FIT5212 Data analysis for semi-structured data -S1 2021

Assignment 1

NAME: PRAJYOT NAGRALE
STUDENT ID: 31132324

Part 1: Text Classification

The content of the data is from computer science, mathematics or physics categories. The Abstract variable contains textual data which is an independent variable. The InfoTheory, CompVis and Math are dependant Boolean variable classified as 1 and 0.

1. InfoTheory: (Information Theory)
2. CompVis: (Computer Vision)
3. Math: (Mathematics)

I'm first dividing my training data size into 1000 and all data size which will help me to analyse the importance of the data size in the evaluation.

1. Data size = 1000
2. Data size = all

There any different text pre-processing using in machine learning like regular expression, NLP and nltk for tokenization, stop words removal, lowercase, stemming, lemmatization, numberers removal, punctuations removal, single character removal, ngram and frequency removal. I'm then using two different text pre-processing steps on the Abstract variable.

1. Tokenizer(NLTK) + Lowercase + Snowball stemming
2. Tokenizer(NLTK) + Stop words removal + WordNet Lemmatization

As our prediction variables are supervised classification I'm using two different machine learning algorithms to predict the dependent variables.

1. Linear SVC (linear support vector classifier)
2. RNN (recurrent neural network

Finally, to evaluate the algorithms on all the above variations.

1. F1 score
2. Precision
3. Recall
4. Precision-recall curve

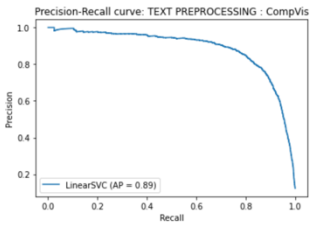
These evaluation matrices will help me to analyse the different variation in predicting the data.

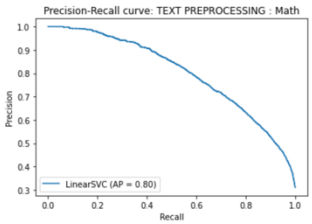
Evaluation matrices of all the variations.

Dataset = 1000						Dataset = All					
Text Preprocessing = 1			Text Preprocessing = 2			Text Preprocessing = 1			Text Preprocessing = 2		
InfoTheory	Linear SVC	RNN	InfoTheory	Linear SVC	RNN	InfoTheory	Linear SVC	RNN	InfoTheory	Linear SVC	RNN
F1 score	0.8512	0.459	F1 score	0.8432	0.4506	F1 score	0.9172	0.4504	F1 score	0.9169	0.4503
Precision	0.9203	0.4897	Precision	0.9232	0.4812	Precision	0.9375	0.4816	Precision	0.9353	0.4756
Recall	0.8096	0.4989	Recall	0.7982	0.4997	Recall	0.8998	0.4998	Recall	0.901	0.4997
Text Preprocessing = 1			Text Preprocessing = 2			Text Preprocessing = 1			Text Preprocessing = 2		
CompVis	Linear SVC	RNN	CompVis	Linear SVC	RNN	CompVis	Linear SVC	RNN	CompVis	Linear SVC	RNN
F1 score	0.7919	0.4724	F1 score	0.7588	0.4736	F1 score	0.9161	0.4725	F1 score	0.9152	0.472
Precision	0.9423	0.4953	Precision	0.9419	0.5565	Precision	0.9562	0.5041	Precision	0.9567	0.4937
Recall	0.7275	0.4999	Recall	0.6945	0.5008	Recall	0.884	0.5	Recall	0.8822	0.4999
Text Preprocessing = 1			Text Preprocessing = 2			Text Preprocessing = 1			Text Preprocessing = 2		
Math	Linear SVC	RNN	Math	Linear SVC	RNN	Math	Linear SVC	RNN	Math	Linear SVC	RNN
F1 score	0.7758	0.412	F1 score	0.7715	0.4119	F1 score	0.8454	0.412	F1 score	0.844	0.412
Precision	0.8198	0.4241	Precision	0.8276	0.4287	Precision	0.855	0.4564	Precision	0.8521	0.4367
Recall	0.7549	0.4989	Recall	0.7482	0.4992	Recall	0.8374	0.4997	Recall	0.837	0.4994

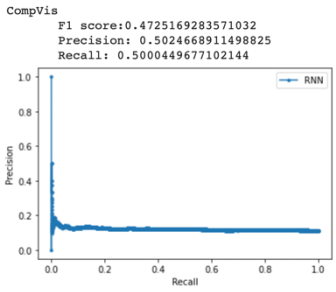
The above tabular data shows the different evaluation matrices for all the variation. We can see that the RNN model is not able to give a good prediction when compared with Linear SVC. We can see that Linear SVC has around 0.8 precision, recall and f1 score in both the text pre-processing, whereas the RNN model has around 0.4 scores. This shows that their Linear SVC around 40% better model than RNN. We can also see the precision-recall curve for Linear SVC is better than RNN. Linear SVC is Linear Support Vector Classification. RNN is a recurrent neural network. The reason why Linear SVC is performing better than RNN is that RNN is underfitting. RNN is a neural network that updates weights in backpropagation and takes a very long time to run. This update of weight happens by the number of epochs time given in the parameter. As I've given epoch as 5 which is updating the weight only 5 times, due to which it's underfitting and giving us a bad evaluation score. If I increase the number of the epoch, which will increase the time for running the code and will increase the RNN to predict better.

