

20NewsGroup Project Report

Singularity System

By Wei Yang

Table of Contents

Summary 1

1. Introduction 1

2. Project Framework 1

3. Data Description and Preprocessing 2

4. Best Performance Summarization 5

5. Experimental Evaluation 6

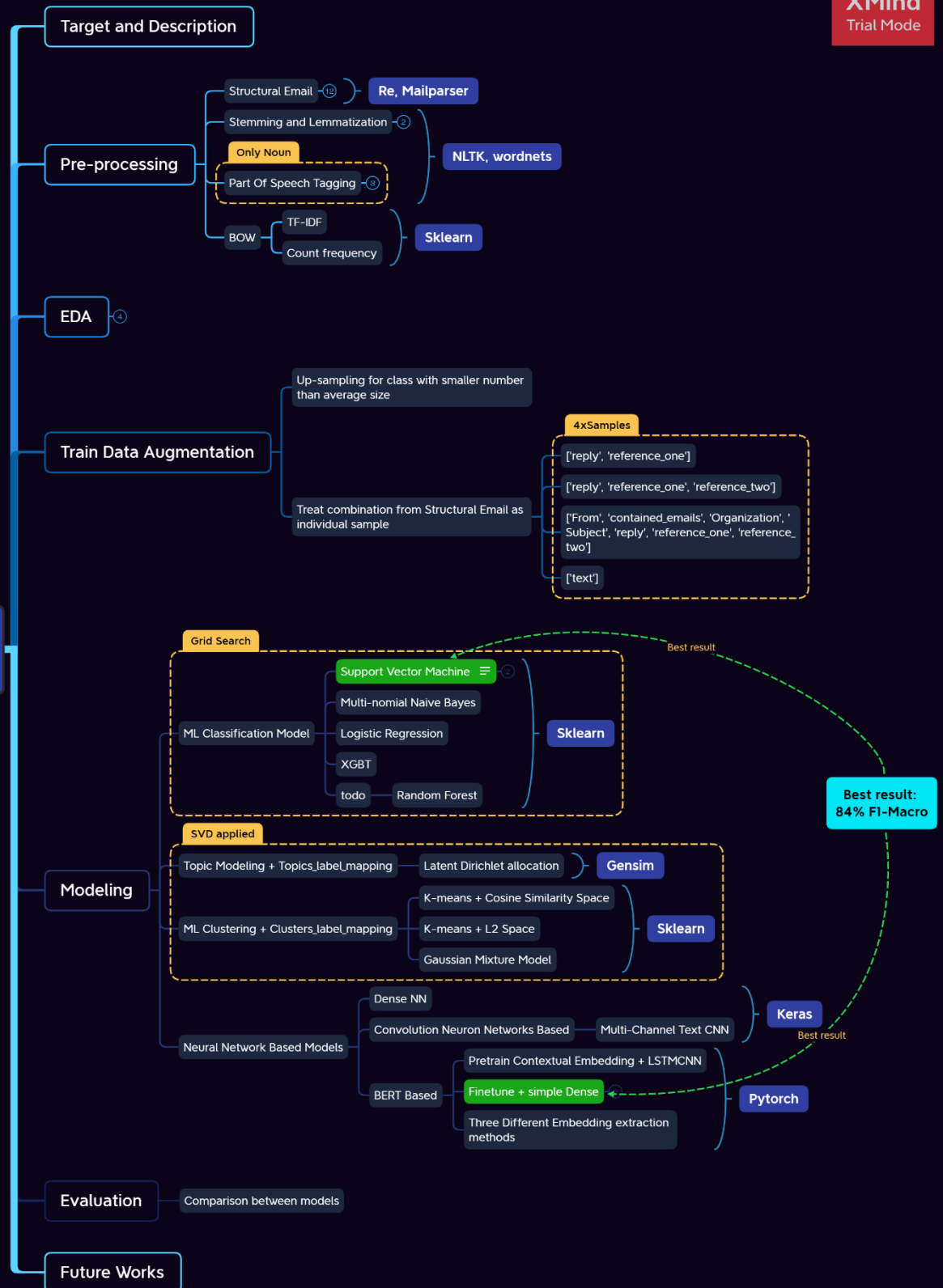
6. Alternative Methods & Future Work..... 11

References 12

Project Framework

XMind
Trial Mode

Singularity System Project Report



Summary

After exploration on machine learning models, probability topic modeling, and deep learning models, this project make a comparison between 10 different models. The best preprocessing procedure are also explored for better performance on each model. With exploring optional setting related to data augmentation, up-sampling for small samples, lemmatization, stemming, and feature selection, this project use grid search, cross validation to find the best setting and preprocessing path for machine learning models. In addition, the clustering, topic modeling, Deep Neural Networks and multi-channel text CNN are implemented and explored in this project. However, the performance is not that acceptable in such models. At last, this project also explores the performance on finetune and pretrain BERT stacked with dense neural networks. The best performance has 0.84 macro F1 average score, which is obtained by both SVM and finetune BERT with a specific preprocessed input.

