Text Classification NLP Task - Report
Dataset: 20 newsgroups
GitHub: link

Vaggelis Lamprou

Introduction

The present report gives a detailed summary of a classification study conducted on the 20 newsgroups dataset. It is divided into four main sections named Dataset, Machine Learning, Deep Learning and Conclusion and is aligned with the accompanying Jupyter notebook enumeration.

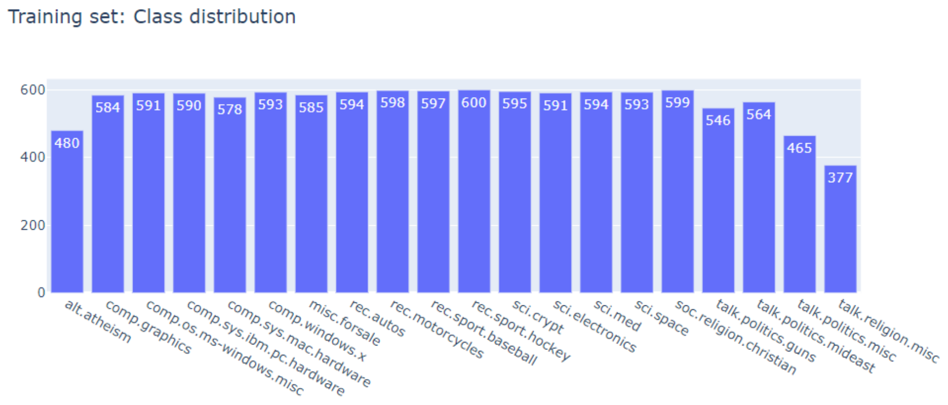
It describes the data exploration and pre-processing, the main ideas coded in the notebook and presents the respective results. The main focus lies in the comparison between the Machine Learning and the Deep Learning approaches; at the level of both the data transformations and the model architectures. A summary of our findings is presented in the Conclusion section, where we eventually see that a GloVe-based BiLSTM model slightly outperforms the selected SVM model among other Machine Learning classifiers.

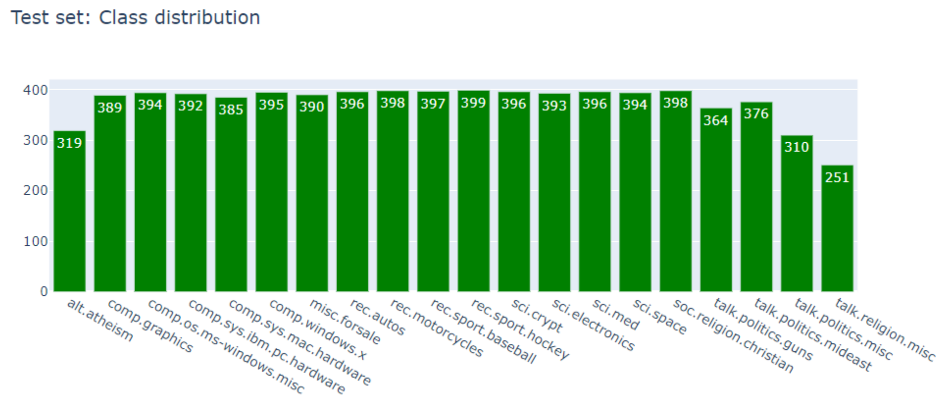
1 Dataset

The 20 newsgroups dataset can be downloaded directly from the sklearn's dataset class. It consists of texts classified in the following categories:

- *alt.atheism*
- *comp.graphics*
- *comp.os.ms-windows.misc*
- *comp.sys.ibm.pc.hardware*
- *comp.sys.mac.hardware*
- *comp.windows.x*
- *misc.forsale*
- *rec.autos*
- *rec.motorcycles*
- *rec.sport.baseball*
- *rec.sport.hockey*
- *sci.crypt*
- *sci.electronics*
- *sci.med*
- *sci.space*
- *soc.religion.christian*
- *talk.politics.guns*
- *talk.politics.mideast*
- *talk.politics.misc*
- *talk.religion.misc*

In total we have **11314 training** and **7532 test** text instances, without 'headers', 'footers' and 'quotes', distributed as follows among the labels:





