Text Classification using XLNet with Infomap Automatic Labeling Process

Author: Triana Dewi Salma, Gusti Ayu Putri Saptawati, Yanti Rusmawati

2021 IEEE 8th International Conference on Advanced Informatics: Concepts, Theory and Applications (ICAICTA)

CSE 4120: Technical Writing and Seminar lab

Submitted by: Alvi Ahmmed Nabil
Roll: 1707009
Computer Science and Engineering
Khulna University Of Engineering & Technology

Submission Date: 27/06/2022

Abstract

Text classification is one of the most important task in Natural Language Processing. As the text data is growing rapidly it needs more computational power to classify the text in a big dataset. Having a good quality text data significantly affects the outcome of the model that has been used to classify them. However to train this huge data in supervised learning, manual labeling is done by the experts in their fields. But this process is expensive, prone to mistakes and very time consuming. If we could automatically label those data, time consumption can be significantly reduced. This experiment attempts to conduct community detection with the Infomap algorithm for authentic labeling in the text classification using XLNet [1]. The accuracy of the model is compared to the baseline, which is data with manual labeling. While the accuracy has not outperformed the overall baseline yer, but the result shows that automatic labeling can improve the data labeling quickly and in high quantity.

Contents

1 Introduction **4**

2 Related Work **4**

3 Methodology of Proposed System **5**
3.1 Dataset 5
3.2 Automatic Labeling of Training Data 5
3.2.1 Preprocessing 6
3.2.2 Addition of Snonyms 6
3.2.3 TF-IDF Transformation 6
3.2.4 Calculation of Cosine Similarity 7
3.2.5 Formation of Sentence Network 8
3.2.6 Infomap Community Detection 8
3.2.7 Testing of Automatic Labeling Result 8
3.3 Text Classification 8

4 Experiment and Analysis **9**
4.1 Comparison between Automatic Labeled Data vs Manual Labeled Data in Text Classification 9

5 Discussion **11**

6 Conclusion **16**

List of Figures

1 Automatic Labeling of Training data Stage 6
2 Closeness Matrix 7
3 Automatic Labeling of Training data Stage 9
4 Text Classification Result(Time vs Epoch) 10
5 Text Classification Result(Accuracy vs Epoch) 10
6 Text Classification Result(Loss vs Epoch) 11
7 Threshold Normalization in Bigram 13
8 Threshold Normalization in Trigram 13
9 Text Classification Result using optimal Threshold(Loss vs Epoch) 14
10 Text Classification Result using optimal Threshold(Loss vs Epoch) 14
11 Text Classification Result using optimal Threshold(Loss vs Epoch) 15

List of Tables

1 Training Data 5
2 Result of Automatic Labeling 9
3 Threshold Test Result Bigram 12
4 Threshold Test Result Trigram 12

1 Introduction

Now a days, the quantity of information that are being dealt with is enormous and with the infrastructure and fast data transmission rate we have in this era, the types and structure of the data varies greatly [2]. In various field, text data is being used for various purpose. Natural Language Processing(NLP), Machine Learning both field widely uses text data, and one of the popular structure of the text data used in NLP is question-answering dataset. In certain task, the system will tries to identify the question category and possibly generate an answer to help with the question [3]. This process of identifying the questions in right category is one of the application of the text classification. To classify the questions the system first needs to be taught the process based on which it will distinguish one category from another. For this we have to train the system with some past data. Having a good quality data to train the system will certainly achieve in higher accuracy. To train with this past data, the data need to be categorized in some class manually by some experts on the field. This manual labeling by human often prone to error, mistakes and results in high cost and very limited quantity high quality train data [4]. Moreover, it is difficult to trace back whether the data has been labeled correctly or not [5]. This is where the idea of automatic labeling the data was generated. This automatic labeling [**?**] is done by unsupervised learning. Community detection algorithm in network science can be a good deal to solve this problem. Community detection helps in data clustering without depending on user parameters by utilizing modularity [6]. The use of community detection specifically in text classification to label training data has been able to improve the performance of text classification using machine learning, namely 3.75% in SVM and 2.68% in Random Forest. Once we get the labeled train data, the next step is to classify them into certain class. In recent years deep learning has become center of attention for various fields such as image processing, NLP and computer vision [7]. Deep learning can take advantage of large datasets to achieve a higher level of accuracy than other classification techniques [8]. There are several models based on deep learning in to solve classification problem - Long Short Term Memory(LSTM), Bidirectional Encoder Representation from Transformers(BERT) and XLNet. XLNet is one of the newer model and has received a state-of-the-art predicate for 18 NLP tasks. This study includes automatic labeling the text dataset using community detection with Infomap and then making a text classification model using XLNet. In addition, research uses bigram tokenization and explains that the opportunity to improve performance can be done by using a higher n-gram because it has a strong role in improving syntactic performance and can capture broader contextual information. Furthermore, we explore the effect of using Bigram and Trigram to their performance in the experimental model.

2 Related Work

Infomap was first introduced in 2008 to uderstand the multipartite organization of large-scale biological and social system [9]. Infomap algorithm is able to group graph with good quality. XLNet

is a pre-training method that uses permutation language modeling objectives to combine the advantages of Auto-regressive(AR) and Auto-encoder(AE) methods and was introduced in 2019 [10]. XLNet has become a state-of-the art task range from NLP and excels at 18 NLP tasks [11].

3 Methodology of Proposed System

The whole process was divided into 2 stages namely

1. The automatic labeling stage of training data using Infomap and
2. The text classification stage using the XLNet model.

