Computational Linguistics CSE 567 Project 1 – Naive Bayes Language Classification

Raj Jaysukh Patel

Person number: 50208278 Ubit name: rajjaysu

INTRODUCTION

In natural language processing, **language identification** is the problem of determining which natural language given content is in. Computational approaches to this problem view it as a special case of text categorization, solved with various 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, **Naïve Bayes classifier** is a simple probabilistic model based on applying Bayes' theorem with strong (naive) 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 and probability, an **n-gram** is a contiguous sequence of n items from a given sequence of text or speech. The items can be phonemes, syllables, letters, words or 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. The reason for this is to considered each starting (n-1) characters independently. 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 ##”.

We can see that it is necessary to add (n-1) '#'s at the beginning and at the end of the sentence to consider 'k', 'ka' and such combinations as well. 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:

$$p(c|d) = \frac{p(d|c)p(c)}{p(d)}$$

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

$$\begin{aligned} C_{MAP} &= argmax_{c \in C} P(c|d) \\ &= argmax_{c \in C} \frac{p(d|c)p(c)}{p(d)} \\ &= argmax_{c \in C}\, p(d|c)p(c) \end{aligned}$$

We here dropped the denominator as it will be same for all $c \in C$. C_{MAP} gives the most likely class c for the document d. Here, class is one of the 21 languages.

$p(d|c)$ 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:

$$p(d|c) = p(x_1, x_2, \ldots\, x_n|c)$$

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

With bag of words comes the assumption, that position doesn't matter. Another assumption made is that the probability of each feature is independent of the other. So $p(d|c)$ can be calculated as follows:

$$p(d|c) = p(x_1, x_2, \ldots\, x_n|c)$$

$$p(d|c) = \pi_{x \in X} p(x|c)$$

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

$$p(x_i|c) = \frac{count(x_i, c) + \lambda}{(\sum_{x \epsilon VC} count(x, c) + \lambda|V|}$$

Where, $count(x_i, c)$ is the count of the n-grams x_i in the language c. so the term $\sum_{x \epsilon VC} count(x, c)$ 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 λ is for smoothing purpose. The basic reason for smoothing is to handle cases like $p(x|c)$ = 0, which will cause the whole product of all the n-

grams to be zero, even if one of the n-grams is not present in the language. And this case is very often as the training data may not be large enough to include all possible n-grams.

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

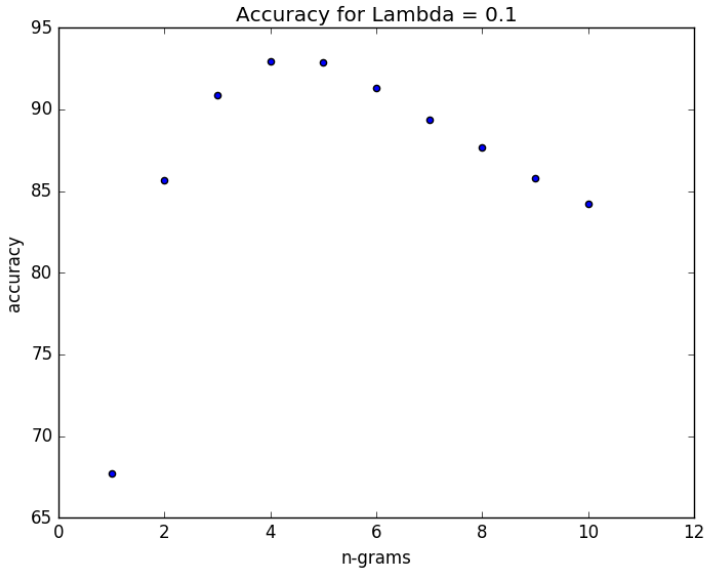
$$p(c_i) = \frac{doccount(C = c_i)}{N_{doc}}$$

Where, doccount($C = c_i$) is the number of documents with language c_i. And N_{doc} 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 λ. 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

