ZPJa: Naive Bayes to Neural Text Classification Report

Alexandra Fernandes
242963@vutbr.cz

Santosh Kesiraju
kesiraju@fit.vutbr.cz

Abstract

Text being a way of communication is rich in many ways. It may be a complex task to evaluate what one may mean with a certain sentence, associating it with a sentiment or just figuring out a topic from a body of text or several text fields. This is exactly what Text Classification covers and in this work three approaches will be explored: Naïve Bayes, Logistic Regression and Convolutional Neural Network. Using the Logistic Regression model the impact of using different feature representations, different number of fields during the classification and getting the top k classifications will be examined.

1 Introduction

Like it is written in the abstract, text is a rich and complex way of communication given its unstructured nature. One may want to classify text for different applications such as sentiment analysis, intent detection, topic labeling and spam detection.

These are all useful applications of the text classification task, lifting the weight of doing these classification in a time-consuming way, such as doing it by hand like it was done in older times. For instance, sentiment analysis is useful to extract positive or negative impressions from reviews and these insights can be used by users or companies for statistical purposes. Spam detection avoids users being swarmed by meaningless information that clutters email inboxs. Just like sentiment analysis, topic labeling is useful to extract several information related to user behaviour for example.

2 Task Definition

The NLP task that will be covered in this work is Text Classification, focusing on topic labelling. This taks basically consists in giving a text input and then the model will output a label or set of labels (or tags), as can be seen in figure 1.

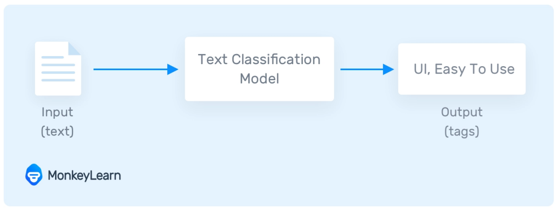


Figure 1: Text Classification, source:(tex, a)

Three different methods will be used: Naïve Bayes, Logistic Regression and Convolutional Neural Network. Using Logistic Regression, it will be explored the impact of using different feature representations, specifically Binary weighting, Term-frequency of words (or just counts, for simplicity) and Term frequency–inverse document frequency (TF-IDF). Additionally, the impact of different number of fields during the classification and also how much influence it makes to get the top k classifications, instead of just one. To clarify, getting the top k-classifications applies when data can belong to more than one topic, getting more than one label after classification correspond to hits that can be ranked, which means if the classifier identifies several correct labels, but one might be more related to the text's body than the following labels. This will be explained better further.

3 Method

There are several approaches to this task such as Naïve Bayes, Logistic Regression, Convolutional Neural Network (CNN) which will be used in this work. Software from the following sources were used and then changed for the purpose of this work, all the source code can be found on (nb; lr; cnn, a). The code after altering can be found in the following github repository (rep).

The 20 Newsgroups and HuffPost datasets were used for Topic Classification/Labelling. The Yelp, Amazon and imbd datasets for Sentiment Analysis just to see how the CNN model performs in general.

4 Experimental Setup

4.1 Environment

The Anaconda Navigator was used to easily create virtual environments and the Spyder editor was used when dealing with the Naïve Bayes and Logistic Regression models. It was necessary to create two virtual environments due to packages' compatibility, one with python 3.7.11 and the other has 3.8.12. Besides this the kaggle website was used since it allows to use jupyter notebooks and have more computing power when dealing with the CNN model. Each session on kaggle can last up to 9h, it has 73.1GB of space available and 16GB of RAM. The reason for using two different environments is that I only found the second environment in a late stage of the development of the project.

4.2 Datasets

Different datasets were used to experiment and evaluate different aspects and hypothesis.

- **20 Newsgroups** (20n, a) has just one field that the full body of text of 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups.

  This dataset was used to evaluate the impact of different feature representation and the top-k classifications analysis.

- **HuffPost** (huf) contains around 200k HuffPost articles and has several fields available, namely:

  - category: Category article belongs to
  - headline: Headline of the article
  - authors: Author of the the article
  - url: Link to the post
  - short_description: Short description of the article
  - date: Date the article was published

  This dataset will be used to evaluate the impact of using more than one field during classification. The fields that are going to be used are: headline, url and the short description.

- **Yelp, Amazon and IMBD** (yel; ama; imb) are datasets that consist on reviews from users and have a positive (1) or negative (0) sentiment label associated with it. This data is read from txt file in the kaggle environment, then is turned into a dataframe with sentences and labels for columns.