We are also able to see that first Text Pre-processing is giving us a little better score than second text pre-processing. For example, in the 1000 dataset for CompVis task the F1, precision and recall score is 0.79, 0.94, 0.73 respectively for first text pre-processing, whereas for second pre-processing in the same dataset and task the scores are 0.76, 0.94, 0.69 respectively. In most scores, we can see that the first text pre-processing is better than the second one, but for some score second is better with a very minute percentage. For example, in the 1000 dataset for the Math task, the precision score for first text processing is 0.8198, and for second text pre-processing the score is 0.8276, which is 0.078 less. For most test score first text pre-processing is better than the second one. So, we can say that the first text pre-processing is better than the second one text pre-processing.





From the precision-recall curve we are able to see that the model is predicting good for some classifier and not for other classifiers.



We are also able to see that the f1, precision, recall score gets better when the size of the data increases. When compared all the evaluation scores for 1000 and full train data size, we can see that almost all the scores are increased for full data size in both the text pre-processing. For example, in 1000 data size first text pre-processing in for InoTheory task the f1, precision and recall score is 0.8512, 0.9203, 0.8096 respectively, whereas for all data size it is 0.9172, 9.9375 and 0.8998 respectively. By looking at this we can say that the model gets better and better when the data size increases.

Part 2: Topic Modelling

This part contains topic modelling which helps to discover the abstract or topic that occurs in the collection of documents. We are using genism. models.LdaModel() function helps us to build a topic per document model and words per topic model. First, we are dividing the Abstract data from full articles into

1. First 1000 articles
2. First 20,000 articles

We are also using two different texts pre-processing and topic selection step:

1. Text Pre-processing 1:
   Tokenizer(nltk) + remove numbers + remove single character words + snowball stemming + bigram + k = 10 number of topics
2. Text Pre-processing 2:
   Tokenizer(nltk) + lowercase + stop words removal + removing non alphabetic characters + wordnet Lemmatization + k = 25 number of topics

I used pyLDAvis.gensim.prepare() function for interactive visualization of words significant on topics. After using this function I was able to see that:

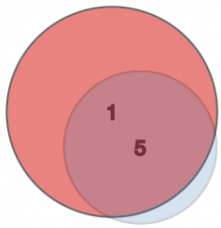
1. Text pre-processing 1 and first 1000 articles:

All the grouping of the topics is segregated differently except bubble 2, 4 and bubble 6, 3. Bubble 2, 4 and bubble 6, 3 are overlapping over each other. This tells that some words like a model, corpus, word and many more are similar between bubbles 2 and 4. Most of the words are not able to properly signify anything about the topic. As it is trained on the very low data set, I'm not properly able to see proper come up with the head for the words in each bubble. But when I adjusted λ

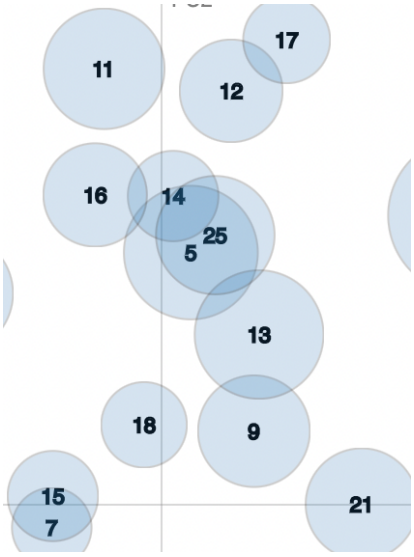
And checked for bubble 10. To know more about the topic I also read the first few lines of the article "An Extended Clustering Algorithm for …". I was able to see that this topic is talking about language encoding.

2. Text pre-processing 1 and first 20,000 articles:

When I checked the same text pre-processing in the first 20,000 articles I was more clearly able to distinguish the topics from the words. Like bubble 4 was talking about networks. I was also able to see that bubble 1 and bubble 5 are mostly similar topic and are about bound algorithm. Most of the words are also similar in both of them with some bigrams like lower_bound. This can tell us that we can reduce a topic selection number of k by one. I'm also able to see a lot of stop words in the topics. These stop words are making it hard to distinguish information about the topics. I was also able to see some words like pars which was stemmed but does not give any meaning for the topics. I was also able to see some punctuations as words. So for the second text pre-processing I have removed stop words and used word net lemmatization from nltk.



3. Text pre-processing 2 and first 1000 articles:



For text pre-processing of this task, I've also taken only the alphabetic characters lowercased all the words and increased the number of the topic to 25. When used the model for the first 1000 articles, I was able to get more detail about each topic. The words were able to help us to more easily distinguish the topics for each bubble. Like, Bubble 3 can be detailed information about the English language. Bubble 2 can be about text analysis. Bubble 6 is machine learning algorithms. But still, the grouping about topics are more gathered together. Like bubble 16, 4, 5 and 25 are very close to each other.

4. Text pre-processing 2 and first 20,000 articles:

To overcome this challenge we can use this text pre-processing on more articles. This time I used the same text pre-processing step in the first 20,000 articles. After doing this I was more easily able to distinguish the topics. Like, Bubble 21 is talking about social being. Bubble 8 is talking about power transmission. Bubble 24 is talking about machine learning and many more. By using lemmatization I was also able to know the word pars was talking about parsing.

I further did keywords topics percentage contribution. To understand the importance of words in that topic for the significance of the article. I was able to see that topic number 8 has a 77% topic contribution for the article "Incremental Parser Generation for Tree Adjoin...".

I was able to analyse from the above steps the significance of text pre-processing and article size for topic modelling.

Thank You