Automated Language Recognition Tool

Ankita Singh, Nimisha Srinivasa

UCSB CS 273 Fall 2016

Abstract

Building a Language Recognition tool forms the first step for most of the Natural Language Processing techniques. Language classification for a large text corpus has been a well-researched topic today. However, with a short contents of text that is available in social media articles such as tweets in Twitter, language classification poses it's own challenges. With the growing popularity of social media among people, people tend to express their thoughts in a language that is more suitable to them. Thus, the number of languages to be identified is huge. The research so far has mostly been on a small subset of these languages that require specific techniques to differentiate. We implement a model that is generic enough to recognize this large number of languages in short texts.

1. Introduction

The number of people using social media to connect with others across the world is increasing everyday. They use it as the means to express their feelings at real-time to their friends. The challenge however arises when there is a limit on the number of characters used for communication. Some social media platforms limit the number of character in their posts. This forces people to use acronyms and abbreviations. Thus, it becomes even more challenging to extract words from the tweet and recognize the language of that tweet.

Analysis of short texts is gaining importance in Social Media. Understanding the content of a post is considered important as it helps to grasp the interests of the user. This could help the Social Media platforms in future for better content-targeted advertising as well as fighting spam. Language identification forms the first step towards understanding the content.

Much research has been done for classification for languages based on huge text-corpus. The techniques applicable in such a scenario does not correspond well with identification of languages in short texts.

2. Dataset

For our project, we decided to work on Twitter data to recognize the languages in tweets. Twitter has one such dataset that has a pre-labeled data for tweets and their corresponding languages [1]. This can be accessed using the Twitter Developer API. The tweets are available in a json format while the mapping of the languages with the tweets is stored in a tab-separated-value (tsv) format. The data contains tweets from 70 different languages. We used about 70,000+ tweets for our classification. However, the fetched data calls for a some preprocessing before using it for our language classification.

2.1. Data Preprocessing

We removed the noise in the dataset by removing the emoticons which did not add much value towards help identify the language of the tweet. Secondly, we removed the digits and punctuation from the data, since they seem standard for different languages. Next, we removed the @-handles and hashtags since most of them tend to be in English. Lastly, we also removed all the URLs mentioned in a tweet as well. Also, the dataset was skewed with the tweets not having uniform samples per language. Some had languages less than 3 tweets which we removed since the amount there wasn't enough data/features to train a model for these languages. Thus, we finally ended up with 58 languages having at least 3 tweets or more for this project.

3. Features

3.1. Top-k words

The most intuitive approach to the problem would be to build a dictionary of the words for each language. The presence or absence of the word in the tweet can help classify it into a language. For resolving conflicts, we tried to use the frequency of the words as weights to help classify it into a language. This method works well with most of the languages which have a space between them to differentiate between words. However, there exists languages that follow Scriptio continua, a style of writing without spaces, or other marks between the words or sentences. Chinese and Javanese follow this style and pose a problem when this approach was used.

3.2. N-gram features

Thus, we concluded that using characters instead of words would be a better feature for including languages without a space in between them into consideration. Since character n-grams had shown good results in the past [2] when used on large text corpus, we decided to incorporate this as a feature and explore the results. The optimum value of n for the classification in discussed in results section.

4. Methodology

In this section, we discuss the algorithms we used for the classification task.

4.1. Baseline

We decided to explore the language classification task with the most intuitive approach to a person. We used the most frequent word for each language to help classify it into a language. We found that some languages had a common most frequent word which increased the possibility of the misclassification of a text. Next, we implemented a top-k most frequent words features for the classification. This did give us better results compared to the top-1 most frequent word. The comparisons in accuracy will be discussed in the discussion section. However, as mentioned in Section 3, using words posed problems for languages without a space between the words. Additionally, if all words in a language were used to build a dictionary, it would not be a scale-able approach since the vocabulary size would increase drastically for every new language included in the class labels. Hence, for all further algorithms, we have used character n-grams as features.