  This dataset is used to see how CNN's work and try a bit of hyperparameter tuning.

4.3 Metrics

A description for the metrics used can be found on the appendix A. The focus will be on Accuracy, F1 Score and Mean Reciprocal Rank (MRR). Different metrics were used according to the data used and what hypothesis is being tested.

4.4 Feature Representations

One of the things that is going to be explored is the impact of using different feature representations. These are the three that are going to be examined:

- Binary weighting - if a word is present in a document, the weight is '1', otherwise the weight is '0'.

- Term-frequency of words (counts) - the frequency of a word becomes its corresponding weight.

- Term frequency–inverse document frequency (TF-IDF) - words that are unique to a particular document have higher weights compared to words that are less common across documents.

4.5 Models

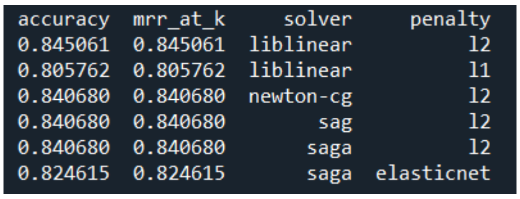
4.5.1 Naïve Bayes (NB)

Starting with NB, the 20 Newsgroup dataset was used and Spyder. The scikit-learn Multinomial Naive Bayes was used (link). 20 Newsgroup data was fetched using scikit-learn.datasets package, meaning that it already comes separated into training and testing sets. It has 11314 training samples and 7532 test samples, across 20 classes. The feature representation chosen was TF-IDF, because overall it seems better since it values more unique words to documents.

4.5.2 Logistic Regression(LR)

For LR, the 20 Newsgroup and HuffPost datasets were used. The first was fetched the same way as the previous model. The second is uploaded from a json file that was obtained from the repository of (lr) (direct link to dataset in the repository).

The scikit-learn Logistic Regression was used (link). The hyperparameters were set to verbose=1, solver='liblinear',random_state=0, C=5, penalty='l2',max_iter=1000. According to the package's page, verbose parameter is just to print more detailed information on the terminal, solver is the algorithm to use in the optimization problem and penalty is the specific norm of the penalty used. There are a few combinations of solver and penalty that were tried (the results are on 2). Also C corresponds to the inverse of regularization strength, so a small C was chosen to avoid overfitting. These will be the hyperparameter used during the experiments.



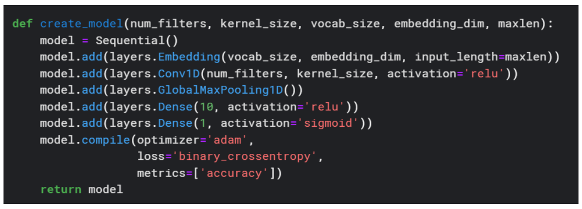
accuracy	mrr_at_k	solver	penalty
0.845061	0.845061	liblinear	l2
0.805762	0.805762	liblinear	l1
0.840680	0.840680	newton-cg	l2
0.840680	0.840680	sag	l2
0.840680	0.840680	saga	l2
0.824615	0.824615	saga	elasticnet

Figure 2: Accuracy for different combinations of solver and penalty

4.5.3 Convolutional Neural Network(CNN)

Finally, with the CNN the 20 Newsgroup and Yelp, Amazon and IMBD datasets were used with the kaggle website.

The hyperparameters to tune are the number of neurons, activation function, optimizer, learning rate, batch size, and epochs. Using the three last datasets mentioned for sentiment analysis, tuning was done using RandomizedSearchCV from scikit-learn package. The values explored were: number of filters=[32, 64, 128] and kernel size=[3, 5, 7]. The structure of the network is:

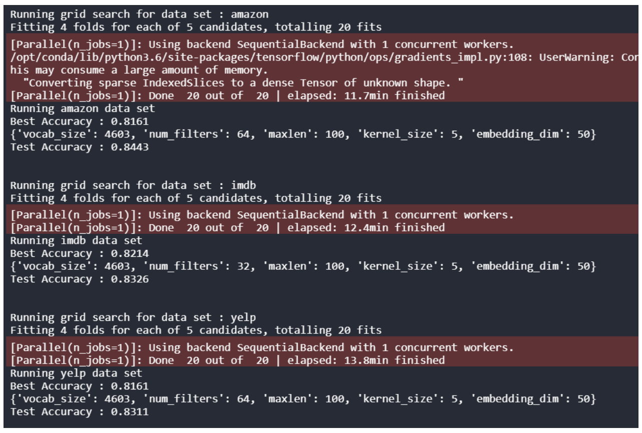


```
def create_model(num_filters, kernel_size, vocab_size, embedding_dim, maxlen):
    model = Sequential()
    model.add(layers.Embedding(vocab_size, embedding_dim, input_length=maxlen))
    model.add(layers.Conv1D(num_filters, kernel_size, activation='relu'))
    model.add(layers.GlobalMaxPooling1D())
    model.add(layers.Dense(10, activation='relu'))
    model.add(layers.Dense(1, activation='sigmoid'))
    model.compile(optimizer='adam',
                  loss='binary_crossentropy',
                  metrics=['accuracy'])
    return model
```

