Naive Bayes, assumes a Bernoulli Distribution of features.
10. DecisionTreeClassifier: Creates a rule based decision tree inferred from data for predict.
11. AdaBoostClassifier(n_estimators=170): Fits N($n\_estimators$) Decision Tree Classifiers to samples from training data. Then assigns them weights based on errors. Using those weights, casts a voting to find the best prediction.

We have tried those 22 combination of algorithms and parameters, with 3 different data configuration parameters;

1. Binary Classification
2. Include user history
3. Ngram

Resulting with 8 different data configurations and 176 different combinations.

3.4.1 Precision, Recall, Fscore

To represent the results of those 176 combination, micro precision, recall and fscore are used.

3.4.2 Execution

To execute those tasks, we have decided to use Google Cloud instances. Google Cloud offers 300$ trial usage for users who want to try out their services and it allows up to 32 cores in this trial period. We have implemented an argument to execute all those tasks on multiple google cloud instances, in parallel. Created a template operating system including an executable version of our implementation. Created 15 two core execution instances from this template, 1 MongoDB instance to serve data, store results and progress. It took 35 mins to finish the execution of those 176 tasks.

4. Results

Using Scikit-learn[PVG+11] we tried location classification using 22 models on 8 different settings. Those results have been shared in Figure 1 and 2. Figures contain 12 subfigures, showing precision, recall and f1scores for different combinations of ngram(1,2) and include user history(0,1) parameters. Each figure contains the relative micro results for 22 different models using the Box plots[MTL78]. Each box plot shows outliers with plusses(+), max and min value as dashes, a bounding box of upper and lower quartiles, a red dash showing the mean(instead of median). Because each class contains equal amount of data, mean of micro precision, recall and f1score can be interpreted as the macro precision, recall and f1score. Each subfigure contains a green line indicating the baseline macro precision, recall or f1score. In subfigures, models have been labelled with numbers ranging between 0 and 21. Those labels are given in Table 1.

Tall boxes in the figures represent unstable results, mostly caused by models predicting one or more classes better than others or failing to classify one or more classes not as good as others.

0	LogisticRegressionCV	1	SGDClassifier(loss="hinge")
2	SGDClassifier(loss="log")	3	SGDClassifier(loss="modified_huber")
4	SGDClassifier(loss="squared_hinge")	5	SGDClassifier(loss="perceptron")
6	SGDClassifier(loss="squared_loss")	7	SGDClassifier(loss="huber")
8	SGDClassifier(loss="epsilon_insensitive")	9	SGDClassifier(loss="squared_epsilon_insensitive")
10	Perceptron	11	PassiveAggressiveClassifier(loss="hinge")
12	PassiveAggressiveClassifier(loss="squared_hinge")	13	SVC(kernel="poly")
14	SVC(kernel="sigmoid")	15	LinearSVC
16	KNeighborsClassifier(weights="uniform")	17	KNeighborsClassifier(weights="distance")
18	MultinomialNB	19	BernoulliNB
20	DecisionTreeClassifier	21	AdaBoostClassifier

Table 1: Classifier Model Labels

4.1 Parameters

4.1.1 Ngram

Effects of including token bigrams with unigrams did not have major implications in the results. This parameter has changed the average f1scores of models by +0.04 to -0.05.

4.1.2 Including User History

Including history in the training and testing phases reduced the classifier performances in almost all cases. Overall average decrease in f1scores was 0.076 with values ranging from 0.21 to -0.09.

4.2 Multi-Class Classification

Baseline for Multi-Class classification is 0.125. Since we have 8 classes in total for this task, if a model has predicted only one class, that baseline value would be the macro precision, recall and f1score. For this task, top result was achieved by LogisticRegressionCV with an fscore of 0.443 and a recall of 0.50.

4.3 Binary Classification

Baseline for this task is 0.5. With bigrams, BernoulliNB, SGD with modified huber loss function and SGD with hinge loss function have achieved the top 3 f1scores of 0.734, 0.726, 0.72, respectively. Only with unigrams BernoulliNB, MultinomialNB and PassiveAgressive-Classifier with hinge loss function achieved the top 3 f1scores of 0.726, 0.719 and 0.717, respectively.

5. Conclusion and Discussion

In this research we have seen that binary classification on tweet locations can get up to 73% macro precision. Compared to previous studies dealing with this problem in a multi-class perspective, if the task is to predict if a tweet is belonging to one particular location, this looks like a promising result. But the fact of all our data belonging to geotagged tweets may compromising the validity of these results.

Refining the training dataset with less application generated content and more data would yield with more authentic research result.

