Bachelor thesis

Native language identification on Twitter user-generated data via ensemble learning

Fachbereich INW
Hochschule Merseburg
August 2019

vorgelegt von: Serhii Hamotskyi
Matrikel-Nr.: 21995
shamotskyi@gmail.com

eingereicht bei: Prof. Dr. Michael Schenke
Zweitgutachter: Prof. Dr. Eckhard Liebscher

Contents

Chapter 1

Introduction

1.1 Vocabulary / Terms used

- Mention – As used in this Bachelor's thesis, a mention is a Twitter user's username preceded by "@" used inside a Tweet. (@example)
- Hashtag - A word or phrase preceded by a hash sign ("#"). Used inside messages in many social networks to identify a specific topic.
- POS – Part of speech – a category to which a word belongs, usually divided on the basis of their meaning, form or syntactic function, such as (in English) noun, pronoun, verb, adjective etc.

1.2 Scope of the topic and hypotheses

In this thesis, I attempted to classify tweets written in English by the mother tongue of their author. The geographical location of the tweet was used as a proxy of the mother tongue. To test how much the two correlate, two of the target languages chosen, Spanish and Portuguese (Mexico and Brazil) are from the same language family (even though the countries themselves are geographically distant). Classification was attempted in two ways: firstly, based on punctuation, parts of speech, word count, and similar features not containing the actual words used, and secondly, using the words. It was hypothesized that the first classification attempt would be less precise than the second one, due to the lack of language- and country- specific word cues; that nevertheless the first set of features would be enough to offer at least

some success in the classification, and, if the geographical location is a good indicator of the mother tongue of the author, Brazil and Portugal would be the highest in misclassification (and possibly UK and India).

1.3 Background

This topic interested me because I noticed that the way people from a different linguistic background use a foreign language is different, and this difference goes deeper than accent or phonetics. A person's culture shapes the way they see the world (for example, perceptions of time [1]), and some argue that even the native language of a person does, insofar as the two can be separated, even though the latter idea (linguistic determinism) is much more controversial. (An often-quoted example of this is how native Spanish speakers were more likely to describe a bridge as "big", "dangerous", "long", "strong", "sturdy", "towering" and German speakers as "beautiful", "elegant", "peaceful" and "fragile". A bridge's grammatical gender is masculine in Spanish and feminine in German[2]).

Even if languages don't shape how we see the world, different languages use different paradigms to describe it. For example, in English the difference between "cup" and "glass" is mainly based on material, while in Russian shape plays a bigger role. Another one is how in German a dog "beißt" and a fly "sticht", and in Russian one single verb ("kusat") describes both[1]. If one learns a foreign language that divides the world into categories in different ways than their mother tongue, the lack of 1-to-1 correspondence between concepts might create errors when speaking the foreign language.

I was always interested in the way different people use language, and people of different linguistic backgrounds using similar words and making similar errors was one of the things I kept noticing. In the case of Germany and the English language, examples range from both obviously derived from German ("plaything" (=Spielzeug), "we meet *us* at", overuse of punctuation) to subtler ones (in my experiences German people are much more likely to call "dinner" a noon meal as opposed to an evening one, as would have been natural for me). This bachelor's thesis is one way to explore these differences using machine learning and a user-generated dataset.

[1] Mother Tongues, available online at http://slavenorth.com/columns/sanskrit.htm, is a fascinating essay exploring this kind of connections in Sanskrit.

1.4 Native language identification

Native-language identification (NLI, NLID), also known as First language identification, is the task of determining an author's native language (L1) based only on their writings in a second language (L2). This is usually framed as a supervised classification task where the set of L1 is known. It works under the assumption that an author's linguistic background will dispose them towards particular language production patterns in their L2. [3]

Related fields and topics include cross-linguistic interference (CLI, also "language transfer"), that describes the effects of one's mother tongue on the acquisition of other languages, and is part of Second language acquisition (itself part of linguistics).

NLI is a relatively recent but rapidly growing area of research, and has many applications. The identification of typical usage patterns of L1 speakers can influence the way foreign languages are taught, and allow to create teaching resources tailored to the native language of the learners. Another practical area where NLI is applied is forensic linguistics and authorship identification (for example, in case of a ransom note and no known suspects, clues about the writer's linguistic background might be valuable).

1.5 Related work

The NLI shared task [4] is a NLI competition and the source of most papers [3] on the topic. The state of the art paper [5] uses stacked generalization and multiple features, and uses the TOEFL11 corpus[6] as does most other research on NLI. The report on the TOEFL11 corpus contains a description of the other existing NLI corpora.

"Native Language Identification with User Generated Content" [7] is a very relevant paper focusing on much longer Reddit source texts and uses some social-network specific features, such as subreddits.

On the day before printing I found "Predicting Foreign Language Usage from English-Only Social Media Posts" [8], which uses Twitter data, but does not call itself NLI, more correctly stating that it predicts the other languages spoken by the person. (This thesis, strictly speaking, predicts the country and not the L1.) The criteria is if the person tweets in other languages

except English, and the classification is done on English tweets. Since NLI is not mentioned it wasn't found during my initial literature review, and the results are neither compared to nor used nor mentioned anywhere in this thesis due to time constraints.

Chapter 2

Methods

2.1 Classification

Classification, in machine learning and statistics, is the process of identifying to which category a new observation belongs[9]. For example, given a dataset of petal length, number of petals etc, attempt to classify which kind of plant it is. Or, in the case of text, classifying incoming mail as questions, invoices, spam, etc. For this a number of algorithms exist, some of them described below.

2.1.1 Random Forest

Random forests are an ensemble learning method for classification, regression and other tasks. It is a classifier building a forest of decision trees, each with a different subset of features. This improves the accuracy in the case of many input variables[10]. Each tree "votes" for a class, and the class with the most "votes" is chosen. Intuitively each tree is a series of yes/no questions.

Lastly, this is relatively fast algorithm (on my datasets too) and it does not suffer from overfitting.

2.1.2 SVM

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outlier detection.

The advantages are [11]:

- Effective in high-dimensional areas (such as POS-ngrams used here)
- Still works when the number of dimensions is more than the number of samples (in my balanced dataset three were 49876 tweets and, for example, 156353 POS 1-4 n-gram features)
- Uses only a subset of features (support vectors) in the decision function
- Versatile

It works by finding the boundary between different classes in such a way as to maximize the distance between the boundary and the classes.

In the case of multi-class classification SVCs implemented in sklearn use one-against-one approach.

Among the parameters, one of the important ones is C, the regularization parameter. It basically sets how sensitive should the classifier be to outliers: a high C means a smaller margin is chosen if the results of the classification are better. If the C is low, it will choose a hyperplane that maximizes the margin, even if it means more points are misclassified. Usually, for noisy datasets a lower C works better.

Fig. 2.1 in a demonstration of the above, taken from [12]. While the red line minimizes classification errors, it will not generalize. The green boundary is a much better one, it maximizes the margin between it and the classes it separates, and it will probably deal much better with unseen data. But to set it some tolerance to misclassified semi-outlier elements is needed.

2.1.3 Stochastic Linear Descent

SGD, or stochastic gradient descent, is an optimization method. SVM or Logistic regression can be thought as classification algorithms that define a loss function, and SGD is a method that optimizes/minimizes it.

SGD is implemented in scikit-learn and is quite popular in for large-scale sparse ML problems often encountered in text classification and natural

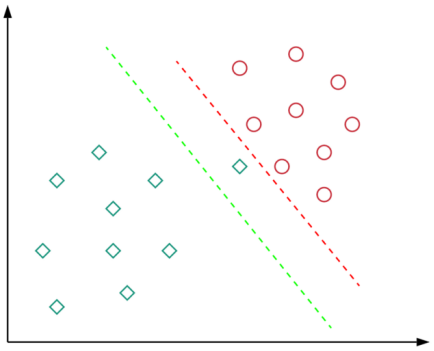


Figure 2.1: Demonstration of C and decision boundaries in a SVM [12]

language processing [13]. It's very efficient, but requires careful tuning of hyperparameters.

In this thesis, two loss functions are used: loss and hinge. The first one implements logistic regression, the second one a soft-margin SVM. A soft-margin allows SVMs to make more mistakes in classification to keep the margin as wide as possible, this is very helpful for noisy and/or linearly inseparable data.

SGDClassifier also supports multi-class classification, in a one-versus-all scheme.

2.1.4 Neural networks

A neural network is a model based on the structure of biological neural networks. It usually consists of an input layer, zero or more hidden layers, and one output layer. Each layer is composed of neurons. A neuron has an activation function, and it processes the inputs from the previous layer and transmits them to the next. The connections between the neurons have a weight. The inputs to a neuron are multiplied by the weight of the connection, summed, and then passed through the activation functions that controls which values will the neuron return.

A 5-layer feedforward network was used in this thesis, composed of dense layers (100, 500 and 50 neurons each) and dropout layers, for the purpose of predicting a final classification score based on the predictions given by

the other classifiers. Dropout is a technique for reducing overfitting, and it consists of ignoring some neurons during parts of the training. As a result, not all neurons are trained on all the training data, and the network learns much more robust features that are more likely to generalize. A dropout rate of 0.2 was used, and it meant that each input to the dropout layer hat 20% probability of being set to zero during each update.

2.1.5 Ensemble learning

Ensemble learning is the use of multiple learning algorithms to obtain better results than the use of one algorithm alone. In this thesis, an ensemble was used at the end, when results of the different classification algorithms were combined. The main idea is that the different algorithms ("weak learners") can be combined in a way that compensate for the weaknesses of one using the strengths of another; this is sometimes framed as variance and bias.

