May 29, 2014

Universitat Politècnica de Catalunya

ANLP
Automatic Language Classification

Authors:
Iosu Mendizabal Borda,
Daniel Horowitz,
Jeroni Carandell Saladich

Abstract

In this paper, we will analyze and experiment with different methods for Tweet language classification. We will first start with a state-of-the-art analysis of short text classifiers. Then we will explain how we created the data set of Tweets followed by an explanation of the methods we used which are basically using Lindstone Smoothing and ranking methods. We then apply different experiments to optimize our models and to compare them.

1 Introducing the Problem

Language identifications of tweets is vital as a preliminary step of any natural language processing applications on Twitter(translation, sentiment analysis, etc.). This is however not a trivial task. Twitter limits users' number of characters to 140. This has a huge impact on how Twitter users write and synthesize text, often not worrying about spelling mistakes, using plenty of emoticons and sharing URLs. This means that sometimes with only one or two misspelled words, a good classifier should be able to infer the language of the tweet.

Another challenge that we have faced is multilingual texts, like Spanish and Catalan but especially English in all languages.

2 State of Art

Various previous approaches have been made for automatic language identification. One of the most basic yet most effective are using N-gram Based Cumulative Frequency Addition. [1]

One paper, works with an identification of a language based on a test set with text that contain from 5 to 21 characters. Bayes classifier based on character n-gram models. We will try this method on twitter.

The ranking method by Cavnar and Trenkle (1994) consists of generating a list of the most common n-grams. These are then compared by computing the distance by its "out-of-place" measure. This measure determines how far an n-gram from the text profile has been displaced with respect to the language profile. [2]
Language Identification of Short Text Segments with N-gram Models [3] shows a very large description of different methods two of which we have based this work.

3 Our Approach

In this section we are presenting our language classifier approach. Section 3.1 is going to present the creation of the database, in 3.2 we are going to explain the classification algorithms we will apply: Lidstone Smoothing and Ranking models.

3.1 Creation of the Database

Instead of using regular databases, we've decided to use an online framework to host our database.The framework chosen was Parse.com. By using this framework, we are able to easily access and perform operations in our online database, using only a couple of code lines. In order to obtain the

items of our database of languages, we use the twitter API for python, to search for tweets using the tweet's language as a search parameter. For performing the search, we've used 4 different languages: English, Spanish, French and Portuguese. Then, once we've retrieved the results, we've refined them using different techniques.The techniques we've used will be described in the following sections. The process for creating the database can be divided into 3 steps.

1. Creating the database: To create the database in Parse.com, we needed to create an account and an application at Parse.com, as so to retrieve our credentials uswd in authentication. After the app and was created, we've defined the tables, using regular attributes as rows and columns. As you can see in Figure **??**.

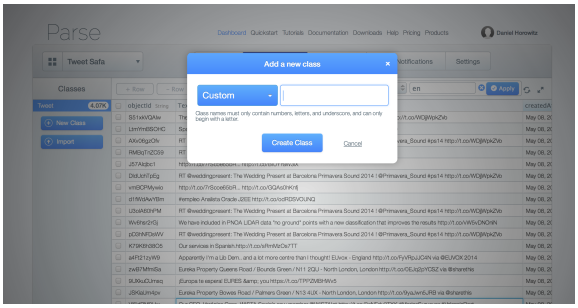

Figure 1: Example of out-of-place measure

After setting the classes, we have integrated the application with the parse.com API wrapper for python, using only a couple of code lines to do so.

2. Twitter API integration: In order to start working with the Twitter API, we had to create a twitter application and generate our credentials. Once this step was over, we've integrated our code with the Twithon API, a wrapper of the twitter API for python. The twitter API enables performing searches using language as the search parameter. However, this feature is still in testing, therefore, it was prone to errors. In order to generate more reliable results, we've used the google translate API to label the tweets by language.The image below illustrates the database after performing the search and inserting the results.