3.1 Dataset

The data used in this study was taken from TREC-QA, one of the open domain question-answer datasets that are often used in research [12]. The dataset consists of 2 columns. The columns are named Questions and Class Lables. There 6 class labels, namely ABBR(Abbreviation), DESC(Description), ENTY(Entity), HUM(Human), LOC(Location), NUM(Numeric Value). The training data was collected from three sources and labeled manually, while the test data si 500 questions.

Table 1: Training Data

Layers	**Parameters**
What fowl grabs the spotlight after the Chinese Year of the Monkey ?	1(Entity)
How can I find a list of celebrities ' real names ?	0(Description)
What sprawling U.S. state boasts the most airports ?	5(Location)
How many Jews were executed in concentration camps during WWII ?	4(Number)
What does the abbreviation AIDS stand for ?.	2(Abbreviation)
Name 7 famous martyrs .	1(Human)

3.2 Automatic Labeling of Training Data

The flowchart of Labeling the training data automatically is given below in Fig 1.

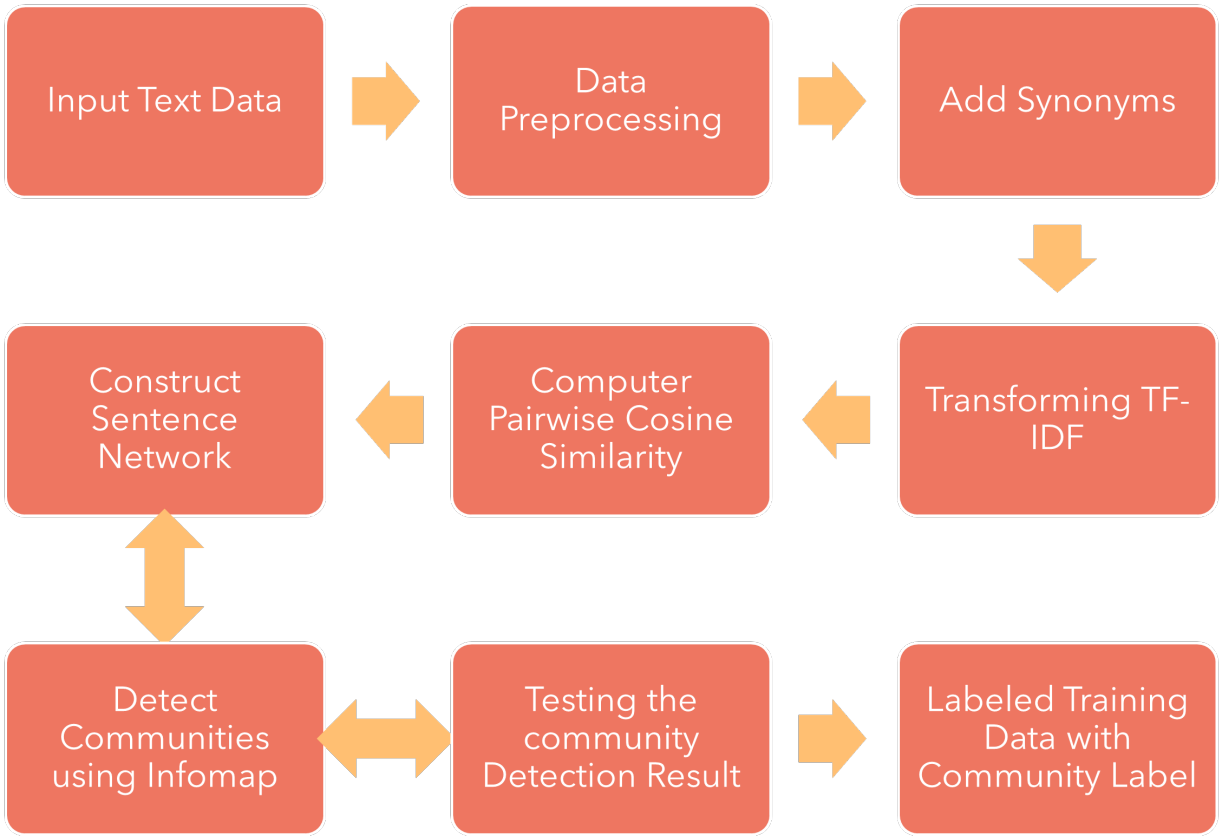


Figure 1: Automatic Labeling of Training data Stage

3.2.1 Preprocessing

Preprocessing steps include expanding Contractions or restoring contractions to the original word order, deleting URLs, removing punctuation marks, deleting stopwords, and stemming.

3.2.2 Addition of Snonyms

The addition of synonyms aims to increase the variety of words and is expected to improve the understanding of the resulting classification model for different words but describe the same meaning. Adding synonyms makes use of the NLTK-wordnet in Python to add synonyms in different contexts.

3.2.3 TF-IDF Transformation

Each sentence is transformed into a vector representation by transforming TF-IDF calculation. It considers the word or Term Frequency(TF) in the document as well as how unique or infrequent(IDF) a word is in the corpus. It gives a higher score for unique words and devalues common words [13]. The TF-IDF transformation consisted of two types of Ngrams, namely bigram and trigram. One basic form of TF-IDF could be denoted by the weight value of TF-IDF $w_{t,d}$ for word t in the document d in form of an equation:

$$w_{t,d} = tf_{t,d} * x * log_{10}(\frac{N}{df_t})$$

Where,
$tf_{t,d}$ is the frequency of the word t in the document d
N is the number of document in the collection
$df_{t,d}$ is the number of documents where the word t appears

