Natural Language Processing for Big Data by Using Data Fusion Approaches in a Multi-Classifier Fusion System Based on Hadoop

Mohammad Hossein Badiei, Behzad Moshiri

Abstract— ***Natural Language Processing is the technology used to aid computers to understand the human's natural language. It's not an easy task teaching machines to understand how we communicate so there are several models which able to do this in order to understanding how humans communicate with each other. N-grams represent a continuous sequence of N elements from a given set of texts. They are statistical models that predict the next word in a text by using the previous n-1 words. In this paper, we investigate how these models can be implemented and how to fuse them as a new model called backoff model. Also we study the role of naïve Bayes algorithm in natural language processing classification. The result of this work is based on the HDFS system of Hadoop. We*** **Exploit MapReduce algorithm in order to reduce the loads of training and validating the models.** ***Therefore, we study how N-grams and multi-classifier system can be computed efficiently using a MapReduce for distributed data processing.*** **The system, which is implemented, is a system of recognizing the poet based on a stanza of the poem, which is given to it. This system has the ability to distinguish the poet from a single stanza with a maximum accuracy of about 82.5% by analyzing the words in the stanza.**

Index Terms— **Natural Language Processing, N-grams model, Naïve Bayes, Hadoop, Multi-Classifier Fusion System**

I. Introduction

To complete the Natural Language Processing tasks such as text classification, we provide a measure to enable comparison operations and thus assessment method for grading our models. This measure is probability. Actually, the goal of a language model is to compute a probability of a sequence of words. The probability of a sentence *S* as a sequence of *words wi* is: *P*(*S*) =*P* (*w*1, *w2, w3*,…,*wn*). Now it is important to find the probability of upcoming word. It is an everyday task made e.g. while typing our mobile keyboard. We will settle the *conditional probability* of *w*3 depending on all previous words. For a 3-word sentence this conditional probability is:

$$P(S) = P(w1, w2, w3) \equiv P(w3|w1, w2)$$

To calculate the joint probability of a sentence as a word sequence *P*(*W*) the rule of probability below will be used.

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \Rightarrow P(A \cap B) = P(A|B)P(B)$$

So For a item sequence we get:

$$P(S) = P(w_1, \dots, w_n) = \prod_i P(w_i|w_1, \dots w_{i-1})$$

Therefore, to estimate each probabilities, we can use simple counting shown below.

$$P(w4|w1, w2, w3) = \frac{N(w1, w2, w3, w4)}{N(w1, w2, w3)}$$

However, this gives us too many possible sequences to ever estimate. It is so many data (occurrences of each sentence) we would have to get. Therefore, the role of big data and HDFS system is obvious here. Hadoop already has build in text processing support. There are many approaches being applied to the data sets based on that, particularly inverted indices, co-location scores and N-Gram modeling. Therefore, you can simply find out how the latest tools, techniques, and algorithms driving modern and predictive analysis can be applied to produce powerful results.

II. HAD OOP MAPREDUCE FRAMEWORK AND HBASE

Hadoop is an open source framework for coding and running distributed applications that process a massive amount of data [5]. MapReduce is a programming model which supports the framework model [6]. When loading, the file is dividing into multiple data blocks with the same size, typically 64MB blocks named FileSplits, and to guarantee fault-tolerance each block is triplicated [7]. A general MapReduce architecture can be illustrated as Figure 1.

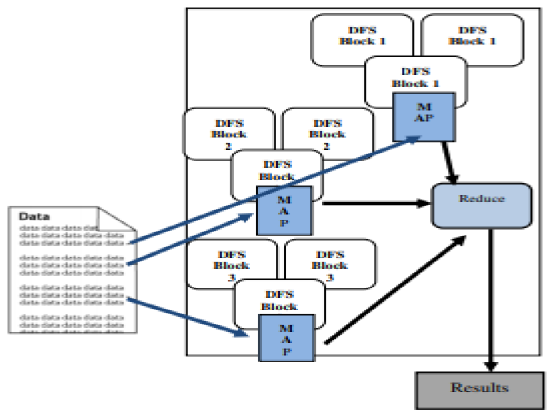


Fig 1. MapReduce Architecture. [3]

A MapReduce *job* splits the input dataset into independent chunks, which are processed by the *map tasks* in a parallel manner. The framework sorts the outputs of the maps, which are then input to the *reduce tasks*. Typically, both the input and the output of the job are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks. The MapReduce framework consists of a single master JobTracker and one slave TaskTracker per cluster-node. The master is responsible for scheduling the jobs' component tasks on the slaves, monitoring them and re-executing the failed tasks. The slaves execute the tasks as directed by the master.