Figure 3: Structure of the network used on sentiment analysis

The results of this grid search for each dataset can be found on figure 4. An attempt to do it also with the 20 Newsgroup but it is unfinished.

The structure of the CNN used for the 20 Newsgroups is shown image 5, please note that this network was not designed by me.



```
Running grid search for data set : amazon
Fitting 4 folds for each of 5 candidates, totalling 20 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
/opt/conda/lib/python3.6/site-packages/tensorflow/python/ops/gradients_impl.py:108: UserWarning: Cor
his may consume a large amount of memory.
  "Converting sparse IndexedSlices to a dense Tensor of unknown shape. "
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed: 11.7min finished
Running amazon data set
Best Accuracy : 0.8161
{'vocab_size': 4603, 'num_filters': 64, 'maxlen': 100, 'kernel_size': 5, 'embedding_dim': 50}
Test Accuracy : 0.8443


Running grid search for data set : imdb
Fitting 4 folds for each of 5 candidates, totalling 20 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed: 12.4min finished
Running imdb data set
Best Accuracy : 0.8214
{'vocab_size': 4603, 'num_filters': 32, 'maxlen': 100, 'kernel_size': 5, 'embedding_dim': 50}
Test Accuracy : 0.8326


Running grid search for data set : yelp
Fitting 4 folds for each of 5 candidates, totalling 20 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed: 13.8min finished
Running yelp data set
Best Accuracy : 0.8161
{'vocab_size': 4603, 'num_filters': 64, 'maxlen': 100, 'kernel_size': 5, 'embedding_dim': 50}
Test Accuracy : 0.8311
```

Figure 4: Results of grid search for sentiment analysis

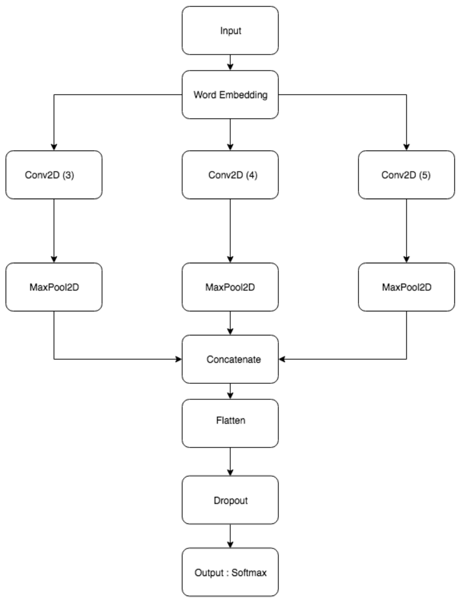


Figure 5: Structure of the network used on topic labelling

In the source codes there is an explanation of the structure of each neural network, since I did not alter this code much I will just leave a reference and use their results (cnn, a,b). In the github repository ((rep)) of this project it is possible to check the attempt that was made.

The CNN will only be used to see a different approach, supposedly better and to try out the tuning of hyperparameters (even if is was with different datasets used on the previous models).

5 Results and Analysis

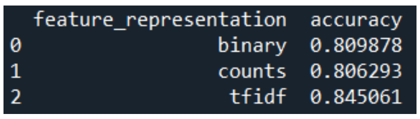
5.0.1 Feature representations

Starting with the impact of feature representations, for this study the 20 Newsgroup dataset was used.

Looking at the confusion matrixes on appendix

B, it is possible to see that all of the confusion matrices obtained (the values correspond to hits) are sparse matrices, having greater values in the main diagonal. This means that all of them do not make many misclassifications. Just looking at these matrices is not enough to get a good overview of the models, so other metrics were used. Since 20 Newsgroup is pretty much a balanced dataset, it is possible to look at other metrics like accuracy or F1-score.

On appendix C it is possible to see a more extensive look into the results of each feature representation, taking into account the macro-average of accuracy and f1-score, it can be infered that the best one is TF-IDF. Additionaly, on a more summarized table, figure 6, it is shown that the representation that gets best acccuracy is TF-IDF.