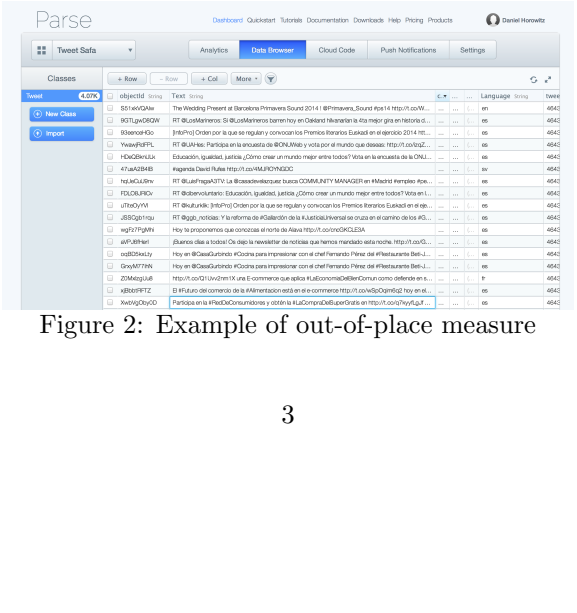

Figure 2: Example of out-of-place measure

3. Refining the results: Once we had inserted our search results into the database, we used some techniques so as to avoid noise in the data and refine our results. Initially, we have removed all special characters and numbers from our tweet's. Then, we removed double spaces,duplicated tweets and also the tag 'RT', which stands for re-tweet, this tag was removed because it had the same meaning in all languages and it was generating noise in the data.In the same matter, we've removed the URL's from the text. Finally, we've converted all of our tweets to lower case letters only.

3.2 Classification Algorithms with N-gram models

We decided to compare two different approaches to classifying languages. The criteria we consider is speed and, of course, accuracy.

This method used n-gram models to define a probability of a sentence being generated by a language. An n-gram is a contiguous sequence of n items from a given sequence of text. For each language L_i, a distribution D_i of n-gram frequency will be learned. Experimentally, it becomes obvious that each language has its own distribution of n-grams.

3.2.1 Lidstone Smoothing

In order to perform a smoothing for our algorithm we choose the Lidstone smoothing for categorical data. With this method we can compute the probability of the tweet taking into account the n-grams that we have in our database and making a smoothing of the ones that are not in the database, considering all the possible combinations.
The smoothing formula is the next one:

$$P_{LID}(X) = \frac{count(X) + \lambda}{N + B\lambda} \tag{1}$$

Where X is the n-gram of the text we want to identify, λ is the Lidstone smoothing parameter ($\lambda = 0$ corresponds to no smoothing), N is the count of all the n-grams of the model (L_i) and finally B is referring to all the n possible combination of characters.

To classify the tweets with the highest probability we used the distributions, D_i, of n-gram frequency that we learned for each language, L_i, and we look for the probability of the target tweet within these distributions. The probabilities of the tweets are computed using both, bi-grams and tri-grams, and after computing the probability for the two of them we finally get the language with the highest probability.

3.2.2 Ranking Model

As an alternative to compare with the previous method, we chose to use an N-gram-ranking-based model for the classification of languages.

Standard Ranking Models

Given a set of Tweeter concatenated texts $(T_1, ..., T_k)$ in languages $(L_1, ..., L_k)$ respectively, we extracted their N-gram distributions and got a set of ranks $(R_1, ..., R_k)$ of the m most frequent N-grams in each Language. To find the language of a new external Tweet T_T, we got a ranking R_T of its n most frequent N-grams. All these ranking are ordered from most to least frequent. The next step is to compare the ranking of tweet with the ranking of all the languages. To do so, we will use the *out-of-place measure*. This measure is a distance based on the distance of indexes between equal N-grams. The idea is that the ranking in the Tweet will probably be similar to the ranking of the language it belongs. We measured this "similarity" by computing the sum of distances of all the N-grams in the rank R_T with the ones in the ranks R_i $\forall i$ in $\{1, ..., k\}$. The lower the sum the more similarity. Supposing an N-gram is in the p^{th} place of rank R_T and in the q_i^{th} place in rank R_i, we define the distance as $\mid p^{th} - q_i^{th} \mid$. If the N-gram were not to appear in rank R_i, we would simply return the maximum possible distance m. In Figure 4 shows an example of how the *out-of-place measure* is computed. Figure 3 shows the flow of the algorithm.

