Automated detection of code switching in multi-lingual twitter communities

Mihael Simonič
Cognitive Science (M.Sc.)
Matriculation no.: 3779260

Nikolas Zeitler
Computer Science (M.Sc.)
Matriculation no.: 3782257

{mihael.simonic|nikolas.zeitler}@student.uni-tuebingen.de

Abstract

In this paper we consider the problem of labeling the languages of words in mixed-language tweets using English-Spanish data from EMNLP 2014 shared task. The problem is approached in an unsupervised fashion, ...

obtianing an accuracy of ...

1 Introduction

Code switching (CS) is "the use of two or more linguistic varieties in the same conversation or interaction" (Scotton and Ury, 1977). It is a linguistic phenomenon often present in bilingual societies.

Through new technologies and the fall away of border is it easier then ever to communicate with people from different countries. And of course we use english more and more on a daily basis. In the german language are words like sexy, laptop, fast food, wellness and many more widely spread[CITE FEHLT]. Furthermore could advertising not live without english anymore (e.G. Douglas - Your partner in beauty). The german language goes even a step further and invents its own english words (e.G Handy is german for mobile-phone).
It is therefore interesting to look at code-switching between english and other languages. In this paper we will look at tweets containing english and spanish words. Our goal is it to find out the language for each word.

In this paper we perform language identification using machine learning techniques. Those techniques are able to indeficate the language of a single word without looking at the sentence. After the machine learning algorithm has indetificated the languages of each word, we try to improve the result using statistical facts of code-switching. We have with example observed that code-switching occurs very rarely in between two other words of the same language.
VERGLEICH VERSCHIEDENER ANSÄTZE? LINEARE REGRESSION / NEURONALE NETZE
The data we are using are provided by *First Workshop on Computational Approaches to Code Switching*(wor,). We have 11.400 Spanish-English tweets. Some of them are deleted or invalid thus we have XXX tweets and XXX different words. We know the language of each word in every tweet.
To evaluate our methods the workshop(wor,) provided us with addional test- and trial-tweets. SECTION X DOES LALALALA SECTION Y DOES LALALALA

2 Data

The data provided by (wor,) contains english and spanish tweets. For each tweet are provided with how to split the words and which language this word is. Tweets contain a lot of special characters. Thus the data not only has labels for english *lang1* and spanish *lang2*. But also *other* for emoticons, nicknames or gibberish. With example ":)", "@Ody12", or "Zaaaas" are labeled as other. The lable *mixed* describes if a word contains both, spanish and english words, "ClutterDesordenLook" is such a word. The next label is *ne*, ne is a name entity. Those are proper names refering to people, places, organizations, locations or titles. The difficulty for those is that they could span above multiple words "West Coast" for example.
The data provides information of how to split the tweet into words. This is important in cases where punctuation marks or emoticons are directly

at a word without whitespace in between with example "HEY!:)".
Lets say the data gives us the tweet: "@snapchateame No! Jason you look bueno!:)". The data provides us furthermore with the information of how to split the tweet into words. This gives us something like: "snapchateame, No, Jason, you, look, bueno, :)". And finaly the data tells us which label every word gets "mixed, ambiguous, ne, lang1, lang1, lang1, lang2, other".

3 Approaches

This chapter explains our aproach in solving the task. First of all it gives a short introduction of how we are going to solve it. Then it explains the two approaches we performed to solve this problem.
We want to find the language for every word inside a tweet using mashine learning algorithms. We try two different approaches. The first is the Linear Regression. Our approach ignores the sentence in its entirety and looks only at distinct words. Using character-level uni- and bigrams as features will the be used to train the Linear Regression.
The second approach is the using the word bigram frequency count proposed in the paper Detecting Code-Switches using Word Bigram Frequency Count(Nathaniel Oco, 2014). This approach looks at the whole sentence not only at a single word.

3.1 Linear Regression

The Linear Regressions goal is it to look at a single word and decide which language it is. It works on a character-level N-Gram basis. The unigrams are every character used in the tweets, and the bigrams is a combination of every character with every character. Next we looked up every distinct word in the provided tweets. Aftwerwards we wrote down a table which showed for each word which uni- and bigrams it contains. Additional we provided the table with the language from which the word is. This way we have a table with one column containing each word. Then one column for each possible bigram and if each word contains this bigram. Lastly a column with the label from which language the word is from. We havent used trigrams since the paper (Kind and Abney,) already showed that trigrams wont improve the machinelearning algorithm. The next improvement is reached when using nGrams in the length of the words. But since the trigram table would have had required around 20Gbyte we did not perform this step either. The nGrams in length of words would have been resulting in a hardware problem.
This table is then useful in combination with the Python libraries Pandas and Sklearn. We used pandas to read the table and performed the Linear Regression using sklearn.

3.2 Word Bigram Frequency Count

4 Evaluation

In this section we are going to talk about how the different approaches perform solving different tasks.

