Project 1 – Naive Bayes Language Classification

# INTRODUCTION

In [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing), **language identification** is the problem of determining which [natural language](https://en.wikipedia.org/wiki/Natural_language) given content is in. Computational approaches to this problem view it as a special case of [text categorization](https://en.wikipedia.org/wiki/Text_categorization), solved with various [statistical](https://en.wikipedia.org/wiki/Statistical) methods. The data set is lines of movie subtitles from 21 different languages. The task here is to train a model based on training data set, learn the hyper parameters on validation set and identify the language of subtitles in unknown languages. The model implemented for this task is Naïve Bayes model. Each line in training data set is considered as a single document. All documents are represented as Bag of n-grams.

# NAÏVE BAYES MODEL

In [machine learning](https://en.wikipedia.org/wiki/Machine_learning), **Naïve Bayes classifier** is a simple [probabilistic](https://en.wikipedia.org/wiki/Probabilistic_classifier) model based on applying [Bayes' theorem](https://en.wikipedia.org/wiki/Bayes%27_theorem) with strong (naive) [independence](https://en.wikipedia.org/wiki/Statistical_independence) assumptions between the features. Input to the model is the set of documents and fixed set of classes. The bag of n-grams representation is used for considering features for this model. Each document is given as docid, sentence and its language separated by pipe symbol. Below is a sample from training data set.

train.s1|kaÅ¾dopÃ¡dnÄ› kdybych nekÃ½vl tomu bezdomovci , tak nikdy nepotkÃ¡m allison .|cze

Here, train.s1 is the document-id, The sentence next to it is subtitle and “cze” is the language (class) of the document.

## Bag of n-gram representation

In the fields of [computational linguistics](https://en.wikipedia.org/wiki/Computational_linguistics) and [probability](https://en.wikipedia.org/wiki/Probability), an **n-gram** is a contiguous sequence of n items from a given [sequence](https://en.wikipedia.org/wiki/Sequence) of text or speech. The items can be [phonemes](https://en.wikipedia.org/wiki/Phoneme), [syllables](https://en.wikipedia.org/wiki/Syllable), [letters](https://en.wikipedia.org/wiki/Letter_(alphabet)), [words](https://en.wikipedia.org/wiki/Word) or [base pairs](https://en.wikipedia.org/wiki/Base_pairs) according to the application. For this model the sentence is converted to n-grams. Before converting sentence to n-grams (n-1) times ‘#’ is appended at the beginning and at the end of each sentence. Process of converting a sentence to n-grams is explained below with an example from training data set.

Below is a sample line from training document

###kaÅ¾dopÃ¡dnÄ› kdybych nekÃ½vl tomu bezdomovci tak nikdy nepotkÃ¡m allison ###

Considering n = 4, the n-grams for above sentence will be generated as:

“###k”, ”##ka”,”#kaÅ”, ”kaÅ¾”,……., “son ”,”on #”,”n ##”.

The document is represented as count of each n-gram as follows.

|  |  |
| --- | --- |
| ###k | 1 |
| ##ka | 1 |
| #kaÅ | 1 |
| . . . | . . . |

The Bayes’ rule applied to documents d and class c is as follows:

Where, d is the document and c is the class (language in our case). The Naïve Bayes classifier is as follows:

gives the most likely class c for the document d. Here, class is one of the 21 languages.

is the probability of the document given the language. And the document is considered as a bag of n-grams. And each n-gram is considered as a feature of the document d. So it can be represented as follows:

Where, the are the n-grams in document d.

So can be calculated as follows:

Probability of each n-gram given the language is calculated as follows:

Where, is the count of the n-grams in the language c. so the term is the sum of count of all the n-grams in the language c. The term |V| is the size of whole vocabulary that is the number of unique of n-grams in all the languages. And the term is for smoothing purpose.

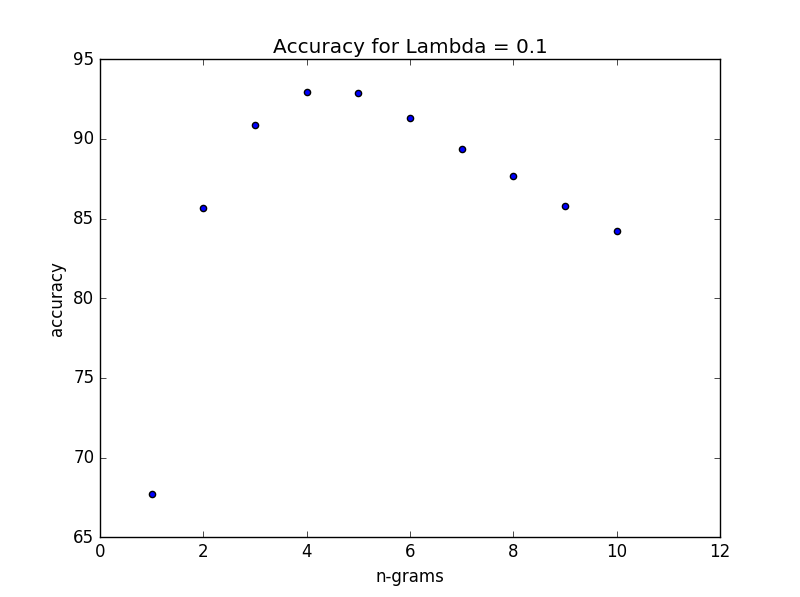
The term p(c) in the main equation, is the marginal probability of the class (language) c, and is calculated as follows:

Where, doccount) is the number of documents with language . And is total number of documents in training data set.

