Language Identification - Programming in Julia Final Project

Milan Miletic, University of Tübingen

October 2020

1 Overview

Given my current knowledge of the matter, my approach relies on the use of a Naive Bayesian Classifier using character bigrams for the purpose of identifying a language of a single given word. For the sake of accomplishing this task, I made use of the following Julia packages: `DelimitedFiles`, `IterTools`, and `StatsBase`.

2 Corpus

In order to improve the results of the program, I started by filtering the given corpus first. I removed all non-word tokens from the corpus (punctuation, numbers etc.) since they do not provide any useful information for our purposes. For the same reason, I normalized the word tokens by
turning all the letters to lowercase. Another 'problem' I faced and had to deal with was the fact that in some languages certain words were written with a single space separating each letter of the word. For instance, *pasai*, the word meaning 'article' in Achehnese-Latin1, was originally written as *P a s a i* and had to be normalized.
In order to do all this, I used the perks of Linux's command line tools (mainly `awk`). Here are the exact commands that I used for filtering the corpus:

```
more training.tsv |
awk -F "\t" '{$2=tolower($2); print}' OFS="\t" |
awk -F "\t" '{gsub(/[[:punct:]]/, "", $2); print}' OFS="\t" |
awk -F "\t" '{gsub("[0-9]+", "", $2); print}' OFS="\t" |
awk -F "\t" '{a = gensub(/\w{2,}/, "€\\0€", "g", $2); print $1, a}' OFS="\t" |
awk -F "\t" '{b = gensub(/€\s((\w)|(\w\s\w))\s€)/, "€\\1€", "g", $2); print $1, b}' OFS="\t" |
awk -F "\t" '{c = gensub(/€(\w)\s(\w)€/, "€\\1\\2€", "g", $2); print $1, c}' OFS="\t" |
awk -F "\t" '{d = gensub(/\s/, "", "g", $2); print $1, d}' OFS="\t" |
awk -F "\t" '{e = gensub(/€+/, " ", "g", $2); print $1, e}' OFS="\t" |
awk -F "\t" '{f = gensub(/(^ | $)/, "", "g", $2); print $1, f}' OFS="\t" >
training_edited.tsv
```

3 Approach

As mentioned above, I used a Naive Bayesian Classifier based on character bigrams to approach the problem. I decided to create a program that would predict the language of a word based on the
probabilities of its bigrams occurring in that language. The predicted language would be the
language with the highest total probability. The 'naive' assumption made here is that all bigrams are statistically independent.
To be more precise, for all languages provided, I stored the bigram counts for all the words appearing in the corpus. Therefore, when the user enters the word, I would divide it into bigrams and find the probability of each of those bigrams in all 190 given languages. Afterwards, I would multiply the probabilities of each bigram in each language (independence assumption) and get the final
probability for each language. Finally, the language with the highest final probability is the one that will be returned.

3.1 Laplace Smoothing

In order to take into account the possibility that a bigram from the input word (entered by the user) doesn't occur at all in a certain language, I used the Laplace (add-1) Smoothing technique. In other words, I added 1 to all the bigram occurences when calculating the probabilities, meaning that the bigrams that occured 0 times, would now have occured 1 time. Consequently, I would divide the bigram counts not just by the total number of bigram tokens, but by this number plus the number of unique bigrams in the language, in order to take this addition into account.

3.2 Optimization

In order to make the program a bit more efficient I decided to do the following - I wouldn't just immediately rush into calculating the probabilities, but first make a couple of checks. I would first simply check whether the input word already appears as such in any of the languages. In case that the word appears in exactly one of the given languages, the program will immediately predict this as the correct language, without going through all the calculations. Furthermore, if the word appears in more than one language, I would do the described calculations only for those languages in which the word appears, and not for all, in order to improve efficiency. Only if the input word was not to be found in any of the 190 languages would I do the complete process as mentioned.

4 Concerns

This approach is probably not the most statistically precise one. I am not yet skillful enough to know how exactly I could calculate the confidence level in this case, thus I am not able to
correctly estimate the accuracy of the result. However, for higher accuracy, statistical linguistic knowledge of the given languages would significantly improve the approach. That, however, would require much more time and contemplation and, most likely, much more experience.