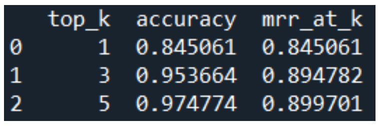
```
  feature_representation  accuracy
0                 binary  0.809878
1                 counts  0.806293
2                  tfidf  0.845061
```

Figure 6: Accuracy of different representations

I think this was expected since there are some unique words to some documents that should have more weight/stand out somehow, and TF-IDF has that into account.

5.0.2 Top k-classifications

Moving onto the impact of top k-classification, also using the 20 Newsgroup dataset. Two metrics were taken into account: accuracy and MRR (check A for explanation MRR), the reasoning for this will be explained next. The trials were done using the IF-IDF feature representation and the values tried are k=[1,3,5].

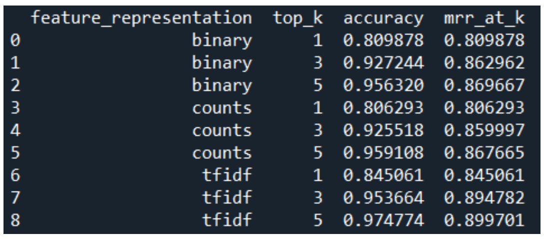


```
   top_k  accuracy  mrr_at_k
0      1  0.845061  0.845061
1      3  0.953664  0.894782
2      5  0.974774  0.899701
```

Figure 7: Accuracy and MRR of different k for top-k classification

Even though the accuracy is higher using the top-5 classification, as can be seen on figure 7, it is important to note that the top is unranked, so the first label has the same weight as the third label when computing this accuracy. But the MRR metrics takes the rank of the first correct answer into consideration, so the higher the rank of the correctly predicted category/label, the higher the MRR. Having this in mind, it seems the greater the k, better are the results. This makes sense because usually one body of text may belong to more than one category and the more it is know about a text the more it is identifiable. Just like an animal for instance, knowing more and more about an individual and more individuals allows for identifying different categories more easily.

Varying both these two variables leads to the results on figure 8. As expected the best results were achieved using TF-IDF and k equal to 5.



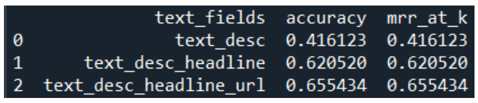
```
  feature_representation  top_k  accuracy  mrr_at_k
0                 binary      1  0.809878  0.809878
1                 binary      3  0.927244  0.862962
2                 binary      5  0.956320  0.869667
3                 counts      1  0.806293  0.806293
4                 counts      3  0.925518  0.859997
5                 counts      5  0.959108  0.867665
6                  tfidf      1  0.845061  0.845061
7                  tfidf      3  0.953664  0.894782
8                  tfidf      5  0.974774  0.899701
```

Figure 8: Accuracy and MRR of different feature representations and k for top-k classification

5.0.3 Number of fields during the classification

Next analysing the impact of the number of fields during classification, the Huffpost dataset was for the reasons explained before - it has several fields available. The metrics used are accuracy and MRR.

Using the top-1 classification and trying the TF-IDF, the best result was achieved with more fields as shown in figure 9. o it seems using more fields is better, when not analysing the articles themselves.

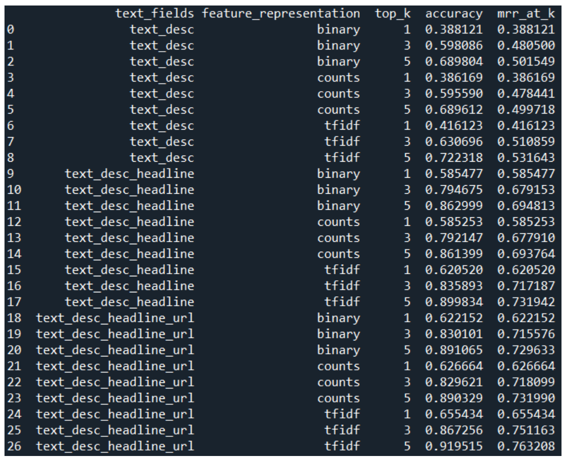


```
            text_fields  accuracy  mrr_at_k
0             text_desc  0.416123  0.416123
1    text_desc_headline  0.620520  0.620520
2  text_desc_headline_url  0.655434  0.655434
```

Figure 9: Accuracy and MRR with HuffPost dataset, k=1 and TF-IDF

Now trying several combinations of k values and feature representations, it is possible to check that with more fields analysed the better the MRR (also the accuracy, but as explained before, it does not take into account the rank of labels). As can be seen in figure 10, the best result is achieved with TF-IDF, k=5 and the three fields chosen.