In the future experiment on similar dataset, considering the exploration cost, SVM should be tested at first to know the baseline of model's performance in the dataset. It is well implemented by Sklearn and can be quick trained with sparse matrix form input, which will reduce the risk of OOM as well. If the further exploration needed, the BERT related model should be explored. With more detail hyper-parameters tuning tricks and experience, the BERT model could have even better performance than 84% in this dataset. Furthermore, the finetuned model can also be transferred in other similar dataset to finish the classification task. For future reference, all the utils are wrapped and upload into GitHub¹ repository.

1. Introduction

This project is mainly focus on exploration on Machine learning model and Deep learning model performance on the 20 News Group (20NG) dataset. Besides, with organized the related tools in Natural Language Processing (NLP) field, the related python packages are wrapped as a higher level utils function in the GitHub for personal future reference, such as the utils for preprocessing, clustering, classification, neural networks and Bidirectional Encoder Representations from Transformers (BERT). With such utils, similar tasks could be easily implemented and tested in the future. With the wrapped utils, the computational expensive training task related to Neural Networks were trained by GPU in Google Colab. The exploration of model is implemented and demonstrated in Jupyter notebook.

2. Project Framework

To make the accurate prediction on the 20 labeled classes, this project explores the following models:

- Machine learning based models: K-means, Logistic Regression, Multinomial Naive Bayes, Support Vector Machine (SVM), gradient boost tree.
- Topic models: Latent Dirichlet allocation (LDA).
- Deep Learning based models: Fully connected Deep Neural Networks, Multi-channel Convolutional Neural Network, pretrained BERT with LSTM-CNN, finetune BERT with two linear dense layers.

Other parts of the project are showed as the previous mind map page and will be discussed in detail in the Experimental Evaluation part of this report.

¹ <https://github.com/InscribeDeeper/Text-Classification>

3. Data Description and Preprocessing

3.1 Data Description

The dataset in this project contains 11,083 training samples, and 7,761 test samples. In this dataset, the 1060 and 770 indexes of file are duplicated in training and test samples. For future sample checking and further analysis, the global unique index is created along with the dataset. The graphs below show the basic summary statistics of the dataset and the basic content in one document.

label	count_train	percentage_train	count_test	percentage_test
0 alt.atheism	480	0.043310	319	0.041103
1 comp.graphics	584	0.052693	389	0.050122
2 comp.os.ms-windows.misc	591	0.053325	394	0.050767
3 comp.sys.ibm.pc.hardware	590	0.053235	392	0.050509
4 comp.sys.mac.hardware	578	0.052152	385	0.049607
5 comp.windows.x	593	0.053505	395	0.050896
6 misc.forsale	585	0.052784	390	0.050251
7 rec.autos	593	0.053505	395	0.050896
8 rec.motorcycles	598	0.053957	398	0.051282
9 rec.sport.baseball	597	0.053866	397	0.051153
10 rec.sport.hockey	172	0.015519	827	0.106558
11 sci.crypt	595	0.053686	396	0.051024
12 sci.electronics	591	0.053325	393	0.050638
13 sci.med	792	0.071461	198	0.025512
14 sci.space	593	0.053505	394	0.050767
15 soc.religion.christian	599	0.054047	398	0.051282
16 talk.politics.guns	546	0.049265	364	0.046901
17 talk.politics.mideast	564	0.050889	376	0.048447
18 talk.politics.misc	465	0.041956	310	0.039943
19 talk.religion.misc	377	0.034016	251	0.032341