2 Machine Learning (ML)

2.1 Preprocessing

In this section we explore the dataset from the machine learning perspective. At first, we are concerned with data preprocessing by defining two functions, named "Preprocess1" and "Preprocess2" with the following characteristics :

- Preprocess1
  - Lower all characters

- Use nltk's word_tokenize function to obtain words

- Preprocess2
  - Lower all characters
  - Use nltk's word_tokenize function to obtain words
  - Remove words with length smaller than 3 characters
  - Remove stopwords as determined by nltk
  - Stem the words with nltk's PorterStemmer

The functions convert each text to a list of modified tokens and will be used in section 2.2 to tune the tokenizer of the Tfidf vectorizer. At this point we can observe, that by construction, Preprocess2 seems to focus better to more useful words. This is depicted in the below example as well, as it ignores words like 'in' and 'us' which do not probably contribute to the class characterization.

Sliced output of Preprocess1: ['do', "n't", 'be', 'so', 'sure', '.', 'look', 'what', 'happened', 'to', 'japanese', 'citizens', 'in', 'the', 'us', 'during', 'world', 'war']

Sliced output of Preprocess2: ["n't", 'sure', 'look', 'happen', 'japanes', 'citizen', 'world', 'war', "'re", 'prepar', 'say', 'let', 'round', 'peopl', 'stick', 'concentr', 'camp', 'without']

2.2 Classification (Training and Testing)

This subsection describes the evaluation of multiple pipeline models in order to determine the one that performs optimally on the training and the test sets.

Each pipeline consists of the following two components:

- The "TfidfVectorizer" part, which is responsible for transforming the text words into vectors based on their text frequency. In this study, we tune the following hyper-parameters:

  - tokenizer: By considering Preprocess1 and Preprocess2 defined above

  - ngram_range: By considering the alternatives of uni-grams only, unigrams & bigrams and bigrams-only

  - norm: By considering either the 'l1' or the 'l2' norm to normalize the resulting word vectors (such that they sum to 1)

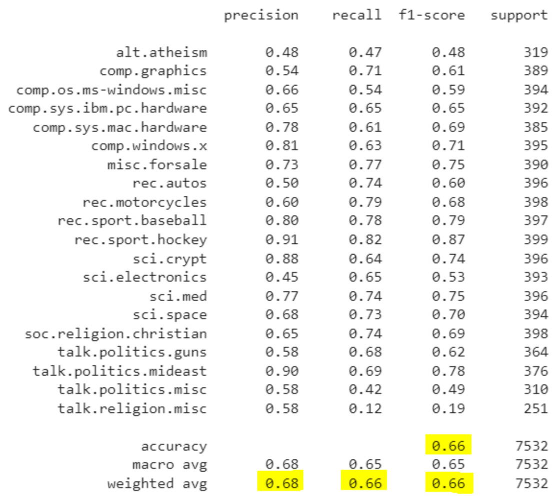
- The "Classifier" part, which is tuned with the following classifier options:
  - Support Vector Machine
  - Multinomial Naive Bayes
  - Random Forest

In the interest of time and the limited available Colab resources, we note that the classifier hyper-parameters were left as the default ones. For instance, C=1 for the SVM and n_estimators=100 for the Random Forest.

The aforementioned tuning choices result in 36 different model pipelines in total. Their evaluation is implemented by the "evaluate_and_test_ml" function and the optimal model is selected via its validation accuracy score. The validation size is 20% of the training data.
Eventually, a pipeline with

- Preprocess2
- uni-grams
- 'l2' norm
- SVM classifier

is chosen. The function also trains from scratch the model on the entire training dataset and then tests it on the untouched test dataset. Its classification report is as follows:



```
                          precision    recall  f1-score   support

             alt.atheism       0.48      0.47      0.48       319
           comp.graphics       0.54      0.71      0.61       389
 comp.os.ms-windows.misc       0.66      0.54      0.59       394
comp.sys.ibm.pc.hardware       0.65      0.65      0.65       392
   comp.sys.mac.hardware       0.78      0.61      0.69       385
          comp.windows.x       0.81      0.63      0.71       395
            misc.forsale       0.73      0.77      0.75       390
               rec.autos       0.50      0.74      0.60       396
         rec.motorcycles       0.60      0.79      0.68       398
      rec.sport.baseball       0.80      0.78      0.79       397
        rec.sport.hockey       0.91      0.82      0.87       399
               sci.crypt       0.88      0.64      0.74       396
         sci.electronics       0.45      0.65      0.53       393
                 sci.med       0.77      0.74      0.75       396
               sci.space       0.68      0.73      0.70       394
  soc.religion.christian       0.65      0.74      0.69       398
      talk.politics.guns       0.58      0.68      0.62       364
   talk.politics.mideast       0.90      0.69      0.78       376
      talk.politics.misc       0.58      0.42      0.49       310
      talk.religion.misc       0.58      0.12      0.19       251

                accuracy                           0.66      7532
               macro avg       0.68      0.65      0.65      7532
            weighted avg       0.68      0.66      0.66      7532
```