4.2. Naive Bayes Classification

Naive Bayes Classification (NBC) is a simple probabilistic model that works on the assumption that the different features used for classification are independent of one another. It works on the Bayes theorem that states that,

$$P(A \mid B) = \frac{P(B|A)P(A)}{P(B)}$$

where, P(A) and P(B) are the probabilities of observing A and B without regard to each other.

$P(A|B)$, a conditional probability, is the probability of observing event A given that B is true.
$P(B|A)$ is the probability of observing event B given that A is true.

NBC algorithm has been one of the most popular algorithm used for text categorization. Traditionally, word features were used for the classification. But with our problem mentioned in section 3.1, we used the NBC algorithm with character n-grams as features. With multiple classes in our task, we used the Multinomial Naive Bayes classification (MNBC) algorithm. The formula for MNBC is given by:

$$P(L_k|x_1, x_2, ..., x_n) = P(L_k) \prod_{i=1}^{n} P(x_i|l_k)$$

Where L_k is the language under consideration, $P(L_k)$ is the prior-probability of the text belonging to L_k among the text corpus. $x_1, x_2, ...x_n$ are the n features used for the classification.

Thus, a probabilistic model of each language L_k is created during the training phase. For testing, the language with the highest probability is the predicted language of the tweet.

4.3. Logistic Regression

Logistic Regression (LR) model is a regression model where the dependent variable is categorical. Hence, it is often used for classification tasks. This model also assumes independence between the different features used for the task. It is a generative model, that uses Bernoulli's distribution for computing the conditional probabilities:

$$P(Y = k|x_{1k}x_{2k}...x_{nk}) = \frac{e^{\beta_k \cdot x_k}}{\sum_{i=1}^{m} e^{\beta_i \cdot x_k}}$$

We calculate the weights of each feature during the training and the predicted class of the data is the class with the highest likelihood.

5. Results

All our results are summarized below:

5.1. Baseline

Our baseline system selected the most frequent words from each language. If any of the most frequent words were found in the input tweet, it would be classified into the corresponding language. We got an accuracy of 31.85%

by using only the one most frequent word for the languages. However when we increased the count of most frequent words to 10 then our accuracy was increased upto 42.5%. We tried to increase the count of most frequent words further upto 100, however the results were stabilized and their was not much performance gain.Figure 1 summarizes the results for our baseline.

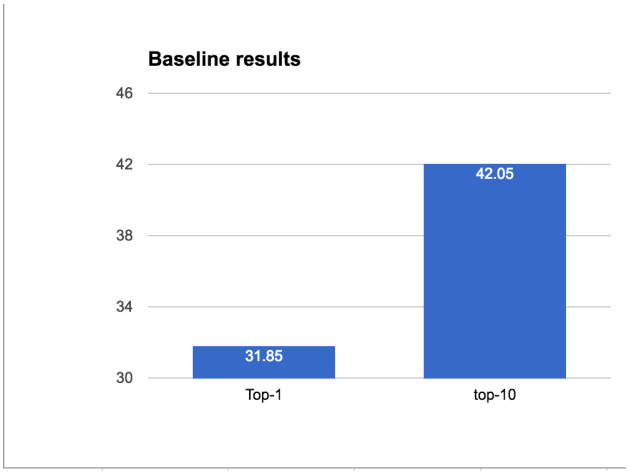

Figure 1: Baseline

5.2. Naives Bayes

We used different n-gram lengths to analyze the performance of Naive Bayes classifier. We got the best results for 3-4 n grams. Bigrams had a lower accuracy (58%) because the probability of a bigram of a language overlapping with the bigram of another language is high. We got the best results when the length of each gram was between 3 to 4 (61%). As the length of grams was increased further to 5 and 6, the accuracy dropped(47%). This can be explained by stating that as the length of a gram increases, the total number of features decreases. Also since twitter has a cap of 140 on the total number of characters, people usually tend to use acronyms and the number of words with length greater than 5 is usually low.Figure 2 summarizes the results for Naive Bayes using different n-gram lengths.