Both bias and variance can be reduced by the right ensemble. The hypothesis is that the different classifiers don't perform equally well on all test-cases or on all classes, and with the use of an ensemble the model will not suffer by predictable errors from a single source.

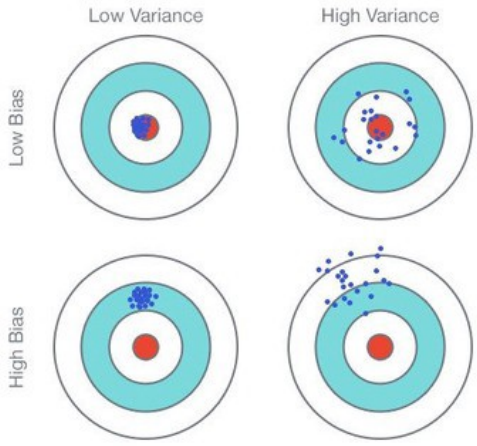


Fig. 1: Graphical Illustration of bias-variance trade-off , Source: Scott Fortmann-Roe., Understanding Bias-Variance Trade-off

Three basic approaches exist: bagging, boosting, stacking.

Bagging decreases the model's variance by training multiple similar models

in parallel but independently from each other, then combines the predictions.

Boosting learns similar models sequentially, usually emphasizing instances misclassified by the previous model. Basically, a new model is created and trained from the errors of the previous learner.

Stacking is an ensemble model where a new model is trained from the combined predictions of the previous models.

In this thesis, stacking is used, and a meta-learner (a neural network) is trained on the predictions of weaker classifiers.

2.2 Features

2.2.1 TF-idf

TF-idf is a way to score terms by their importance in the dataset. It's a product of two parts, TF and IDF. **TF is the term frequency – how much is the term used, relatively.** For example, in the previous sentence, 'the' his used twice, there are 12 words, so $tf(t,d) = 2/12 = 0.167$. IDF is the inverse document frequency – it's the measure of how significant is that term over the entire corpus. It's given by the number of sentences divided by the number of sentences containing the term.

In the above paragraph there are 7 'the', 6 sentences and 70 words. IDF is calculated by the log of total number of sentences divided by the number of them containing the term – in this case, 5 sentences contain 'the'. $df(t,D) = log(6/5) = 0.18$.

The TF-IDF is calculated as $tfidf = tf(t,d) * idf(t,D)$, where d is the sentence and D is the paragraph. The TF-IDF then is $0.167 * 0.18 = 0.03$.

The same calculation for "term", in that same sentence, is $tfidf = tf * idf = 2/12 * log(6/4) = 0.04$.

The same calculation for "frequency" is $tfidf = tf * idf = 1/12 * log(6/2) = 0.09$.

Even on such a small scale it describes the relative 'importance' of these words in the paragraph quite well. Sometimes different formulas are used both for TF and IDF, but the idea is the same.

The same holds for n-grams.

2.2.2 N-grams

Words cannot be directly fed to an algorithm, and language models are used to transform some input text to numbers. The simplest language model is Bag of words, which describes the text by counting (and possibly normalizing, such as TF-IDF) the tokens inside it. While this works for some settings, such as topic classification, such a representation completely loses the relative position of words. The relative position can influence the meaning, for example "alcohol free" vs "free alcohol".

N-grams are one way to solve this problem. A n-gram is a sequence of items from a source text in the order they appeared, be they words, characters, syllables etc. For example, a series of 2-grams would be: 'N-grams are', 'are one', 'one way', 'way to', 'to solve', 'solve this', 'this problem'.

TF-IDF N-grams were used in this thesis to represent both POS-tags and tokens.

2.2.3 Part-of-speech tags

Part-of-speech (POS) tagging is the process of marking up a word as corresponding to a particular part of speech (such as noun, verb, adjective, etc.), by definition and by context.

Except the usual noun, verb, article, adjective, preposition, pronoun, adverb, conjunction and interjection, many more categories can be distinguished – such as singular/plural, case of the word, grammatical gender, tense. While just by tagging the words themselves without any contextual information gives an accuracy of 90%, usually a number of more complex techniques are used, and not all words can be clearly marked as part of a class. ("Refuse" can be both a verb or a noun, in "The Duchess was *entertaining* last night" 'entertaining' may be a verb or an adjective).[14]

The POS tagging used here was the one implemented in NLTK (Natural language toolkit), a suite for NLP for English written in Python. An example output is:

> ('The', 'DT'), ('Duchess', 'NNP'), ('was', 'VBD'), ('entertaining', 'VBG'), ('last', 'JJ'), ('night', 'NN')

It interprets 'Duchess' as a proper noun (like Bob) and 'entertaining' as a verb.

The following POS tags are used, list and examples from [15]:

- CC coordinating conjunction
- CD cardinal digit
- DT determiner
- EX existential there (like: “there is” … think of it like “there exists”)
- FW foreign word
- IN preposition/subordinating conjunction
- JJ adjective ‘big’
- JJR adjective, comparative ‘bigger’
- JJS adjective, superlative ‘biggest’
- LS list marker 1)
- MD modal could, will
- NN noun, singular ‘desk’
- NNS noun plural ‘desks’
- NNP proper noun, singular ‘Harrison’
- NNPS proper noun, plural ‘Americans’
- PDT predeterminer ‘all the kids’
- POS possessive ending parent’s
- PRP personal pronoun I, he, she
- PRP$ possessive pronoun my, his, hers
- RB adverb very, silently,
- RBR adverb, comparative better
- RBS adverb, superlative best
- RP particle give up
- TO to go ‘to’ the store.
- UH interjection errrrrrrrm
- VB verb, base form take
- VBD verb, past tense took
- VBG verb, gerund/present participle taking
- VBN verb, past participle taken
- VBP verb, sing. present, non-3d take
- VBZ verb, 3rd person sing. present takes
- WDT wh-determiner which
- WP wh-pronoun who, what
- WP$ possessive wh-pronoun whose
- WRB wh-abverb where, when

POS tags added during dataset preparation:

- HASHTAG #hashtag

- URL http://whitehouse.gov
- MENTION @mention

2.3 Evaluation

I needed to evaluate the results of the classifications in some way, to be able to compare different classificators and datasets to each other and to the baseline. The task was a multi-class classification on a balanced dataset.

2.3.1 Accuracy, precision, recall, F-score

Accuracy is one of the simplest performance measures, and it's **the overall effectiveness of a classifier**. It is a ratio of the correct predictions to the total number of predictions. It's much more useful in the case of balanced datasets, and I used it as the main metric, since most of the classifications were on a balanced dataset. A **balanced accuracy score** is one way to use accuracy in the case of class imbalance, and in it the contribution of each sample is weighted according to the inverse prevalence of its true class (see Fig. 2.2).

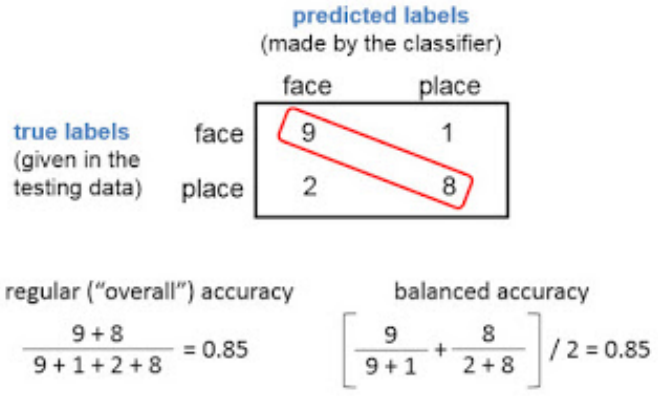


Figure 2.2: Accuracy vs balanced accuracy.[16]

Precision for a given class X is the number of items correctly predicted out of all predicted labels. Intuitively, precision is the ability of the classifier not to label as positive a sample that is negative[17]. For example, for English, it

would be the number of correctly predicted UK tweets divided by all tweets predicted as being UK ones.

Recall for a class X is the number of true predicted members of X divided by all instances that *should have been labeled X*.It measures **the effectiveness of a classifier to identify positive labels** [18]

For example, a precision of 0.6 and recall of 0.2 for class X means that out of the times X was predicted, 60% of times it was correct; and out of all the times X should have been predicted, this happened only 20% of times.

F-score is the weighted harmonic mean of the precision and recall, and it's a measure that takes both precision and recall into account.

2.3.2 Confusion matrix

In the case of multi-class classification, a confusion matrix visualizes the performance of the algorithm. As the name implies, it makes it easy to see which classes are the hardest for an algorithm, and which classes tend to be confused.

Fig. 2.3 is an example of a confusion-matrix of a near-perfect classification. The main diagonal is clearly seen, and it contains the number of correct predictions. Incorrect predictions are located outside of this diagonal. The "10" in the last row means that 10 tweets from Mexico were (incorrectly) classified as being from Brazil. Section 4.2.2 contains an example of worst-case confusion matrix.

Section 4.3 has more realistic confusion matrices, and confusions (such as between BR-MX or between UK-IN) are clearly seen. This would not have been clear from any other typical metrics.

The confusion matrices used in this thesis were drawn using adapted source code from the scikit-learn documentation[19].

