MEDICAL DOCUMENT CATEGORIZATION USING SVM

Siddharth Pandey, Pranshu Ranjan Singh, Eduard Anthony Chai
Nyon Yan Zheng, Tan Kok Keng

Institute of System Science, National University of Singapore

ABSTRACT

Medical document classification important for medical database management. The Ohsumed collection included medical abstracts from the MeSH categories of the year 1991. The aim of this study is to first instrument a baseline methodology to classify the documents followed by implementing a more sophisticated approach using feature engineering and hyperparameter tuning to contrast the difference in performance of the models.

1. INTRODUCTION

The Ohsumed dataset used by Joachims[1] is divided into two sets, training and test, with each set containing 10,000 abstracts. For the purpose of this experiment, a subset of 5 categories out of 23 is taken. The categories taken are *Bacterial Infections and Mycoses, Digestive System Diseases, Respiratory Tract Diseases, Nervous System Diseases, Urologic and Male Genital Diseases*. As Support Vector Machines (SVM) has the capability to handle a large number of attributes adeptly, they are well suited for text analytics tasks which usually have a huge input space. Thus, SVMs are used as the machine learning models. The F1 score, micro-averaged across all categories, will be used as the performance score to compare different models. For the purpose of comparing the models, all abstracts from standard conventional training set split, will be used for learning. The evaluation set is generated from all abstracts from the test set.

1.1. Exploratory Data Analysis

A preliminary exploratory analysis is performed to have a better understanding of the dataset. *Figure 1(a)* shows the number of documents in each of the category. From *figure 1(b)* some of the common words across the corpus is visualized. *Figure 1(c)* shows common bigram phrases in the corpus.

1.2. Toolkits Used

Scikit-learn, 0.19.1 is used for bag-of-words, Tf-Idf, SVM implementation. NLTK 0.33 is used for stemming and lemmatization. As different toolkits have will different default parameter, to replicate the experiment results, refer to the definition of baseline approach in Section 2.

2. BASELINE APPROACH

The main idea of the baseline approach is to use the default settings available in toolkits and produce a reference score to compare other models. As our problem is of text categorization, the raw data cannot be fed directly to SVM, hence some preprocessing is required. But to still keep this as a baseline model, the simplistic bag of words approach is used to convert the documents into a feature matrix.

$$d_j = (w_{1j}, \ldots, w_{|T|j})$$

Here d_j is the jth document, w_{ij} is the weight term, which in this case is the frequency for i_{th} word in the document [3]. SVM with a linear kernel is used as the baseline model.

3. PROPOSED APPROACH

To improve the model performance, multiple approaches can be employed such as modifying the dataset, applying feature engineering and hyperparameter optimization. As this experiment is a comparative study, no changes can be made to the dataset. Thus later two approached are explored in this experiment to improve the performance.

3.1. Feature Engineering

Feature engineering is used to enrich the attribute space by modifying existing and/or creating new features based on domain knowledge, problem nature to make machine learning algorithm fit the data better.

3.1.1. Using Tf-Idf Feature Matrix

Term frequency-Inverse document frequency is used to generate the feature matrix instead of the bag-of-words approach. The inspiration of this idea lies within the fact