3 Deep Learning (DL)

3.1 Preprocessing

The data pre-processing was implemented with the Keras Tokenizer function which is similar to the Preprocess1 function build in section 2.1, in the sense that they do not consider removal of small or stopwords or any stemming.

Once the tokens are extracted the Keras Tokenizer learns 200-dim sequence representations per text by exploring the frequences of the 20000 most common words (vocabulary size). It assigns them an integer representation in descending frequence order. (i.e. the most common is indexed with "1", the second most common with "2" etc). Finally, when applied to test data, the unseen words are indexed with "0".

3.2 Glove Embeddings

For the purposes of the word vectorization process we consider the famous pre-trained GloVe embeddings. Each indexed word is assigned to a learned vector of dimension 100 resulting in an embedding matrix of shape (20000, 100). This matrix is responsible for determining the weights of the embedding layer of the neural network model.

3.3 Classification

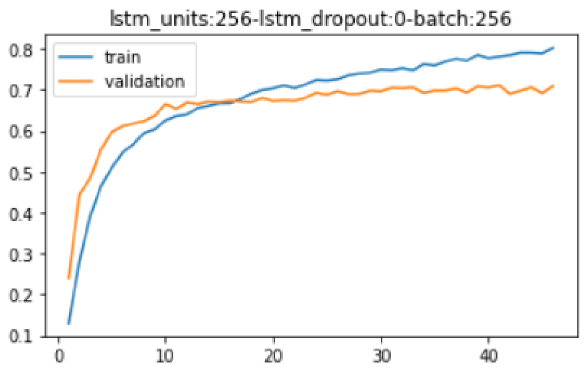
Analogously to the ML case once the tokenizing and vectorization tasks are clear we proceed by determining the final neural network classifier.

We note that in contrast to the ML case, code-wise speaking, the evaluation, training and test is not performed in a single function. At first, multiple models are evaluated in the "dl_model" function (with validation size again 20% of the training dataset) and then the optimal model is trained and tested separately.

