SCIENTIFIC DOCUMENT CLASSIFICATION

Created By: Rakesh Nain

Document classification is a problem in which different documents need to be classified using a machine learning model. Documents are classified in this project according to their scientific fields. Abstracts of different documents are given in each row in the dataset and the prediction is to be made using the given abstract.

Document classification helps to identify the field to which a particular document belongs to. This can help to sort the document based on their fields. The abstracts are crawled from arXiv. It is given that this is a fine-grained classification task which is very different from the sentiment analysis. Fine grained classification is the classification with categories which are very similar. The categories are different but share a very common part structure. [1]

Aim:
The aim is to develop a classifier that can classify different abstracts to their correct labels. For this purpose, different classification models need to be tested and the model with the highest accuracy can be used to classify depending on the dataset.

Table of Contents

1 INTRODUCTION..3

2 PRE-PROCESSING AND FEATURE GENERATION...................................3

2.1 PRE-PROCESSING...4

2.2 FEATURE GENERATION..5

3. MODELS..6

3.1 MODEL 1...6

3.2 MODEL 2...7

3.3 MODEL 3...7

3.4 DISCUSSION OF MODEL DIFFERENCE(S)...7

4 EXPERIMENT SETUPS...8

5 EXPERIMENTAL RESULTS...9

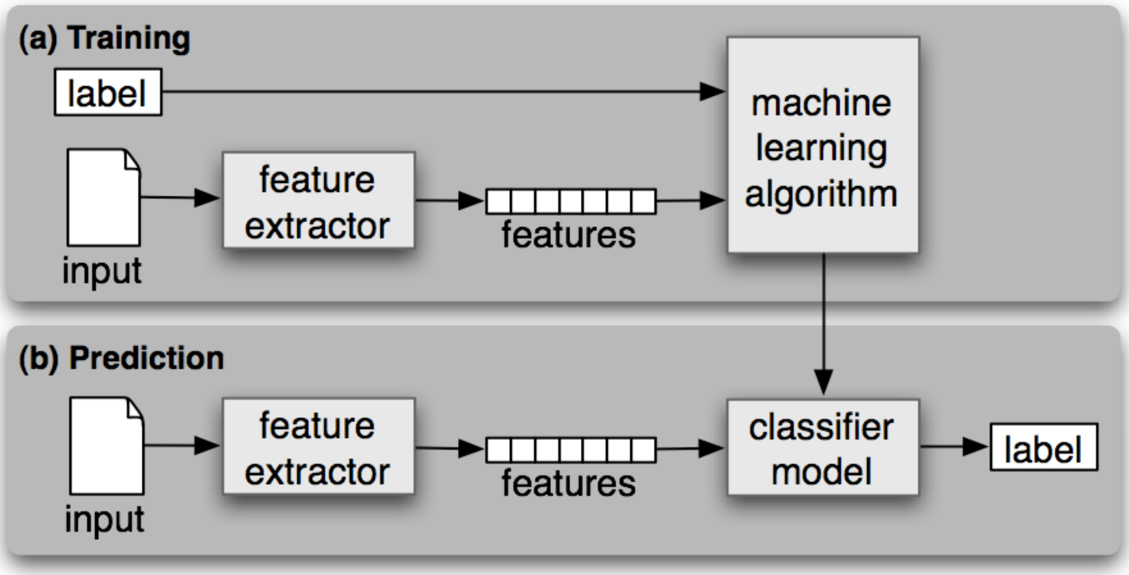
6. CONCLUSION...10

REFERENCES..11

1 Introduction

Scientific document classification is a key step for managing research articles and papers in forums like arxiv, Google Scholar and Microsoft Academic. In this project, we have given some abstracts crawled from arXiv, the task is to develop classification models which can make predictions and return the corresponding scientific fields of the source documents. Different from coarse grained classification tasks like sentiment analysis, this is a fine grained classification task where there are 100 filed classes in total. There are many machine learning methods that can be used in the classification task.

General framework of this project will be:



As shown in the figure, there are three major steps, including generating features, developing a proper classifier, and applying the classifier to the unseen data. The feature extractor is shared by both training and prediction, which tells us that data used in training and prediction should share the same feature space.

The aim of this project is to develop a classifier that can assign a set of scientific abstracts to their corresponding labels as correctly as possible.

<u>Dataset</u>

Data Source	Type	classes	num. training examples
arXiv	Scientific field	100	29,638

Table 1

We provide the following data sets (Table 1):

- train_data_labels.csv contains training ids, abstracts and labels. It contains abstracts from 29,638 articles and acts as the training data.