This aligns with what was seen in the previous trials and it makes sense that looking at more fields gets better results, since just like a person or animal for instance, knowing more and more about an individual allows for identifying it more easily.



```
                 text_fields feature_representation  top_k  accuracy  mrr_at_k
0                  text_desc                 binary      1  0.388121  0.388121
1                  text_desc                 binary      3  0.598086  0.480500
2                  text_desc                 binary      5  0.689804  0.501549
3                  text_desc                 counts      1  0.386169  0.386169
4                  text_desc                 counts      3  0.595590  0.478441
5                  text_desc                 counts      5  0.689612  0.499718
6                  text_desc                  tfidf      1  0.416123  0.416123
7                  text_desc                  tfidf      3  0.630696  0.510859
8                  text_desc                  tfidf      5  0.722318  0.531643
9         text_desc_headline                 binary      1  0.585477  0.585477
10        text_desc_headline                 binary      3  0.794675  0.679153
11        text_desc_headline                 binary      5  0.862999  0.694813
12        text_desc_headline                 counts      1  0.585253  0.585253
13        text_desc_headline                 counts      3  0.792147  0.677910
14        text_desc_headline                 counts      5  0.861399  0.693764
15        text_desc_headline                  tfidf      1  0.620520  0.620520
16        text_desc_headline                  tfidf      3  0.835893  0.717187
17        text_desc_headline                  tfidf      5  0.899834  0.731942
18    text_desc_headline_url                 binary      1  0.622152  0.622152
19    text_desc_headline_url                 binary      3  0.830101  0.715576
20    text_desc_headline_url                 binary      5  0.891065  0.729633
21    text_desc_headline_url                 counts      1  0.626664  0.626664
22    text_desc_headline_url                 counts      3  0.829621  0.718099
23    text_desc_headline_url                 counts      5  0.890329  0.731990
24    text_desc_headline_url                  tfidf      1  0.655434  0.655434
25    text_desc_headline_url                  tfidf      3  0.867256  0.751163
26    text_desc_headline_url                  tfidf      5  0.919515  0.763208
```

Figure 10: Accuracy and MRR with HuffPost dataset with all combinations of feature rep. and values of k

5.0.4 Overall comparison of models

Finally, each model has pretty good results, but the best one according to some research seems to be CNN. Even thought it was not shown in this work due to not optimizing the network analysed, with the 20 Newsgroup dataset (this was the dataset that I tried to use in all the models, so it would be the base to compare).

The Logistic Regression result was the on obtained with k=1 and TF-IDF to be in the same conditions as the Naïve Bayes model.

Model	Accuracy
Naïve Bayes	77.39%
Logistic Regression	84.51%
CNN	56.15%

Table 1: Accuracy of different models

As can be seen in table 1, the best model was the Logistic Regression. The CNN was expected to have better results. I believe the poor results obtained are due to poor tuning, since as can be seen in figure 4, a tuned network gets much higher accuracy (with Yelp, Amazon and IMBD datasets and a tuned network for each, the accuracy is aorund 80%).

6 Conclusion

The overall goal of this work was to get acquainted with the Text Classification problem, the focus was on Topic Labelling and a hint of Sentiment Analysis. After this work, I think I understand more of the subject and find it really interesting and useful in many ways.

Going deeper into Topic Labelling, it was possible to experiment with different models, feature representations, ranks of classification, having more than one field analysed, metrics, etc. Concluding that :

- TF-IDF was the best feature representation, since it takes into account that words that are unique to documents should have more weight than more common words in documents;
- When having more than a label attributed to the text, taking more labels into account seems better;
- And when given the ever increasing speed society we live in, headlines and descriptions of articles are made in such a way that analysing just their fields is enough to correctly classify them in the correct topic.

On the evaluation of the impact of using more than one field during classification, it would be interesting to compare the results with the full body of the articles. This was not done in this work due to lack of time to fetch the full body of text from the link available. I am not sure what would be expected, but I think giving the overall tendency to have short and concise descriptions and headlines nowadays (these were two of the fields used), these two fields are generally representative of the content of the articles, so looking into the full body of text itself should have better results, but looking at more fields without having access to the text itself would not stay too much behind it.

I would have liked to explore a bit more of the CNN and RNN models, due to their increasing popularity in the last years. And it would have been interesting to tune their structure and hyperparameters, seeing the impact of each on the results of the model.

