CMT316 Applications of Machine Learning - Coursework I Part 2

Zhiliang Xiang 2028447

1st April, 2021

Part 2 Report

For the given BBC news dataset[**4**], we aim at building a machine learning program to classify news articles. This report will give details and justifications of each step that follows the experimental design pipeline, then show the overall performance of the model, and finally discuss how the model could be improved with respect to potential problems.

1. Text Preprocessing

Starting from loading raw text document locally, once the raw data are loaded, the primary task of text preprocessing is to filter all the texts and symbols that are irrelevant to our classification task. we then conduct tokenisation and lemmatisation to the loaded data, eliminate all the stopwords and symbols from the token set. Here we only filtered some trivial symbols which commonly used in any contexts. Symbols which are related to certain contexts like '$' or '@' won't be eliminated.

2. Feature Engineering

In this step, we extract 3 representative features:

1. Token frequency of news contents
2. Token frequency of news headlines
3. Frequency distribution of Part of Speech Tags

and represent them as vectors.

We first consider a typical choice of document representation, the *bags of words* approach, it could capture how much does a word contribute to the semantics of a certain document[**1**]. We deemed the set of top frequency-ranking tokens from each topic as vocabulary, such that contents of a news can be represented as a vector, each entry of the vector records the occurrence count of a token from the vocabulary.

However, in the setup of raw tokens frequency, all words are considered equally important, that is, some irrelevant words that've been commonly used could be ranked to the top. In this case, we introduced the TF-IDF weighting[**2**] scheme to apply on each token when building the vocabulary. Consider the following equation, tf_i is the frequency of i in topic T, cf_i denotes the occurrences of i within all topics, n_i is the number of topics that contain i, and N is the total number of topics, then the ranking score s_i of token i in topic T is:

$$s_i = \text{tf}_i * \frac{\text{tf}_i}{\text{cf}_i} * log(\frac{N}{n_i})$$

By using TF-IDF weighting scheme, we can achieve that:

- (High Relevance) Scores of tokens with high frequency in a certain topic will be scaled up
- (Low Relevance) Scores of tokens which occurs fewer times in a topic, or occurs in many topics will be scaled down
- (Lowest Relevance) Scoring 0 when a token appears in all topics

By choosing the top-ranking tokens and combining sub-vocabulary from each topic, we can get the general vocabulary within all the documents. After the vocabulary was built, we iterated the each line of the documents and stored the scores into a set of arrays such that the entries represent the frequencies of top-ranking tokens in the vocabulary.

In addition, considered the news headlines could contain key information of which topic the news belongs to, for example tech news headlines usually contain 'computer' or other topic related keywords, in that case one can easily tell which topic the news is by only looking at the headline. To implement this, we separately built a top-frequency ranked vocabulary of news titles, followed the same manner as in processing news contents.

In terms of the third feature, as our first two features are all word-based features, introducing the POS features could give us some context information of the words occurred in documents, this might help us making the word-based features more stronger. For the simplicity of features, we only took noun, pronoun, verb, adjective and adverb these five word types into account. For each document in all topics, counting every token according to which type of word it is, and storing the counting numbers in an array that represent the five word types respectively.

BUSINESS	ENTERTAINMENT	POLITICS	SPORT	TECH
0	1	2	3	4

Figure 1.: mappings between topics and labels

Finally, we mapped the five topics to labels ranging from 0 to 4 shown as Figure 1, after which the collection of vectors that represent the given documents can be obtained.

3. Feature Selection, Training and Evaluation

3.1 Feature Selection

Although the top-ranking vocabulary scheme have reduced the potential dimensionality of representation vectors (30000^+ without top-ranking selection), we still ended up with a dimensionality of almost 4000. In order to make the dataset less noisy and the model less expensive to train, feature selection is needed. In this regard, we considered the number of selected features k as a hyper-parameter and integrated feature selection process into the pipeline of model training and fine-tuning as following steps: normalisation $\rightarrow$ feature selection $\rightarrow$ training. As for the selection algorithm, we tried both χ^2 and mutual information schemes, although the selection process ran much faster while using χ^2 scheme, it's still worth trying both since performance of them have relatively close[**3**].