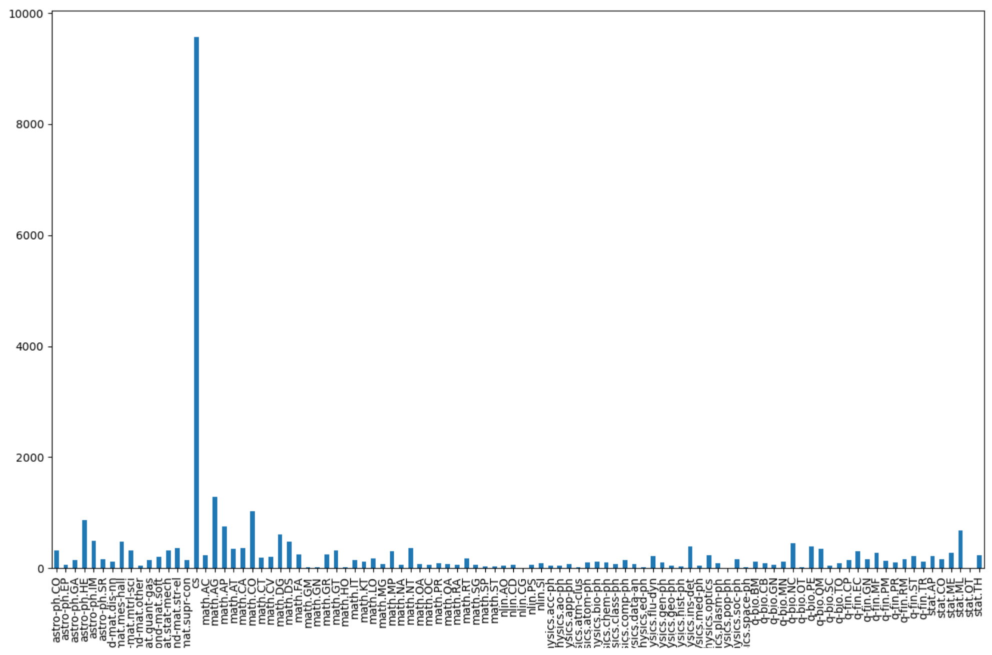
Note: need to do random train, test split.

2 Pre-processing and feature generation

In the text pre-processing step [2], multiple text cleaning techniques are used like making tokens, removing special characters or unnecessary characters, removing digits and stop words, changing text to lower case, removing most frequent and least frequent words, and lemmatizing. In this process, the texts are converted to more feasible format. To perform this step, 'NLTK' python library is used.

In addition to this, a new column "label_id" is added which is the encoding of the "label" column in the integer format. This is done because categorical variables must be converted from string to integer.

In the feature generation step, features are extracted for document classification. The features are selected by measuring how important the word is to the document. This is done by calculating the TF-IDF after POS tagging the unigrams. To perform this step, 'sklearn' python library is used.



From the plot it can be observed that the classes are imbalanced however under sampling or oversampling are not performing to balance the dataset because the high prediction accuracy can be achieved for majority classes keeping a sensible prediction accuracy for the minority classes. [3]

2.1 Pre-processing

1. **Case Normalisation**
   All the text is converted to lower case using the lower function in python. This step is done to avoid having the same tokens in different cases.

2. **Tokenisation**

Tokenisation is the task of breaking down a sentence to single units called tokens where token is a sequence of character when grouped together forms semantic unit. Tokens are used to understand the context and develop models for Natural Language Processing. Here, the unnecessary characters and numbers are removed before making tokens using 'RegexpTokenizer'. The tokens of length 1 are also removed as they have no meaning.

3. **Stop Words Removal**

Stop words are common words like 'the', 'is', 'an', 'are', they are available in abundance and they do not give any unique information which can be useful. Hence, the stop words are removed using the NLTK library of python.

4. **Lemmatization**

Lemmatization is the method to return the base form of a token known as lemma. This step is required to reduce the dimensionality by collapsing the similar tokens, by removing suffixes, prefixes and pluralisation. 'WordNetLemmatizer' is used to perform this task.

2.2 Feature generation

- **N-Gram Features**

  N-Gram is a continuous sequence of N words. They are the co-occurring words found in the text. N-rams are used to develop features for the machine learning models. But it does not guarantee any significant improvement in model Accuracy.

  ngram_range= (1,1) indicates only unigram is considered
  ngram_range= (1,2) indicates unigrams and bigrams are considered
  ngram_range= (2,2) indicates only bigrams is considered

  In this task, only unigrams are used as there was no improvement observed in accuracy when bigrams were used.