Table 1: dataset summary statistics

```

From: pyron@skndiv.dseg.ti.com (Dillon Pyron)
Subject: Re: Founding Father questions
Lines: 35
Nntp-Posting-Host: skndiv.dseg.ti.com
Reply-To: pyron@skndiv.dseg.ti.com
Organization: TI/DSEG VAX Support

In article <1993Apr5.153951.25805@eagle.lerc.nasa.gov>, pspod@bigbird.lerc.nasa.gov (Steve Podleski) writes:
>arc@cco.caltech.edu (Aaron Ray Clements) writes:
>>Wasn't she the one making the comment in '88 about George being born with
>>a silver foot in his mouth? Sounds like another damn politician to me.
>>
>>Ain't like the old days in Texas anymore. The politicians may have been
>>corrupt then, but at least they'd take a stand. (My apologies to a few
>>exceptions I can think of.)
>>
>>News now is that the House may already have a two-thirds majority, so
>>her opposition out of her concern for image (she's even said this
>>publicly) may not matter.
>>
>>Do people expect the Texans congressmen to act as the N.J. Republicans did?

There is a (likely) veto proof majority in the house. The Senate,
unfortunately, is a different story. The Lt.Gov. has vowed that the bill will
not be voted on, and he has the power to do it. In addition, the Senate is a
much smaller, and more readily manipulated body.

On the other hand, the semi-automatic ban will likely not live, as at least
fifty per cent of the house currently opposes it, and it is VERY far down in
the bill order in the Senate (I believe it will be addressed after the CCW
bill).

And I thought my TX Political Science class was a waste of time!

Dillon Pyron
TI/DSEG Lewisville VAX Support
(214)462-3556 (when I'm here)
(214)492-4656 (when I'm home)
pyron@skndiv.dseg.ti.com
PADI DM-54989
  
```

Fig 1: sample document before parsing

The samples is imbalance in classes including rec.sport.hockey and sci.med. For better performance on the test set, the weights for them may need to be adjusted. Or, we can just up-sampling the rec.sport.hockey class to make sure our model will equally fit the features underlying each class. This project chooses the latter one as the option in the program.

3.2 Preprocessing

The preprocessing mainly works for two different directions depended on models. The first direction is treated it as a sequence with order information, which means more patterns could be recognized. The second direction is treated the words in the document as a bag of words, which means the order of words is not important.

3.2.1 Data Cleaning

The original unstructured corpus contains lots of noise in the dataset. With utilized the mail parser, the email can be separate as main body and header part. The header can be quite clearly separated and extracted, but for the main body of the email, some noise still existed. For example:

- Due to the reference of the previous emails, the reply of the email is not separated by the reference with the “>” symbol before the line of sentence. And the email address also exists before every reference.
- Some of them contain binary data, which is image attachment that cannot be decoded by utf-8. To parse such binary data, some rules-based method implemented with regular expression.

After using multiple regular expressions strategically, two previous problems are fixed. For the previous sample, the program can generate the following structural json file:

```
{
  "From": { "0": "pyron@skndiv.dseg.ti.com (Dillon Pyron)" },
  "Subject": { "0": "Re: Founding Father questions" },
  "Lines": { "0": "35" },
  "Nntp-Posting-Host": { "0": "skndiv.dseg.ti.com" },
  "Reply-To": { "0": "pyron@skndiv.dseg.ti.com" },
  "Organization": { "0": "TI/DSEG VAX Support" },
  "reply": { "0": "There is a veto proof majority in the house. The Senate, unfortunately, is a different story. The Lt. Gov.
  "reference_one": { "0": "Do people expect the Texans congressmen to act as the N.J. Republicans did?" },
  "reference_two": { "0": "Wasn't she the one making the comment in '88 about George being born with a silver foot in his mou
  "date": { "0": null },
  "delivered_to": { "0": [] },
  "to_domains": { "0": ["skndiv.dseg.ti.com"] },
  "error_message": { "0": null },
  "contained_emails": { "0": [ " <1993Apr5.153951.25005@eagle.lerc.nasa.gov>, ", "pspod@bigbird.lerc.nasa.gov ", " >arc@cco.ca
  "long_string": { "0": [] },
  "tag_reply": { "0": "veto proof majority house story bill power addition body hand ban cent house bill order Science class
  "tag_reference_one": { "0": "people Texans congressmen Republicans" },
  "tag_reference_two": { "0": "comment foot mouth damn politician days politicians stand majority opposition concern image"
}
```

Fig 2: sample document after parsing

The reply contains all the reply body information of this email. The reference_one contains the previous received information. The reference_two contains all the contents before the previous email send to this sender. The long_string is used for binary data detection.

The reason why this project separates the reply and reference is because some models in there will need to encode the sentence information. With order information, language model and BERT based model can recognize more patterns from corpus in a correct way.