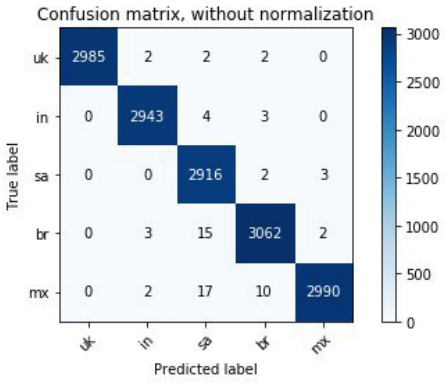


Figure 2.3: Near-perfect confusion matrix

Chapter 3

Construction of training data

3.1 Dataset

The dataset contains geo-tagged tweets from Twitter. Twitter is a social network with 126 million active users, on which users interact with each other with short messages (tweets). Tweets originally were restricted to 140 characters, but in 2017 the limit was doubled to 280 for most languages. Users can optionally specify a location in their tweets, and when searching tweets can be filtered by their location. The location can either be specified as exact coordinates or a Twitter "Place", which is a polygon, and has additional semantic information connected to it, like a city name[20][21].

Additionally, Twitter automatically identifies the language of tweets, and it's possible to filter tweets by their location (if they are geo-enabled) and their language at the same time.

3.1.1 Advantages

- Tweets are ephemeral in nature, and Twitter is not the place for balanced in-depth analysis. The "here-and-now" of the content may mean that it's less thought-out and less studied, which may influence the way the language is used. NLI is very often done on essays written in a foreign language for the purpose of language testing, where an effort is made to use correct grammar and punctuation. User-generated social media content may have the advantage of having L1-grammar and

language patterns more clearly visible, since less effort and attention is dedicated to them.

- As opposed to usual NLI-datasets, which may be contextually limited (essays are written on certain topics), user-generated content may be seen as a slice-of-life source of material that is closer to what people think and experience in real life – very different people in different experiences and different lives. While it may be argued that Twitter users are not a representative demographic, it's not-representative in a different way from the others (most are essays written either in the context of tests such as TOEFL or by students; students writing essays in a foreign language for grades is only one context where a foreign language is used, Twitter is another one).

3.1.2 Drawbacks

A user-generated dataset like the one used here is not a "clean" dataset.

3.1.2.1 Twitter-specific issues

- For the purposes of this thesis, I assumed that the tweet location is a good proxy of the first language of the author, which is not always the case — people writing English tweets while being located in a certain country does ot mean that they are a native speaker of that country's language. This *will* impact the results of any classification. One way to overcome this is to use the last N tweets of a user, check how many of them were written in a certain country, and assume that it's their country, and not that they, say, are on vacation. Or to assume that followers of accounts like @timesofindia are Indians. Or classify based on Tweets of users who mostly tweet in their L1 (as detected by Twitter), and use their tweets in English for the dataset.
- Not all tweets are written by real people — Twitter has a big number of bots and tweets posted automatically. By some estimates, as much as 24% of tweets are created by bots [22], and 9% to 15% of active users are bots [23]. Some of them are weather bots, financial aggregation bots, or novelty bots with text that is not representative of real life language used by L1 native speakers.
- In line with the point above, tweets can be automatically generated. For example, some users choose to configure automatic cross-posting

of their activity in other social networks. This creates tweets that are very similar to each other ("%@username% just posted a photo on his Instagram account: %link%"). Twitter used to have an API feature to show and/or filter posts made from other clients, but this option was removed as part of a policy aimed at focusing on their official website and clients (as opposed to third party apps)[24]. This means that such content has to be found and filtered manually, which was attempted but was not too effective. In my dataset, which initially contained 341353 Tweets, 5889 (~1.7%) had identical text.

3.1.2.2 Issues stemming from the type of data used

- Tweets (in the general case) can be up to 280 characters long, but in fact are usually much shorter (half of the tweets were less than 74 characters long). This means that a tweet may not contain enough information to distinguish the L1 of the author. As mentioned above, usually NLI is done on much longer texts, such as essays.
- Twitter's language detection is not perfect, especially with shorter texts. I did not quantify this, but visually about 5% of the supposedly English tweets were not in English (Brazilian/Portuguese seemed to be the language most often misclassified as English by Twitter).

Fig. 3.1 is a random selection of unprocessed tweets from the dataset. Note the length, how challenging the grammar for POS-tagging is, how not all of them are in English, and in general how little material there is compared to, for example, essays or longer posts.

One way to overcome part of these disadvantages would be to get a number of tweets from the same user, thereby getting a synthetic dataset of longer texts, or getting longer texts of *different* users of the same category.

3.1.3 Additional remarks

Usually, NLI done on spoken text transcription is much less precise than NLI done on essays or written text, some sources report as much as a 10% difference. There may be various causes for this, one of them may be that transcribed spoken text contains fewer features such as punctuation, and that people, when speaking, tend to use much simpler words and grammatical constructions. While equating tweets to spoken text transcription would

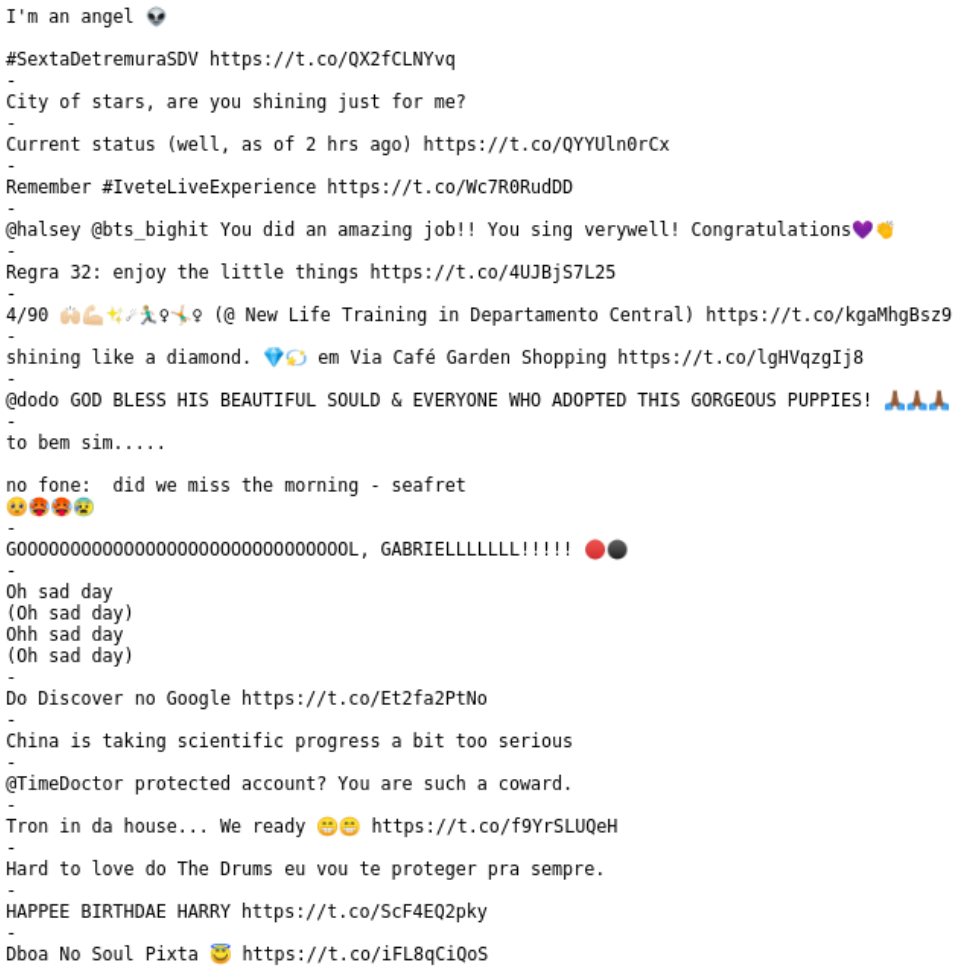


Figure 3.1: Examples of tweets

be questionable, Tweets are usually written much closer to how people actually speak. The format is quite informal, and most people would not spend too much effort on correct grammar and punctuation for a tweet (this more typical for essays, and it's essays that make up typical NLI datasets, such as the TOEFL11 NLI corpus [6]).

Mitigating part of the disadvantages mentioned in this chapter was not attempted, but it would create a much better dataset and should not be hard. It would be an extremely interesting route for further research, because combining most of the ideas suggested would overcome part of the issues of this dataset while retaining most advantages.

3.1.4 Collection

The dataset was collected in April and August 2019 in a period of 4-5 days. It contains Tweets from regions delimited by the bounding boxes (GPS coordinates) in the English language.

3.1.4.1 Tweepy

For communication with the Twitter API, the tweepy ("An easy-to-use Python library for accessing the Twitter API")[1] package has been used. The final script used for collecting the tweets from a Twitter Stream was heavily based on GeoSearch-Tweepy[25], a script which contained examples of how to filter Twitter Streams by geolocation data.

3.1.4.2 Countries and bounding boxes

The countries used in this thesis are **India, Saudi Arabia, Brazil, Mexico, and Great Britain**. These countries were chosen because of their large number of Twitter users (to make data collection easier) and because they represent languages from different language families.

Brazil and Mexico have different languages that nonetheless belong to the same language family, and if the assumptions and process are valid, they should have the highest mutual mis-classification.

[1] https://www.tweepy.org/

The bounding box for Great Britain also contains Ireland, since they are close linguistically (and fit neatly into one bounding box). In India a lot of languages are spoken, but most belong to two unrelated language families: Indo-European->Indo-Iranian and Dravidian, 78% and 20% of speakers respectively.