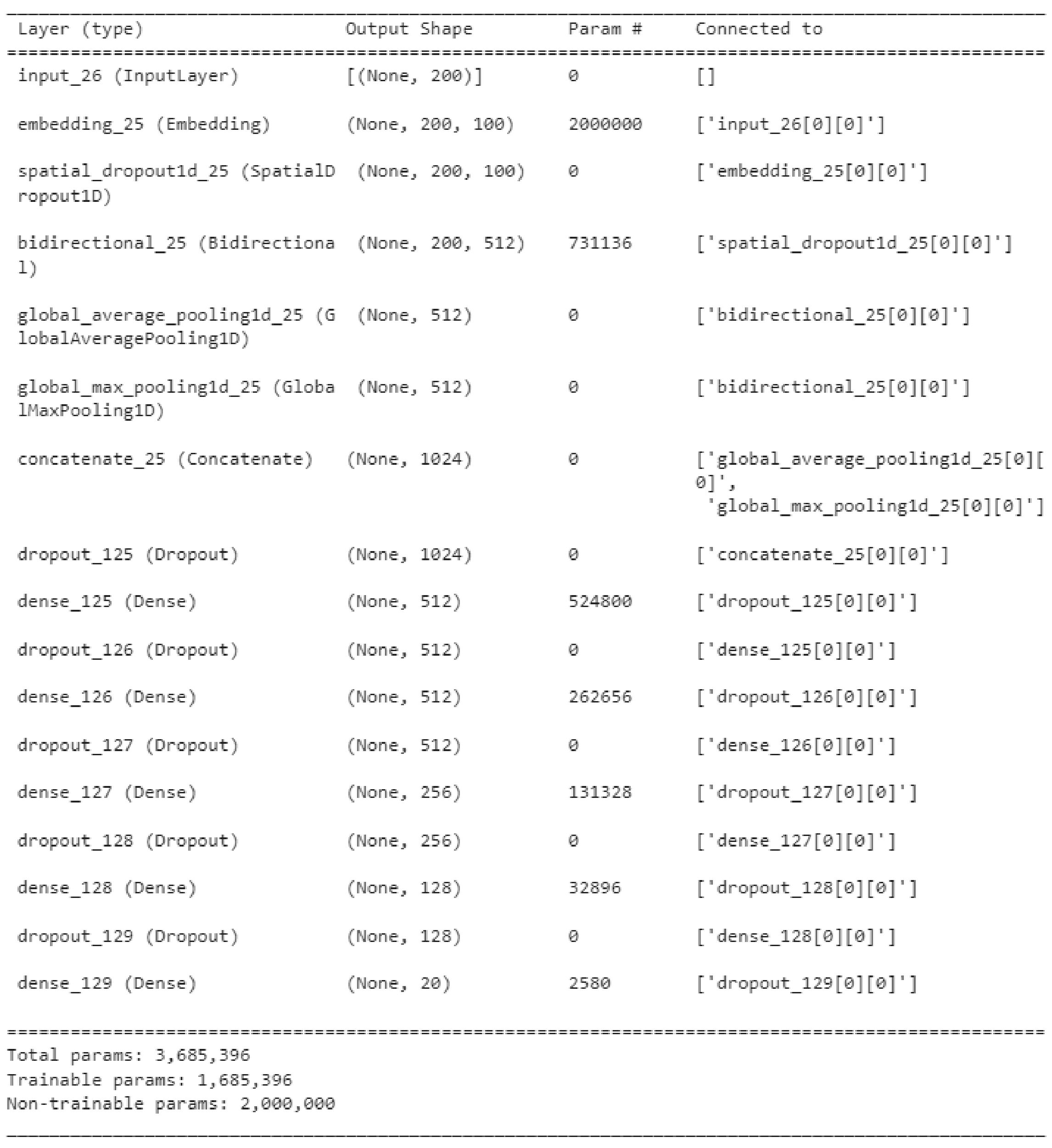
As far as the evaluation task is concerned, we note that -high level speaking- we considered models where the initial (GloVe) embedding layer is followed by a Bidirectional LSTM layer and then a dense sequence of layers. The tuning process was expanded to different values of the LSTM units, the LSTM dropout value, the number of dense layers, the number of nodes in the dense layers, the dropout value between the dense layers, the layers' activation function, the loss function, the optimizer and its learning rate and the training batch size.

In the interest of time and due to Colab's limited resources we eventually present only the structures that looked more promising. Note that during our experiments the MSE loss, the ReLU activation and the SGD optimizer were outperformed by Categorical Crossentropy, the tanh function and Adam optimizer respectively in most of the cases. Same holds for learning rates different than 0.001. Thus, from the list of hyper-params referenced in the above paragraph, the "dl_model" function tunes only the LSTM units and dropout values and the training batch size.

This leads to a total of 18 different models, monitored by Early Stopping, whose training and validation accuracy history is included in the ipynb file (link). Eventually, we pick a model of the following accuracy history :