3.2.4 Calculation of Cosine Similarity

When the data has been represented in a vector using TF-IDF, the equation between the two vectors in each training data is calculated using cosine similarity. Cosine similarity has a value range of 0 to 1 where 0 means no similarity between sentences and 1 mean the sentences are equal. This step is achieved using Closeness Matrix.The result of cosine similarity calculation is a matrix of closeness between sentence vectors with a cosine similarity value of 1 for each sentence vector compared to itself as shown in Fig 2.

$$
\begin{array}{c}
 \\
d1 \\
d2 \\
\vdots \\
d5452
\end{array}
\begin{bmatrix}
d1 & d2 & \cdots & d5452 \\
SIM_C\left(\overrightarrow{t_{d1}}, \overrightarrow{t_{d1}}\right) & SIM_C\left(\overrightarrow{t_{d1}}, \overrightarrow{t_{d2}}\right) & \cdots & SIM_C\left(\overrightarrow{t_{d1}}, \overrightarrow{t_{d5452}}\right) \\
SIM_C\left(\overrightarrow{t_{d2}}, \overrightarrow{t_{d1}}\right) & SIM_C\left(\overrightarrow{t_{d2}}, \overrightarrow{t_{d2}}\right) & \cdots & SIM_C\left(\overrightarrow{t_{d2}}, \overrightarrow{t_{d5452}}\right) \\
\vdots & \vdots & \ddots & \vdots \\
SIM_C\left(\overrightarrow{t_{d5452}}, \overrightarrow{t_{d1}}\right) & SIM_C\left(\overrightarrow{t_{d5452}}, \overrightarrow{t_{d2}}\right) & \cdots & SIM_C\left(\overrightarrow{t_{d5452}}, \overrightarrow{t_{d5452}}\right)
\end{bmatrix}
$$

Figure 2: Closeness Matrix

3.2.5 Formation of Sentence Network

The sentence network is formed from the nodes of each sentence in the training data and the weights obtained from the calculation of the proximity matrix that has been carried out. This sentence network formation utilizes the networkx [14] library in Python.

3.2.6 Infomap Community Detection

The first step is that each community will be coded according to the community level, and then in each community the node will be coded based on the node level. By integrating these two aspects, the coding of one node is confirmed by community-coding and node coding. Therefore, the problem of community detection can be replaced by the problem of coding compression, which makes the length of the encoding the shortest. The best method for community detection would be to obtain the maximum amount of encoding compression.

3.2.7 Testing of Automatic Labeling Result

One test can be carried out by using Newman Girvan Modularity which has a value range of 0 to 1 [15]. This can be calculated from following equation:

$$Q(S) = \frac{1}{m} \sum_{c \varepsilon S} (m_s - \frac{(2m_s + l_s)}{4m})$$

Where,
m is total number of edges in graph
m_s is number of edges in community
l_s is number of edges from node S to nodes outside of S

The second test is by calculating the average value of the class split and class merge. Class split occurs when a manually labeled class is mapped in many communities due to semantic similarity. In contrast, a merge class is when several or many manually labeled classes are detected in the same community [4]. We attempt to minimize both values because if the class split is high, many communities are formed. However, if the merge class has a high value, the results of community detection are considered not grouped correctly. Based on both values, the average is calculated to be taken into consideration in determining the community detection results.

3.3 Text Classification

The pretrained XLNet model is pre-trained on BooksCorpus, English Wikipedia, Giga5, ClueWeb and Common Crawl. Classification is done by dividing the training data into 80% as training and 20% as validation data. Text classification is implemented based on a model that has passed the pre-train process with the XLNet model for sequence classification base cased. This model is then fine-tuned to better understand the training data that is owned and the number of epochs is set, namely 10, learning rate 3e-5, batch size in the range 1-5, max_len 64, and AdamW optimizer. 3

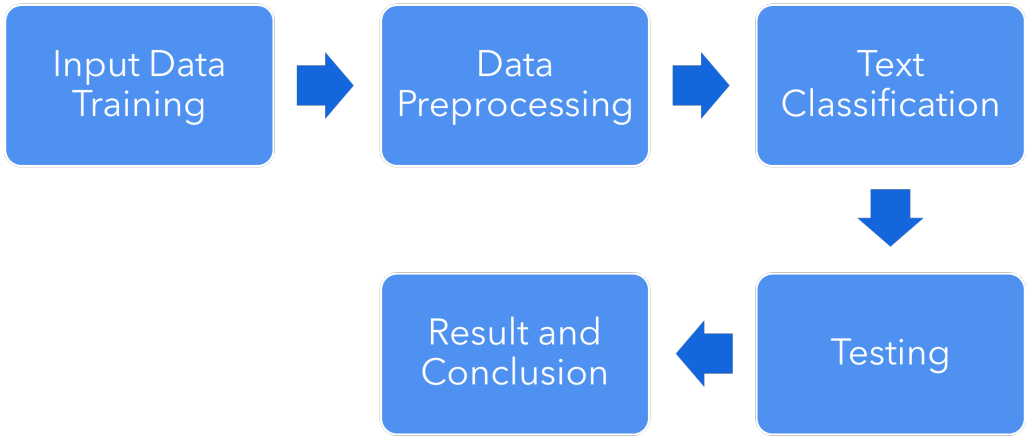


Figure 3: Automatic Labeling of Training data Stage

4 Experiment and Analysis

The number of nodes formed is 5452 nodes with a number of edges of 1,232,614 edges. This network of sentences then enters the community detection process, which is carried out using the cdlib library. Community detection will produce training data that has been labeled as a community, the number of communities formed, modularity, and the value of class split and class merge.After performing community detection using Infomap we were left with dataset having the form below in bigram and trigram.