Country	Bounding box	Language	Language family
India	(67.46, 5.43, 90.71, 27.21)	122 major languages	78% Indo-European; 19.64% Dravidian; some English
Saudi Arabia	(34.77, 12.86, 49.84, 30.19), (48.1, 13.93, 60.25, 24.77)	Arabic	Afro-Asiatic → Semitic
Brazil	(-58.55, -30.11, -35.26, 2.5), (-67.3, -13.03, -34.38, 1.53)	Portuguese	Indo-European → Italic
Mexico	(-112.59, 17.98, -85.38, 27.75)	Spanish (de facto)	Indo-European → Italic
Great Britain	(-10.68, 50.15, 1.41, 59.69)	English	Indo-European → Germanic

3.1.4.3 First collection results

341353 Tweets were collected in April and August 2019.

3.1.5 Cleanup and dataset preparation

To simplify further analyses, I started with a cleaned-up version of the dataset. The tweets were changed or removed, as described below, and some initial features were added. All of this was done using pandas, a data analysis framework/library for Python.

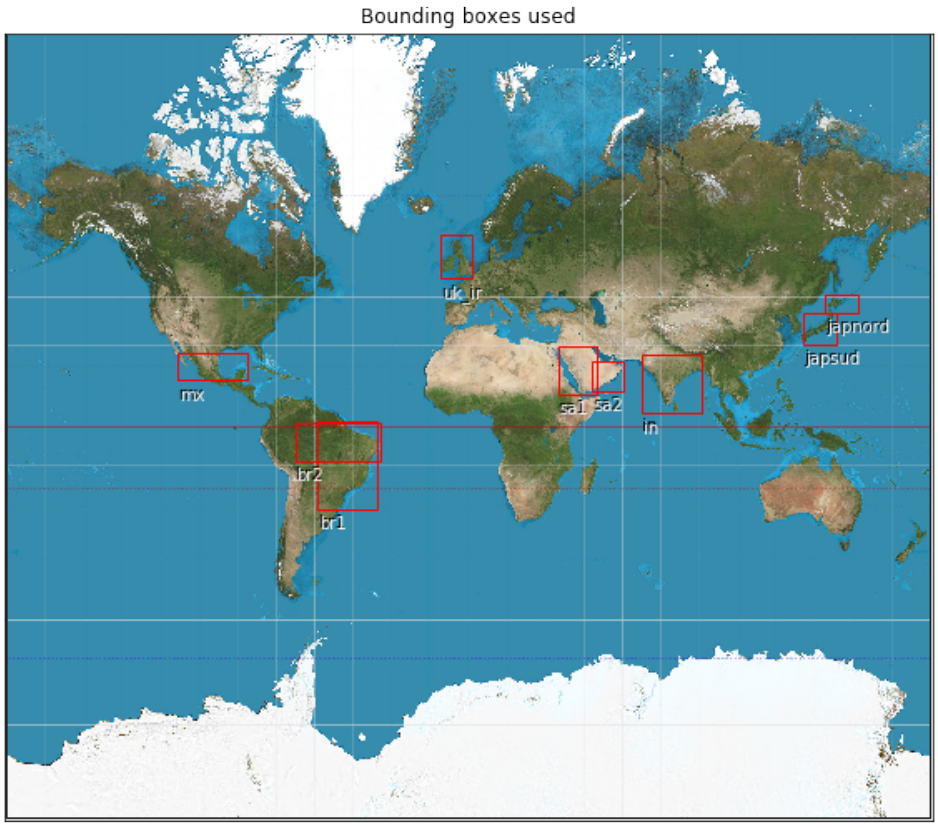


Figure 3.2: Bounding boxes used. Background image © Daniel Strebe, 15 August 2011, CC BY-SA 3.0. Japan was later removed from classification.

3.1.5.1 Original dataset

The csv file with the dataset had the rows "userid, username, location, datetime, lat, long, lang, text", containing the author's Twitter user ID, @username, location as provided by the user themselves, the date and time of the tweet in UTC, latitude, longitude, language of the tweet as detected by Twitter, and the text of the tweet.

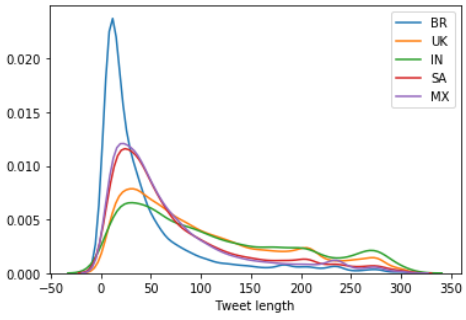
3.1.5.2 Removing duplicates, short and possibly automatically generated tweets

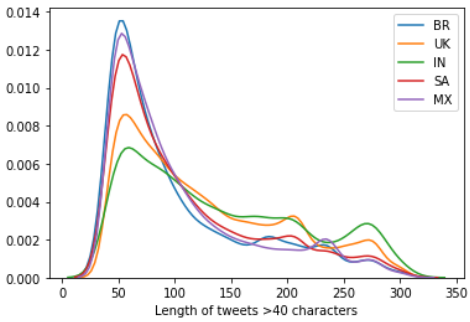
First, tweets containing the exact same value in "text" were removed. There were 5889 (~1.7%) such tweets.

Tweets containing substrings indicative of automatic posting ("Just posted", "has just posted", "Want to work at", "I'm at") were also removed (also about 6%).

A number of tweets did not fit neatly inside bounding boxes. There may be various explanations for this, and they (77619, or ~22%) were also removed from the dataset.

Then, Tweets shorter than 40 characters (inside the `clean` column, so without counting mentions, see below) were deleted (78439 of the remaining 194351 ones, so ~40%).[2] The figures below show how the length of the tweets changed by country after this. The mode for `char_count` in this final dataset ranged from 42 for Brazil, Saudi Arabia and UK to 46 characters in India.





[2]This would make classifying real tweets shorter than 40 characters complicated, but I believe they would have been complicated to classify regardless.

This left a dataset of 104001 tweets, or just **30%** of the initial number.

3.1.5.3 Location and native language (L1)

Then, based on the latitude and longitude data, each tweet was labeled with one of the L1 categories. This was done by checking in which bounding box the tweet's GPS coordinates were located. The results can be seen on Fig. 3.2.

3.1.5.4 Balancing the dataset

The dataset at the beginning was unbalanced - as one can see on Fig. 3.3, the languages (=target classes) were not represented uniformly. Two versions of the dataset were created, a balanced and an unbalanced one, to observe their effects on classification.

The unbalanced dataset contained 104001 tweets, the balanced one 49876.

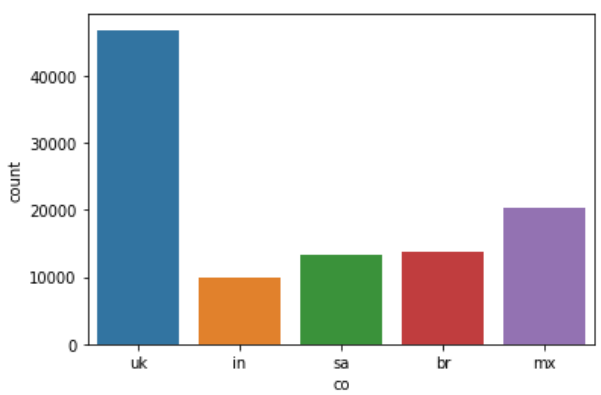


Figure 3.3: Language distribution of final unbalanced dataset

After basic testing, it was concluded that the balanced dataset would be a better choice, and all further tests and classifications were based on it.

3.2 Feature engineering

3.2.1 Removing superfluous mentions

In Twitter, it's possible to "mention" another user by prefixing their username with an "@" (`@example`). This is also done automatically when a user is replying to a tweet or otherwise taking part in a conversation with multiple users, in this case mentions are added to the beginning of the tweet (the user is free to override this though). The number of mentions does not count towards a tweet's limit of 280 characters (longest tweet in the dataset is 964 characters long). This meant that even though the raw character count was sometimes much more than 280 characters, the tweet was still useless from a NLI perspective. If a tweet is even 350 characters long and 300 of those characters were usernames, there's little data to work with.

Still, I felt that completely removing mentions was not a solution. Semantically, they represent something close to proper nouns, and their location in a tweet/sentence (and their number) might be significant.

It was decided to remove only the ones at the beginning, leaving maximum two mentions. For example, if a tweet has one or two mentions at the beginning, nothing changes, otherwise if there are n>2 mentions n-2 get removed (see Fig. 3.4 for an example).

This meant that superfluous mentions were removed while preserving as much of the information they give as possible. Leaving two mentions instead of one allows to preserve the grammatical number of that part of the tweet, which might be significant.

On this step line breaks were also removed.

137878	br	@50years_music @JeffJ1961 @magsbelfast72 @NancyBourque55 @karlakakes56 @Peterst06328885 @kimberly_booher @alicesings @Sunfirekisses @VGrantly @PikeStanzak @PeterKieviet @marytheluckyone @retrositcom @auticomics @WOJBCarpenters @DickTiz94 @Lisamarie61 @LisaJmcqueen1 Good morning, beautiful angel.\nGod bless your Friday.\nThat's where you walk.\nYour path is covered.\nOf beautiful flowers... https://t.co/gn8BiK96aR	@Lisamarie61 @LisaJmcqueen1 Good morning, beautiful angel. God bless your Friday. That's where you walk. Your path is covered. Of beautiful flowers... https://t.co/gn8BiK96aR

Figure 3.4: Results of removing superfluous mentions from the beginning of tweets.

The original tweets were moved to the column `otext`, and the ones with fewer mentions were added in `text`.

3.2.2 Masking mentions, hashtags and URIs

A version of the tweets with *all* the mentions, hashtags and URIs removed was written to `clean`, and features like `char_count` and punctuation were calculated based on it. This left the question of what to do with mentions, hashtags and URIs in the actual training data. For further analysis, a column "masked" was created with the changes described below.