3.2.2 Bag of Words features

Excepts for considering the sequence information, we can use only the words frequency information. With the well-built NLP techniques, we can directly tokenize all the sentences in emails with regular expressions or well-implemented Tokenizer class from Sklearn. With treated each email as a sample, we can build the term of frequency matrix (TF matrix) or term frequency–inverse document frequency (TF-IDF matrix). And further preprocessing option describe as follows:

- Lemmatization and Stemming: without any filter, the number of unique tokens will be 127,414, which is not acceptable since the curse of dimensionality. It will consume too many computational resources, which probably lead to OOM. Therefore, before tokenization, the lemmatization and stemming for each vocabulary is included, which is also implemented as an option in the program. And only 10,593 vocabularies will be considered.
- The Part of Speech tag parser also implemented in case that the NOUN already covers lots of information in the documents. They are treated as optional extra features in this project.
- For stop words, we need to remove it in this direction since it is the noise and will not add any information for helping prediction task. The later grid search experiment result also proved this.
- The high frequency words will be multiplied by inverse-document frequency (IDF) weight to lower down the importance of the columns, which is quite useful for reducing the noise signal underlying features.

Before the preprocessing, the highest frequent word in this dataset shows as the left word clouds. And the right one is after filtering symbols, lemmatization, stemming and removing stop words.



Fig 3: wordcloud before cleaning

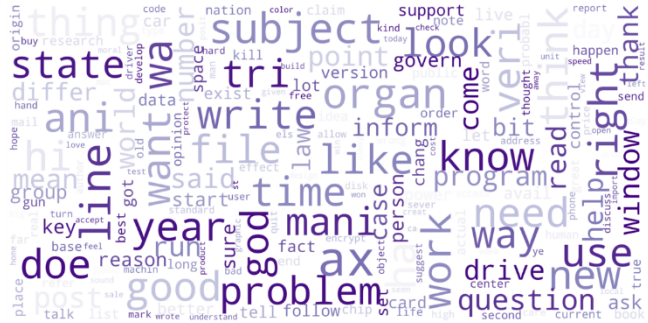


Fig 4: wordcloud before cleaning

3.3 Document Length and Word Frequency

After previous preprocessing, the following graph shows the frequency distribution of the words across all documents. The x-axis is the word index generated by Keras Tokenizer class, which will treat the higher frequency word with lower index value. The y-axis is the cumulative frequency ratio of the words in all corpus. When setting the threshold equal to 98% of the frequent words, we need to consider 4587 words in vocabulary. With 95% threshold, we only need to consider 2899 words.

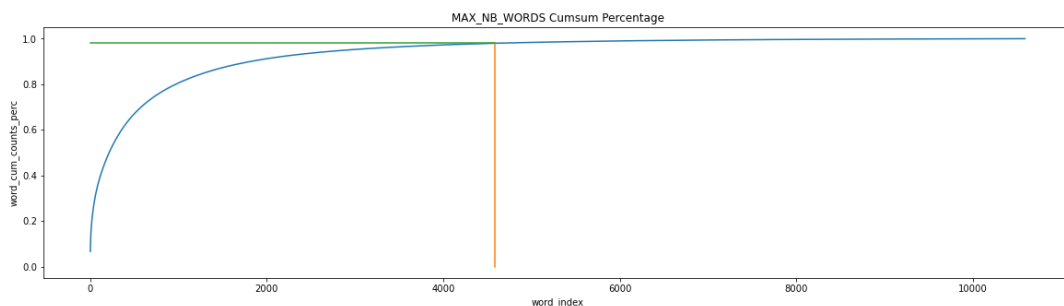


Fig 5: word frequency cumsum distribution

For the document length, the following graph shows the document length distribution across all documents. When setting the threshold equal to 95% of the frequent length of document, we need to consider 210 as the max length of all document. In other words, it means 95% of document has length less than 210 after preprocessed. The x-axis is the length of the document, the y-axis is the frequency ratio of the correspond length. Some of extreme long documents are the email attached the binary image data, for example, the file comp.os.ms-windows.misc named as 9986 only contains roman.bmp binary data, which has 956 lines and 59,105 characters.

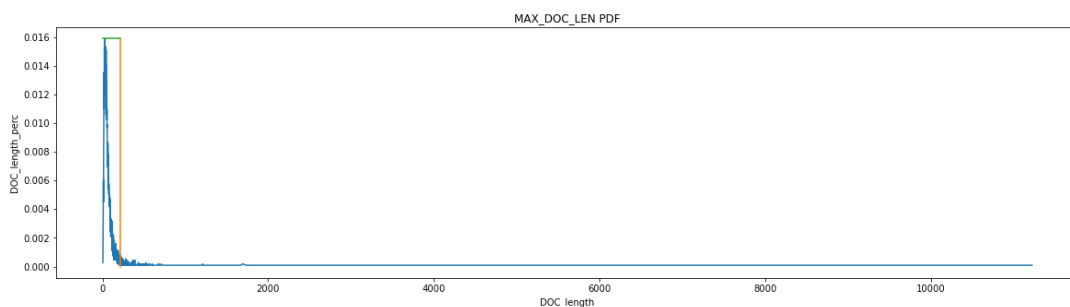


Fig 6: document length frequency distribution

4. Best Performance Summarization

All the models are trained only with the train data. But with different preprocessing methods, the training input will be slightly different. The Jupyter notebook files keep the evaluation record inside of it. The models are reproducible in a short time, so the models' parameters did not save locally in this period. The augmentation means double the training set by extract reply and reference one columns separately to double the training set.