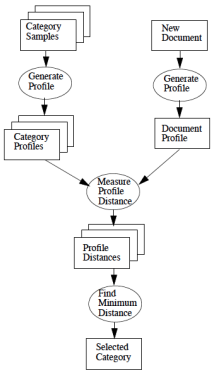

Figure 3: Flow of ranking method.

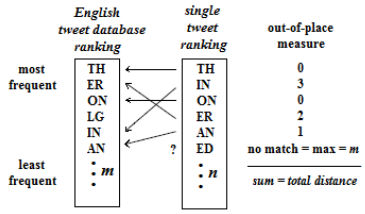

Figure 4: Example of Out-of-place measure.

After a few experiments, we discovered that just as we had expected, both m and n values have a huge impact on the results, as it is shown in Experiment.

Augmenting Ranking models with voting

There is enough reason to believe that each type of N-gram adds a certain characteristic to a language. Obviously, the lower the N, the less specific the description of the language. This led us to believe that by changing the N-gram used for the ranking would influence the results greatly. We observed that bi-grams in general were the best to classify languages in general. However, results were not good enough(more than 50% error). An alternative had to be found.
We decided to make different evaluations for uni-grams, bi-grams,tri-grams, 4-grams and even 5-grams and return the most voted evaluation, with bi-grams as default in case no repetition was found.

4 Experiments

The following experiments have been done with leave-one-out cross validation. This is because of the small size of the data set that we had in our disposal.

4.1 Experiment 1: Lidstone

In the first experiment, we try to detect the language of new tweets using n-gram models, trained with the database we created from real tweets. But as we can have an n-gram, in the tweet we want to detect the language, that it is not appearing in the n-gram model trained before, we are going to loose all probability for the given sentence.

To avoid with this problem we introduce a smoothing algorithm in section 3.2.1, Lidstone. As we know the smoothing value of this method is the λ number, if λ is equal 0 we are not going to have any smoothing and in the other hand while we increase this value the less stable will be the algorithm.

In the table of below we can see the experiment we made to get the classification error. We can notice that while the smaller is λ, the higher will be the accuracy of the language detector.

	λ				
	0.1	0.3	0.5	0.7	1
Error	0.40625	0.490625	0.528125	0.56875	0.60625

Table 1: Results of Different configurations of λ

4.2 Experiment 2: Ranking Method

To find the most suitable m, and n values, we searched for the lowest error in a mesh.

		m			
		20	50	80	110
	50	0.33125	0.280898876404	0.274774774775	0.345945945946
n	80	0.331168831169	0.209302325581	0.224299065421	0.134831460674
	110	0.32	0.322580645161	0.346153846154	0.325581395349
	150	0.416666666667	0.3	0.32	0.333333333333

Table 2: Results of Different configurations of n and m values

As we previously said before, after optimizing the m and n values, we took a look at how we could lower the error of the method. As we previously said, if we did a voting between consecutive

N-grams, the error got lower. As we predicted, we got a better accuracy than without voting. In the following table we can see the different voting error of consecutive N-grams.

	uni-grams bi-grams tri-gram vote	bi-grams tri-grams 4-grams vote	tri-grams 4-grams 5-grams vote	4-grams 5-grams 6-grams vote
Error	0.646875	0.15625	0.121875	0.171875

It turned out that using this method with the combination of tri-grams, 4-grams, and 5-grams greatly outperformed the others

4.3 Experiment 3: Lidstone vs ranking

As we previously said, one of the main characteristics we consider is computational cost, because of that we have computed the mean time interval of the two algorithms we use for tweet classification. We chose the optimal parameters for both. The Lidstone method takes an average of 5.03123188019 seconds while the Rank-based voting method takes only 1.43000006676 seconds.
This means that not only is the Lidstone method for language classification worse in terms of accuracy than the second method, but it is also more expensive in computational cost.

