Data Wrangling – Report – Coursework part B

Napier University 2019/2020 - Alessio Williams Gava - 40212064

1. Method

The model adopted is a simple Convolutional Neural Network, tuned for outputting classification results. A very similar model has been followed, as presented in the literature by Kim Yoon (2014). It is a simple and popular model that is referenced in many other literatures works, where a CNN is utilized to classify text sequences.

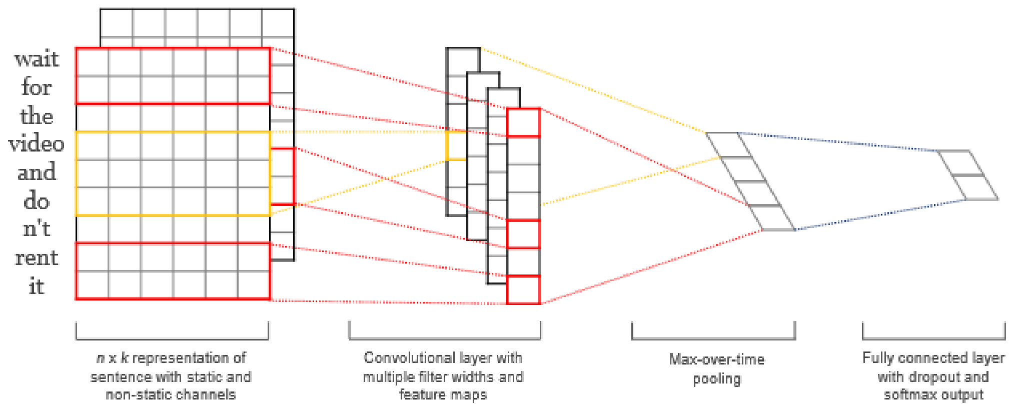

wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Figure 1 model used for text classification, taken from "Convolutional Neural Networks for Sentence Classification" (Kim 2014)

Data is inputted in the embedding layer, which is created by using a word embedding matrix, no further trainable once added to the model. One-dimension filters are convoluted along the word vectors and their outputs are stored in the convolutional layer. A Pooling layer is applied to the convolutional layer to reduce and summarize the data in the previous layer. A drop out layer is applied to attempt to reduce overfitting. Finally, a dense layer with a SoftMax activation function is employed to classify the text. The number of neurons is the same as the number of unique labels in the training input data.

Similarly to the paper proposed, it was attempted to apply this model in three different scenarios: utilize pre-embedded word vectors (Glove 50 dimension from 1.2M vocab, uncased), calculate the word vectors from the dataset (word2vec), utilize random initialized word vectors.

2. Evaluation of improvements

2.1. Preprocessing and word vector matrix

The evaluations utilize the accuracy metric and the average loss. A KFold has been used (k = 5) to more precisely evaluate the accuracy obtained, reducing randomness in the evaluations.

Below are presented the initial accuracy results by running the CNN previously presented with the raw dataset.

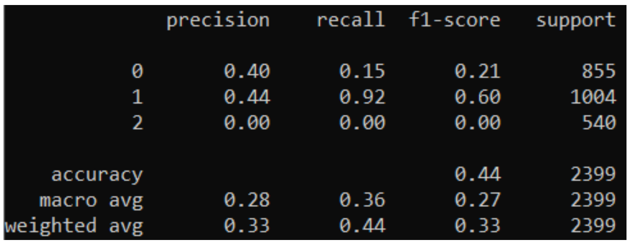
	Training accuracy	Testing accuracy	Words random initialized
Rand-init	55.53%	46.80%	100%
Embedded	44.19%	43.68%	0%
Pre-embedded	65.37%	52.60%	60%

Table 1 Initial accuracies using raw dataset. Parameters: max_len: 100, embedding_length : 50, filter_n : 100, filter_high : 3, strides : 1, batch_size : 32, epochs : 10, min_count wv : 1

Unusually, the pre-embedded has 60% of words initialized at random, meaning that more than half of the words in the dataset are unknown. This is because punctuation and case affect the Glove pre-embedded matrix.

Additionally, the embedded results were surprisingly low, performing worse than a random initialized one.

One recurrent pattern is the difficulty in correctly predicting the sentences for label 'OAG'. As shown in the figure below, obtained from the confusion matrix, (OAG is label 2), the lowest frequent label is never correctly predicted in the Embedded scenario, leading to very low accuracy.



```
              precision    recall  f1-score   support

           0       0.40      0.15      0.21       855
           1       0.44      0.92      0.60      1004
           2       0.00      0.00      0.00       540

    accuracy                           0.44      2399
   macro avg       0.28      0.36      0.27      2399
weighted avg       0.33      0.44      0.33      2399
```

Figure 2 precision, recall, f1score and support metrics

	Predicted 0	Predicted 1	Predicted 2
Actual 0	124	731	0
Actual 1	79	925	0
Actual 2	107	433	0

Table 2 difficulty in predicting label 2 - OAG

It was decided to improve the word2vec representation.