- **POS Tags**

  Identifying and tagging each word's part of speech in the context of a sentence is called Part-of-Speech Tagging, or POS Tagging. [4]

  The four parts of speech identifies with the use of pos tagging are:
  - Noun
  - Verb
  - Adjective
  - Adverb

  A function is created to identify if it is a Noun or any of the other four parts of speech. Pos tagging is important in data pre-processing as pos tagging is done before lemmatization and lemmatization is done depending on the part of speech of the token.
  This step was removed in the last as the computing time was very high and there was not much improvement in the accuracy of the model.

- **TF-IDF**

  TF-IDF features are generated for the given document abstracts. If the value of idf is close to zero it can be said that the word is common while the value close to 1 says it is rare. This is useful in finding the important relevant words.

  Below are the parameters set in TfidfVectorizer to calculate TF-IDF. [5]

  - **sublinear_df** = true, so that a logarithmic form is used for frequency
  - **ngram_range =** (1,1), only unigrams are considered as bigrams were not increasing the accuracy
  - **analyzer =** word, as the word feature is taken for unigrams
  - **input =** content, as input is expected to be a sequence of items that can be of type string or byte
  - **lowercase = true,** for case normalisation
  - **max_df =** 0.95 and **min_df** = 0.001 (Max_df is the maximum number of documents a word needs to be present in and min_df is vice-versa) The numbers are selected as they were increasing the accuracy.
  - **tokenizer =** lemmatokenizer() In this case tokenizer is class in which token are created and stopwords are removed. Basically, the noise is removed using a regular expression.

3. Models

Before selecting models, the following three points are considered:

1. **Response variable:**
   Since our response variable is discrete, classification models are used.
2. **Classification type:**
   The classification could be either binary classification or multi-class classification. In the given dataset as the number classes for prediction are 100, multi-class classification is performed.
3. **Data set:**
   Since the text needs to be labelled, text classification is performed.

For multi-class classification of textual data, appropriate model must be found. There can be many models for this scenario, but following are the models considered:

- Logistic regression
- Linear support vector machine
- Support vector machine with sigmoid kernel
- Stochastic gradient descent on Linear SVM

Reason for selecting only these models than the other multi-class text classification models is given in details below.

3.1 Model 1 Logistic Regression

Linear Regression is one of the most interpretable and a simple machine learning model but, it is hard to use linear regression for classification because linear regression might produce probabilities less than zero or greater than one. For a response variable with three possible values, say positive, negative, and neutral, coding them as 1,2,3 implies an ordering to the outcomes. So, to solve this issue Logistic regression [6] uses Sigmoid function to change all the values between range 0 to 1 and

then, it can easily find class/label based on the threshold. Multi-class classification can be used for one-vs-all or one-vs-one classification approach.
For classification problem in machine learning, logistic regression is widely used because it is easy to understand and is accepted as a good start in natural language processing. [7]

3.2 Model 2 Support Vector Machine with nonlinear Kernel

Support vector machine is based on support vector classifier and support vector classifier is based on maximal margin classifier. Basically, maximal margin classifier tries to find the hyperplane that makes the biggest gap or margin between the two classes but maximal margin classifier is a hard classifier as it does not allow miss classification, so new improvement was support vector classifier which is a soft classifier as it allows a few mis classifications. But to do non-linear classification, SVC was upgraded to SVM (support vector machine) because it is flexible and can make nonlinear boundaries using kernels. With kernel in SVM, nonlinearity can be introduced in support-vector classifiers in a more elegant and controlled way.
SVM is a powerful model for text classification. Therefore, SVM is used. [8]

3.3 Model 3 Linear SVM

Linear SVM [9] is a type of SVM in which the kernel of SVM is linear. Basically, kernel can take any form and this property makes SVM nonlinear. Non-linear kernel is beneficial when it is known that nonlinear boundary is needed. Linear SVM is used as the boundary type is not known.

3.3 Model 4 Stochastic gradient descent on Linear SVM

Stochastic gradient descent [10] is a method for optimising any algorithm or model by iterating. In Stochastic gradient descent, stochastic approximation [11] of gradient descent optimization is used. This stochastic approximation of gradient descent is used to reduce heavy computational in gradient descent. As in the previous model linear SVM is used, and the required accuracy was not achieved, Linear SVM is optimised using Stochastic gradient descent.

3.4 Discussion of model difference(s)

Stochastic gradient descent is just an optimisation algorithm, and it is used on linear SVM. And SVM is also used with linear and nonlinear kernel. So, basically there are two models to discuss, Logistic Regression and Support Vector Machine. Below is the discussion of how Stochastic gradient descent improving SVM at the end and before that the differences of SVM with linear and non-linear kernel, are discussed.