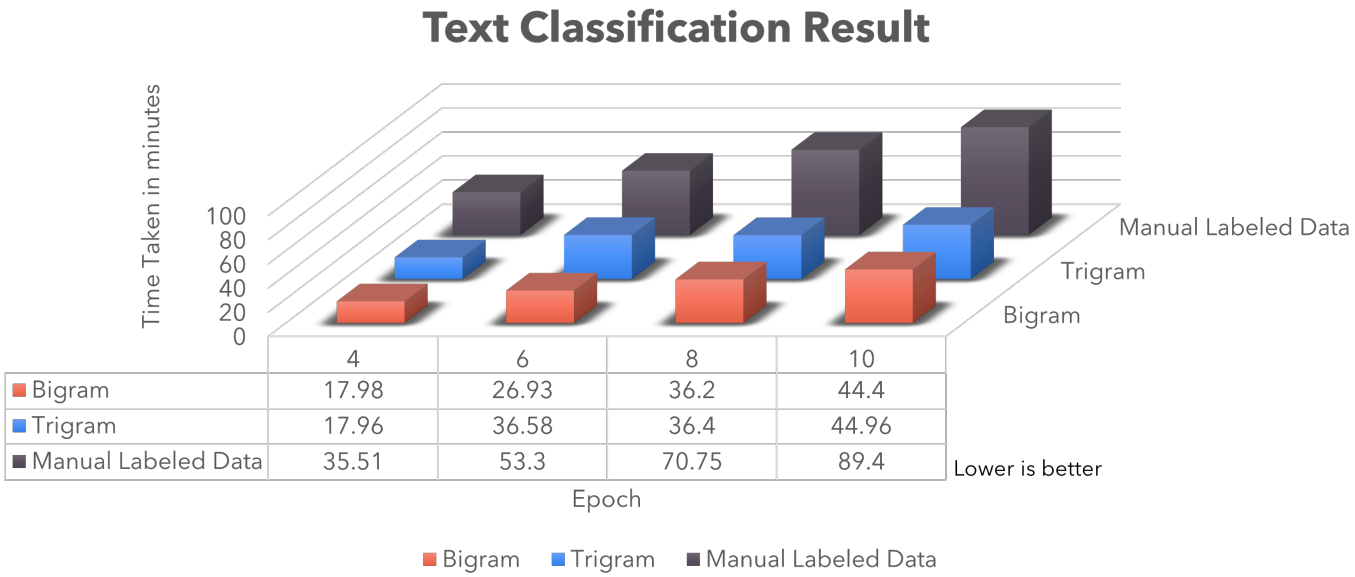
Table 2: Result of Automatic Labeling

Ngram	Number of Community Formed	Modularity	Class Split	Class Merge
Bigram	1007	0.559	365.833	2.171
Trigram	706	0.5793	309.330	2.626

4.1 Comparison between Automatic Labeled Data vs Manual Labeled Data in Text Classification

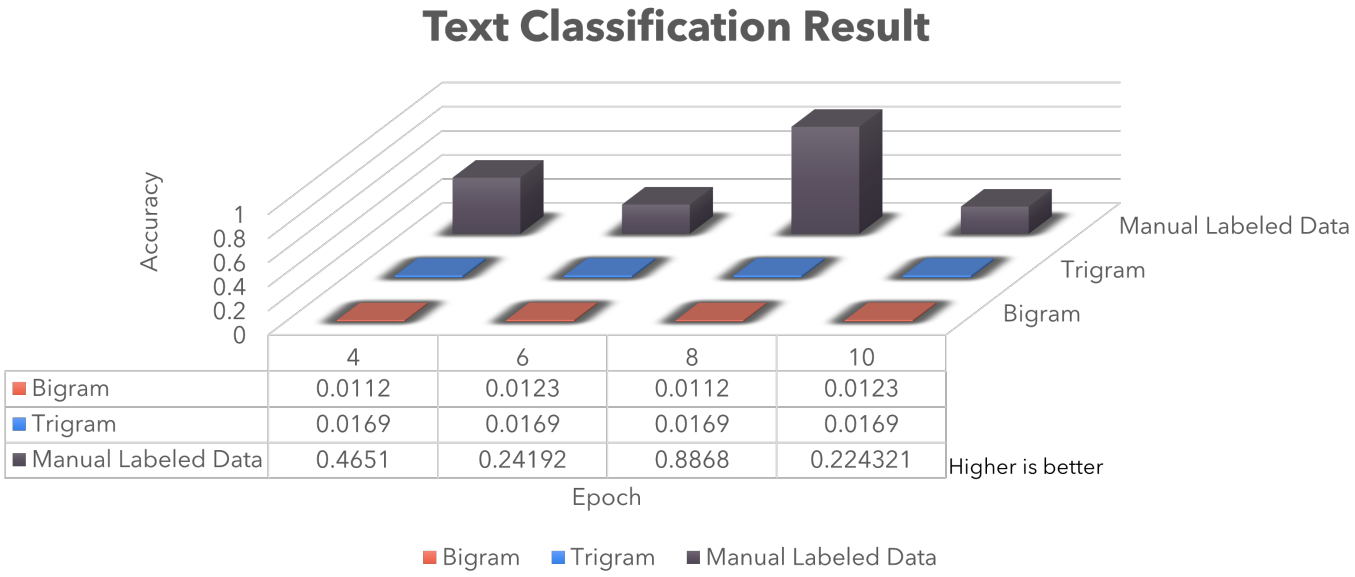
The results of automatic labeling training data using Bigram and Trigram are used in text classification using the XLNet model as previously explained. The hyperparameters used are max len

64, batch size 2, Adamw optimizer, and learning rate 3e-5. The results of recording experiments carried out can be seen in the following figures 4 5 6