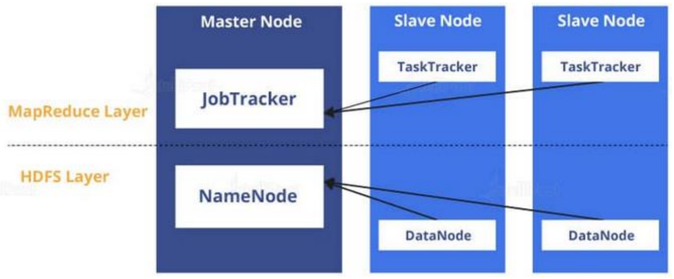


Fig 2. Sketch of the Hadoop architecture

There are two layers in the Hadoop architecture. First, we will see how data is stored in Hadoop and then we will move on to how it is processed. While talking about the storage of files in Hadoop.

HDFS is based on Google File System (GFS), which provides a distributed system particularly designed to run on commodity hardware. The file system has several similarities with the existing distributed file systems. HDFS is mainly responsible for taking care of the storage parts of Hadoop applications. In HDFS, files will be split into chunks, called blocks. The default size of each block in Hadoop 1 is 64 MB; on the other hand, in Hadoop 2 it is 128 MB. For example, in Hadoop version 1, if we have a 100 MB file, it will be divided into 64 MB stored in 1 block and 36 MB in another block. In addition, each block is given a unique name, i.e., blk_n (n = any number). Each block is uploaded to one DataNode in the cluster. On each of the machines or clusters, there is something called as a daemon or a piece of software that runs in the background. You can get more information from https://hadoop.apache.org.

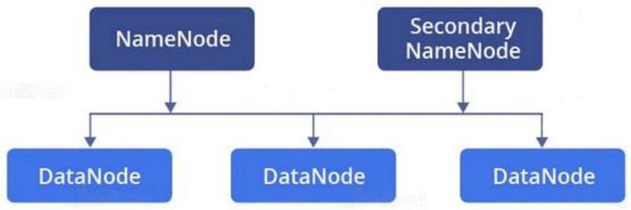


Fig 3. Sketch of nodes in Hadoop architecture

III. N-GRAM

Now, we must consider on N-Gram modeling and its fusion and classification as the main approach which is used in this paper, let us describe how it works.

An N-gram is a contiguous (order matters) sequence of items, which in this case is the words in text. The n-grams depends on *the size of the* prefix, which is represented by a conditional statement. The simplest case is the Unigram mode.

A. *Unigram or 1-gram model*

A unigram is a unary word sequence. Consider the sentence "I love reading blogs about data science on Analytics Vidhya.". For this sentence, the unigrams would simply be: "I", "love", "reading", "blogs", "about", "data", "science", "on", "Analytics", "Vidhya". The simplest case of *Markov assumption* is case when the size of prefix is one.

$$P(w_1, \dots, w_n) \approx \prod_i P(w_i)$$

B. *Bigram or 2-gram model*

A bigram is a two-word sequence of words, like "I love", "love reading", or "Analytics Vidhya". Therefore, if we consider 2-word bigrams correlations where we condition each word on previous one we get some sens of meaning between couples of words.

$$P(w_1, \dots, w_n) \approx \prod_i P(w_i|w_{i-1})$$

$$P(w_i|w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

This way we get a sequence of tandems that have meaning tandem-wise. This still is not enough. Se we define another model as the backoff model.

C. *Backoff model*

In the backoff model, like the deleted interpolation model, we build an N-gram model based on an (N-1)-gram model. The difference is that in backoff, if we have non-zero trigram counts, we reply solely on the trigram counts and don't interpolate the bigram and unigram counts at all. Therefore, we only backoff to a lower-order N-gram if we have zero evidence for a higher-order N-gram.
The trigram version of backoff might be represented as follows:

$$P(w_i|w_{i-2}w_{i-1}) = \begin{cases} P(w_i|w_{i-2}w_{i-1}) & ,if\ C(w_{i-2}w_{i-1}w_i) > 0 \\ \alpha_1 P(w_i|w_{i-1}) & ,if\ C(w_{i-2}w_{i-1}w_i) = 0\ and\ C(w_{i-1}w_i) > 0 \\ \alpha_2 P(w_i) & otherwise \end{cases}$$

Interpolation is another technique in which we can estimate an n-gram probability based on a linear combination of all lower-order probabilities. For instance, a 4-gram probability can be estimated using a combination of trigram, bigram and unigram probabilities. The weights in which these are combined can also be estimated by reserving some part of the corpus for this purpose. While backoff considers each lower order one at a time, interpolation considers all the lower order probabilities together.