Characteristic	**Logistic Regression**	**Support Vector Machine**
General Idea	It has different decision boundaries which are near to the optimal point with different weights.	SVM tries to find maximum margin between the hyperplane and support vectors. Thus, SVM find a solution which is as fare as possible for the two categories while LR has not this property.
Outliers	It is more sensitive to outliers.	It is less sensitive to outliers, we can control SVM sensitivity to outliers.
Outcome	It produces probabilistic values and we set threshold accordingly for 0 or 1.	It produces 0 or 1.

Efficiency	It is computational efficient.	It takes more computation resource that Logistic Regression.
General use	Generally used for linear classification.	Can be used for linear and non-linear boundaries.

The major difference between SVM with linear kernel and SVM with non-linear kernel is that SVM with nonlinear kernel draws non-linear boundaries and hence it can be used for nonlinear classifications but in linear kernel SVM, only linear boundaries are used. [12]
Last thing is how Stochastic gradient descent on Linear SVM makes it different from Linear SVM. Stochastic gradient descent on Linear SVM optimises Linear SVM and improves its performance using stochastic approximation of gradient descent optimization.

4 Experiment setups

Below are the steps for building the models:

(a) Finding hyper para meters for each model to do regularisation and bias variance trade off. And then GridSearchCV function with 10 cross validation folds is used to find perfect tuned parameters.

Selected hyper parameters for each model to do regularisation and bias variance trade off.	**Used GridSearchCV function with 10 cross validation folds to find perfect tuned parameters.**
Logistic Regression ➢ Penalty: ['l1', 'l2', 'elasticnet', none] ➢ C: [0.001, 0.01, 0.1, 1, 10, 100]	**Logistic Regression** ➢ Penalty: ['l2'] ➢ C: [1]
SVM with non-linear Kernel ➢ Kernel: [‘poly’, ‘rbf’, ‘sigmoid’] Based on Kernel other parameters: ➢ C = [0.001, 0.01, 0.1, 1, 10, 100] ➢ Gamma = [‘scale’, ‘auto’, 0.01, 0.1, 1] ➢ Degree (if poly) = [2, 3, 4, 5]	**SVM with non-linear Kernel** ➢ Kernel: [‘sigmoid’] Based on Kernel other parameters: ➢ C = [1] ➢ Gamma = [‘scale’]
LinearSVM ➢ Loss: [‘hinge’, ‘squared_hinge’] ➢ Penalty: ['l1', 'l2']	**LinearSVM** ➢ Loss: [‘squared_hinge’] ➢ Penalty: ['l2']
Stochastic gradient descent on Linear SVM ➢ Warm_start: [True, False] ➢ Average: [True, False] ➢ Max_iter: [1000, 2000, 4000, 6000, 8000] ➢ Penalty: [l1, l2, 'elasticnet']	**Stochastic gradient descent on Linear SVM** ➢ Warm_start: [True] ➢ Average: [True] ➢ Max_iter: [8000] ➢ Penalty: [l2]

❖ Brief introduction about hyper parameters:
- **Penalty:** It is a regularization parameter, used to give penalty if model complexity increases.
  - l1 – Lasso Regression
  - l2 – Ridge Regression
  - elasticnet – Elasticnet Regression
- **C:** It is inverse of regularisation parameter, so smaller the value of C, stronger the regularisation will be.

- **Kernel:** These are mathematical functions which if used with SVM, can introduce nonlinearity in support-vector classifiers in a more elegant and controlled way. [12]
    - **Poly:** It is a polynomial function; we can set its degree.
    - **Rbf**: Radial kernel, its mathematical form is: $\exp(-\gamma||x-x'||\hat{}2)$, where γ is specified by gamma.
    - **Sigmoid**: Sigmoid kernel, its mathematical form is: $\tanh(\gamma<x,x'>+r)$ where r is specified by coef0.

- **Warm_start:** if True, then reuse last state to fit initialisation on previous call, if False then remove last solution. [10]
- **Average:** computes average SGD weights if True
- **Max_iter:** On training data it represents maximum number of passes

(b) Calculated Accuracy of all the tuned models and then compared them. And finally selected the best model which is SVM with sigmoid kernel.

Extras:

- In linear SVM one-vs-all multiclass classification approach is used, and it is known that it is time consuming and computationally heavy to use one-vs-all when data is big but Linear SVM is the simplest as compared to other SVMs so one-vs-all is tried for better results.
- In SVM with Sigmoid Kernel one-vs-one approach for multiclass classification is used because SVM with Sigmoid Kernel is computational heavy and using one-vs-all will make it more complicated. Hence, one-vs-one in SVM with Sigmoid kernel is used.