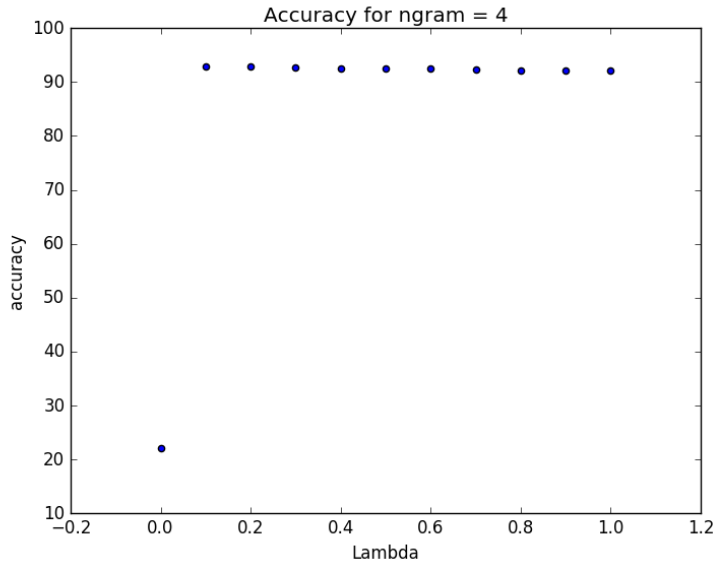
The change in accuracy of the model with $\lambda = 0.1$ is as follows:



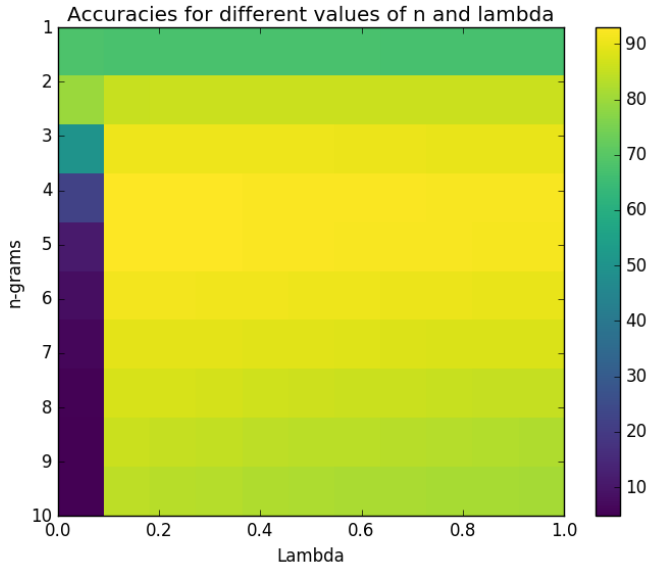
We can observe that the accuracy is maximum at n = 4. It can also be observed that with very small value of n, the accuracy decrease for example at n = 1, accuracy is less than 70%. The reason is that with n = 1, we are considering only single characters which means there is no importance given to combination 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 because it's very less likely to find such large combination of characters in training data set. 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 λ in performance

The change in accuracy of the model with $n = 4$ is as follows:



We can see at $\lambda = 0$, 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 λ. Below is the plot for change in accuracy for different combinations of n and λ. We can see that maximum accuracy of 93.604% is obtained at n = 4 and λ = 0.11.



Performance across Languages

Language	Accuracy	Number of unique n-grams(n=4)	Total number of n-grams/ unique n-grams
Hun	93.001	691	35
Ell	100	277	223
Dan	87.983	564	49
Swe	91.805	588	50
Slo	88.952	746	36
Nor	83.140	559	44
Ita	96.946	468	60
Fin	93.898	466	62
Fre	94.972	549	52
Pol	94.917	675	37
Rum	96.907	554	46
Cze	89.001	758	35
Ind	94.019	462	63
Por	88.863	551	47
Dut	95.992	521	53
Tur	93.732	648	44
Spa	91.895	577	45
Eng	96.030	511	54
Vie	80.752	805	41
Ice	99.049	629	46
Ger	95.063	573	49
OVERALL	93.604	579.619	

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. As shown in table above, it is clear that language “Ell” with minimum number of unique n-grams that is only 277, has highest accuracy of 100%, while the language with maximum number of unique n-grams equal to 805 has minimum accuracy of around 80%. It is also observant that languages with number of unique n-grams greater than the average of 579.619 has accuracy less than the average accuracy of 93.604%. Last column in above table represent ratio of total number of n-grams in that language and unique n-grams in that language. Higher values mean same n-grams are repeated more times and less value means n-grams are less often. So the case with the language “Ell” can be that it has very particular n-grams that rarely belongs to other languages. And case with the language “Vie” can be that it has n-grams that often matches with that of other languages.