The training part involves reading training data set, creating n-grams and calculating frequencies of each n-gram as per language. Validation part involves reading validation data set and training value of n and . Accuracy is simply calculated as percentage of number of correct prediction of documents to total number of documents in validation set.

# Effect of n in performance

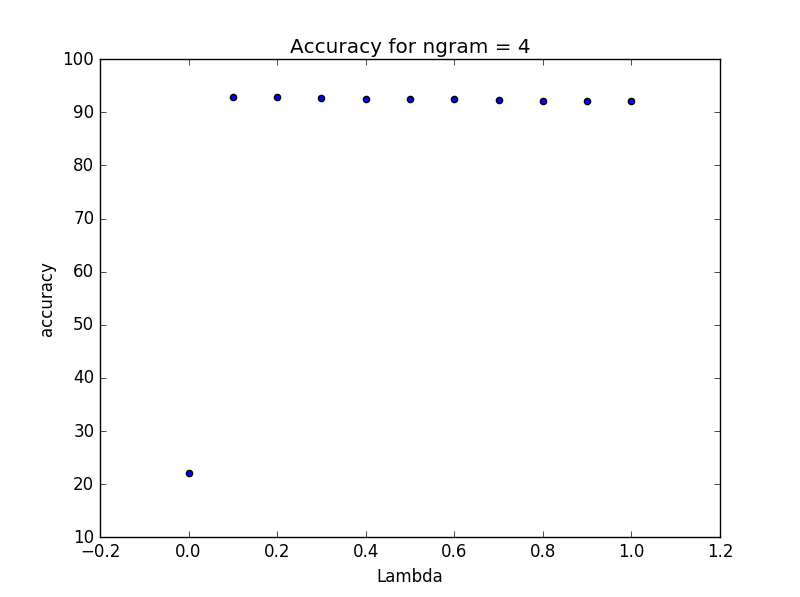
The change in accuracy of the model with is as follows:



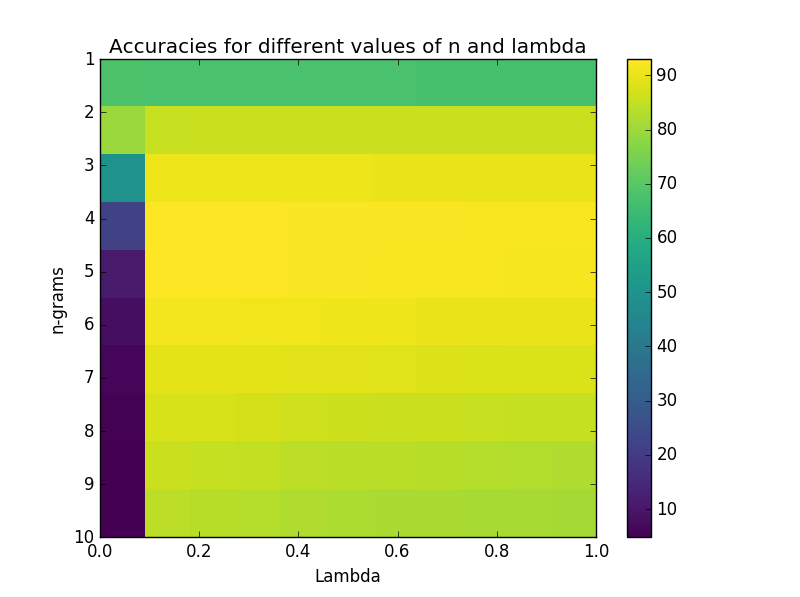
We can observe that the accuracy is maximum at n = 4. It can also be observed that with very small value of n, the accuracy decrease for example at n = 1, accuracy is less than 70%. The reason is that with n = 1, we are considering only single characters which means there is no importance given to combination of characters. With increase in n after n = 4, the accuracy decreases gradually, it indicates that with increase in n, we are considering combination set of larger number of characters in the sentence, which makes it less likely to occur. So it becomes more random. So, these are the main reasons for decrease in accuracy of the model with value of n less than 3,4 and for values greater than 4.

# Effect of in performance

The change in accuracy of the model with is as follows:



We can see at , accuracy falls down to around 20%, the reason is that even if a single n-gram is not found in the document, then the whole probability for that language will be zero. And generally, there are few n-grams which are not in any language or in some languages only, even though they actually belong to different language. But we can see that accuracy doesn’t change much with increase in . Below is the plot for change in accuracy for different combinations of n and . We can see that maximum accuracy of 93.604% is obtained at n = 4 and = 0.11.



# Performance across Languages

|  |  |
| --- | --- |
| **Language** | **Accuracy** |
| Hun | 93.001 |
| Ell | 100 |
| Dan | 87.983 |
| Swe | 91.805 |
| Slo | 88.952 |
| Nor | 83.140 |
| Ita | 96.946 |
| Fin | 93.898 |
| Fre | 94.972 |
| Pol | 94.917 |
| Rum | 96.907 |
| Cze | 89.001 |
| Ind | 94.019 |
| Por | 88.863 |
| Dut | 95.992 |
| Tur | 93.732 |
| Spa | 91.895 |
| Eng | 96.030 |
| Vie | 80.752 |
| Ice | 99.049 |
| Ger | 95.063 |
| **OVERALL** | 93.604 |

There can be many reason for difference in the accuracy of individual language. One reason can be the data itself, Data can available for that particular language can be very ambiguous or may have n-grams common to another language. This can lead to transfer of prediction from one language to another language decreasing accuracy of first language. Another reason can be number of unique n-grams in each language.