For bag of words and n-grams, the actual content of the mention is either irrelevant or counterproductive. In the first case, because a mention is the username of an account, and there are a lot of accounts, this would create 'words'/tokens that are all different between each other but mean the same semantically. In the latter case, mentions used as words/tokens might be counterproductive for the stated purpose of **native-language identification** based on user-generated data. For example, India's largest English-language newspaper is Times Of India (@timesofindia), and someone mentioning it has a high likelihood of being from India, regardless of the other linguistic features. So while this would improve the classification results, they would not generalize as they are largely based on social-network-specific features as opposed to linguistic ones.

The same may be said about hashtags – #LokSabhaElections2019 (Lok Sabha is the lower house of India's Parliament) was trending (a lot of users were tweeting about it) as I was gathering the dataset, and it would offer an easy way to "cheat". On the other hand, sometimes words inside sentences are used as hashtags (for example, in the tweet "what are we waiting for #MPN #NowUnited #MPN #Uniters" the first hashtag is part of the sentence itself, th e last three are just hashtags), and completely removing them is not a solution.

For URIs mostly the same applies – URIs are different, mean mostly the same thing, and in case when they are significant they are not the features we want to use for classifying using *language*.

So all mentions, hashtags and URIs replaced by the words "REPLMENTION", "REPLHASHTAG", "REPLURI" in the column "masked".

3.2.3 Basic features added during cleanup

All of these features were based on the column `clean`, the one with the Tweets after removing the all mentions and newlines.

- `char_count` - the number of raw characters in the tweet.
- `word_count` - the number of words in the tweet.
- `word_density` - the number of characters divided by the number of words.
- `punctuation` - number of punctuation characters divided by the number of words.
- `title_words` - number of words starting with an uppercase letter divided by number of words.
- `upper_case_words` - number of words written in all caps divided by number of words.

3.2.4 Tokenisation

The next step was to break the tweets in tokens. This was done on all columns and tests were ran, and at the end only the column "masked" was tokenized.

For this, a custom tokenizer was used. A standard tokenizer, for example one provided by `ntlk`, would not have detected the Twitter entities, such as @mentions or hashtags. A Python regex-based script was used for this, adapted from Marco Bonzanini's tokenizer[26].

3.2.5 Punctuation

I used Python's `string.punctuation` definition of punctuation, `'!"#$%&\'()*+,-./:;<=>?@[\\]^_\{|}~'`. For each one, a column was created, starting with "p" followed by symbol. In the dataset many more characters were present in this role, especially UTF8 characters, but they were not added separately.

3.2.6 Function words (a, the) counts

- `a` – number of occurrences of the word (not the letter) "a" divided by number of words.
- `the` – number of occurrences of the word "the" divided by number of words.

3.2.7 Parts of speech tagging

POS tagging in social media has its own complexities, well described in the dissertation of Tobias Horsmann "Robust Part-of-Speech Tagging of Social Media Text". [27] In the dissertation, the example on Figure 5 is given. I tagged parts of speech using nltk, which may be another reason of possible suboptimal performance, this is another source of noise in the source data.

Figure 3.5: Complexities of POS tagging user-generated data, figure from the dissertation of Tobias Horsmann on the topic [27].

We have used nltk's POS tagger, that encodes POS short strings. For each POS tag found in the entire dataset, a column was created, with the value of the number of occurrences of that POS in the Tweet.

This left the question of what to do with **mentions, hashtags and URIs**. I decided to tag them as their own parts of speech. While they don't exist in classical English grammar and tagging them would have been problematic, they carry important information, and their location and number may also be significant.

The example tweet below shows the steps from a tweet, to the masked text, to tokens, to the tagged POS:

- A huge thank you to @ROsterley from @CharityRetail for joining us today at the #HUKRetailConf held @ChesfordGrangeQ today. Fantastic insights into the #CharityRetail sector @hospiceuk #conference #hospice
- A huge thank you to REPLMENT from REPLMENT for joining us

today at the REPLHASH held REPLMENT today. Fantastic insights into the REPLHASH sector REPLMENT REPLHASH REPLHASH

- [‘A’, ‘huge’, ‘thank’, ‘you’, ‘to’, ‘REPLMENT’, ‘from’, ‘REPLMENT’, ‘for’, ‘joining’, ‘us’, ‘today’, ‘at’, ‘the’, ‘REPLHASH’, ‘held’, ‘REPLMENT’, ‘today’, ‘:’, ‘Fantastic’, ‘insights’, ‘into’, ‘the’, ‘REPLHASH’, ‘sector’, ‘REPLMENT’, ‘REPLHASH’, ‘REPLHASH’]
- [‘DT’, ‘JJ’, ‘NN’, ‘PRP’, ‘TO’, ‘MENTION’, ‘IN’, ‘MENTION’, ‘IN’, ‘VBG’, ‘PRP’, ‘NN’, ‘IN’, ‘DT’, ‘HASHTAG’, ‘VBD’, ‘MENTION’, ‘NN’, ‘:’, ‘JJ’, ‘NNS’, ‘IN’, ‘DT’, ‘HASHTAG’, ‘NN’, ‘MENTION’, ‘HASHTAG’, ‘HASHTAG’]

3.3 N-grams

A version of the tweets with *all* the mentions, hashtags and URIs removed was written to `clean`, and features like `char_count` and punctuation were not part of the dataset proper, and were calculated after the dataset described in this chapter was read from disk.Scikit-learn’s CountVectorizer was used, with **n-gram range of 1 to 4** for both POS n-grams and token n-grams (after testing, this was the value that performed best using virtually all classificators). Additionally, removing English stop-words seemed to improve predictions (which was surprising, at the beginning I thought that the number of stop-words and use patterns of stop-words would be significant), so this was done too when building n-grams.

3.4 Final feature sets

From here on, I’ll use the following definitions:

- Basic features:
  - Counts of various features: `char_count` `word_count` `word_density` `punctuation` `title_words` `upper_case_words` `the` `a` `pos_count`;
  - Counts of each POS tag seen in the dataset in the tweet
  - Number of punctuation marks: `p! p"` `p#`p$p%p&p’p(p)p*p+p,p-p.p/p:p;p<p= `p>` `p?` `p@` `p[` `p\` `p]` `pˆ` `p_` `p``p{p|p}p~‘
- POS n-grams: N-grams created from POS tags
- Token n-grams: N-grams created from tokens based on the column “masked” (the one not containing hashtags, mentions or URIs.

Datetime, location, language, latitude, longitude and datetime were never used. While date and especially time might have been interesting to take into account, they are too social-network specific, and I wanted to concentrate on the linguistic features. Social features for NLI and tweets classification have been used and very well described in [28] and [23].

Chapter 4

Classification

4.1 Classification goals

There were two main goals:

- classify using POS-tags, punctuation, word count, etc. - everything not using the actual word content
- classify using features above and word content

The reasoning for this was described in the Introduction: I wanted to see how successful would be classification if any not purely linguistic features are avoided.

It could be argued that the issues with hashtags described in Section 3.2.2 apply also to words, albeit a bit less. Since there was an election in India in the time the dataset was collected, some of the models might confer additional meaning to the usually neutral word "election" and give biased predictions to Tweets containing this word. This might be partly fixed by a much larger dataset gathered over longer periods of time. But the issue of some topics or words appearing more often in the tweets of certain countries simply because the concept they represent is more likely to be talked about in certain countries is much harder to mitigate this way.

This became clear when, at the beginning, during initial testing, I wanted to see if there are words much more likely to be used by certain L1-speakers. I first trained a NN on the same dataset using simple Bag of words features. Then I classified each single word seen in the dataset as if it were a tweet,

then sorted them by how confident the NN was in its classification. Unsurprisingly, the result was either city (or personal) names or words clearly belonging to a particular culture (such as 'lakh'/'crore' for big numbers in the case of India).

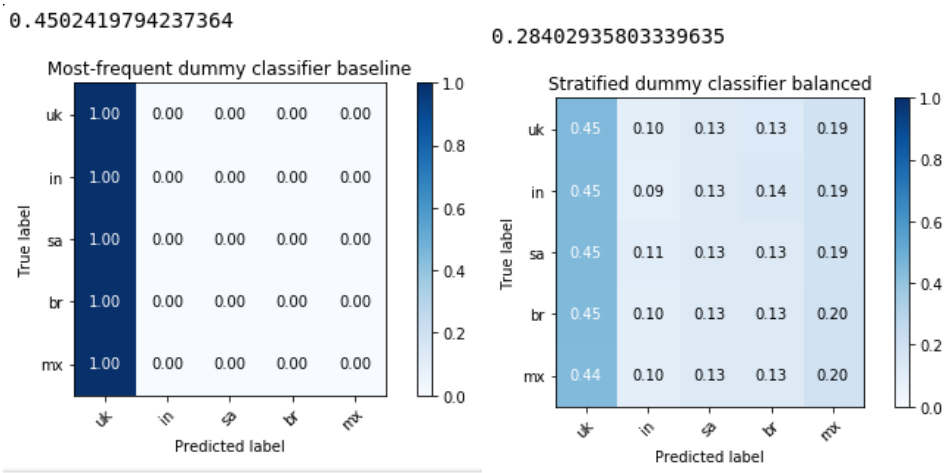
4.2 Baselines

First, baseline results were calculated, for both the balanced and unbalanced dataset.

Baseline results are needed to have a meaningful point of comparison. The final accuracy of the classifiers alone would not say much, since a 40% on a binary classification is much worse than a 40% on a multi-class classification. Lastly, as in the example of anomaly detection, if I have 0.01% anomalies and create a classifier that always returns "no anomaly" I'd have a 99.9% success rate and only the 99.9% baseline would show that the result is actually meaningless.