	4	6	8	10
Bigram	17.98	26.93	36.2	44.4
Trigram	17.96	36.58	36.4	44.96
Manual Labeled Data	35.51	53.3	70.75	89.4

Figure 4: Text Classification Result(Time vs Epoch)



	4	6	8	10
Bigram	0.0112	0.0123	0.0112	0.0123
Trigram	0.0169	0.0169	0.0169	0.0169
Manual Labeled Data	0.4651	0.24192	0.8868	0.224321

Figure 5: Text Classification Result(Accuracy vs Epoch)

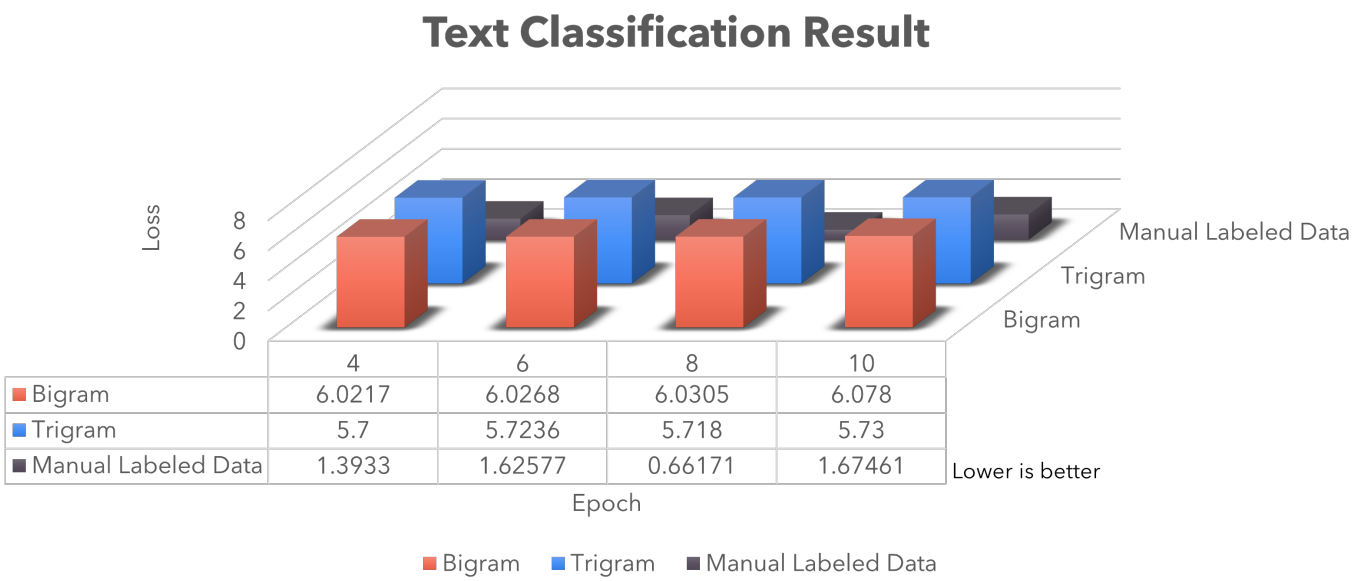


Figure 6: Text Classification Result(Loss vs Epoch)

5 Discussion

The experimental results according to the scenario cannot outperform the baseline, namely text classification with manually labeled training data. Things that affect the results can be analyzed in the following points.

1. Threshold: The experimental results according to the scenario cannot outperform the baseline were caused by the weight of the cosine similarity used in the construction of the edge of the sentence network to form an edge of 1,232,614 for 5452 nodes. This also results in a high class split and class merge values. If the class split is high, it affects the effectiveness of the classification because more and more communities are formed with only a few members in each community, even only one node. On the other hand, if the merge class is high, the community detection results cannot group nodes properly because it maps many classes from manual labels in the same community. Therefore, the value of class split and class merge must be minimized for optimal community detection results. The experimental threshold is intended to limit the edge that has a weight below the threshold. In this experiment, the threshold test was carried out in a value range of 0.0 to 0.6 with the following results.

Table 3: Threshold Test Result Bigram

Thresold	Number of Community Formed	Modularity	Class Split	Class Merge
0.0	1007	0.559	365.833	2.171
0.1	311	0.69251	161.1666	3.09
0.2	194	0.827	101.833	3.118
0.3	350	0.8009	210	3.582
0.4	348	0.624	134.834	2.3074
0.5	97	0.425	35.5	2.134
0.6	22	0.2698	13.666	3.4545

Table 4: Threshold Test Result Trigram

Thresold	Number of Community Formed	Modularity	Class Split	Class Merge
0.0	706	0.5793	309.33	2.626
0.1	283	0.7637	144.334	3.03886
0.2	447	0.82031	286	3.8255
0.3	789	0.6225	236	1.7872
0.4	83	0.4702	42.664	3,012
0.5	55	0.2837	21	2.1818
0.6	12	0.168	9.666	4.333

Here we can see the class split and class merge value is lowest around threshold 0.5 to 0.6 for bigram and threshold 0.3 to 0.4 for trigram.

