UNSW
AUSTRALIA

Group Project

COMP9417 Machine Learning and Data Mining

T1, 2020

Group: MLife

Members: z5205896 Cristian Bernal

Z5206808 Simon Flodin

z3442505 Dung Anh Nguyen

z5281578 Chuang-Yin Wang

z3290805 Pei Wang

University of New South Wales

Faculty of Computer Science and Engineering

April 2020

Introduction

The aim of this project is to build, compare, evaluate and analyze text classification models in order to take in unlabelled articles, classify them and predict the most relevant articles for each of 10 provided topics in the context of news articles being suggested to users.

The problem we are trying to address in this project is to derive machine learning models for classifying articles and provide most relevant article recommendations. Several models are subjected to be built from 9500 training samples, with a portion used for validation. Then the model would be tested against the testing data which contains 500 samples. Evaluation metrics would also be generated for comparison and evaluation. Then provide recommendations of each article type accordingly.

The main challenges we face include:

- Technical knowledge on the application of the SKLearn libraries
    - Transform the list of words into features
    - Building and fitting data into models that incorporate multilabel classifications
    - Generation of evaluation metrics
    - Find most relevant articles from the test data
- Our choice of models most suitable for text classification
- Comparing and evaluating the models in order to find the one with the most competitive performance based on the metrics generated
- Future improvements on the approach
- Collaboration and organisation between team members
- Time management over the course of the project

Exploratory data analysis

Our approach started with a discussion with members of the team sharing different ideas for our approach to the problem. We began noticing diverging views and ideas through this discussion and decided to split up the code with different raw data classifiers to start.

It was found there are almost 36,000 distinctive words in the training data set. As such having them all as individual features is definitely not feasible. An effort to create a subset of these words were made selecting the most frequent words in each sample. The built model did not perform as well as the one built from using Vectorizers upon validation.

In addition, the ‘Irrelevant’ type which takes 48% of training data was another issue to be addressed. It is covered in more details in the following section.

Methodology

Pre-processing

The first step would be preprocessing the data. It was noticed that the ‘Irrelevant’ type takes a significant portion of the whole dataset, more specifically 48%. Having this proportion is troublesome because it leads to imbalanced dataset and insufficient training on the article types that actually matters. Using un-preprocessed data would likely lead to poorly trained models. Hence, the samples of the 10 article types in the training data set were duplicated 3 times to create a bag of words, to size of over 19,000 samples. The improved model enjoyed a significant better performance as shown in the result section later.

During this step, the group has also experimented with CountVectorizer and TfidfVectorizer which are existing libraries built in for text processing. They yielded much better performance than efforts to reinvent the wheel. Experiments were also done on certain parameters such as “stop_words” which was expecting to have improvements on the result. Later on it was found that the provided data may already have been processed. Hence the application of such hyperparameters did not benefit the result.

Choice of Models

Three models have been chosen for analysis: naive bayes, decision tree and random forest.

Naive Bayes

- High performance in multi-class classification
- Insensitive towards irrelevant features
- Easy to implement and fast to build the model
- Requires less training data

There are two kinds of NB based text classifiers. The first one is called multivariate Bernoulli NB, which uses a binary vector to represent whether a term is present or not. The second one is multinomial NB, which counts how many times a term appears in the document. Both Bernoulli NB and Multinomial NB for text classifiers were used.

Decision Tree

- Works well with categorical data.
- Once built, it is able to quickly generate results. This is advantageous in a practical setting for faster search times. So if it were to work well a decision tree would be a useful methodology to use.

- Handles noisy data automatically which is very useful for this task since a lot of the data is irrelevant.

Random Forest

- Introduce extra variability in the learners and more randomness to the procedure
- For every learner, only a subset of features are used
    - This creates diversity
- Take average/majority vote over trees (learners)

Support Vector Classifier

While doing research, it was found the vectorizer objects provided by Scikit-Learn are quite reliable right out of the box, they allow us to perform all the above steps at once efficiently, and even apply preprocessing and rules regarding the number and frequency of tokens. Different vectorizers were applied with their respective results compared. The result indicates that CountVectorizer produced the best performance.

Evaluation Metrics

In our study, we have considered Precision Score, Recall Score and F1 Score instead of Accuracy Score. A model trained with an imbalanced dataset would predict all the results to be the major class which is a counterproductive but it would still get a high accuracy score. Only considering accuracy scores leads to poor performance at classifying minor classes. Therefore we evaluated the performance of models with their Precision Score, Recall Score and F1 Score instead of accuracy.

Finding Most Relevant Articles

The method used for finding most relevant articles is using predict_proba, which returns estimates for all classes ordered by the label of classes. The probabilities of the predicted samples are collected and sorted in reversed order. The top 10 articles of each type would be the top 10 predicted samples with the highest probabilities to each article type respectively.