Model	Preprocessing	Input	Hyper-params	Best F1-macro Avg till Dec 13th
SVM	Augmentation ²	TFIDF matrix	C=3, penalty='l2'	0.84±0.01
Multinomial Naïve bayes	Augmentation	TFIDF matrix	-	0.787±0.01
Logistic Regression	Augmentation	TFIDF matrix + Dimension reduction	-	0.807
Xgboost	Augmentation	TFIDF matrix	-	0.835
Deep Neural Networks	Up-sampling, Lemmatizing, Stemming, one-hot for Subject Organization	TFIDF matrix + extra one-hot encoding	Input dim 3236+5691 Batch size 256 Optimizer Adam lr 0.005	0.73±0.01
Multi-channel Text CNN + Dense Neural Networks	Up-sampling	document -> embedding	Max Doc length 560 Max Number word 15235 Embed dim 200 Filter Size [1,3,5] Number filters 64 Optimizer RMSprop 0.0005 Batch size 32	0.746
Finetune BERT + Dense Neural Networks	Up-sampling, Feature selection ³	document -> embedding	Max Doc length 300 Embed dim 200 Filter Size [1,3,5] Linear Hidden units 64 Optimizer AdamW 0.00005 Batch size 32	0.84±0.01
Pretrain BERT embedding + LSTM-CNN	-	pretrain embedding	-	less than 0.6
Kmeans + Majority vote	-	TFIDF matrix + Dimension reduction	-	less than 0.5
Latent Dirichlet allocation + Majority vote mapping	-	TFIDF matrix + Dimension reduction	-	less than 0.5
Multi-layer Bidirectional LSTM	-	document -> embedding	-	Not implemented

Table Table 2: Summary of model performance

² Treat combination [['text'], ['reply', 'reference_one']] separately to double the training set

³ Select columns: ['Subject','Organization', 'reply', 'reference_one']

5. Experimental Evaluation

4.1 Preprocessing

The model input is the TF-IDF matrix, and the output is the prediction labels for each test sample. To achieve better result, the first step in this project is defined preprocessing procedure, such as up-sampling for small class, data augmentation, string normalization. Select proper preprocessing methods will increase the final performance at least 5%.

- Up-sampling is a bootstrap technique that sampling with replacement from original train set. In our case, this is used especially for handling the imbalance problem in the small group class like `rec.sport.hockey`, `talk.religion.misc`. Model may treat the features underlying classes differently because the sample size are different. Another method is apply the class weight in the training process, for example increase the weight of `rec.sport.hockey` as 2, and the others are 1.
- Data augmentation in there is based on the preprocessing that this project already done before. The original email is splitted into subject, reply, reference one and reference two etc. This project will treat them as different samples for different combinations of them. For example, with the text and the label pair, the reply and the label pair, and the reply and all other reference pair, the model's input will be triple for training.
- The string normalization is for removing stop words and parsing special symbols.

4.2 Machine Learning Models

4.2.1 Comparison between SVM, Naïve Bayes and Logistic Regression

The following table summary the best model performance on different preprocessing methods.

model_preprocessing	mean_fit_time	mean_test_score	std_test_score
SVC-augmentation ⁴	9.607	0.992	0.005
NB-augmentation	3.437	0.939	0.008
SVC-upsampling+normalstring_stage2 ⁵	4.301	0.924	0.012
SVC-upsampling+normalstring	3.716	0.924	0.012
SVC-upsampling	3.933	0.923	0.012
LR_1-upsampling+normalstring_stage2	56.258	0.893	0.003
SVC-upsampling+normalstring+augment_stage2	12.126	0.893	0.124
NB-upsampling+normalstring	2.007	0.892	0.01
NB-upsampling+normalstring_stage2	1.953	0.892	0.01
NB-upsampling	2.289	0.889	0.01
NB-upsampling+normalstring+augment_stage2	3.508	0.861	0.125
LR_1-upsampling+normalstring+augment_stage2	41.108	0.847	0.085
SVC-upsampling+stem_voc_stage2	1.651	0.839	0.024
NB-upsampling+stem_voc_stage2	1.025	0.813	0.014
LR_1-upsampling+stem_voc_stage2	1.489	0.806	0.002
LR_2-upsampling+stem_voc_stage2	162.465	0.802	0.001