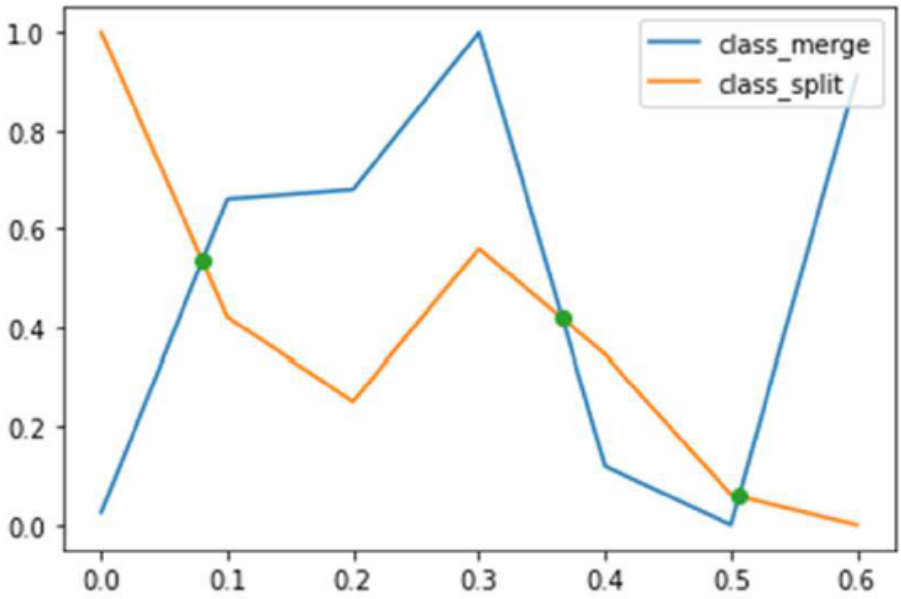


Figure 7: Threshold Normalization in Bigram

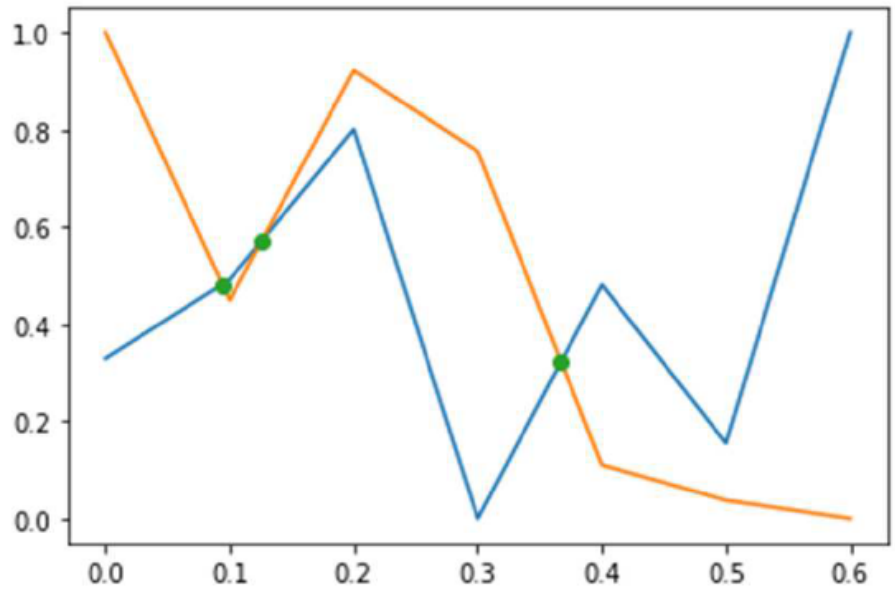


Figure 8: Threshold Normalization in Trigram

That is why we chose an optimal threshold for bigram and trigram Fig 7 8.

Let us now see the same comparison between automatic labeled data and manually labeled data in the following figures 9 10 11 .