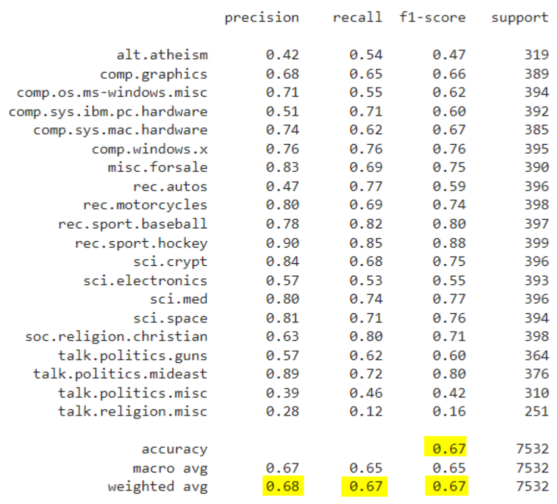


and architecture:



```
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_26 (InputLayer)          [(None, 200)]        0           []

 embedding_25 (Embedding)       (None, 200, 100)     2000000     ['input_26[0][0]']

 spatial_dropout1d_25 (SpatialD  (None, 200, 100)    0           ['embedding_25[0][0]']
 ropout1D)

 bidirectional_25 (Bidirectiona  (None, 200, 512)    731136      ['spatial_dropout1d_25[0][0]']
 l)

 global_average_pooling1d_25 (G  (None, 512)         0           ['bidirectional_25[0][0]']
 lobalAveragePooling1D)

 global_max_pooling1d_25 (Globa  (None, 512)         0           ['bidirectional_25[0][0]']
 lMaxPooling1D)

 concatenate_25 (Concatenate)   (None, 1024)         0           ['global_average_pooling1d_25[0][
                                                                 0]',
                                                                  'global_max_pooling1d_25[0][0]']

 dropout_125 (Dropout)          (None, 1024)         0           ['concatenate_25[0][0]']

 dense_125 (Dense)              (None, 512)          524800      ['dropout_125[0][0]']

 dropout_126 (Dropout)          (None, 512)          0           ['dense_125[0][0]']

 dense_126 (Dense)              (None, 512)          262656      ['dropout_126[0][0]']

 dropout_127 (Dropout)          (None, 512)          0           ['dense_126[0][0]']

 dense_127 (Dense)              (None, 256)          131328      ['dropout_127[0][0]']

 dropout_128 (Dropout)          (None, 256)          0           ['dense_127[0][0]']

 dense_128 (Dense)              (None, 128)          32896       ['dropout_128[0][0]']

 dropout_129 (Dropout)          (None, 128)          0           ['dense_128[0][0]']

 dense_129 (Dense)              (None, 20)           2580        ['dropout_129[0][0]']

==================================================================================================
Total params: 3,685,396
Trainable params: 1,685,396
Non-trainable params: 2,000,000
__________________________________________________________________________________________________
```

Finally, the classification report on the untouched test set is as follows:



```
                          precision    recall  f1-score   support

             alt.atheism       0.42      0.54      0.47       319
           comp.graphics       0.68      0.65      0.66       389
 comp.os.ms-windows.misc       0.71      0.55      0.62       394
comp.sys.ibm.pc.hardware       0.51      0.71      0.60       392
   comp.sys.mac.hardware       0.74      0.62      0.67       385
          comp.windows.x       0.76      0.76      0.76       395
            misc.forsale       0.83      0.69      0.75       390
               rec.autos       0.47      0.77      0.59       396
         rec.motorcycles       0.80      0.69      0.74       398
      rec.sport.baseball       0.78      0.82      0.80       397
        rec.sport.hockey       0.90      0.85      0.88       399
               sci.crypt       0.84      0.68      0.75       396
         sci.electronics       0.57      0.53      0.55       393
                 sci.med       0.80      0.74      0.77       396
               sci.space       0.81      0.71      0.76       394
  soc.religion.christian       0.63      0.80      0.71       398
      talk.politics.guns       0.57      0.62      0.60       364
   talk.politics.mideast       0.89      0.72      0.80       376
      talk.politics.misc       0.39      0.46      0.42       310
      talk.religion.misc       0.28      0.12      0.16       251

                accuracy                           0.67      7532
               macro avg       0.67      0.65      0.65      7532
            weighted avg       0.68      0.67      0.67      7532
```

4 Conclusion

To sum up, above is presented a study that builds a Supoprt Vector Machine and a GloVe-based BiLSTM classifier for the "fetch 20newsgroups" dataset.

The ML approach considers a hand-written preprocess function that is based on nltk's word_tokenize and PorterStemmer functions and also lowers characters and removes small and stop-words (via nltk). The text words are vectorized via the Tfidf vectorizer which uses uni-grams and the 'l2' norm to compute the vector representations. On the other hand, the DL approach focuses on Kera's build-in tokenizer and builds word vector representations via the pre-trained GloVe embeddings.

The resulting models came up after multiple model evaluations and their overall test results are summarized in the below table :

Test Summary (Weighted metrics)				
Model	Accuracy	Precision(W)	Recall(W)	F1-score(W)
SVM	0.66	0.68	0.66	0.66
BiLSTM	0.67	0.68	0.67	0.67