IV. OWA

The Yager's OWA operators are used to aggregate the crisp values in decision making schemes. We use three method of these operators for assigning weights to the estimated probability of each model in the backoff model, which is consisted of O'Hagan's OWA, Optimistic Exponential OWA and Pessimistic Exponential OWA. In Exponential OWA, the weights are determined by the value of alpha and orness, which is calculated from the table shown below.

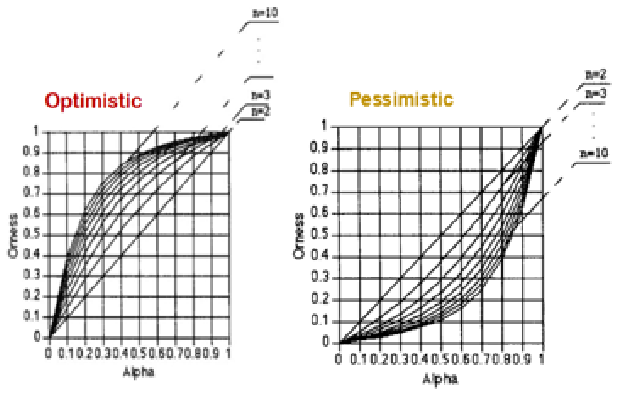
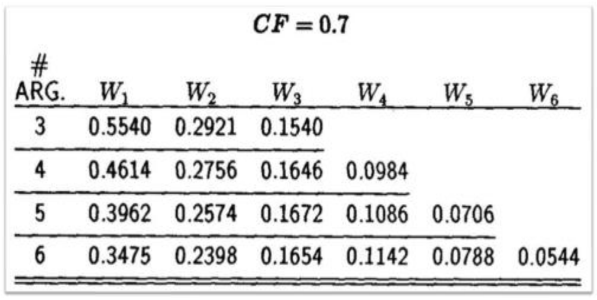


Fig 4. Optimistic and Pessimistic Exponential OWA

Also for O'Hagan's OWA, we determine the weights from O'Hagan's table shown below.



$CF = 0.7$

# ARG.	W_1	W_2	W_3	W_4	W_5	W_6
3	0.5540	0.2921	0.1540			
4	0.4614	0.2756	0.1646	0.0984		
5	0.3962	0.2574	0.1672	0.1086	0.0706	
6	0.3475	0.2398	0.1654	0.1142	0.0788	0.0544

Fig 5. O'Hagan's OWA

V. Naïve Bayes Classifier

Bayes theorem provides a way of calculating the posterior probability, P(c|x), from P(c), P(x), and P(x|c). We use naïve Bayes classifier for classification.

$$l^* = argmax\ P(l|c_{1:N}) = argmax\ P(l)P(c_{1:N}|l)$$

$$\Rightarrow\ l^* = argmax\ P(l)\prod_{i=1}^{N} P(c_i|c_{i-2:i-1}, l)$$

$$P(l|c_{1:N}) = \frac{P(l)P(c_{1:N}|l)}{P(c_{1:N})}$$

VI. Majority Voting

A voting ensemble (or a "majority voting ensemble") is an ensemble machine learning model that combines the predictions from multiple other models. It is a technique that may be used to improve model performance, ideally achieving better performance than any single model used in the ensemble. We use both method of majority voting consist of hard and soft majority voting.