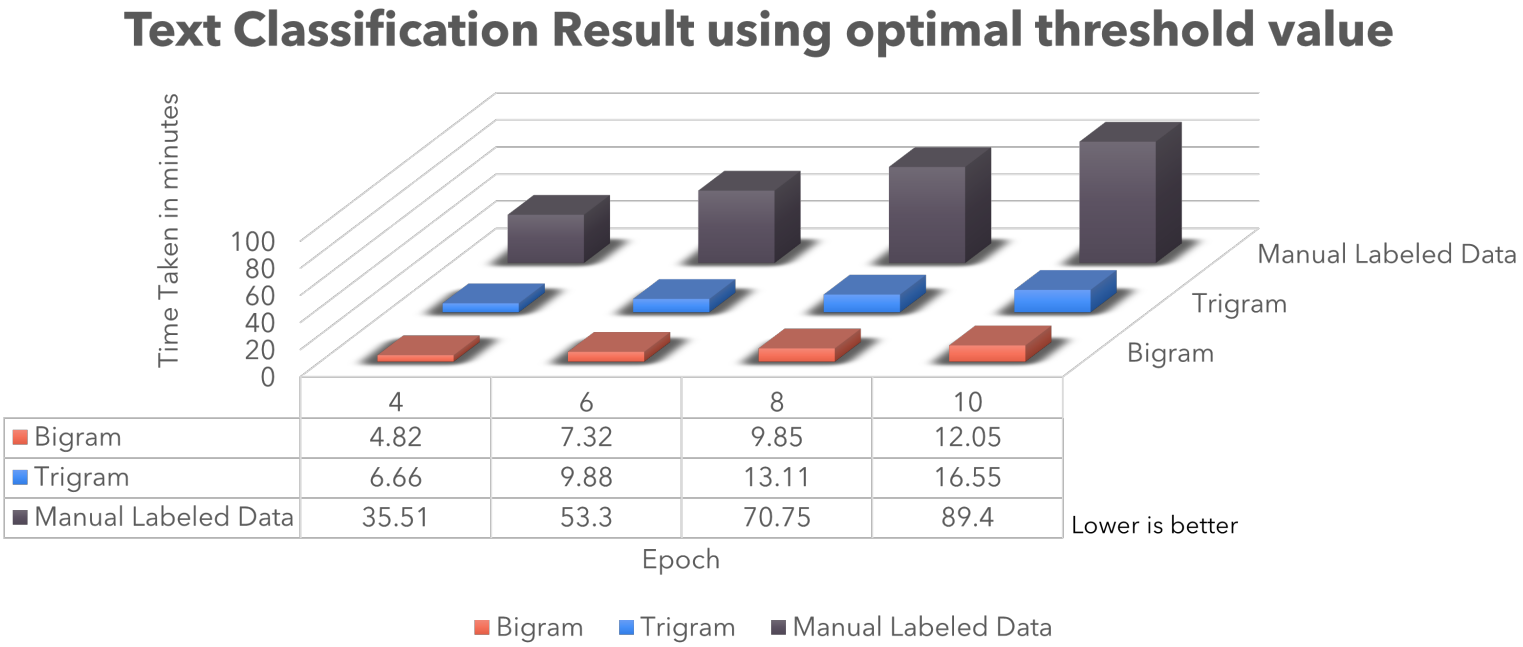


Figure 9: Text Classification Result using optimal Threshold(Loss vs Epoch)

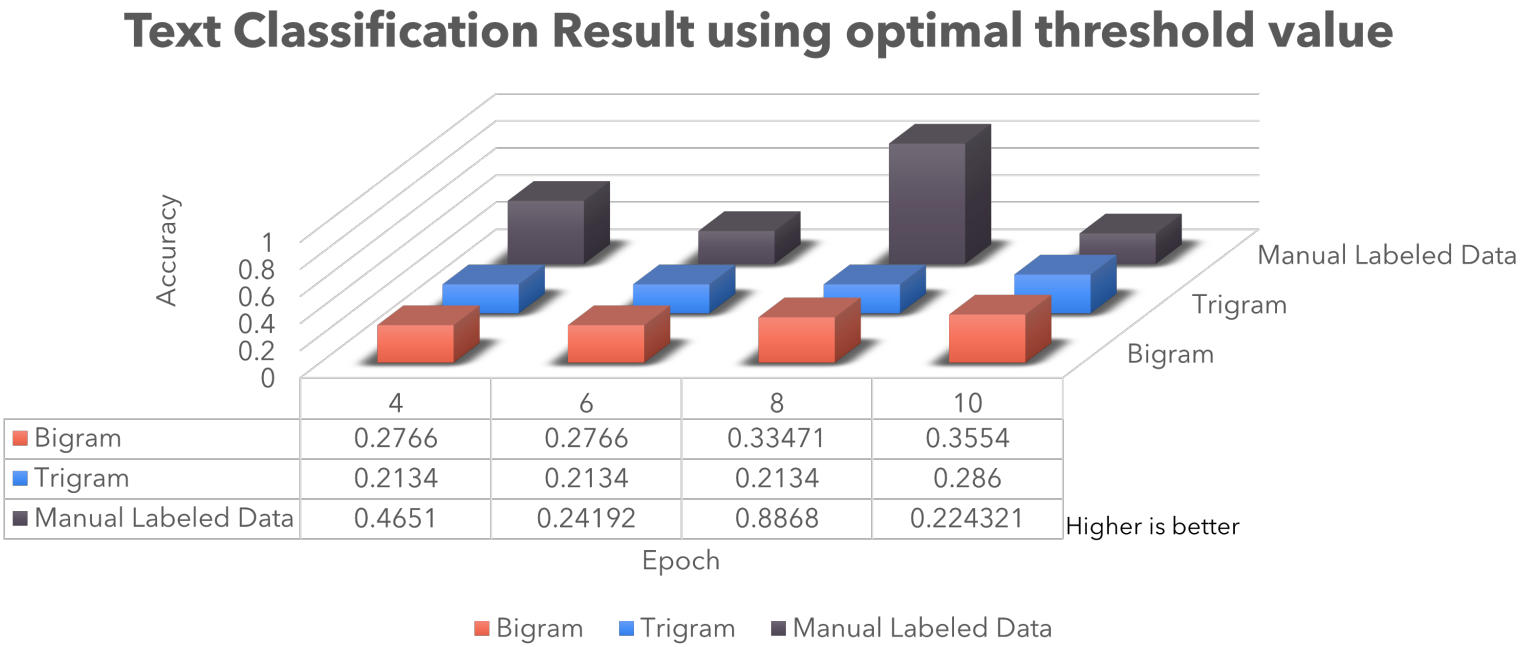


Figure 10: Text Classification Result using optimal Threshold(Loss vs Epoch)