4.1 Test Setup

We want to know the accuracy of both approaches. The usual formula to compute the accuracy would be:

$$Accuracy = \frac{TruePos + TrueNeg}{TotalWords}$$

But we have no "TrueNegative" values.
Our problem does not resolve in "yes" and "no" answers because of the multiple labels "lang1","lang2","other","ne", and "mixed". Thus we compute the accuracy by computing the relative amount of correct labelled words.

$$Accuracy = \frac{TruePos}{TotalWords}$$

Our test-base consists in 5000 tweets. For every tweet we do know the correct labelling.

4.2 Length Of Sentences

The first evaluation looks at how good the accuracy is depending on the length of the sentence. Thus we label sentences of tweets and count the number of words which got correct labelled
Table 4.2 shows how many tweets got checked for each sentence length. The number of tweets is not constant, since we picked the tweets which fitted to our desired sentence length out of our testbase. Figure 4.2 shows Logistic Regession's results. The first intuitive result would be that the sentence length does not change the results. As our Logistic Regression performs on a character level. Thus ignores the sentence as a unit and looks only at single words. But the accuracy graph shows a difference. The accuracy improves with a

Sentence Length	Number of Tweets
3	145
4	211
5	265
6	304
7	363
8	314
9	275
10	261
11	210
12	230
13	181
14	168
15	212

Table 1: Tweets used to evaluate the result

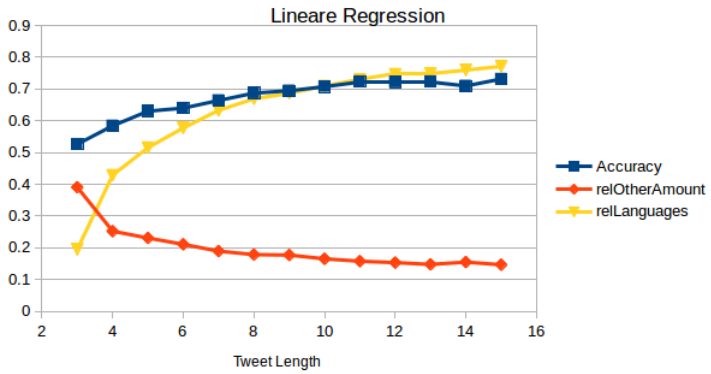

Figure 1: Dependency of accuracy to sentence length together with relative amount of the label ”other” and relative amount of the label ”lang1” or ”lang2”

growing sentence length.
But with the increasing accuracy and sentence length do two factors also change. The relative amount of ”language” tags increases and the relative amount of ”other” tags decreases.
Therefore it looks like that the Logistic Regression has problems in labelling ”other”, ”ne”, ”mixed” and ”ambiguous” words. Since when those words have the highest relative amount at the sentence length of three, are the results the worst. The best results occure when most of the tweet are actual words. This point is at an tweet length of 16words. Here we have 80% spanish or english words and only 20% different words.
The relative amount of ”other” changes rather quickly since usually is the ”@UsersNickname” word (To talk to someone in twitter it is necessary to write @UsersNickname) labelled as ”other”. And usually occurs such a word only once per tweet. Also long tweets tend often to tell an actual story and not to show of emotions as short tweets tend to do. With example the three word tweet: ”@Nickname :DD !!!!!” Has no ”lang” tag but three times ”other”.

4.3 Accuracy labelling Gibberish

To take a look at the strengths and weaknesses of the Logistic Regression we labelled 5000 tweets containing 69960 words. We compute the LRs accuracy per label. Our trainingsdata has six different labels. Two of them are actual langauges. Every label is shown in table 4.3. To compute each labels accuracy we computed the relative amount of correct labelled word for each label:

$$Accuracy_{Label} = \frac{numberOfCorrectLabels}{numberOfLabel}$$

The results are shown in table 4.3
The table shows that the LR has problems labelling words which are no natural language (other, mixed) and name enteties (ne). The worst performance has the LR with ambiguous word (Words which are equal in spanish and english e.g. ”no”) with only 9%. But only 0.24% of the labels are ambiguous. Thus is not much improvement for the overall system possible. Another good solution shows the LR for *Language Detection*. This is the abbillity to recognise that the word has to be labelled with ”english” or ”spanish”. This could be usefull to brute-force decript cypher texts with unkown keys. The LR can be used to check if the decription includes actual words or is just a failed decription containing only gibberish. The best accuaracy has the LR with english words (87%). It seems that the bigramms perform really good with english and wors with spanish (74%).

5 Summary

Both evaluations. Showed that the LR has problems with the label ”other” but perfroms good when it comes to languages.

References

Ben Kind and Steven Abney. Labeling the languages of words in mixed-language documents using weakly supervised methods.

Label	Accuracy
Other	0.47
Ne	0.44
Mixed	0.61
Ambiguous	0.09
Language Detection	0.83
Spanish	0.74
English	0.87

Table 2: Accuracy for different labels

Joel Ilao Rachel Roxas Nathaniel Oco, Jason Wong. 2014. Detecting code-switches using word bigram frequency count.

Carol Myers Scotton and William Ury. 1977. Bilingual strategies: The social functions of code-switching. *International Journal of the sociology of language*, 1977(13):5–20.

First workshop on computational approaches to code switching.