5 Experimental results

The prediction was performed on the test data using the following four models:

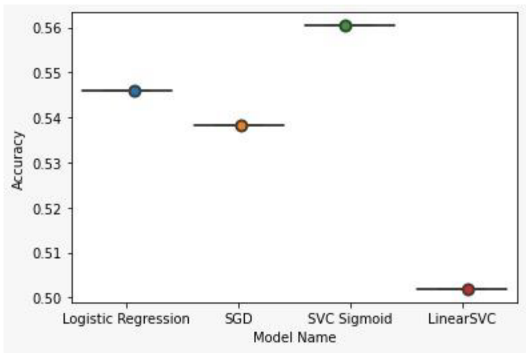
1. Linear SVC
2. Logistic Regression
3. SGD Classifier
4. SVC Sigmoid Classifier

The table below compares the above four models based on several values as follows:

Characteristic	Linear SVC	Logistic Regression	SGD Classifier	SVC Sigmoid Classifier
Accuracy	0.5019	0.5459	0.5382	0.5604
Precision	0.2483	0.2725	0.2483	0.2743
Recall	0.2148	0.1823	0.2085	0.2238
F1-score	0.2225	0.1989	0.2187	0.2345

Note: Above values are generated on test data
Accuracy – It is the number of correct predictions / numbers of all predictions
Precision – Probability of True positives over Total number of True positives and False Positive
Recall – Ratio of positive classes correctly detected.
F1score – Weighted average score of recall and precision

Below is a graph of the accuracy on the train data for all the models.



From the above graph it can be said that the accuracy of the SVC sigmoid classifier is the highest, therefore that is the selected model for this dataset.

6. Conclusion

To summarise,

- The dataset was trained with four classification models and during classification, different parameters were tested for the four models used. The parameter setting was done based on the accuracy of the models for different parameters. As it can be seen from the boxplot given above that the test accuracy is best when SVC sigmoid classifier is used for classification. The difference between the accuracy of the three models is clearly noticeable. Linear SVC has the least accuracy among the four models.
- Before applying any classification model, the main task is data pre-processing and feature extraction. The accuracy of the model mainly depends on these two tasks. The focus was to extract the best features for the model so that high accuracy can be achieved.
- Among the four models tested, SVC sigmoid classifier was the best one and the other model that was close to SVC sigmoid was Logistic Regression

For the future purpose, Neural network model with Convolutional Neural Network or Recurrent Neural Network can be used.

References

[1] “Fine-Grained Classification,” [Online]. Available: http://www.cs.umd.edu/~djacobs/CMSC733/FineGrainedClassification.pdf. [Accessed 22 10 2020].

[2] scikit, “6.2. Feature extraction,” [Online]. Available: https://scikit-learn.org/stable/modules/feature_extraction.html. [Accessed 3 11 2020].

[3] S. Li, “Multi-Class Text Classification with Scikit-Learn,” [Online]. Available: https://towardsdatascience.com/multi-class-text-classification-with-scikit-learn-12f1e60e0a9f. [Accessed 3 11 2020].

[4] K. X. Neo, “Using POS tag to aid textual data pre-processing,” [Online]. Available: https://www.kaggle.com/neokaixiang89/using-pos-tag-to-aid-textual-data-pre-processing. [Accessed 1 11 2020].

[5] scikit, “sklearn.feature_extraction.text.TfidfVectorize,” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html. [Accessed 25 10 2020].

[6] Scikit, “sklearn.linear_model.LogisticRegression¶,” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html. [Accessed 2 10].

[7] K. Ganesan, “A General Supervised Approach to Segmentation of Clinical Texts,” [Online]. Available: https://kavita-ganesan.com/wp-content/uploads/2017/10/stat-segment.pdf. [Accessed 4 11 2020].

[8] Wikipedia, “Support vector machine,” [Online]. Available: https://en.wikipedia.org/wiki/Support_vector_machine#Applications. [Accessed 4 11 2020].

[9] Scikit, “sklearn.svm.LinearSVC,” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html. [Accessed 1 11 2020].

[10] Scikit, “sklearn.linear_model.SGDClassifier,” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html. [Accessed 3 11 2020].

[11] Wikipedia, “Stochastic approximation,” [Online]. Available: https://en.wikipedia.org/wiki/Stochastic_approximation. [Accessed 3 11 2020].

[12] Scikit, “1.4. Support Vector Machines,” [Online]. Available: https://scikit-learn.org/stable/modules/svm.html. [Accessed 5 11 2020].