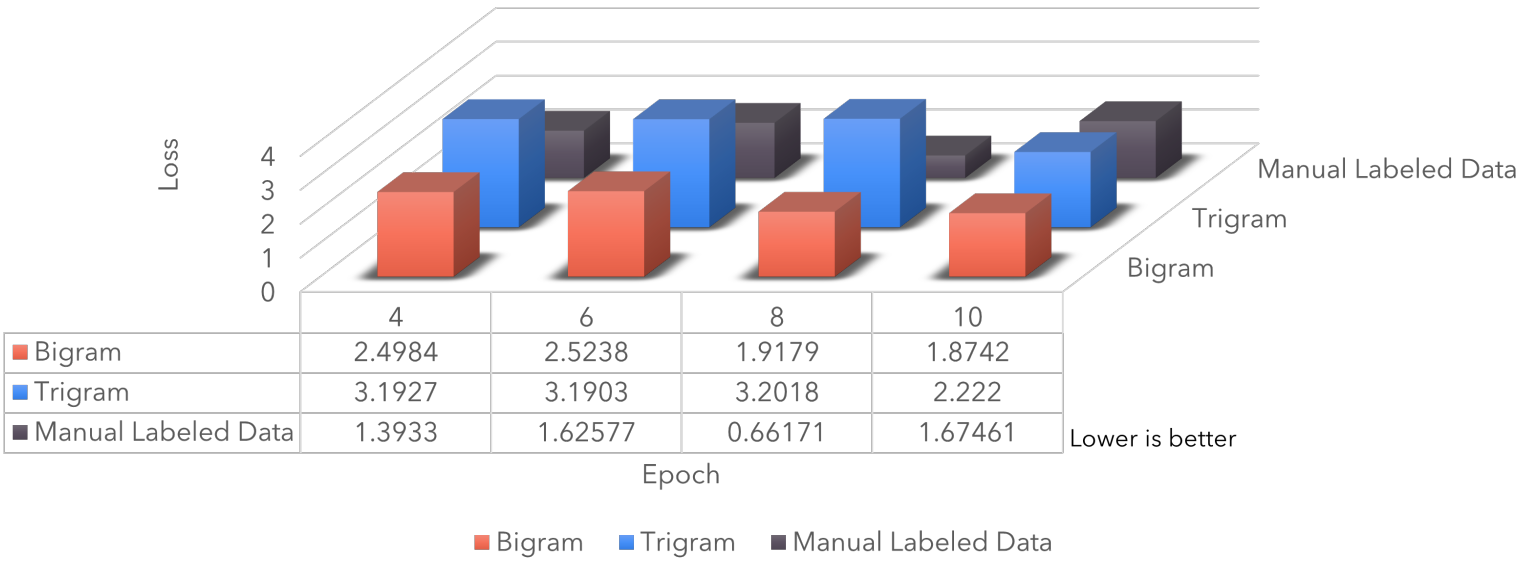


Figure 11: Text Classification Result using optimal Threshold(Loss vs Epoch)

2. Nodes Without Community: In automatic labeling there are node those do not have any community.When using the optimum threshold, it is found that training data that has a community is 1446 on bigram and 2010 data on the trigram of the total training data of 5452. This means that 73.4data is on bigram, and 63.13% of the training data is on the trigram has no community.

3. Class Split and Class Merge: As there are large amount of data that do not have community, there are also some nodes that belongs to multiple community due to high value of Class Split and Class Merge. Some of them even belongs to 6 manually labeled classes simultaneously.

4. Bigram and Trigram: Comparison of the effect of bigram and trigram on the experiments conducted shows the general results that the addition of trigrams in training data increases accuracy. However, when viewed after the optimal threshold experiment was carried out, the accuracy of the trigram data is not superior to the accuracy of Bigram data. This is due to the results of determining the different threshold cutoff points in the two data. Bigram data has the optimal cutoff point at 0.506382 and while in trigram data, the intersection point is at 0.367058.

5. Use of Keywords: The use of available keywords from the training data will also increase the closeness of sentences that have the same keywords.

6. Use of Synonyms: The addition of synonyms aims to have words with the same context to increase the effect of closeness between sentences with the same context. Experiments were carried out with scenarios without synonyms and using keywords to determine the effect of adding synonyms. This affects all epoch count of bigram data greatly. This is because of high class split in bigram data with more community.

6 Conclusion

From this study we can see that automatic labeling can not surpass the accuracy of manual labeling yet it can greatly improve time consumption in high quantity of data. The accuracy depends on various factors like threshold value, n value in ngram data, class split and class merge. It was also discussed that the accuracy can be greatly improved by tweaking this parameters on particular problems. Moreover using community detection labeling allows us to use data that is not community based. That is why a significant amount of data was lost or we can say there were significant amount of nodes without any community. This also affects the accuracy score of this study.

References

[1] Xlnet. [Online]. Available: https://github.com/zihangdai/xlnet

[2] R. Yu, "Reform in the teaching model of english writing in the big data era," pp. 252–259, 2020.

[3] S. Yilmaz and S. Toklu, "A deep learning analysis on question classification task using word2vec representations," *Neural Computing and Applications*, vol. 32, no. 7, pp. 2909–2928, 2020.

[4] F. Prior, J. Almeida, P. Kathiravelu, T. Kurc, K. Smith, T. J. Fitzgerald, and J. Saltz, "Open access image repositories: high-quality data to enable machine learning research," *Clinical radiology*, vol. 75, no. 1, pp. 7–12, 2020.

[5] M. Kim and H. Sayama, "The power of communities: A text classification model with automated labeling process using network community detection," pp. 231–243, 2020.

[6] E. K. Mikhina and V. I. Trifalenkov, "Text clustering as graph community detection," *Procedia computer science*, vol. 123, pp. 271–277, 2018.

[7] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Computing*, vol. 22, no. 1, pp. 949–961, 2019.

[8] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," pp. 372–387, 2016.

[9] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proceedings of the national academy of sciences*, vol. 105, no. 4, pp. 1118–1123, 2008.

[10] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *Advances in neural information processing systems*, vol. 32, 2019.

[11] K. Nagda, A. Mukherjee, M. Shah, P. Mulchandani, and L. Kurup, "Ascent of pre-trained state-of-the-art language models," pp. 269–280, 2020.

[12] (Accessed: 26. 6, 2022) Trec-qa, 'question-answer dataset', 2004. [Online]. Available: https://trec.nist.gov/data/qa/t2004_qadata.html

[13] Y. Zhang, Y. Zhou, and J. Yao, "Feature extraction with tf-idf and game-theoretic shadowed sets," pp. 722–733, 2020.

[14] Networkx library in python. [Online]. Available: https://networkx.org/

[15] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review E*, vol. 69, no. 2, p. 026113, 2004.