3.2. Training and Validation

Our prior knowledge of the given task includes:

- it is a multi-class classification problem;
- sample size is relatively small (only contains 2223 documents);
- dimension of the feature space is high

From the factors above, both SVM and Softmax Regression could have the capacity to fit well on dataset. To confirm this, we first used a simple train-dev-test(80%-10%-10%) dataset splitting with regular parameter setup to perform a one-shot evaluation (refer to `one_shot_eval.py`). If chosen algorithms are underfitting, mis-classification will happen on training set prediction; if algorithms are overfitting, performances on validation set will be poor. Following code snippets show the evaluation report for both Linear SVM classifier and Softmax Regression, we are therefore confident to say that both methods are capable to perform the task.

```
            One-shot test on Softmax Regression

Training Fitting Score:         100.00%

Validation Score:               94.62%

Test Accuracy:                  0.98

             One-shot test on Linear SVM

Training Fitting Score:         100.00%

Validation Score:               94.62%

Test Accuracy:                  0.96
```

Figure 2: One-shot Validation

Furthermore, to exploit the dataset and find out which setup of the two algorithms is the best (fine-tuning), we conduct evaluation using Grid Search with 5-fold cross-validation (refer to `main.py` and `fs_n_train.py`). Started from splitting the whole dataset into training set and test set in 8:2 ratio, we used the training set to perform `GridSearchCV` and fine-tune parameters of both algorithms. This will cross-validate each of the hyper-parameter setup we provided , and return the predictor which achieved the best overall performance scored by accuracy. Figure 3 and 4 show the cross-validating performance of both algorithms in each parameter grid. As can be seen, overall performance of Softmax Regression in each parameter setup is more stable than SVM classifier setups, the best setup of Softmax Regression (96.87%) scored slightly higher than the best SVM setup (96.47%) as well.

params	rank_test_score	split0_test_score	split10_test_score
{'selector__k': 500, 'softmax__C': 0.1, 'softmax__max_iter': 6000}	34	0.9550561797752809	0.949438202247191
{'selector__k': 500, 'softmax__C': 0.1, 'softmax__max_iter': 8000}	34	0.9550561797752809	0.949438202247191
{'selector__k': 500, 'softmax__C': 0.1, 'softmax__max_iter': 10000}	34	0.9550561797752809	0.949438202247191
{'selector__k': 500, 'softmax__C': 1, 'softmax__max_iter': 6000}	28	0.9606741573033708	0.9634831460674157
{'selector__k': 500, 'softmax__C': 1, 'softmax__max_iter': 8000}	28	0.9606741573033708	0.9634831460674157
{'selector__k': 500, 'softmax__C': 1, 'softmax__max_iter': 10000}	28	0.9606741573033708	0.9634831460674157
{'selector__k': 500, 'softmax__C': 10, 'softmax__max_iter': 6000}	25	0.9662921348314607	0.9606741573033708
{'selector__k': 500, 'softmax__C': 10, 'softmax__max_iter': 8000}	25	0.9662921348314607	0.9606741573033708
{'selector__k': 500, 'softmax__C': 10, 'softmax__max_iter': 10000}	25	0.9662921348314607	0.9606741573033708
{'selector__k': 500, 'softmax__C': 100, 'softmax__max_iter': 6000}	31	0.9662921348314607	0.9578651685393258
{'selector__k': 500, 'softmax__C': 100, 'softmax__max_iter': 8000}	31	0.9662921348314607	0.9578651685393258
{'selector__k': 500, 'softmax__C': 100, 'softmax__max_iter': 10000}	31	0.9662921348314607	0.9578651685393258
{'selector__k': 1000, 'softmax__C': 0.1, 'softmax__max_iter': 6000}	22	0.9606741573033708	0.9550561797752809
{'selector__k': 1000, 'softmax__C': 0.1, 'softmax__max_iter': 8000}	22	0.9606741573033708	0.9550561797752809
{'selector__k': 1000, 'softmax__C': 0.1, 'softmax__max_iter': 10000}	22	0.9606741573033708	0.9550561797752809

```
Score:          96.87%
Parameters:
{'selector__k': 2000, 'selector__score_func': <function chi2>,
'softmax__C': 1, 'softmax__max_iter': 5000}
```