From other works, RNN (Recurrent Neural Networks) have better results than the methods anaylsed in this work, but it is important to note that it always depends on the use case. RNN are preferred for text analysis, while CNN are better for image. "CNNs are preferred in interpreting visual data, sparse data or data that does not come in sequence," explained Prasanna Arikala, CTO at Kore.ai, a chatbot development company. "Recurrent neural networks, on the other hand, are designed to recognize sequential or temporal data. They do better predictions considering the order or sequence of the data as they relate to previous or the next data nodes." - quoting from (rnn, a)

References

a. The 20 newsgroups dataset.

b. The 20 newsgroups text dataset on scikit.

The amazon dataset.

a. Comparison between cnn and rnn.

How to encode text data for machine learning with scikit-learn.

The huffpost dataset.

The imdb dataset.

Lstm for text classification in python.

Multi-class text classification with scikit-learn.

b. Recurrent neural networks (rnn) with keras.

Repository with all the code.

a. Source of cnn code with 20 newsgroup dataset.

b. Source of cnn code with yelp, amazon and imbd datasets.

Source of logistic regression code.

Source of naïve bayes code.

c. Text classification with an rnn.

a. Text classification with machine learning & nlp. From MonkeyLearn website.

b. Working with text data with scikit-learn.

The yelp dataset.

Jason Brownlee. 2017. What are word embeddings for text?

Kavita Ganesan. Build your first text classifier in python with logistic regression.

Dan Jurafsky and James H. Martin. 2021a. Chapter 4 - naive bayes and sentiment classification. In *Speech and language processing*. Association for Computational Linguistics.

Dan Jurafsky and James H. Martin. 2021b. Chapter 5 - logistic regression. In *Speech and language processing*. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics*.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*.

A Metrics used

The metrics used can all be found in this link, (lr) for MRR, also the Wikipedia page of MRR was consulted and for confusion matrix check this link. Here is a short description of each:

- **Confusion Matrix** : The number of correct and incorrect predictions are summarized with count values and broken down by each class. Each entry of the matrices in this work correspond to hits. The lines correspond to the predicted labels and the columns to the true labels.
- **Accuracy**: the percentage of texts that were categorized with the correct tag.
- **Precision**: the percentage of examples the classifier got right out of the total number of examples that it predicted for a given tag.
- **Recall**: the percentage of examples the classifier predicted for a given tag out of the total number of examples it should have predicted for that given tag.
- **F1 Score**: the harmonic mean of precision and recall
- **Mean Reciprocal Rank (MRR)**: The reciprocal rank of a query response is the multiplicative inverse of the rank of the first correct answer: 1 for first place, 1⁄2 for second place, 1⁄3 for third place and so on. The mean reciprocal rank is the average of the reciprocal ranks of results for a sample of queries - quoting from Wikipedia. The expression for MRR is:

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i}. \quad (1)$$

Where $rank_i$ refers to the rank position of the first relevant document for the i-th query.