Table 3: Best model average score with different preprocessing methods

⁴ Treat combination `['text']`, `['reply']`, `['reference_one']` separately to double the training set

⁵ Stage 2 meaning the additional grid search setting for the models

This table show the data augmentation is quite useful in Support Vector Classifier (SVC). Since the training set is also enlarged, which mean the model will be evaluated in larger validation set. However, the model average performance still increases with such techniques.

The second step is for finding the best hyper-parameters setting. In each preprocessing setting with grid search and cross validation. This project applies the grid search to find the best hyper-parameters settings to obtain the TF-IDF matrix and the best model settings. Based on F1-macro score with 3-fold cross validation, the following table briefly summarizes the performance with different hyper-parameters setting for SVC. The grid search is performed on the TF-IDF hyper-parameters, including binary, minimum document frequency, stop words, inverse document frequency. It also combined with the hyper-parameters on SVC, including regularization weight C and the coefficient penalty norm.

param_c lf_C	param_clf__ penalty	param_tfidf_ _binary	param_tfidf_ _min_df	param_tfidf__st op_words	param_tfidf_ _use_idf	mean_test _score
3	l2	TRUE	1		TRUE	0.9919
3	l2	TRUE	1	english	TRUE	0.9913
2	l2	TRUE	1		TRUE	0.991
2	l2	TRUE	1	english	TRUE	0.9905
3	l2	TRUE	3		TRUE	0.9905
2	l2	TRUE	3		TRUE	0.9901
3	l2	TRUE	3	english	TRUE	0.99
3	l2	TRUE	5		TRUE	0.9898
2	l2	TRUE	3	english	TRUE	0.9897
3	l2	FALSE	1	english	TRUE	0.9894
2	l2	TRUE	5		TRUE	0.9893
3	l2	FALSE	1		TRUE	0.9893
1	l2	TRUE	1		TRUE	0.9889

Table 4: Grid Search setting for SVC

The table shows the binarize the term of frequency is useful in this case. It will treat the words equally as the word appear in one document, even maybe some of words appear several times. The IDF is needed for lower weight of some trivial words like conjunction, article etc. The stop words are not that influential is because the IDF and binarize setting already flatten the signal that stop words have. For the penalty term and coefficient C in the SVC, the higher C will add more penalty on the coefficient norm, which will improve the generalization capability of the model. The SVD dimension reduction transformation is also tested in this experiment. After shrinking dimension to 1000, the performance will only reduce at most 5%. The option is adjustable, and it is dependent on the deployment environment.

The related tools are wrapped as `classification_utils.py` and models are implemented in Jupyter notebook files in the repository. And in this direction, the best SVC model has F1 macro average score is 0.835, F1 micro average score is 0.847 on the test set. The hyper-parameters C is 3 and penalty norm is L2. The best Multinomial Naïve Bayes (NB) has F1-macro with 0.815 and F1 micro average score is 0.830 on the test set. The hyper-parameters alpha is 0.6 for NB.

4.3 Topic modeling and K-means Clustering

This project also make exploration on K-means clustering and the topic modeling with Latent Dirichlet allocation model. With the labeled train dataset in each group, the majority rule will map the highest frequent labels in each cluster or each topic into the ground truth label. In other words, all the samples in the group will be treated as the label with largest size in the group. However, both are performed not good in such dataset. Even considered the cosine similarity between each sample, the models still performed bad. The highest accuracy can only achieve 45% in experiments. In addition, the feature space is large and such clustering methods is computational expensive. Even after the truncated singular value decomposition (SVD), the model cannot perform well after transformation in the 500 largest eigen vector space. At the last stage, this direction is disposed, but it might perform well in the small size dataset and small number of classes. The toolkit is prepared and wrapped as `clustering_utils.py` for future reference.

4.4 Neural Network Based Models

In this part, the project explores three types of Neural Networks, including the Deep Neural Network with only dense layers (Dense NN), Multi-Channel Text Convolution Neural Network (TextCNN), and the BERT based models. The RNN and its variants are not explored since the training process cannot be computed by GPU, which means the model is slow to converge to get the evaluation result. All the training process applied the early stopping techniques based on the categorical accuracy on the validation set, which is stratified samples from 20% of the training data.