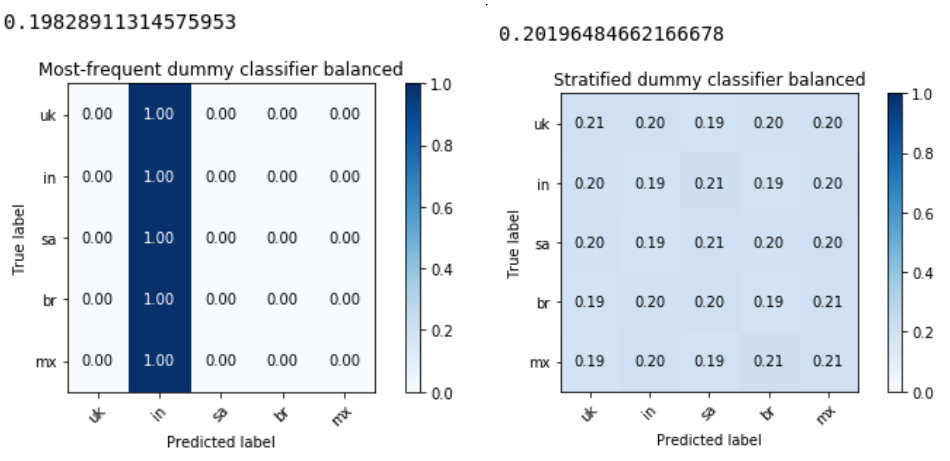
4.2.1 Baseline for unbalanced dataset

A stratified and a most-frequent dummy classifier was used, both provided by sklearn.dummy.DummyClassifier. [29] A stratified classifier returns values with the same distribution as the classes in the source data. It had **28% accuracy**. A most-frequent classifier returns only the most-frequent class, in this case UK, and it had **44% accuracy**.



4.2.2 Baseline for balanced dataset

For the balanced datasets, all classifiers had ~**20% accuracy**, that is 100% divided by the number of the target categories.



We will consider 20% as a good baseline for all future classification on the balanced dataset.

4.3 Classification on basic features

A number of tests were done on basic features as defined in Section 3.4.

The dataset was shuffled and then divided into a train and test set as usual, with a split of 0.7/0.3 training/testing, and it resulted into 34912/14963 and 72800/31201 train/test Tweets for balanced/unbalanced respectively. Same ration was done in the next section for POS and token n-grams. Additionally, the features were scaled to (-1, +1) using the standard settings of the Standardscaler of sklearn, since SGD is very sensitive to this.

This section has a better and more precise description of the tests taken to demonstrate the process, in the next section the process and confusion matrices are only shown for the ones used in the final estimator.

Having calculated the baselines, it's possible to start classifying the tweets. Three algorithms were used for this initial classification – a DNN classifier as implemented in Tensorflow's Estimators API, a Random forest algorithm and a SVM algorith, both available in scikit-learn.

4.3.1 Grid-search

A grid-search of optimal parameters was performed, using from `sklearn.model_selection.GridSearchCV`, using crossvalidation of 5.

For SVC the following parameters were tested:

```
{'C': [1, 10,100,1000], 'kernel': ['linear']},
{'C': [1, 10,100,1000], 'gamma': [0.001, 0.0001], 'kernel': ['rbf']}
```

For SGD the following was used:

```
{'penalty': ['none','l1','l2','elasticnet'], 'loss': ['hinge',
'log','modified_huber'],'alpha': [1e-1,1e-2,1e-3,1e-4,1e-5]},
```

For Random Forest, the parameters were:

```
{'n_estimators': [1, 10,100,1000], 'criterion': ['gini','entropy'],
'min_samples_split':[2,5,10],'min_samples_leaf':[1,5,10],
'max_features':['auto','sqrt','log2']},
```

For DNN, no automatic parameter search was performed, but many different options and architectures were manually tried.

Unless otherwise specified, the results are usually the ones of the classifier found to perform best on the data in question.

4.3.2 DNN on unbalanced dataset

I used Tensorflow's DNNClassifier implemented as part of the Estimators API. Two hidden layers were used, of 30 and 20. I did not use any cross-validation or hyperparameter tuning to choose the parameters. The choice of using two layers was partly motivated by [30] and the rule of thumb "less than half of the input layer" (89 features in this case), but I tried a lot of combinations even with hundreds of neurons, and there was a ceiling of accuracy at 52%.

The result was **52% accuracy**.

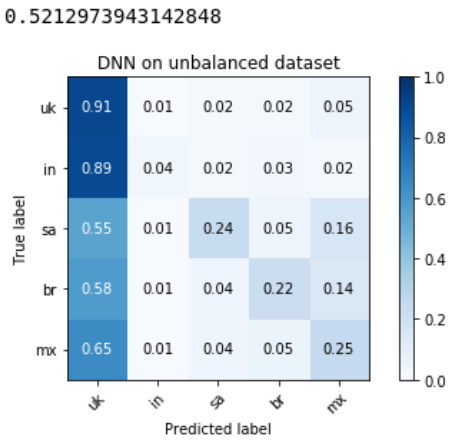
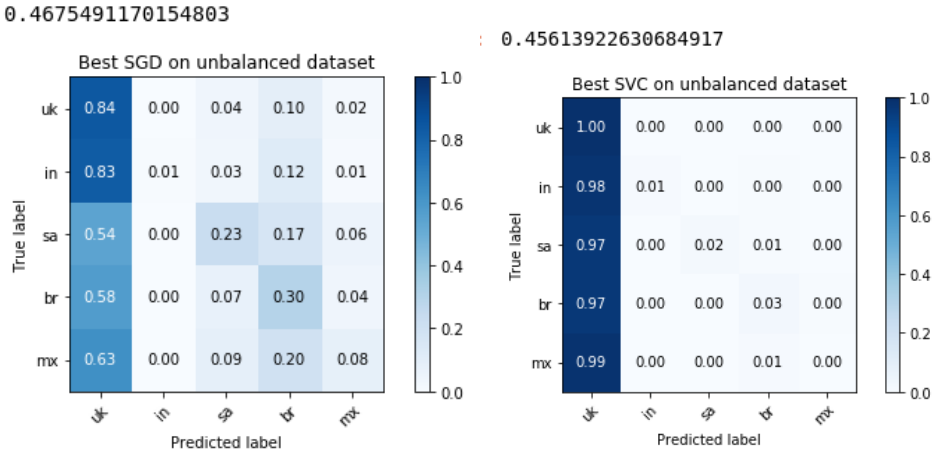


Figure 4.1: Confusion matrix of DNN classifier running on unbalanced dataset

4.3.3 SGD and SVM on unbalanced basic features



4.3.4 Random Forest on unbalanced dataset

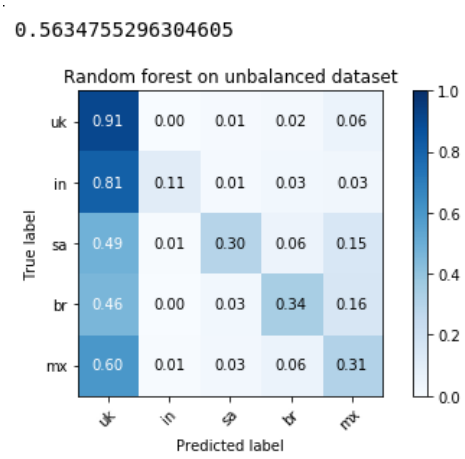


Figure 4.2: Confusion matrix of Random Forest classifier with 1000 estimators running on unbalanced dataset

Results are on Fig. 4.2. It's interesting to note that RF handles class imbalance better than all the other algorithms in this section; it was also the fastest.

4.3.5 Deep Neural Network on balanced dataset

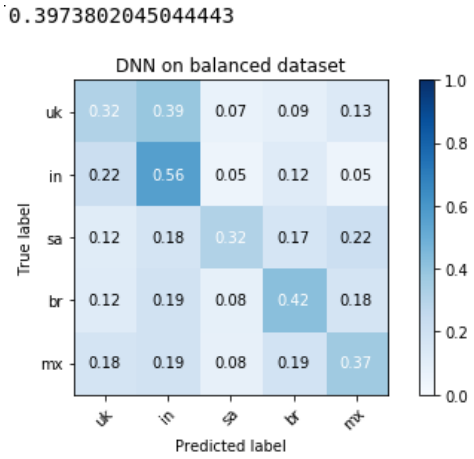


Figure 4.3: DNN on balanced dataset

Accuracy was 0.39, results on Fig. 4.3.

4.3.6 Random forest on balanced dataset

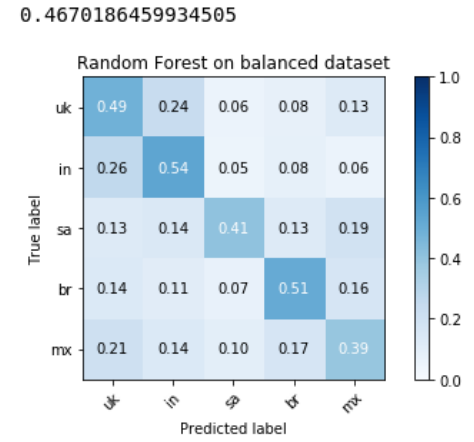


Figure 4.4: Confusion matrix of Random forest on balanced dataset

Accuracy: 46.2%

```
rfcl=RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                    max_depth=None, max_features='auto', max_leaf_nodes=None,
                    min_impurity_decrease=0.0, min_impurity_split=None,
                    min_samples_leaf=10, min_samples_split=5,
                    min_weight_fraction_leaf=0.0, n_estimators=2000,
                    n_jobs=None, oob_score=False, random_state=None,
                    verbose=0, warm_start=False)
```

4.3.7 SGD and SVM on balanced basic dataset

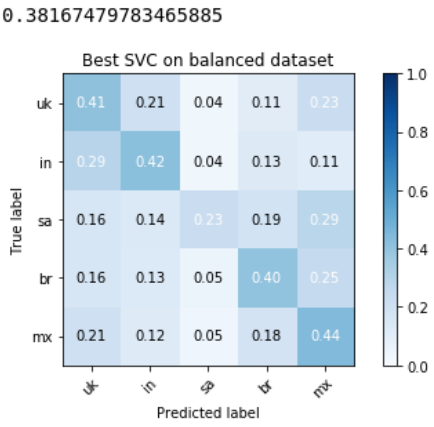


Figure 4.5: Confusion matrix of SVM classifier running on balanced dataset

Optimal SVC had the parameters C=1.0, kernel='linear'. The fact that a low C was chosen by the grid search implies that the dataset is noisy.