B Confusion matrices obtained

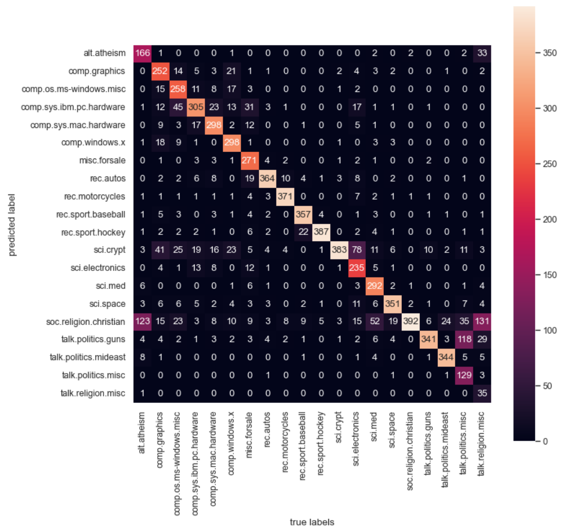


Figure 11: Confusion matrix of NB

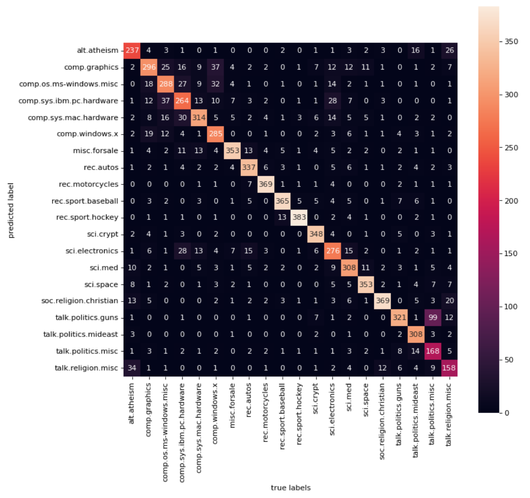


Figure 12: Confusion matrix of LR with binary weighting

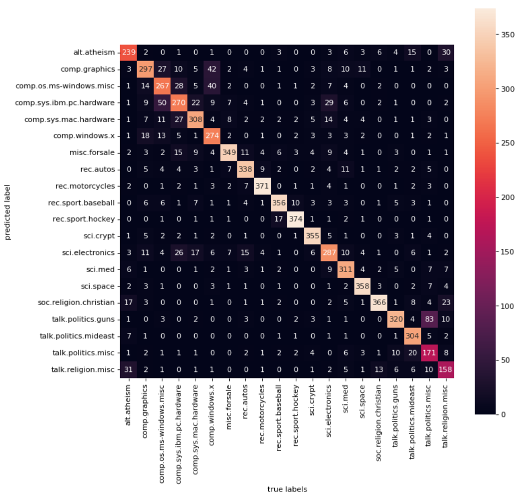


Figure 13: Confusion matrix of LR with counts

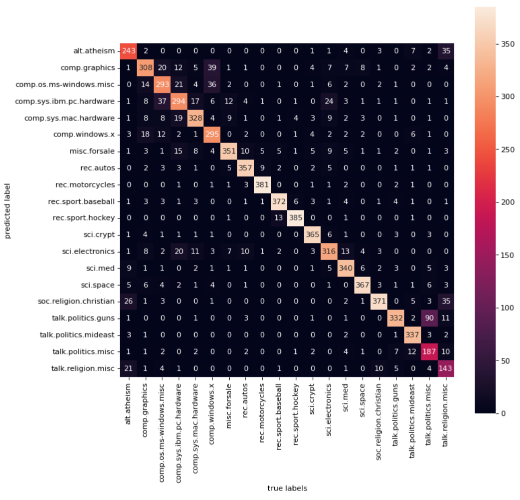
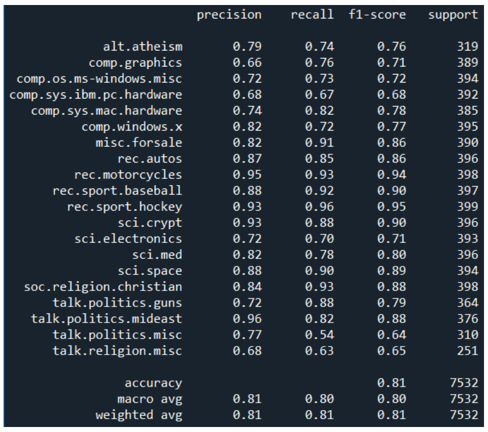


Figure 14: Confusion matrix of LR with TF-IDF

C Full tables of results for LR

The full tables of results for each feature representation using LR and using top_k=1, having the values for several metrics discriminated by class for the 20 Newsgroup dataset and then with the macro-average in the end.

Given the large amount of trials donem only these three full tables are shown for demonstrarion purposes.

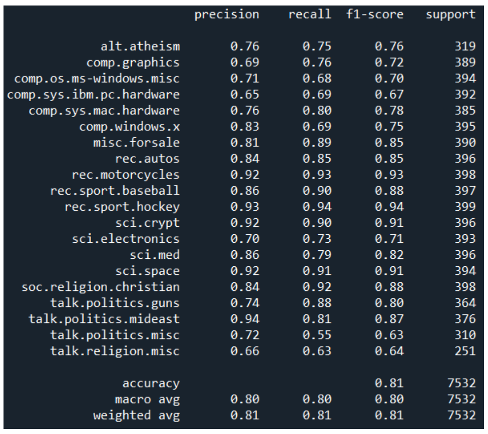


```
                          precision    recall  f1-score   support

             alt.atheism       0.79      0.74      0.76       319
           comp.graphics       0.66      0.76      0.71       389
 comp.os.ms-windows.misc       0.72      0.73      0.72       394
comp.sys.ibm.pc.hardware       0.68      0.67      0.68       392
   comp.sys.mac.hardware       0.74      0.82      0.78       385
          comp.windows.x       0.82      0.72      0.77       395
            misc.forsale       0.82      0.91      0.86       390
               rec.autos       0.87      0.85      0.86       396
         rec.motorcycles       0.95      0.93      0.94       398
      rec.sport.baseball       0.88      0.92      0.90       397
        rec.sport.hockey       0.93      0.96      0.95       399
               sci.crypt       0.93      0.88      0.90       396
         sci.electronics       0.72      0.70      0.71       393
                 sci.med       0.82      0.78      0.80       396
               sci.space       0.88      0.90      0.89       394
  soc.religion.christian       0.84      0.93      0.88       398
      talk.politics.guns       0.72      0.88      0.79       364
   talk.politics.mideast       0.96      0.82      0.88       376
      talk.politics.misc       0.77      0.54      0.64       310
      talk.religion.misc       0.68      0.63      0.65       251

                accuracy                           0.81      7532
               macro avg       0.81      0.80      0.80      7532
            weighted avg       0.81      0.81      0.81      7532
```