5.3. Logistic Regression

Logistic regression gave us the best results (66%) as compared to Naive Bayes and the baseline.As in the case of Naive Bayes, we got the best results

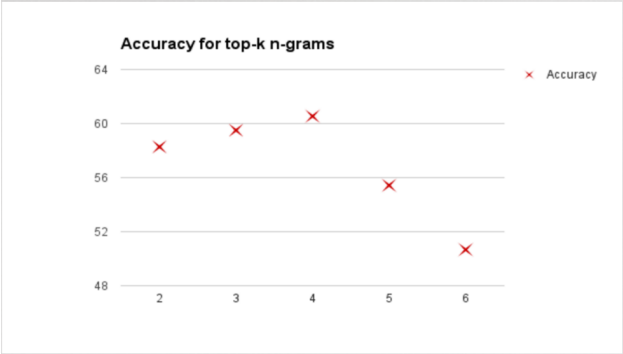

Figure 2: Naive Bayes

when the length of n-grams was between 3 to 4. Figure 3 summarizes the performance of Logistic Regression for different lengths of a gram.

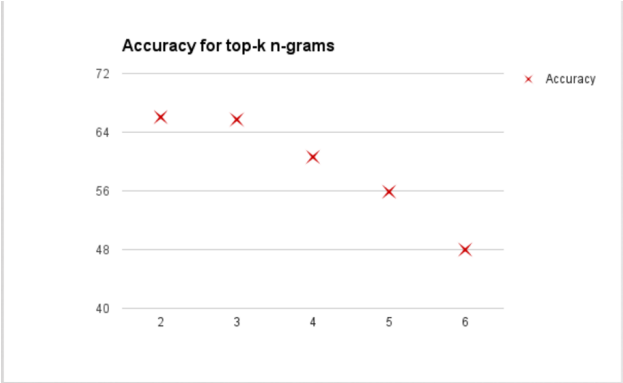

Figure 3: Logistic Regression

5.4. Comparison of different models

Baseline performed the worst as compared to Naive Bayes and Logistic Regression. As both Naive Bayes and Logistic Regression are linear classifiers and both use the same set of features,the performance is very similar. Figure 4 compares the results of all three classifiers.

6. Conclusion & Future Work

We built a classification system capable of classifying 58 languages with a performance better than the simple dictionary approach. The best accu-

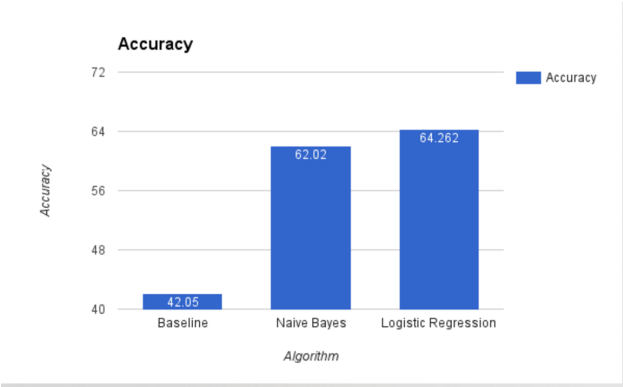

Figure 4: Logistic Regression

racy we achieved was 64.26% with the Logistic Regression model using 3-4 n-grams as features. This, does not however include all the exhaustive set of languages supported by twitter. In future, we plan to simulate a dataset in future with less than 140 characters for those languages not considered in this project. Also, the dataset was skewed with the ratio of English:German samples being 10:3. We feel that our system would have performed better if we had access to uniformly sampled data with ample samples for each language. A comparatively lower accuracy can be attributed to the issue of mis-classification amongst very similar languages. Having a hierarchical classifier with a general model at the root and specific classification models for handling similar languages could give us better results. The success of Recurrent Neural Network (RNN) recently for text-related machine learning calls for using RNN on our task as well. We cannot say much about the accuracy of the model since our texts are very small. However, the comparison between our existing approaches with RNN might be insightful.

References

[1] Twitter dataset *https://blog.twitter.com/2015/evaluating-language-identification-performance.*

[2] Cavnar, William B. and Trenkle, John M. *N-Gram-Based Text Categorization*