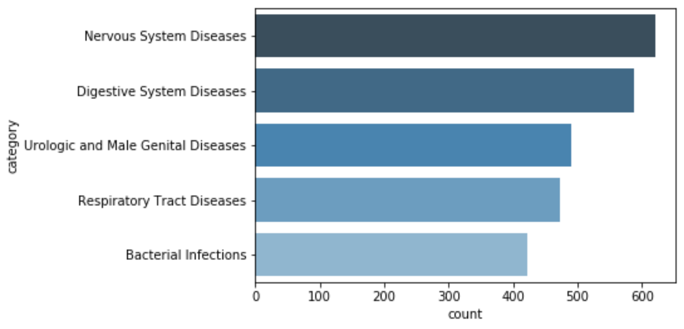


Figure 1(a) Number of documents in each category

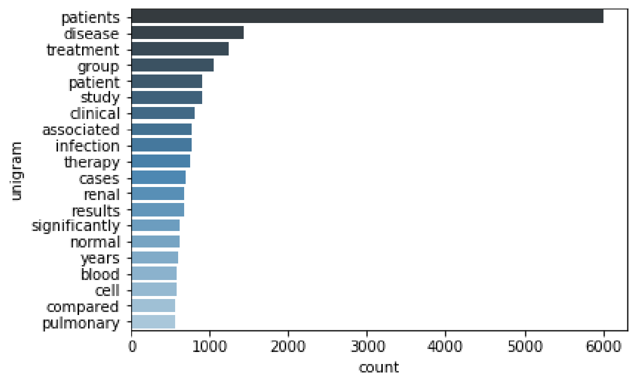


Figure 1(b) Most commonly used words in the corpus

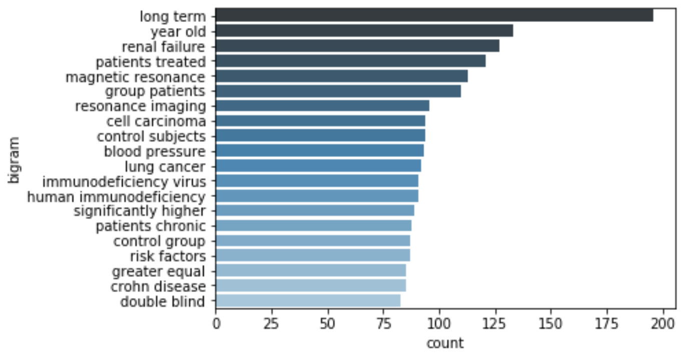


Figure 1(c) Most common phrases in corpus, after stemming and lemmatization, and removing stop words.

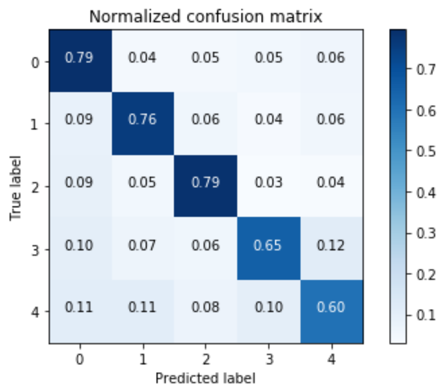


Figure 2. Normalized confusion matrix for baseline approach.

that in a large corpus some words will be present in all documents. These common words carry very little meaningful information about the contents of the documents. Using this approach such words are given less weights. It is calculated as follows.

$\text{tf-idf(t,d)} = \text{tf(t,d)} \times \text{idf(t)}$ where tf is the term frequency and idf is inverse document frequency.

$$\text{idf}(t) = log\frac{1+n_d}{1+\text{df}(d,t)} + 1$$

where n_d is the total number of documents, and $\text{df}(d,t)$ is the number of documents that contain term t. The resulting document vector is normalized.

3.1.2. Stemming and Lemmatization

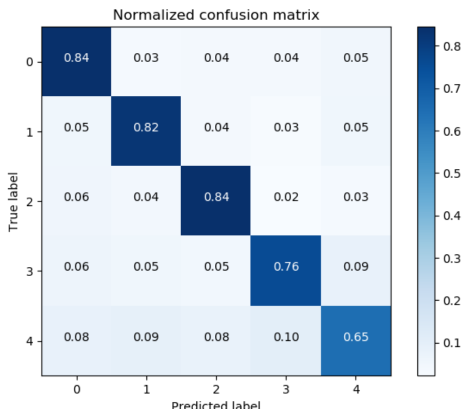
The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. For instance, with stemming and lemmatization all these words will be considered different terms with different weights, but in essence, they refer to the same entity.

car, cars, car's, cars' $\Rightarrow$ car

To avoid such situations, the following preprocessing is carried out. *Stemming* usually refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time, and often includes the removal of derivational affixes. *Lemmatization* usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the *lemma.*

3.1.3. Removing stop words

Stop words are commonly used words like *is, the, their, them,* etc. which do not carry any information about the topic. Such words, can confuse the learning models and may hinder in generating useful n-gram models, are removed to decrease the size of feature matrix and improve the quality of feature space.