Figure 3: GridSearchCV Result of Softmax Regression

params	rank_test_score	split0_test_score	split10_test_score
{'SVC__C': 0.1, 'SVC__kernel': 'linear', 'selector__k': 500}	13	0.9353932584269663	0.9438202247191011
{'SVC__C': 0.1, 'SVC__kernel': 'linear', 'selector__k': 1000}	7	0.9691011235955056	0.9550561797752809
{'SVC__C': 0.1, 'SVC__kernel': 'linear', 'selector__k': 2000}	1	0.9747191011235955	0.9634831460674157
{'SVC__C': 0.1, 'SVC__kernel': 'rbf', 'selector__k': 500}	22	0.2303370786516854	0.2303370786516854
{'SVC__C': 0.1, 'SVC__kernel': 'rbf', 'selector__k': 1000}	22	0.2303370786516854	0.2303370786516854
{'SVC__C': 0.1, 'SVC__kernel': 'rbf', 'selector__k': 2000}	22	0.2303370786516854	0.2303370786516854
{'SVC__C': 1, 'SVC__kernel': 'linear', 'selector__k': 500}	12	0.9634831460674157	0.9606741573033708
{'SVC__C': 1, 'SVC__kernel': 'linear', 'selector__k': 1000}	8	0.9662921348314607	0.9606741573033708
{'SVC__C': 1, 'SVC__kernel': 'linear', 'selector__k': 2000}	5	0.9662921348314607	0.9578651685393258
{'SVC__C': 1, 'SVC__kernel': 'rbf', 'selector__k': 500}	19	0.32865168539325845	0.3342696629213483
{'SVC__C': 1, 'SVC__kernel': 'rbf', 'selector__k': 1000}	20	0.23595505617977527	0.2303370786516854
{'SVC__C': 1, 'SVC__kernel': 'rbf', 'selector__k': 2000}	21	0.2303370786516854	0.2303370786516854
{'SVC__C': 10, 'SVC__kernel': 'linear', 'selector__k': 500}	14	0.9550561797752809	0.9466292134831461
{'SVC__C': 10, 'SVC__kernel': 'linear', 'selector__k': 1000}	9	0.9662921348314607	0.9634831460674157
{'SVC__C': 10, 'SVC__kernel': 'linear', 'selector__k': 2000}	3	0.9662921348314607	0.9578651685393258

```
Score:          96.47%
Parameters:
{'SVC__C': 0.1, 'SVC__kernel': 'linear', 'selector__k': 2000,
'selector__score_func': <function chi2>}
```

Figure 4: GridSearchCV Result of SVM Classifier

4. Evaluation

The next step was using the independently reserved test set to evaluate the performance of chosen best models and give final judgment of chosen models in terms of the unseen data performance, we could be certain that the models generalised dataset well if the result is good. As shown in Figure 5.1.1. to Figure 5.2.2., the best SVC model slightly outperformed the best Softmax Regression

model.

To test that how well can the models perform beyond the provided dataset, we used them to predict on a small amount of up-to-date unseen test data (refer to `unseen_test.py` and, 4 samples for each category), from the confusion matrices shown in Figure 5.3 and Figure 5.4 we can see that SVC model achieved a better performance on the selected unseen data.