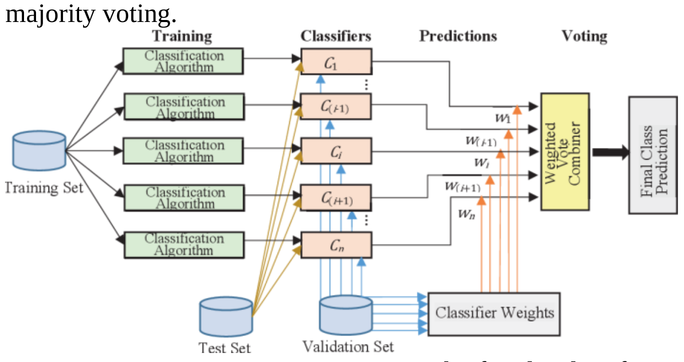
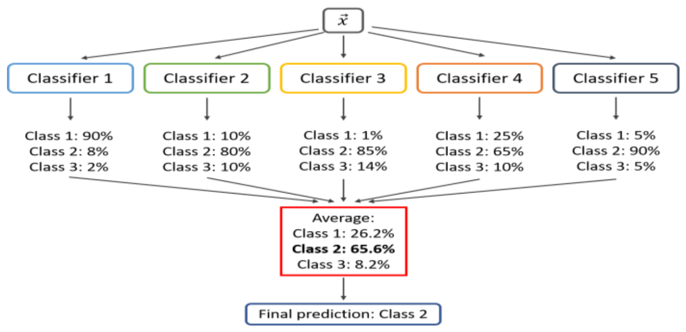


Fig 6. Majority voting as an approach of multi-classifier fusion

A. Soft majority voting

Soft voting ensemble involves summing the predicted probabilities for class labels and predicting the class label with the largest sum probability.

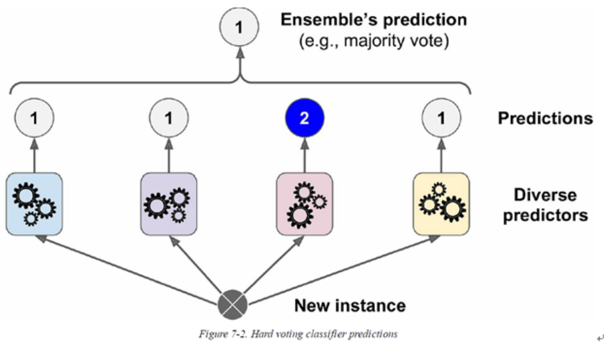


$\vec{x}$
Classifier 1
Classifier 2
Classifier 3
Classifier 4
Classifier 5
Class 1: 90% Class 2: 8% Class 3: 2%
Class 1: 10% Class 2: 80% Class 3: 10%
Class 1: 1% Class 2: 85% Class 3: 14%
Class 1: 25% Class 2: 65% Class 3: 10%
Class 1: 5% Class 2: 90% Class 3: 5%
Average: Class 1: 26.2% Class 2: 65.6% Class 3: 8.2%
Final prediction: Class 2

Fig 7. Soft majority voting

B. Hard majority voting

Hard voting ensemble involves summing the votes for crisp class labels from other models and predicting the class with the most votes.



Ensemble's prediction (e.g., majority vote)
1
2
Predictions
Diverse predictors
New instance
Figure 7-2. Hard voting classifier predictions

Fig 8. Hard majority voting

VII. Results and Discussion

A. Backoff model with O'Hagan's OWA

The results of accuracy curve for 10 different test data and train data for this method is shown below.

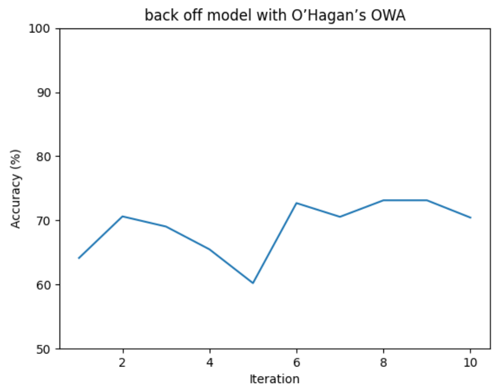


Fig 9. Accuracy curve of Backoff model with OHagan's OWA

The average accuracy is obtained from this method is 68.947 % and the standard deviation of that is 4.320.

B. Backoff model with optimistic exponential OWA

The results of accuracy curve for 10 different test data and train data exactly the same of previous part is shown below.

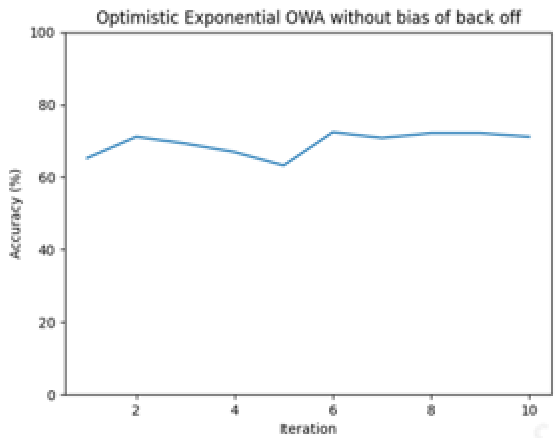


Fig 10. Accuracy curve of Backoff model with optimistic exponential OWA

The average accuracy is obtained from this method is 69.422 % and the standard deviation of that is 3.210.