4.4.1 Dense Neural Network

For the Dense NN, it was constructed by several fully connected dense layer. Rather than use the embedding layer to extract information from text, this model use TF-IDF matrix as it's input. It has the horizontal structure shows as the following simple graph.

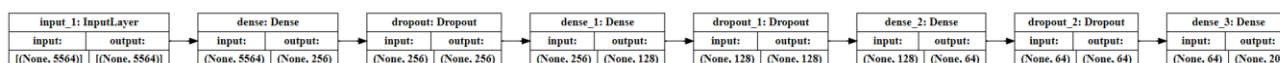


Fig 7 Deep NN structure

In this model, this project also tried to append extra keywords features as one-hot encoding along with the TF-IDF after dimension reduction, but the performance even worst due to the information lost in this case. Since it is kind of model based on Bag of Word, with enough computational capabilities, more information should be included in the input.

The code is implemented with Keras and wrapped as `nn_utils.py` in the repository. After tuning the hyper-parameters, the best F1 macro average score is 0.702, F1 micro average score is 0.712.

4.4.2 Multi-channel TextCNN

For the TextCNN, this project sets the embedding dimension with 200. It applies the several 1-dimension Convolution kernel to filter features from the embedding matrix. The kernel size will determine the observed window in the document, which acting like n-gram in language model. But the

TextCNN will not consider the order information. For example, “cat love dog” is the same as “dog love cat” in this model. As the following graph shows, several channels are used to extract n-gram features from the document input. Finally, the feature maps will be flattened and concatenate together for next full connected dense layers. Basically, the multi-channel CNN acted like a feature extractor. Like pretrain BERT did. But the BERT will consider more detail sequential information rather than only bag of words and n-gram combination in TextCNN. The CNN extractor has structure shows as the following simple graph. And the classifier on the top of this extractor is similar to the Dense NN. The hyper parameters in final version make trivial adjustment.

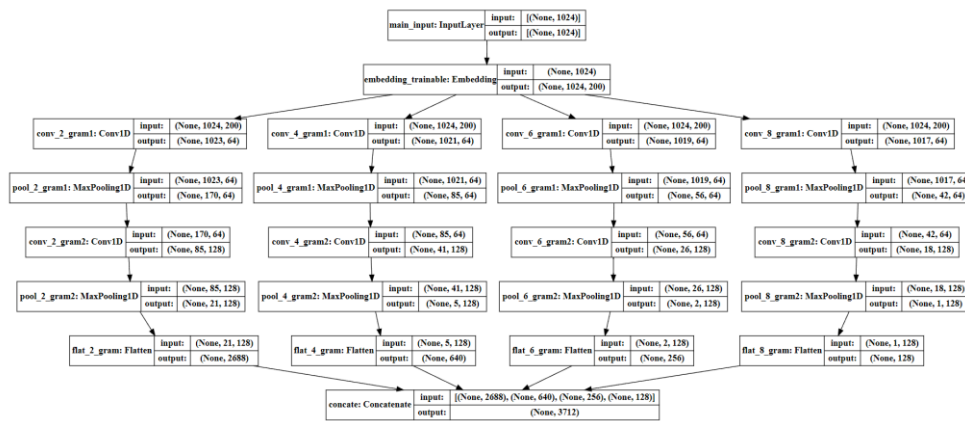


Fig 8 Multi-channel CNN feature extractor structure

In this model, with enough regularization tricks like dropout to prevent overfitting. The model seems cannot extract more information from the train dataset after 10 epochs. In other words, the model achieves its structure limit to exploit more information from the original dataset.

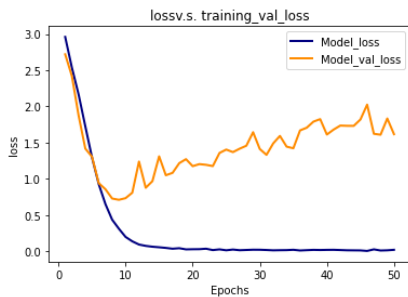


Fig 9 TextCNN training loss history

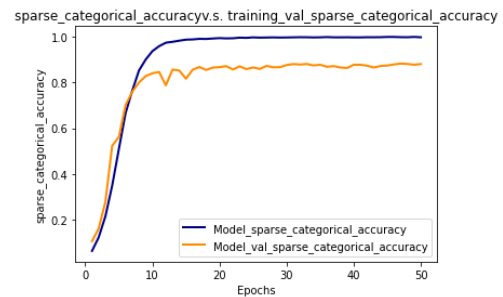


Fig 10 TextCNN training accuracy history

Although the SVC have the same input, TextCNN may be easier influenced by the noise in the dataset. To further improve the performance in this classification task. This project further explores the BERT model structure with previous split reply, reference in the document since those kinds of preprocessing keeps the sequential information on the document.