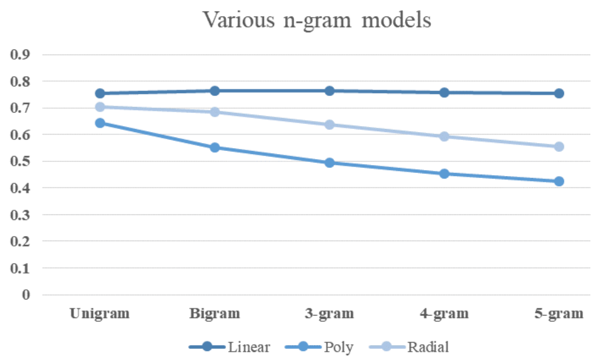
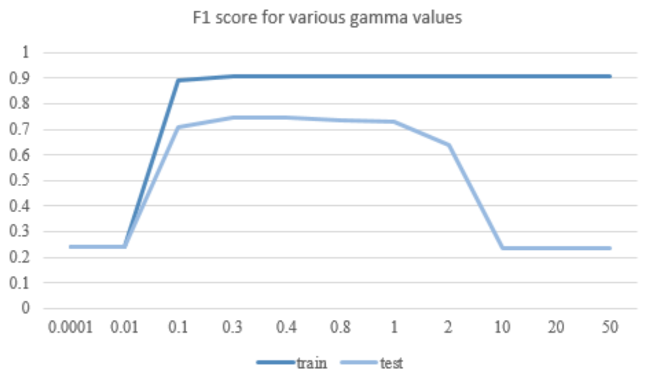


Figure 3. Normalized matrix for the solution found using grid search. There is an improvement seen in all the five categories.

Figure 4(a). Increasing number of n-gram does not improve the average F1 score found using 3 fold cross-validation. For Linear kernel, there is a slight increase in F1 score, till trigram.



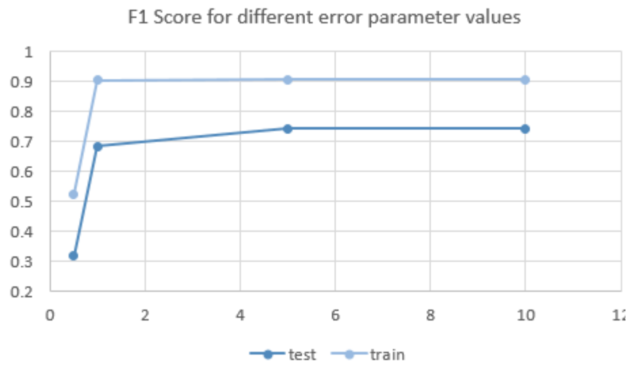


Figure 4(b). As gamma increases the F1 for training, test improves, until 0.8, afterwards the score for training set remains same, but for test set decreases due to overfiting (for radial basis kernel, error parameter = 3, unigram model).

Figure 4(c). Average F1 score on train and test, with different error parameter values for SVM with radial baisi kernel, gamma=0.6

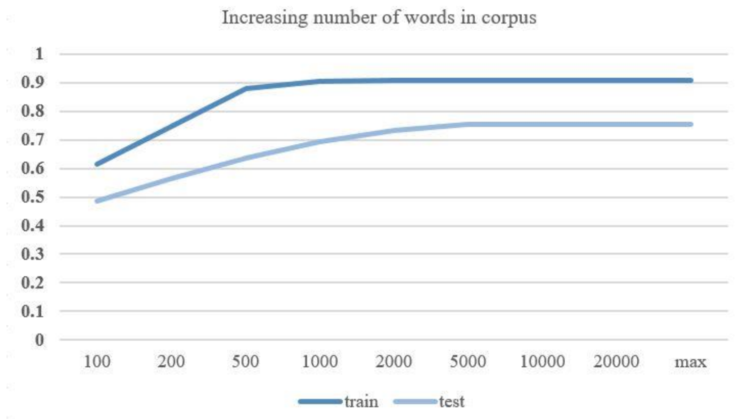


Figure 5. Average F1 score of linear kernel SVM improves with increasing the vocabulary size.

3.1.4. Increasing n-gram range

Bag-of-words approach is essentially a unigram approach, which does not capture phrases and multi-word expressions and ignoring any word order dependence.

Instead of unigrams, one can choose to include bigrams collection of words, which can capture consecutive word occurrences are also counted. Other higher n-gram models can also be explored, as they are more resilient against misspelling and derivation.