C. Backoff model with optimistic exponential OWA and bias

We add bias to the previous backoff model in order to track the effect of that. So, the results of accuracy curve is shown below.

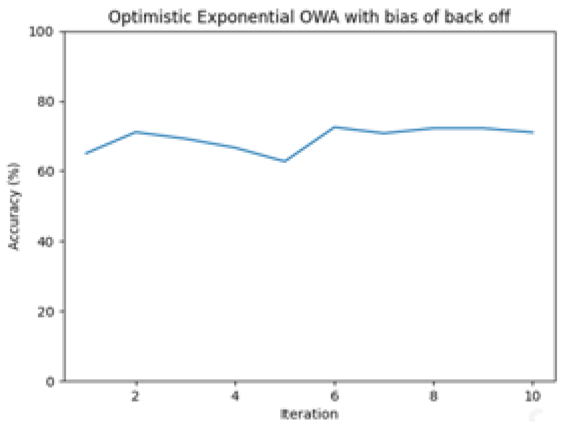


Fig 11. Accuracy curve of Backoff model with optimistic exponential OWA and bias

As you can see here, the average accuracy is obtained from this method is 69.309 % and the standard deviation of that is 3.396. It is obvious that accuracy is reduced a little bit. It shows that bias doesn't always approve the fusion accuracy results. In the following, we'll discuss about it.

D. Backoff model with pessimistic exponential OWA

The results for this method is shown below.

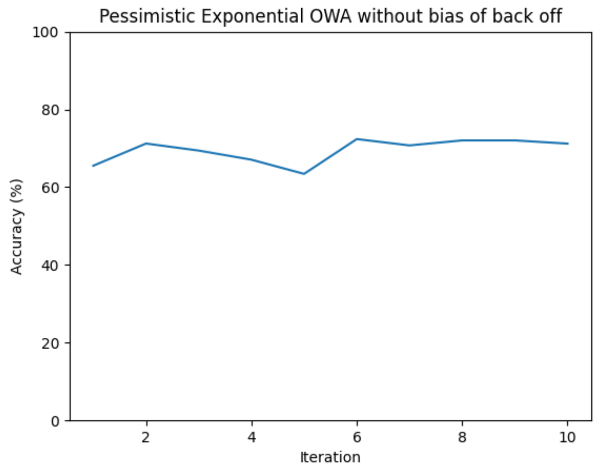


Fig 12. Accuracy curve of Backoff model with pessimistic exponential OWA

The average accuracy is obtained from this method is 69.492 % and the standard deviation of that is 3.111.

E. Backoff model with pessimistic exponential OWA and bias

The last item we discuss about it in the study of owa fusion effects is backoff model with pessimistic OWA and bias. The results is shown below.

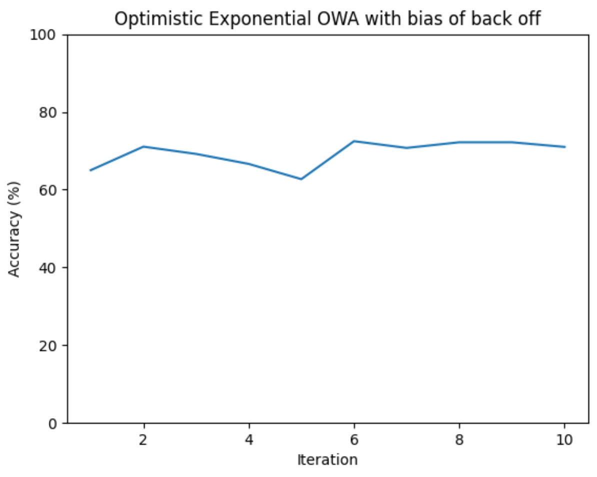


Fig 13. Accuracy curve of backoff model with pessimistic exponential OWA and bias

The average accuracy is obtained from this method is 69.0465 % and the standard deviation of that is 3.762.