The code is implemented with Keras and wrapped as nn_utils.py in the repository. The model is training on the Google colab. After tuning the hyper-parameters, the best F1 macro average score is 0.75, F1 micro average score is 0.76.

4.4.3 Pretrain-BERT

In this direction, this project explores the pretrain BERT contextual embedding. With treated BERT as a document encoder, it will generate hidden state on the last layer for every word, which is also including the special tokens [CLS]. For obtaining better performance, this project explores four different ways to use the BERT output.

- A. Use the [CLS] token embedding, which compressed all the documents information into it.
- B. Use all contextual embedding output for every word in the document, which will consider all the tokens embedding as the input of the next stage. This way is computational expensive. In this case, the classifier on the top of this output is LSTM and CNN, with treated the BERT output only as a special embedding which probably encode the document in a proper way.
- C. Like the previous one, the hidden states from last four layers can be averaged to represent the whole document.
- D. Use the global pooler output. But this output is not a good summary of the semantic content of the input from the experiment outcome.

However, the performance of with pretrain BERT and LSTM-CNN is not acceptable. It cannot even beat the baseline from the SVC. That's why this project further explores the finetune version of BERT, which is far better than pretrain one.

4.4.4 Finetune-BERT

Similar to the pretrain BERT, this project explores different usage of the embedding output from BERT. However, the parameters in there will be adjusted and finetune by the train dataset. This model finally chooses type A embedding, the [CLS] token embedding, for down-stream task. On the top of BERT, the whole model structure further stack two dense layers to handle the final classification task. The following graphs show the training accuracy and loss along with the epochs. The same, early stopping technique applied in there, which is stratified samples from 20% of the training data.

During the training process, the norm clipping is added to make sure the gradient will not explode. The maximum norm of the gradient is limited to 1. The other hyper-parameters of this model are: `learning_rate = 0.00005`, `epochs = 20`, `batch_size = 32`, `hidden_units = 64`, `MAX_DOC_LEN = 300`.

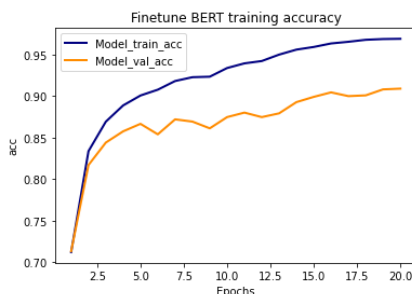


Fig 11 Finetune BERT training loss history

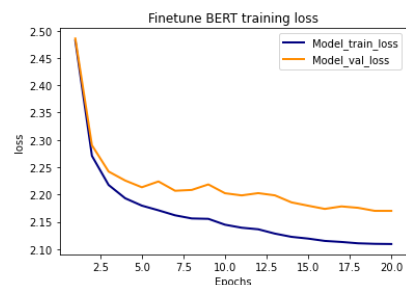


Fig 12 Finetune BERT training accuracy history

The code is implemented with Pytorch and Transformer from hugging face. The related functions are wrapped into `bert_utils` folder in the repository. The model is training on the Google colab. After tuning the hyper-parameters, the best F1 macro average score is 0.83, F1 micro average score is 0.85.

6. Alternative Methods & Future Work

- For this classification task, some well implemented toolkit like Vowpal Wabbit, an interactive machine learning library, could be explored.
- With the larger dataset, for faster training process on machine learning solutions, the model could be deployed with AWS EMR and finished the training process by distributed clusters.
- After further organized and stored the model's parameters, they can be loaded together and ensembled with majority vote for label prediction.
- Implemented Graph based BERT to further exploit language information on the document.
- More detail parameters tuning for large model such as BERT and TextCNN.

References

1. Asim, M.N., Khan, M.U.G., Malik, M.I., Dengel, A. and Ahmed, S., 2019, September. A robust hybrid approach for textual document classification. In *2019 International conference on document analysis and recognition (ICDAR)* (pp. 1390-1396). IEEE.
2. Gomez, J.C. and Moens, M.F., 2014. Minimizer of the reconstruction error for multi-class document categorization. *Expert Systems with Applications*, 41(3), pp.861-868.
3. Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T. and Weinberger, K., 2019, May. Simplifying graph convolutional networks. In *International conference on machine learning* (pp. 6861-6871). PMLR.
4. Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.