Figure 15: Results of LR with binary

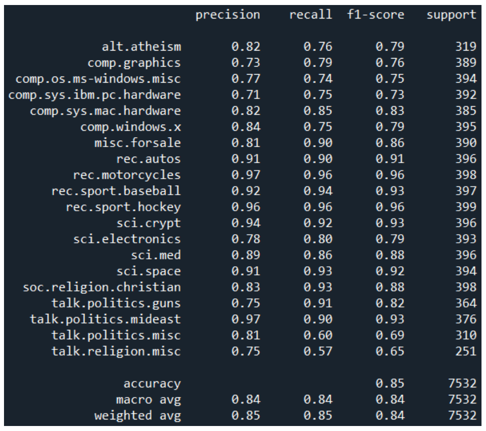


```
                          precision    recall  f1-score   support

             alt.atheism       0.76      0.75      0.76       319
           comp.graphics       0.69      0.76      0.72       389
 comp.os.ms-windows.misc       0.71      0.68      0.70       394
comp.sys.ibm.pc.hardware       0.65      0.69      0.67       392
   comp.sys.mac.hardware       0.76      0.80      0.78       385
          comp.windows.x       0.83      0.69      0.75       395
            misc.forsale       0.81      0.89      0.85       390
               rec.autos       0.84      0.85      0.85       396
         rec.motorcycles       0.92      0.93      0.93       398
      rec.sport.baseball       0.86      0.90      0.88       397
        rec.sport.hockey       0.93      0.94      0.94       399
               sci.crypt       0.92      0.90      0.91       396
         sci.electronics       0.70      0.73      0.71       393
                 sci.med       0.86      0.79      0.82       396
               sci.space       0.92      0.91      0.91       394
  soc.religion.christian       0.84      0.92      0.88       398
      talk.politics.guns       0.74      0.88      0.80       364
   talk.politics.mideast       0.94      0.81      0.87       376
      talk.politics.misc       0.72      0.55      0.63       310
      talk.religion.misc       0.66      0.63      0.64       251

                accuracy                           0.81      7532
               macro avg       0.80      0.80      0.80      7532
            weighted avg       0.81      0.81      0.81      7532
```

Figure 16: Results of LR with counts



```
                          precision    recall  f1-score   support

             alt.atheism       0.82      0.76      0.79       319
           comp.graphics       0.73      0.79      0.76       389
 comp.os.ms-windows.misc       0.77      0.74      0.75       394
comp.sys.ibm.pc.hardware       0.71      0.75      0.73       392
   comp.sys.mac.hardware       0.82      0.85      0.83       385
          comp.windows.x       0.84      0.75      0.79       395
            misc.forsale       0.81      0.90      0.86       390
               rec.autos       0.91      0.90      0.91       396
         rec.motorcycles       0.97      0.96      0.96       398
      rec.sport.baseball       0.92      0.94      0.93       397
        rec.sport.hockey       0.96      0.96      0.96       399
               sci.crypt       0.94      0.92      0.93       396
         sci.electronics       0.78      0.80      0.79       393
                 sci.med       0.89      0.86      0.88       396
               sci.space       0.91      0.93      0.92       394
  soc.religion.christian       0.83      0.93      0.88       398
      talk.politics.guns       0.75      0.91      0.82       364
   talk.politics.mideast       0.97      0.90      0.93       376
      talk.politics.misc       0.81      0.60      0.69       310
      talk.religion.misc       0.75      0.57      0.65       251

                accuracy                           0.85      7532
               macro avg       0.84      0.84      0.84      7532
            weighted avg       0.85      0.85      0.84      7532
```

Figure 17: Results of LR with TF-IDF