In this case, the bias shows the same results as we saw before. Actually, it have a bad effect in the system and it means that the bias of a backoff model with fusion approaches might make the accuracy lower.

Now, we discuss about the application of soft and hard majority voting in model selection between unigram, bigram and backoff. In the following we investigate the effects of soft and hard majority voting in the system.

F. *Hard and soft majority voting*

In this case, we plot the effect of soft and hard majority voting in model selection and archive the best result in hard majority voting. You can see the results below.

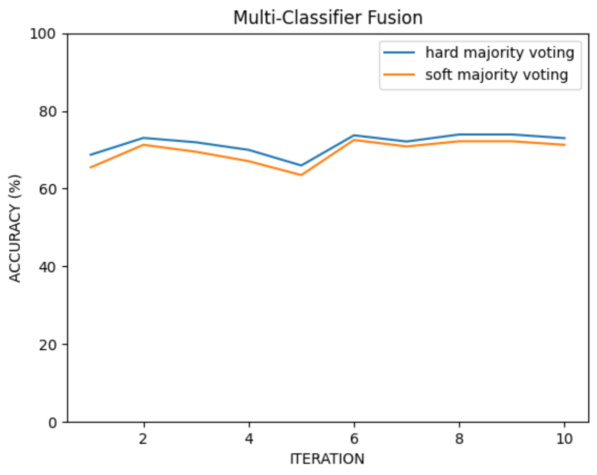


Fig 14. Accuracy curve for hard and soft majority voting

The average accuracy is obtained for hard majority voting is 71.623 % and the standard deviation of that is 2.635. In addition, the average accuracy is obtained for soft majority voting is 69.571 % and the standard deviation of that is 3.160.

G. *Unigram and Bigram model with naïve Bayes classifier*

The result of this part is shown below.

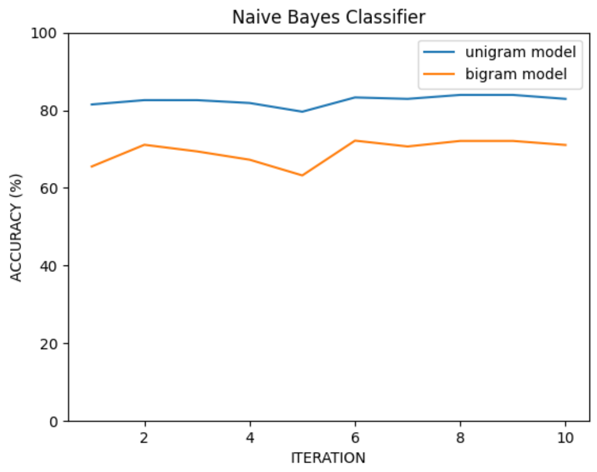
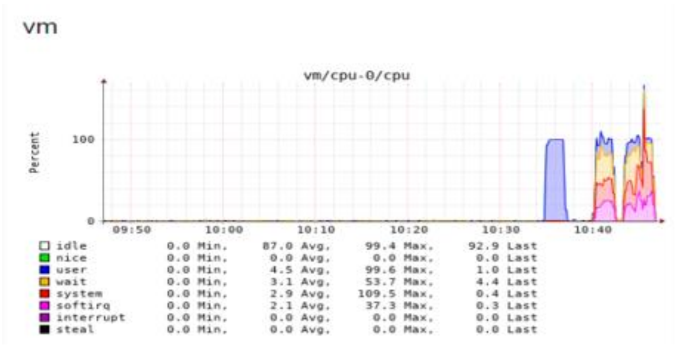


Fig 15. Accuracy curve of unigram and bigram model with naïve Bayes classifier

As you could see here, we obtain the highest average accuracy in unigram model with naïve Bayes classifier. The average accuracy in this case is about 82.518 % and the standard deviation of that is 1.286. Also for bigram model, we obtained the average accuracy equal to 69.459 % and 3.116 for the standard deviation.