The results of SGD seemed to be very unstable and are not included.

4.3.8 Summary of results and effects of unbalanced datasets

Unbalanced datasets give higher (classical) accuracy, but are not helpful for our purposes. From the confusion matrices it's clear that class imbalance is a problem for most of the tried classificators, RF and DNN seemed to be less affected by the class imbalance. The SGD and SVM were used without additional settings such as weights, which might have improved their results.

Balanced accuracy was not used here, and accuracy alone should not be seen as a valid metric, but the confusion matrices make it clear that unbalanced

datasets are an issue. I decided to train only on the balanced dataset, to remove at least this source of complexity from an already challenging classification, and since a test of the possible ways to handle class imbalances was not part of the goals of this thesis.

4.4 Classification with NLP features

After the results of the previous chapter, I decided to work only on the balanced datasets. A comparison was done between the classification on masked text (replacing usernames, mentions and URLs by REPLUSER, REPLMENT, REPLURI), original text, clean text (with all mentions, URIs, hashtags completely removed) etc, all tokenized. At the end, 1-4 n-grams on masked text tokens were used. No scaling was done since all n-grams were represented using TF-IDF, which already returns scaled data.

4.4.1 POS n-grams

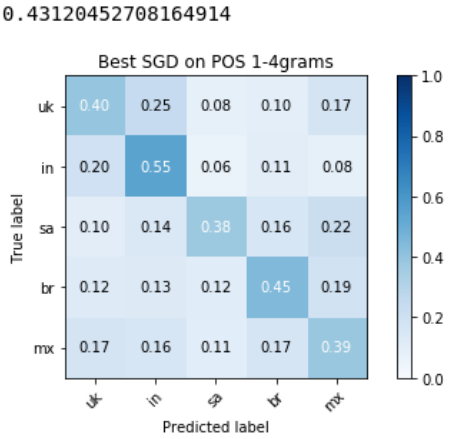


Figure 4.6: SGD with log loss on POS n-grams

On this dataset, the best performing classifier was a SGDClassifier with log loss (a logistic regression model), penalty l2, and alpha of 0.0001, with an **accuracy of 43%**. Same classifier with hinge loss (an implementation of an SVM) performed 2% worse. This is interesting because hinge loss performed dramatically better in the case of of token n-grams.

4.4.2 Token/word n-grams

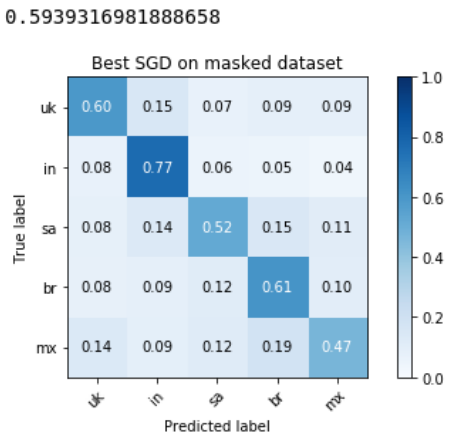


Figure 4.7: SGD with hinge loss on token n-grams

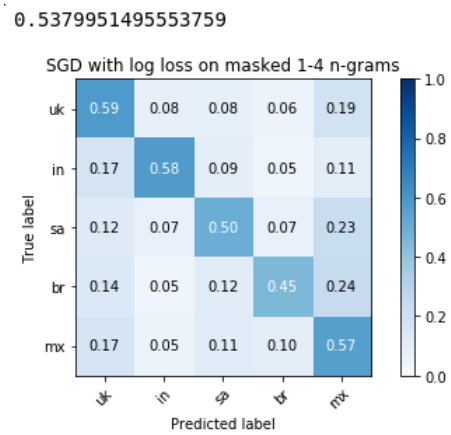


Figure 4.8: SGD with log loss on token n-grams

The best performing classifier was a SGDClassifier with loss='hinge', penalty='l2' and alpha=0.0001. Log loss performed much worse, but was retained because it seemed to improve the results of the ensemble. Also decreasing the alpha seemed to improve the results of the classifier with log loss, but not hinge loss. At the end, a value of 0.0001 was used for both.

Interesting in these classifications is that there's much less misclassification between linguistically similar classes than in the sections above, and that

India is the easiest class to classify correctly. Possibly, because Hindi English has more local words in it than the English spoken in other countries, or that the election meant that a lot of names of cities and politicians were used. The fact that here the misclassifications in general seem to be relatively randomly distributed (in the hinge SGD classifier) may be an indicator of one of the limitations of the dataset: the tweets that can't be well described neither by POS n-grams, nor basic features, nor token n-grams might be an artifact of the way the dataset was collected, and are the people in vacation, the people being in the country for work, quotes, automatically generated tweets etc.

4.5 Ensemble learning

To improve on the maximum accuracy of each of the classificators, ensemble learning was attempted. The fact that most of the classifiers returned not just the prediction, but also the probability estimates for each of the classes, allowed to create a meta-classificator. For each, both DNN, SGD, SVM and RF were tested.

4.5.1 Dataset used for ensemble learning

The class probabilities returned by each classificator were used as features, for each of the five languages. In the case of SGD with hinge loss, that returned only the prediction but not the probabilities, a simple one-hot encoding was used, that doubled as a prediction of 100% probability for the predicted class and 0% for the other ones.

4.5.2 Ensemble using predictions on basic features and POS n-grams

Most papers for the purpose of such meta-classificators use SVMs, but I found a DNN with the following settings to be the most accurate:

```
tf.keras.layers.Input(probs.shape[1], 32),
tf.keras.layers.Dense(100, activation='relu'),
tf.keras.layers.Dropout(0.2),
tf.keras.layers.Dense(500, activation='relu'),
```

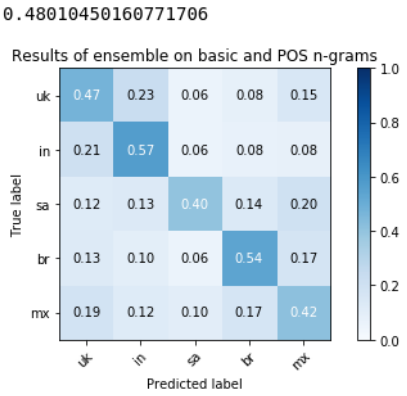


Figure 4.9: DNN Ensemble on basic+POS

```
tf.keras.layers.Dropout(0.2),
tf.keras.layers.Dense(50, activation='relu'),
tf.keras.layers.Dropout(0.2),
tf.keras.layers.Dense(5, activation='softmax')
```

As input the probabilities returned by the RF classifier (46.7% accuracy) on basic features and the SGD with log loss on the POS n-grams (42% accuracy) were used.

This improved by 1.3% the results of best performing RF classifier, increasing the accuracy from 46.7% to **48%**.

4.5.3 Ensemble using predictions on all available features

Input classifiers used were:

- RF on basic features - 46.7%
- SGD with log loss on POS n-grams: 42% accuracy
- SGD with log loss on 1-4 token n-grams: 53.7%
- SGD with hinge loss on 1-4 token n-grams: 59.3%

For example, one row would use all the following probabilities:

classifier	UK	IN	SA	BR	MX
RF:	0.13850509	0.12204897	0.21259514	0.25949008	0.26736071
SGD on POS:	0.168	0.118	0.175	0.254	0.285

classifier	UK	IN	SA	BR	MX
SGD log:	0.16851449	0.19365019	0.17553513	0.25271791	0.20958229
SGD hinge:	0	0	0	0	1
Real class:	0	0	0	0	1

Both SGD with log-loss and SGD with hinge loss on the same features increased the accuracy much more than each of them individually.

I believe that this improvement can be explained by the fact that the RF **used the probabilities as additional information about what the hinge classifier was not sure about.** (A SGDClasifier with hinge loss returns only the prediction, but not the probability estimates for the other classes, while one with log loss can return such probabilities, but performs much worse.)

Additionally, I expected a neural network to be able to use complex relationships between classifiers better than anything else, but this was not the case.