Results

Model building

We first built our models with different methods and evaluated the performance of the data (without doing any data pre-processing) by averaging the scores of K-fold cross validation. Separated our training set into 10 folds so we can use the whole training set, get more metrics and draw important conclusions both about our model and our data. Here are the result for different methods:

bernoulliNB using raw data: accuracy: 0.7038942167520311 precision: 0.37212369276862634 recall: 0.2819154032170111 f1: 0.28388746010825894	**multinomailNB using raw data:** accuracy: 0.7358161832731782 precision: 0.6306736067435463 recall: 0.5526188603489018 f1: 0.5583352913640214
Decision Tree using raw data: accuracy: 0.724 precision: 0.5422977915506378 recall: 0.5555469797174604 f1: 0.5253106096020288	**SVM using raw data:** accuracy: 0.746 precision: 0.47611676793583707 recall: 0.34186073270741457 f1: 0.3722400682974134
Random Forest using raw data: accuracy: 0.734 precision: 0.4380108551371127 recall: 0.29694006413113966 f1: 0.31540531381487563	

Among all the results, MultinomialNB and Decision Tree had the best performance from the raw data.

Pre-processing

We reshaped the training datasets to reduce the effect of imbalanced data. Our approach started with both reducing the irrelevant class and increasing the minor classes.

reducing irrelevant class data (multinomialNB) accuracy: 0.7140522329677184 precision: 0.7135826179053613 recall: 0.6052460319053538 f1: 0.624984852475951	**reducing irrelevant class data (decision tree)** accuracy: 0.656 precision: 0.44052385756425905 recall: 0.5227159758624289 f1: 0.45697979934275923
increasing the minor classes data(multinomialNB) accuracy: 0.8046992057147376 precision: 0.8135680332176245 recall: 0.8751822879751108 f1: 0.8346417211219871	**increasing the minor classes data(decision tree)** accuracy: 0.678 precision: 0.4538188280956876 recall: 0.4951380522604779 f1: 0.44482902224883564

Instead of using bag-of-words models, our team also used a tf-idf model to represent the words of the article. Here are the result using different text vectorize methods:

multinomialNB using Bag-of-words accuracy: 0.8046992057147376 precision: 0.8135680332176245 recall: 0.8751822879751108 f1: 0.8346417211219871	**multinomialNB using tf-idf** accuracy: 0.6519025860434599 precision: 0.6564512514734997 recall: 0.28690327479646266 f1: 0.27558590791252524

Final Result

We chose multinomialNB for our final model, with increased minor classes data to build the model and used bag-of-words for text vectors. Here are the final result for each class on testset:

Topic name	Precision	Recall	F1
ARTS CULTURE ENTERTAINMENT	0.29	0.67	0.40
BIOGRAPHIES PERSONALITIES PEOPLE	0.64	0.47	0.54
DEFENCE	0.41	0.92	0.57
DOMESTIC MARKETS	0.14	0.50	0.22

FOREX MARKETS	0.41	0.60	0.49
HEALTH	0.68	0.93	0.79
MONEY MARKETS	0.49	0.71	0.58
SCIENCE AND TECHNOLOGY	1.00	0.33	0.50
SHARE LISTINGS	0.46	0.86	0.60
SPORTS	0.95	1.00	0.98

Final Recommendation

Topic name	Suggested Articles	Precision	Recall	F1
ARTS CULTURE ENTERTAINMENT	9604, 9789, 9830, 9952, 9526, 9703, 9933	0.29	0.67	0.40
BIOGRAPHIES PERSONALITIES PEOPLE	9724, 9758, 9797, 9878, 9940, 9896, 9645, 9988, 9854, 9956	0.7	0.47	0.56
DEFENCE	9559, 9576, 9579, 9607, 9616, 9731, 9739, 9741, 9770, 9773	0.6	0.46	0.52
DOMESTIC MARKETS	9923, 9989, 9796, 9640, 9762, 9767	0.17	0.50	0.25
FOREX MARKETS	9572, 9577, 9584, 9625, 9693, 9711 9727, 9743, 9823, 9704	0.40	0.08	0.13
HEALTH	9621, 9661, 9735, 9807, 9873, 9929 9937, 9947, 9982, 9810	0.8	0.57	0.66
MONEY MARKETS	9516, 9528, 9534, 9553, 9583, 9586 9602, 9723, 9737,9755	0.7	0.10	0.175
SCIENCE AND TECHNOLOGY	9722	1.00	0.33	0.50
SHARE LISTINGS	9601, 9666, 9972, 9667, 9518, 9501 9562, 9999, 9867, 9846	0.60	0.86	0.71
SPORTS	9508, 9513, 9514, 9520, 9536, 9541	0.9	0.15	0.26

	9568, 9569, 9573, 9574			

Discussion

Methods

After comparing all the methods we used, multinomialNB and decision tree returned the best performance in cross validation, but the precision score, recall score and f1 score are still comparatively low, which could be improved to get to an acceptable standard. Thus we tried to reduce the effect of imbalanced data and found out that by increasing the minor classes data, the performance of multinomialNB was enhanced greatly. We also attempted to use TF-IDF for different text vectors, but it did not work well in our case compared to the CountVectorizer.