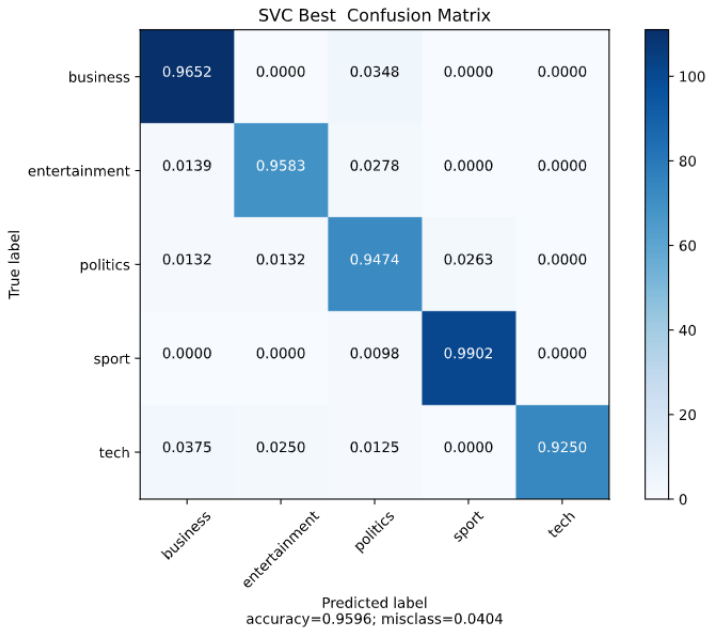

Figure 5.1.1.: Confusion Matrix of the best SVC model

	precision	recall	f1-score	support
0	0.96	0.97	0.96	115
1	0.96	0.96	0.96	72
2	0.90	0.95	0.92	76
3	0.98	0.99	0.99	102
4	1.00	0.93	0.96	80
accuracy			0.96	445
macro avg	0.96	0.96	0.96	445
weighted avg	0.96	0.96	0.96	445

Figure 5.2.2.: Classification report of the best SVC model

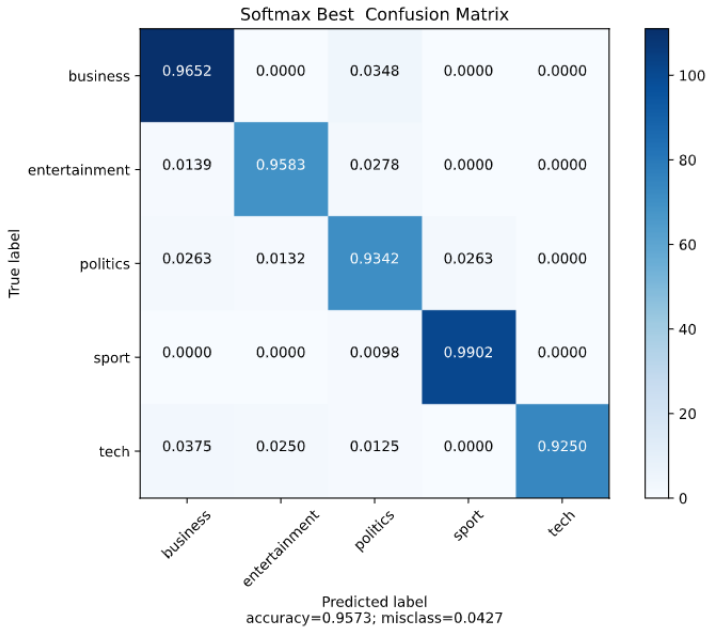

Figure 5.2.1.: Confusion Matrix of the best Softmax Regression model

	precision	recall	f1-score	support
0	0.95	0.97	0.96	115
1	0.96	0.96	0.96	72
2	0.90	0.93	0.92	76
3	0.98	0.99	0.99	102
4	1.00	0.93	0.96	80
accuracy			0.96	445
macro avg	0.96	0.95	0.96	445
weighted avg	0.96	0.96	0.96	445