3.2. Hyper-Parameter Tuning

In our baseline approach, we used out-of-box parameter values for our SVM model with a linear kernel. In this segment, we will explore other kernels, as well as fine tune the hyperparameter options to improve model performance.

A grid search is used to try a various combination of hyperparameters to find the best model

3.2.1. Using Polynomial/Radial basis Kernel

The problem that we were solving may not necessarily be linearly separable. To better fit the dataset, more complicated decision boundaries generated from polynomial and radial basis kernel of SVM were explored

3.3.2. Tuning Degree/Gamma

The degrees of the polynomial used in the polynomial kernel can be modified to generate more complex decision boundaries. In this experiment, polynomial of degrees 2, 3 and 4 was used.

3.3.3. Tuning Error Parameter

The error parameter (C) is a regularization parameter. The following values of C were experimented with: {0.5, 1, 5, 10}.

4. EXPERIMENTAL RESULTS

Using the grid search, a total of 256 support vector machines were trained and tested on using 2 fold cross-validation technique. Models were ranked based on the F1 score metric and averaged across the two folds.

4.1. Evaluation of baseline approach

The baseline model, when tested on evaluation set, produces a 0.75 micro-averaged F1 score. Average accuracy score is 72.9%. Below is the classification report.

Category	*Precision*	*Recall*	*F1-score*
Nervous System Diseases(0)	0.79	0.82	0.81
Digestive System Diseases(1)	0.75	0.77	0.76
Urologic & Male Genital Diseases(2)	0.75	0.78	0.77
Respiratory Tract Diseases(3)	0.75	0.69	0.72
Bacterial Infections(4)	0.65	0.63	0.64
Average	*0.73*	*0.73*	*0.74*

4.2. Evaluation of the proposed approach

The most optimized model found using grid search is an SVM based on a linear kernel with error parameter (C) equal to 5. This model uses a feature matrix created from n-gram ranging from unigram to trigram. SVM with radial kernel (gamma=0.6, c=1, unigram) also show similar performance.

The improved F1 score is 0.792 and the accuracy is 79%. Below is the classification report.

Category	*precision*	*recall*	*f1-score*
Nervous System Diseases(0)	0.85	0.84	0.85
Digestive System Diseases(1)	0.80	0.82	0.81
Urologic & Male Genital Diseases(2)	0.78	0.84	0.81
Respiratory Tract Diseases(3)	0.80	0.76	0.78
Bacterial Infections(4)	0.69	0.65	0.67
Average	*0.79*	*0.79*	*0.79*

5. CONCLUSIONS

The accuracy of the new model has improved from 73% to 79% with the F1 score also improved from 0.75 to 0.79. Thus, we see an improvement in the model performance after applying feature engineering and hyperparameter optimization.

Although there is an improvement, the baseline approach did perform better than our expectation. The improvement after applying numerous pre-processing and parameter optimization is not huge. This also endorses the results of the study [4], which shows that the accuracy of text classification using SVM is not affected much by the presence of noise in the document.

6. REFERENCES

[1] Thorsten Joachims, Text Categorization with Support Vector Machines: Learning with Many Relevant Features. LS8-Report 23, Universitat Dortmund, LS VIII-Report, 1997.

[2] Yiming Yang, An evaluation of statistical approaches to text categorization. Journal of Information Retrieval, Vol 1, No. 1/2, pp 67--88, 1999.

[3] [Sebastiani, 2002] Fabrizio Sebastiani. Machine learning in automated text categorization. ACM Computing Surveys, 34(1):1-47, 2002.

[4] S. Agarwal, S. Godbole, D. Punjani, and Shourya Roy. 2007. How much noise is too much: A study in automatic text classification. In Proceedings of the Seventh IEEE Intl. Conf. on Data Mining (ICDM 2007), pages 3–12.

WEB REFERENCES

- http://ijcsn.org/IJCSN-2014/3-4/Medical-Document-Classification-from-OHSUMED-Dataset.pdf
- http://disi.unitn.it/moschitti/corpora.htm
- https://www.mat.unical.it/OlexSuite/Datasets/SampleDataSets-about.htm
- http://davis.wpi.edu/xmdv/datasets/ohsumed
- http://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction
- https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html
- http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.423.2888