4.4 Most frequent bi-gram and tri-grams in twitter

As we classified using n-gram models we have calculated all the bi-gram and tri-gram frequencies for the database of tweets we created. In the next images we are showing the 10 most frequent bi-gram and tri-grams of the tweets of our database. It seems like the less we consider non-appeared N-grams, the better the classification.

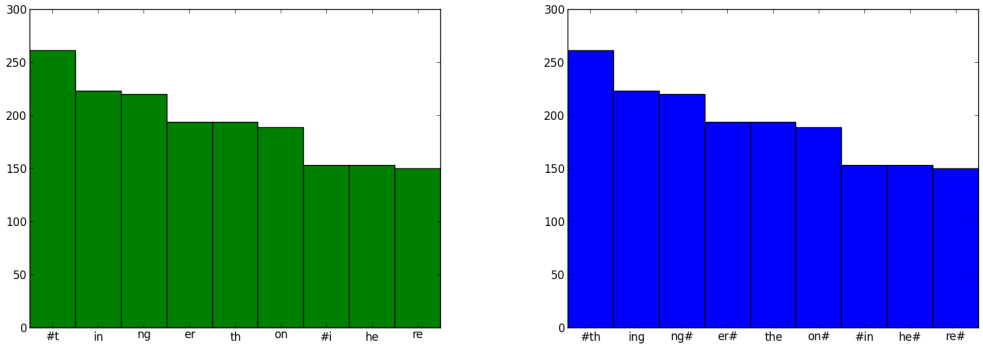

Figure 5: English 10 most frequent bi-gram and tri-grams.

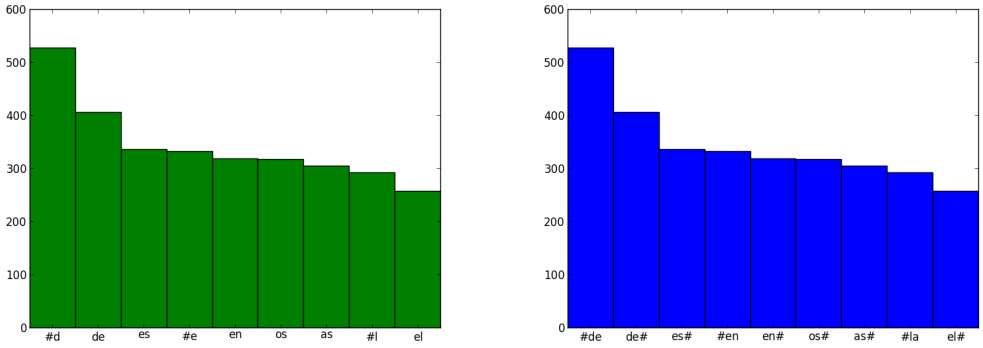

Figure 6: Spanish 10 most frequent bi-gram and tri-grams.