Figure 5.1.2.: Classification report of the best Softmax Regression model

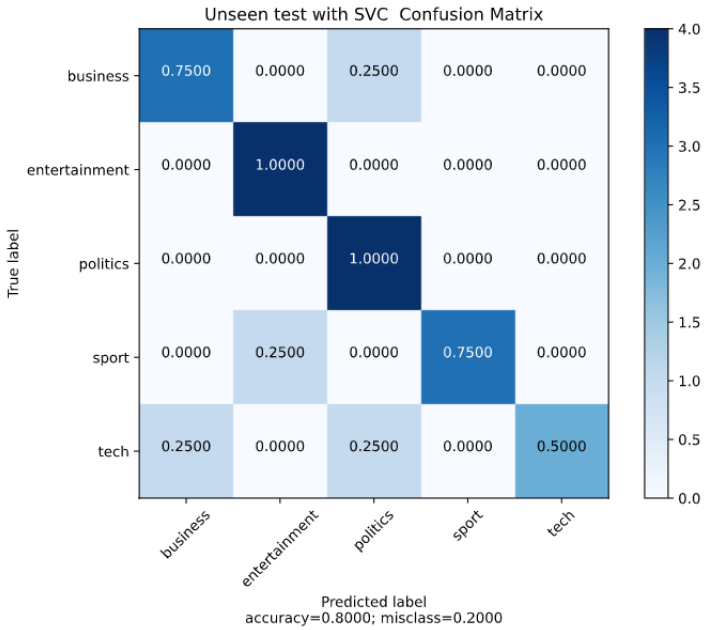

Figure 5.3.: Performance of SVC model on unseen data

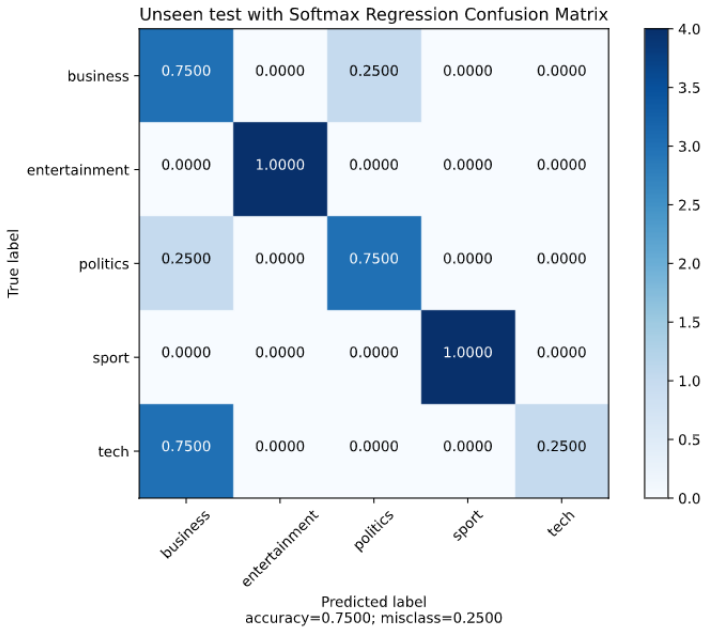

Figure 5.4.: Performance of Softmax Regression model on unseen data

5. Further Improvements

As shown in the predictions of up-to-date data, although our models generalised the dataset well, but when going beyond the dataset, the accuracies significantly dropped. This might due to the following reasons:

1. The length of vocabulary in our models are fixed, words that are out-of-vocabulary but relevant to certain topic cannot be identified.

2. Our representation approach of documents cannot capture semantics, for instance, articles which are represented in similar frequency vectors but belong to different topic in actual will be misclassified.

For improvements, a naive approach is to increase the size of corpus and train the models with new documents, but this could immensely increase the dimensionality of representation and still can do nothing about semantics required factors like synonymy and polysemy. In this case, we can try topic modelling method of representation such as Supervised Latent Dirichlet Allocation[**5**]., which based on generative model and has greater capacities in text representation and semantics capturing .(See section 6 for more details)

6. Literature Review of sLDA(Optional)

Supervised Latent Dirichlet allocation(sLDA) is a derivation of *Latent Dirichlet allocation* (LDA), a generative probabilistic topic modelling approach for

discovering "topics" that occur in a collection of documents. The basic ideas of LDA are modelling the documents in a collection as a mixture over an underlying set of topics, and the set of topics are modelled as probability distributions over words in documents[**5**]. While LDA is an unsupervised approach, sLDA provides a supervised topic model, where each document is paired with a response. After topic model is built, sLDA can be used to infer the topic structure of an unlabelled document[**6**].

Here we illustrate the power of topics modelling by a simple test case (refer to `sLDA_test.py`). When training the news classifier with sLDA representation, the dimensionality of representation vectors reduced significantly comparing to *bag of words* approach(from almost 4000 to only 5). Also, as shown in Figure 6.1 and Figure 6.2, although the untuned model has a slightly lower accuracy than our previous best fine-tuned model, but it performed incredibly well on the up-to-date unseen samples, which can somehow show that topic modelling approach is more flexible and robust than BoW.

	precision	recall	f1-score	support
0	0.92	0.90	0.91	40
1	1.00	0.88	0.94	42
2	0.91	0.93	0.92	42
3	0.98	1.00	0.99	57
4	0.87	0.95	0.91	41
accuracy			0.94	222
macro avg	0.94	0.93	0.93	222
weighted avg	0.94	0.94	0.94	222