Metrics

Among all the metrics, we think that precision score is more important than others. By definition,

$$precision = \frac{tp}{tp + fp}$$

tp is an outcome where the model correctly predicts the positive class while fp is an outcome where the model incorrectly predicts the positive class. In this project our goal is to recommend users the articles they want as correct as possible,which means that if the false positive values are minimized, there will be less chance our model would miss predict. Having the model with a high precision score is more probable in giving the right recommendation to users.

For the final result table, despite science and technology having a precision of 100%, it was the result of a single prediction. More test cases of the class would be welcomed. Sport is the next best performing class with a high precision of 0.94. This also indicates our model is properly trained at handling sport class of article and/or such class uses a more distinctive set of words compared to the rest. On the other hand, Domestic market and Entertainment are two classes with less than satisfying results with precisions of 0.17 and 0.29. We could generate better models if we have more training samples of these classes.

The metrics for the recommendation table are slightly different from the metrics for the final result. Some of the precision scores tend to be higher because it is easier for a model to pick top ten most possible samples to be that class than to pick all the possible samples in that class. The model would have lower fp thus the precision score would be higher. However the recall scores of some classes drop dramatically. That is because in a recommendation table there can only be 10 items to be picked, and lots of samples which are actually true are treated as fn in these cases. But in our work we care much about precision score so we can neglect these small drops of the recall score.

Future Improvements

Given the opportunity to further work on this project in future, we attempt to put more time towards experimenting with the hyperparameters of the classifiers used. As of submission, we have primarily improved our results through preprocessing methods and by applying different classifiers to our data. Modifying hyperparameters for the various classifiers such as min_samples_leaf/split for the Decision Tree may have improved its performance alongside our preprocessed data but was not explored thoroughly due to time constraints. Additionally as previously mentioned, certain article types are not performing well despite a satisfying overall performance. Efforts have been made to mitigate the lack of samples of valid article types by duplicating existing training data. It would still be welcomed to have more real samples to generate better performing models. Lastly, only the collective metric generated by the test dataset has been used. Another way to improve is to put more weight on the miss classified article types and have the model retrained.

Conclusion

This project had our group generate a machine learning model for classifying and recommending news articles to users on a variety of topics from an input of text. Our methodology primarily involved pre-processing the data in a variety of ways and combinations, such as duplicating relevant articles, decreasing the number of irrelevant articles and experimenting with stop words, text vectorizers and so on. We then applied a variety of different classifiers to solve the problem with a main focus on Naive Bayes, Decision Tree and Random Forest. Our final model was a multinomialNB classifier with increased minor classes using countVectorizer to transform the text as it showed the best results in comparison.

Aside from the technical knowledge learned and applied, we learned that machine learning is a practice that requires both knowledge and experimentation. This project simulates a scenario in which much research and self driven learning needs to be done in order to have the most efficient approach, in this case the use of vectorizer. When it comes to selecting classifiers, they have to be compared with metrics to find out the best performing one for the task. While achieving a positive outcome, there is still room for further improvement given more time for further experimentation.

Reference

José Blanco, Feb 16, 2018, Hacking Scikit-Learn's Vectorizers, towards data science, viewed 26 April 2020, <https://towardsdatascience.com/hacking-scikit-learns-vectorizers-9ef26a7170af>

Various authors, 2019, API Reference, scikit-learn developers, viewed 26 April 2020 <https://scikit-learn.org/stable/modules/classes.html>

Chris Albon, 2020, Bernoulli Naive Bayes Classifier, chrisalbon.com, viewed 26 April 2020 <https://chrisalbon.com/machine_learning/naive_bayes/bernoulli_naive_bayes_classifier/>

Geeksforgeeks, 2018, Removing stop words with NLTK in Python, geeksforgeeks.org, viewed 26 April 2020 <https://www.geeksforgeeks.org/removing-stop-words-nltk-python/>

Ezzaldin Mamdouh 2020, Naive Bayes Classifier, Kaggle.com, accessed 23 April 2020 <https://www.kaggle.com/ezzaldin6/naive-bayes-classifier?scriptVersionId=29386958>

Jason Brownlee, July 26, 2017, What is the Difference Between a Parameter and a Hyperparameter?, machinelearningmastery.com, viewed 26 April 2020 <https://machinelearningmastery.com/difference-between-a-parameter-and-a-hyperparameter/>