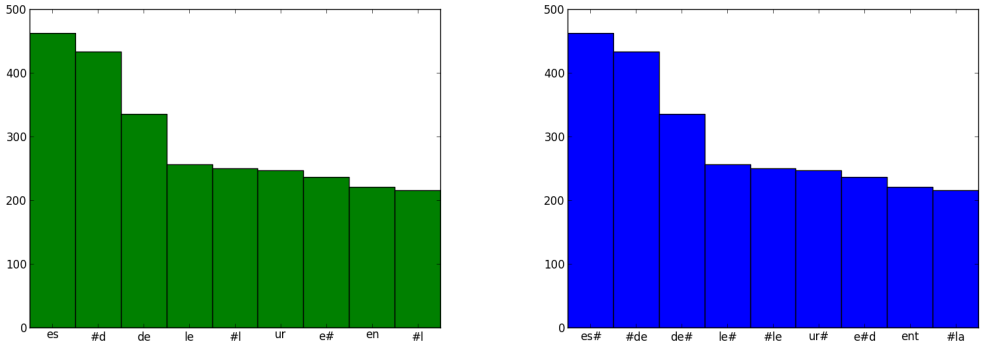

Figure 7: French 10 most frequent bi-gram and tri-grams.

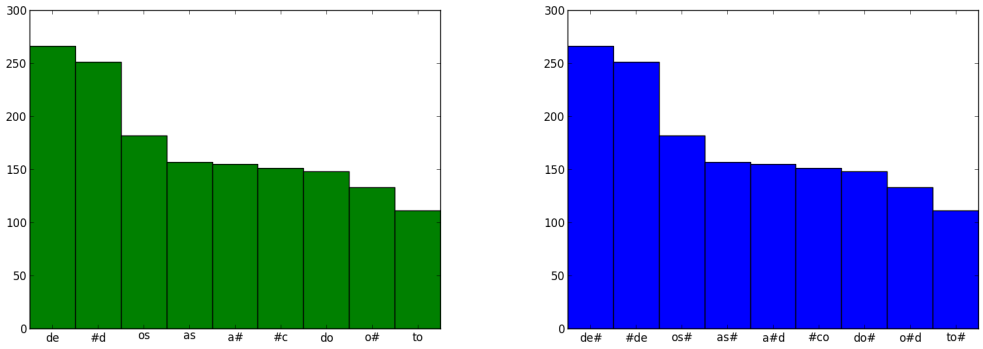

Figure 8: Portuguese 10 most frequent bi-gram and tri-grams.

5 Conclusions & Future work

In this project, we've presented a couple of models for automatic language classification. Our solution was based on using a database of tweets with four different languages. We've presented two different techniques, in order to perform the language classification: Additive smoothing and Ranking models. Also both of the techniques used cross validation and voting as so to generate more accurate results.

We believe that one of the factors that did not allow us to get a higher accuracy was the preprocessing of the data. This is not only seen in repeated characters(e.g. "Hiiiii"), but also in special ones(文章内容). Accents, symbols, Special characters could be converted from utf-8 to Unicode, by doing so, we could take into account different aspects of the languages and generate more accurate n-gram models.

Instead of Lidstone Smoothing we could use other techniques as Linear Discounting or Absolute Discounting. The first one, Linear Discounting method, the non-zero MLE (Maximum Likelihood Estimation) frequencies are scaled by a constant slightly less than one and remaining probability mass is again distributed across novel events.

As for the ranking methods, the voting based approached proved to be much better. This led us to believe that this approach could be refined. We thought that a possible solution to enhance this method would be to do a voting with weight in a way that each N-gram would have a weight that would be learned. This would definitely improve the "unfairness" of voting strategies.

Finally, the project could be expanded to perform automatic language classification in other languages. Of course the question remains of whether with more languages, our algorithms would be robust enough. We believe that, by performing the changes stated in this section, the algorithm would generate good results to a wider set of languages.

References

[1] Bashir Ahmed and Sung hyuk Cha. Language identification from text using n-gram based cumulative frequency addition, 2004.

[2] W.B. Cavnar and J.M. Trenkle. N-gram-based text categorization. *Ann Arbor MI*, 48113(2):161–175, 1994.

[3] Tommi Vatanen, Jaakko J. Väyrynen, and Sami Virpioja. Language identification of short text segments with n-gram models. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may 2010. European Language Resources Association (ELRA).