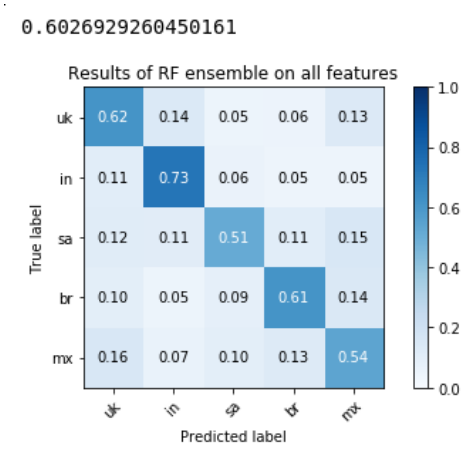


Figure 4.10: RF on all available features in ensemble

Final settings for the RF, as found via grid search, were:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
    max_depth=None, max_features='auto', max_leaf_nodes=None,
    min_impurity_decrease=0.0, min_impurity_split=None,
    min_samples_leaf=10, min_samples_split=5,
```

```
    min_weight_fraction_leaf=0.0, n_estimators=2000,
    n_jobs=None, oob_score=False, random_state=None,
    verbose=0, warm_start=False)
```

Chapter 5

Summary

5.1 Basic results

In this Bachelor's thesis, I attempted to detect the native language of the authors of tweets written in English. I approached this as a classification problem: first, I collected tweets in English, written from 5 different geographical areas, then replaced mentions, hashtags and URIs by tags. After this, using a combination of features like part of speech, punctuation, POS n-grams and word n-grams trained two classifiers – one Random Forest, and the other a Support Vector Machine. Lastly, I trained a meta-classifier (a deep neural network), that used predictions from both and delivered a final prediction, improving the results by about 1%. Two different feature sets were used:

- only parts of speech, punctuation and articles, which yielded 48% accuracy over 5 categories (with a 20% trivial baseline accuracy)
- parts of speech, punctuation, articles and word tokens, with 60% accuracy over the same categories.

I separated both because of two possible different goals in such a classification. The first one is an attempt to classify based on pure language and grammatical features that are independent from the content and word choice of the tweet. A 46% accuracy with 5 categories means that people with different language backgrounds do use different parts of speech, punctuation, articles, and that some degree of information about an author may be gathered using these features alone. The second goal was to classify

using all available *linguistic* features. Mentions, hashtags and URIs were hidden while keeping a tag in their place, because they might allow a way to "cheat" during the classification (it doesn't take machine learning to guess that someone mentioning @timesofindia is probably located in India). There are a number of social-network-specific features that would have improved these results, but this would not have been pure NLI. Lastly, using actual words improves results by picking other cues. If there's an election in India someone speaking about elections is more likely to be from India, even though this would not have increased the accuracy on a dataset collected after a month. And someone mentioning Riyadh (the capital of Saudi Arabia) probably speaks Arabic, always. The first goal was an attempt to see how far would I get by avoiding all such cues completely.

Random Forest has been consistently one of the best working algorithms, and the fastest. The only exceptions are SVMs for the n-grams, which is not surprising due to how well they handle sparse data.

Native Language Identification is a complex task even for humans, and usually it's done on much bigger texts (at least 100 words per instance, 75% of the tweets used in this thesis contain less that 25 words) and datasets with less noise, so a fair comparison to existing results is not possible.

5.2 Discussion

Comparing these results to existing ones is nontrivial because NLI is rarely attempted on such datasets. The most relevant was "Native Language Identification with User Generated Content" [7], but it used Reddit data of >100 words and concentrated on SN features such as subreddits. To my knowledge, NLI was never attempted on such a noisy dataset with such short texts.

Confusion matrices were very enlightening during most stages of classification. The misclassifications between MX/BR and UK/IN were expected. The former because of the same language family, the latter because of British English being used and taught in India for centuries. This confirms the initial hypothesis that these categories would be misclassified, and confirms that using the country where the tweet was written as a proxy of the author's native language was sensible, but I believe is still the biggest limitation of these results. The hypothesis that POS-tags and basic features would be less accurate than tokens was also confirmed.

The dataset can be improved in many ways, and additional testing with the ideas described in Section 3.1 would be interesting (such as the use of a synthetic dataset to create longer source texts, or using more tweets from individual accounts).

It would be also interesting to focus on the smileys used, both the smileys and their location inside the tweet, along with any UTF-8 symbols used. Both were not analyzed and adequately covered by the used features, but to me they seemed to be quite different between the categories.

More and more different categories would also be interesting. My choices were partly dictated by the amount of Twitter users in the countries and the practicality of rectangular bounding boxes around the countries, since I was not sure about the viability of my method at all; and some of my choice regarding algorithms and parameter optimisation were also limited by computing power and time. Now I believe collecting more tweets for a longer time and a more thorough look at algorithms and their parameters is warranted.

Declaration of authorship

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keiner anderen Prüfungsbehörde vorgelegt.

I, Serhii Hamotskyi, declare that this thesis and the work presented in it are my own. All used sources are marked.

Merseburg, the 19.08.19

References

[1] A. Sircova *et al.*, "Time perspective profiles of cultures," in *Time Perspective Theory; Review, Research and Application: Essays in Honor of Philip G. Zimbardo*, 2015, pp. 169–187.

[2] L. Boroditsky, L. A. Schmidt, and W. Phillips, "Sex, syntax, and semantics," *Language in mind: Advances in the study of language and thought*, pp. 61–79, 2003.

[3] S. Malmasi *et al.*, "A Report on the 2017 Native Language Identification Shared Task," in *Proceedings of the 12th workshop on building educational applications using nlp*, 2017.

[4] "NLI shared task." https://sites.google.com/site/nlisharedtask/home [Accessed: 2019-08-01].

[5] S. Malmasi and M. Dras, "Native language identification using stacked generalization," *arXiv preprint arXiv:1703.06541*, 2017.

[6] D. Blanchard, J. Tetreault, D. Higgins, A. Cahill, and M. Chodorow, "TOEFL11: A corpus of non-native english," *ETS Research Report Series*, vol. 2013, no. 2, pp. i–15, 2013.

[7] G. Goldin, E. Rabinovich, and S. Wintner, "Native language identification with user generated content," in *Proceedings of the 2018 conference on empirical methods in natural language processing*, 2018, pp. 3591–3601.

[8] S. Volkova, S. Ranshous, and L. Phillips, "Predicting foreign language usage from English-only social media posts," in *Proceedings of the 2018 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 2 (short papers)*, 2018, pp. 608–614.

[9] E. Alpaydin, *Introduction to machine learning*. MIT Press, 2014.

[10] A. Prinzie and D. Van den Poel, "Random multiclass classification: Generalizing random forests to random mnl and random nb," in *International conference on database and expert systems applications*, 2007, pp. 349–358.

[11] "1.4. support vector machines — scikit-learn 0.21.3 documentation." https://scikit-learn.org/stable/modules/svm.html [Accessed: 2019-08-01].

[12] "Support vector machines — soft margin formulation and kernel trick." https://towardsdatascience.com/support-vector-machines-soft-margin-formulation-and-kernel-trick-4c9729dc8efe [Accessed: 2019-08-01].

[13] "1.5. stochastic gradient descent — scikit-learn 0.21.3 documentation." https://scikit-learn.org/stable/modules/sgd.html [Accessed: 2019-08-01].

[14] Wikipedia contributors, "Part-of-speech tagging — Wikipedia, the free encyclopedia." 2019 [[Accessed 2019-08-02]].

[15] Gianpaul Rachiele, "Tokenization and parts of speech(POS) tagging in python's nltk library." https://medium.com/@gianpaul.r/tokenization-and-parts-of-speech-pos-tagging-in-pythons-nltk-library-2d30f70af13b [Accessed: 2019-08-01].

[16] Jo Etzel, "MVPA meanderings: Balanced accuracy: What and why?" 2015 [[Accessed 2019-08-08]].

[17] "3.3. model evaluation: Quantifying the quality of predictions — scikit-learn 0.21.3 documentation." https://scikit-learn.org/stable/modules/model_evaluation.html#from-binary-to-multiclass-and-multilabel [Accessed: 2019-08-01].

[18] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information processing & management*, vol. 45, no. 4, pp. 427–437, 2009.

[19] "Confusion matrix — scikit-learn 0.21.3 documentation." https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html [Accessed: 2019-08-01].

[20] "Filtering tweets by location — twitter developers." https://developer.twitter.com/en/docs/tutorials/filtering-tweets-by-location.html [Accessed: 2019-08-01].

[21] "Geo objects — twitter developers." https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/geo-objects [Accessed: 2019-08-01].

[22] "Most-active-twitter-user-data | sysomos." https://sysomos.com/inside-twitter/most-active-twitter-user-data/ [Accessed: 2019-08-01].

[23] O. Varol, E. Ferrara, C. A. Davis, F. Menczer, and A. Flammini, "Online human-bot interactions: Detection, estimation, and characterization," in *Eleventh international aaai conference on web and social media*, 2017.

[24] J. Kastrenakes, "Twitter is going to make third-party apps worse starting in august - the verge." https://www.theverge.com/2018/5/16/17362138/twitter-api-third-party-apps-changes-explained [Accessed: 2019-08-01].

[25] "GitHub - ccantey/geosearch-tweepy: Python code, using the tweepy and mysqldb modules, to stream the twitter api." https://github.com/Ccantey/GeoSearch-Tweepy/ [Accessed: 2019-08-01].

[26] Marco Bonzanini, "Mining twitter data with python (part 2: Text preprocessing) – marco bonzanini." https://marcobonzanini.com/2015/03/09/mining-twitter-data-with-python-part-2/ [Accessed: 2019-08-01].

[27] T. Horsmann, "Robust part-of-speech tagging of social media text," PhD thesis, 2018.

[28] "Native language identification with user generated content." https://www.aclweb.org/anthology/D18-1395 [Accessed: 2019-08-01].

[29] "Sklearn.dummy.DummyClassifier — scikit-learn 0.21.3 documentation." https://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html [Accessed: 2019-08-01].

[30] "The number of hidden layers | heaton research." https://www.heatonresearch.com/2017/06/01/hidden-layers.html [Accessed: 2019-08-01].