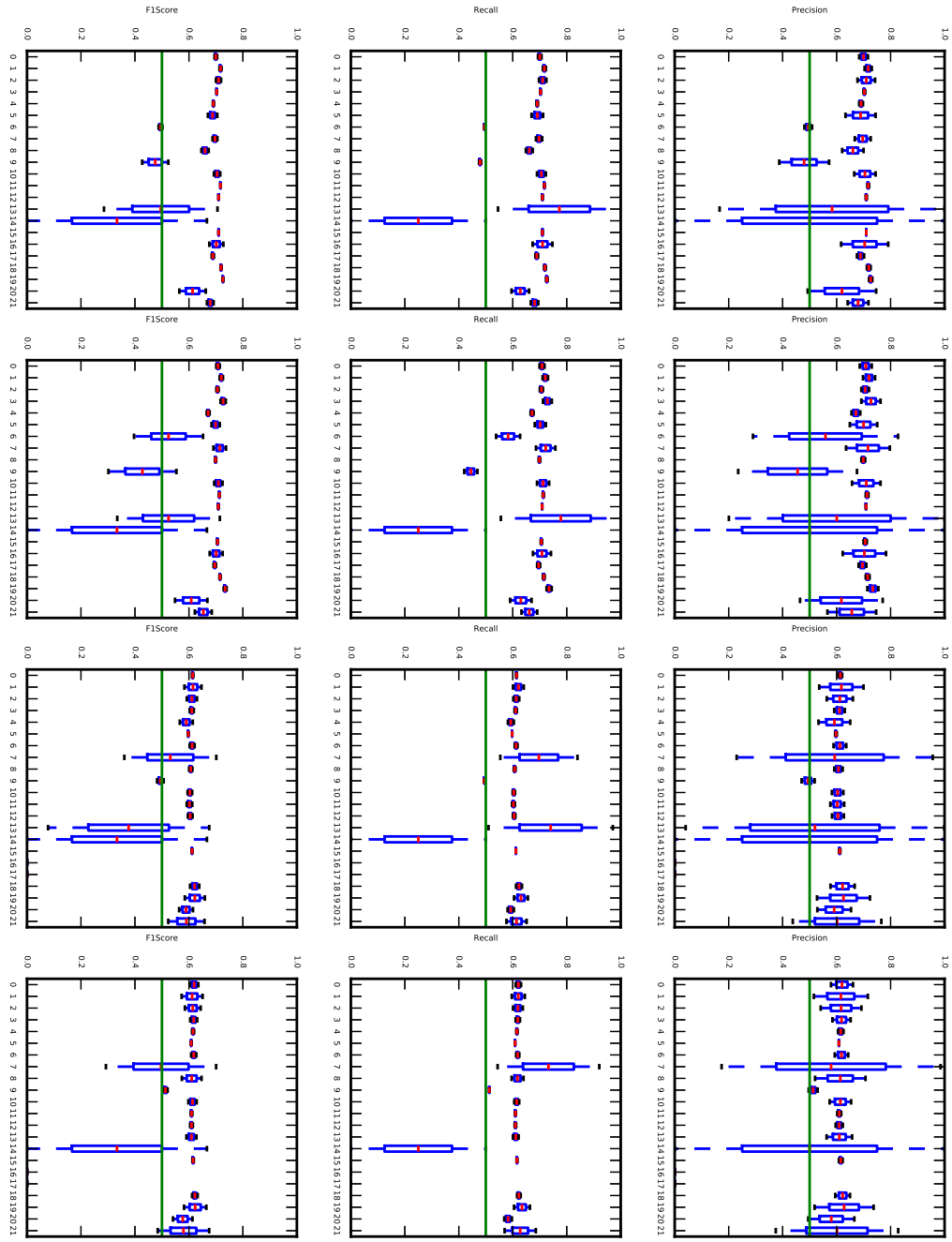

Figure 1: Binary classification results

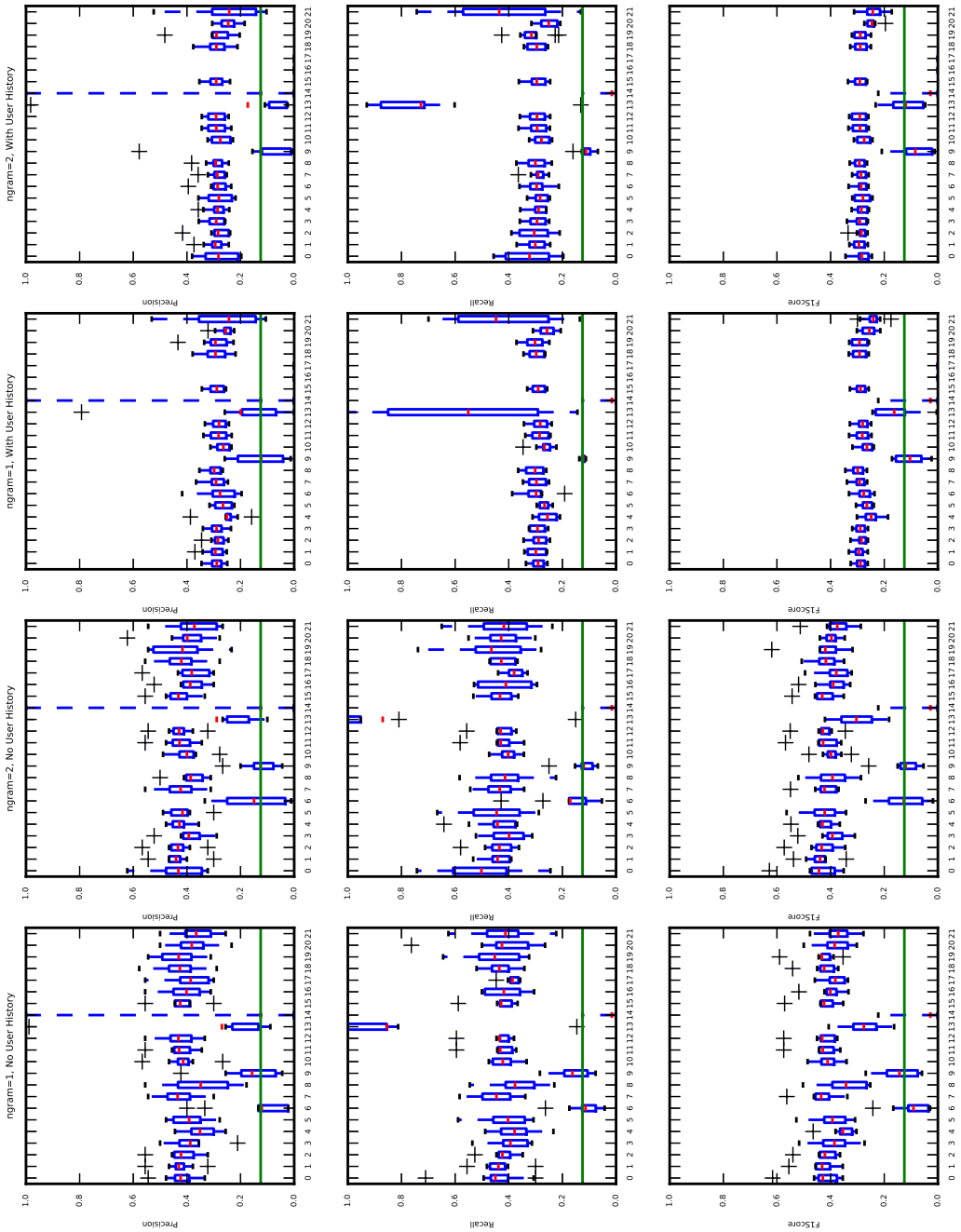

Figure 2: Multi-Class classification results

References

[CCL10] Zhiyuan Cheng, James Caverlee, and Kyumin Lee. You are where you tweet: a content-based approach to geo-locating twitter users. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 759–768. ACM, 2010.

[CL97] Bradley P Carlin and Thomas A Louis. Bayes and empirical bayes methods for data analysis. *Statistics and Computing*, 7(2):153–154, 1997.

[DeG] Morris DeGroot. H (1970): Optimal statistical decisions.

[EOSX10] Jacob Eisenstein, Brendan O'Connor, Noah A Smith, and Eric P Xing. A latent variable model for geographic lexical variation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1277–1287. Association for Computational Linguistics, 2010.

[GHG14] Mark Graham, Scott A Hale, and Devin Gaffney. Where in the world are you? geolocation and language identification in twitter. *The Professional Geographer*, 66(4):568–578, 2014.

[GSO+11] Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A Smith. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 42–47. Association for Computational Linguistics, 2011.

[HHSC11] Brent Hecht, Lichan Hong, Bongwon Suh, and Ed H Chi. Tweets from justin bieber's heart: the dynamics of the location field in user profiles. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 237–246. ACM, 2011.

[MN+98] Andrew McCallum, Kamal Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer, 1998.

[MND12] Jalal Mahmud, Jeffrey Nichols, and Clemens Drews. Where is this tweet from? inferring home locations of twitter users. *ICWSM*, 12:511–514, 2012.

[MND14] Jalal Mahmud, Jeffrey Nichols, and Clemens Drews. Home location identification of twitter users. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(3):47, 2014.

[MTL78] Robert McGill, John W Tukey, and Wayne A Larsen. Variations of box plots. *The American Statistician*, 32(1):12–16, 1978.

[PP10] Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *LREC*, volume 10, pages 1320–1326, 2010.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[SKB12] Adam Sadilek, Henry Kautz, and Jeffrey P Bigham. Finding your friends and following them to where you are. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 723–732. ACM, 2012.

[WB11] Benjamin P Wing and Jason Baldridge. Simple supervised document geolocation with geodesic grids. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 955–964. Association for Computational Linguistics, 2011.