H. *Memory and CPU usage of the software in Hadoop*

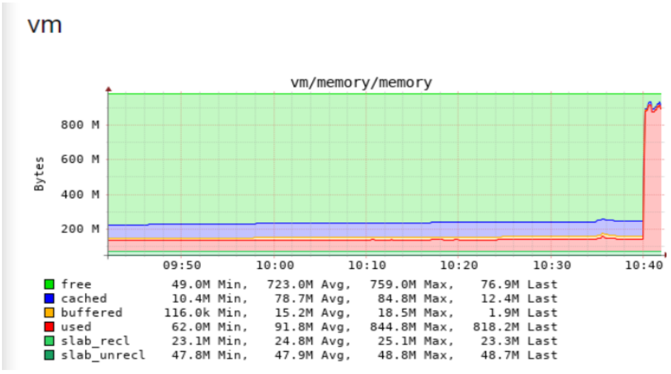
The results of CPU usage is shown below:



```
□ idle        0.0 Min.   87.0 Avg.    99.4 Max.   92.9 Last
■ nice        0.0 Min.    0.0 Avg.     0.0 Max.    0.0 Last
■ user        0.0 Min.    4.5 Avg.    99.6 Max.    1.0 Last
■ wait        0.0 Min.    3.1 Avg.    53.7 Max.    4.4 Last
■ system      0.0 Min.    2.9 Avg.   109.5 Max.    0.4 Last
■ softirq     0.0 Min.    2.1 Avg.    37.3 Max.    0.3 Last
■ interrupt   0.0 Min.    0.0 Avg.     0.0 Max.    0.0 Last
■ steal       0.0 Min.    0.0 Avg.     0.0 Max.    0.0 Last
```

Fig 16. CPU usage of the system in Hadoop

In addition, the results of memory usage of the system is shown below.



```
🟩 free          49.0M Min,   723.0M Avg,   759.0M Max,    76.9M Last
🟦 cached        10.4M Min,    78.7M Avg,    84.8M Max,    12.4M Last
🟧 buffered     116.0k Min,    15.2M Avg,    18.5M Max,     1.9M Last
🟥 used          62.0M Min,    91.8M Avg,   844.8M Max,   818.2M Last
🟩 slab_recl     23.1M Min,    24.8M Avg,    25.1M Max,    23.3M Last
🟩 slab_unrecl   47.8M Min,    47.9M Avg,    48.8M Max,    48.7M Last
```

Fig 17. Memory usage of the system in Hadoop

VIII. CONCLUSION

In this section, we integrated the results in the unit table.

TABLE I

model	AVERAGE ACCURACY OF ACCURACY CURVE	Standard deviation of accuracy curve
Backoff model with O'Hagan's OWA	68.947 %	4.320
Backoff model with optimistic exponential OWA	69.422 %	3.210
Backoff model with optimistic exponential OWA and bias	69.309 %	3.396
Backoff model with pessimistic exponential OWA	69.492 %	3.111
Backoff model with pessimistic exponential OWA and bias	69.0465 %	3.762
Hard majority voting	71.623 %	2.635
Soft majority voting	69.571 %	3.160
Unigram model with naïve Bayes	82.518 %	1.286
Bigram model with naïve Bayes	69.459 %	3.116

According to the above table, we can conclude unigram has the best performance and afterward, hard majority voting considering to its accuracy, places in the second rank etc. Among fusion methods, pessimistic has a better performance compared with the others. It means that each of the models has a god estimation on the test data. Because we get approximately the same weight to each model in this case and also it means we cannot ignore any models in the fusion system or make it's effect low.

Considering to low memory and cpu of the VMs, we have a good performance in the system and I think it's admissible for processing of natural language software and big data.

REFERENCES

[1] J. Gao, P. Nguyen, X. Li, C. Thrasher, M. Li, K. Wang,'' A Comparative Study of Bing Web N-gram Language Models for Web Search and Natural Language Processing".

[2] H. Adel, K. Kirchhoff, N. Thang Vu, D. Telaar, T. Schultz, " Comparing Approaches to Convert Recurrent Neural Networks into Backoff Language Models For Efficient Decoding".

[3] T. M. Allam, H. M. Abdullkader, A. Abdelhameed Sallam,"Managed N-gram Language Model Based on Hadoop Framework and a Hbase Tables".

[4] R. P. Padhy, " Big Data Processing with Hadoop-MapReduce in Cloud Systems".

[5] C. Lam, "Hadoop in Action", Manning Publications Co. Copyright 2011, pages. 3-37

[6] K. Lee, H. Choi, B. Moon, "Parallel Data Processing with MapReduce: A Survey," SIGMOD Record, Vol. 40, No. 4, December 2011.

[7] T. M. Allam; H. M. Abdullkader; A. A. Sallam, "Cloud Data Management based on Hadoop Framework," The International Conference on Computing Technology and Information Management, Dubai, 2014 (SDIWC), April 2014, pp 455-460.