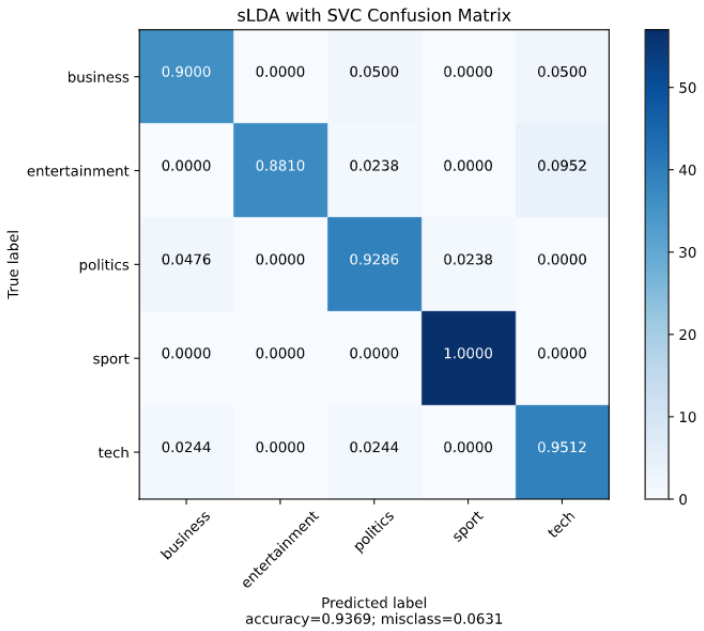

Figure 6.1.: Classification report of sLDA modelling with linear SVC

	precision	recall	f1-score	support
0	0.80	1.00	0.89	4
1	1.00	1.00	1.00	4
2	0.80	1.00	0.89	4
3	1.00	1.00	1.00	4
4	1.00	0.50	0.67	4
accuracy			0.90	20
macro avg	0.92	0.90	0.89	20
weighted avg	0.92	0.90	0.89	20

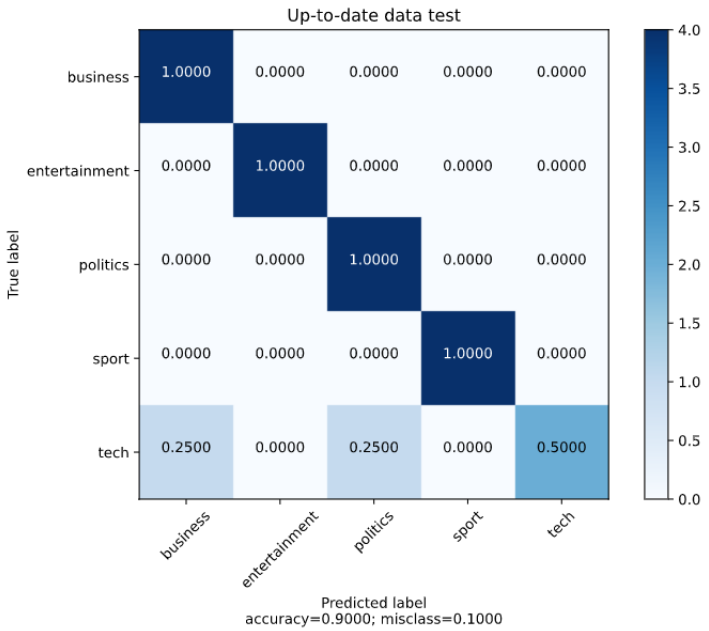

Figure 6.2.: Classification report of sLDA modelling with linear SVC

References

[1] Fabrizio Sebastiani. *Machine learning in Automated Text Categorization. ACM Computing Surveys*, March 2002
https://dl.acm.org/doi/pdf/10.1145/505282.505283

[2] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, ISBN 0521865719. 2008.
https://nlp.stanford.edu/IR-book/

[3] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze. *Introduction to Information Retrieval: Comparison of feature selection methods*. Cambridge University Press, ISBN 0521865719. 2008.
https://nlp.stanford.edu/IR-book/html/htmledition/comparison-of-feature-selection-methods-1.html

[4] D. Greene and P. Cunningham. *"Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering"*, Proc. ICML 2006.

[5] D. M. Blei, '*Latent Dirichlet Allocation*', p. 30.

[6] J. D. Mcauliffe and D. M. Blei, '*Supervised Topic Models*', p. 8.