By modifying the minimum *word count* (words taken into account during creation of word2vec), it was discovered that there is a very high number of rare words. In fact, just by setting that parameter to 2, the word2vec would ignore 16k words (out of 27k total), meaning that 16k are only seen once in the whole dataset. This implies that the word matrix would be mostly dominated by the few high frequency ones. By tuning the parameter *sample,* the impact of high frequency words can be lowered, but no improvement was registered in the CNN accuracy. A second algorithm is available to create the matrix, Skip-Gram, and as one of the authors of word2vec indicates: "Skip-gram: works well with small amount of the training data, represents well even rare words or phrases"(Mikolov et al. 2013). In fact, by switching to the skip-gram algorithm and using *hs* (hierarchical softmax), improvements were registered.

	Average accuracy	Epoch1 loss	Epoch10 loss
Training accuracy	56.92%	1.0548	0.9351
Testing accuracy	50.74%	1.0083	0.9727

Table 3 Average accuracies and losses beginning and end epochs of Embedded scenario

The accuracies increased considerably. The average losses follow a similar descending pattern, but the training has a steeper curve because the model is only using the training data to calculate the losses. The testing average loss is reducing similarly, meaning that the model is not completely overfitting on the training values.

In the other two scenarios, the label 'OAG' precision can be very high, but its recall is usually less than 10%, meaning that it is correctly identified only a few times. This problem is thought to be inherent to the number of labels elements, since the dataset is quite unbalanced: CAG : 4240, NAG : 5051, OAG : 2708. To fix this issue, the training set data has been down sampled, resulting in the same frequency of all labels types (2708 elements each). This was shown to achieve great accuracy for the 'OAG' label, but there is not improvement of the total accuracy, meaning that it was no beneficial for the other labels.

By analyzing the dataset, shortcomings were discovered and tried to be improved by preprocessing the text, utilizing methodologies presented in the papers by Mhatre et. al (2017) and Resyanto et al. (2019).

Preprocessing includes removing duplicates sentences (because using kfold), lowercase, removing common stop words, converting emoji and emoticons to text, removing html tags, removing http url, removing punctuation, lemmatizing the words.

	Training accuracy	Testing accuracy	Words random init
Rand-init	56.50%	47.46%	100%
Embedded	59.06%	52.51%	0%
Pre-embedded	67.10%	53.95%	38%

Table 4 average accuracy using preprocessed dataset

All of the models' accuracies improved by preprocessing the dataset. The highest improvement was registered in the Embedded, supposedly because by cleaning the train data, the resulting word vector is more precise in summarizing the relations between the words.

2.2. CNN model

On each group of Convolutional layers, the filter height can be chosen, therefore creating n-grams where n is the filter height. Yoon, in his paper, utilized 3-4-5. All these combinations were tried, but no evident improvement was noticed. Instead, it was seen that higher filter heights, output a somewhat better accuracy for training, because they are considering whole sentences instead of group of words, essentially overfitting on the training set.

Similar output was registered for the number of filters. The higher number used, a more overfitted model is created.

Filter number	Train accuracy avg	Test accuracy avg
20	0.572058022	0.519982815
50	0.595916152	0.517834127
100	0.575067163	0.529437065
200	0.650080621	0.526858628
400	0.717463732	0.517834127

Table 5 Filters number affecting average accuracy – Embedded model

The global pooling layer can be either set as average or maximum. It was initially thought that an average would be more beneficial to create a higher general module. However, it was discovered that the average performs worse (-2% on testing accuracy). Seeming that higher values contribute more to the detecting of classification.

Dropout layer was examined to understand if the model could be used without it.

Dropout	Train accuracy avg	Test accuracy avg
0	65.62%	50.24%
0.2	64.59%	51.22%
0.5	58.42%	52.56%

Table 6 Dropout percentage layer influence average accuracies – Embedded model

This model is shown to be influenceable by the training data. By using the dropout, it has lowered the training accuracy, but increased the testing one. The aim is to improve the test accuracy, therefore it was preferred 0.5.

A few activation functions were tested to understand their influence on the accuracy, and it was discovered that they perform similarly. The best was 'elu' as shown below.

Activation	Train accuracy avg	Test accuracy avg
relu	58.68%	53.24%
tanh	59.65%	52.69%
elu	60.53%	53.89%

Table 7 Activation functions influence average accuracies – Embedded model

3. Reflection

By utilizing good word embeddings, it can be taken advantage of the property that similar semantical words have similar cosine distance between their word embeddings. Meaning that when convolving with sentences with different words, but similar in their meaning, the output from applying a filter to them, will be similar, leading to analogous activation values. This was noticed by applying the pre-embedding model and observing that without any hyperparameters tuning, performed better.

The algorithm to create word embeddings must be carefully chosen as they are very sensible to the type of input data provided.

Dropout is an essential layer which helps in its generalisation, reducing overfitting.

The model has a steady increase on the training data accuracy over each epoch. A similar pattern is noticed on the validation data, meaning that the model is learning general rules. The average accuracies obtained are not very high. Seemingly because of the difficulty of learning a specific type of label type. The reason could be related to the lower number of element type of the rare label.

The dataset has a very sparse frequency of words, creating many rare words, which it appears to be challenging for the model to represent well. An improvement could be creating a larger model, utilizing more types of filters and hidden layers. Doing so it might give more expressivity to the model, considering also rare words.

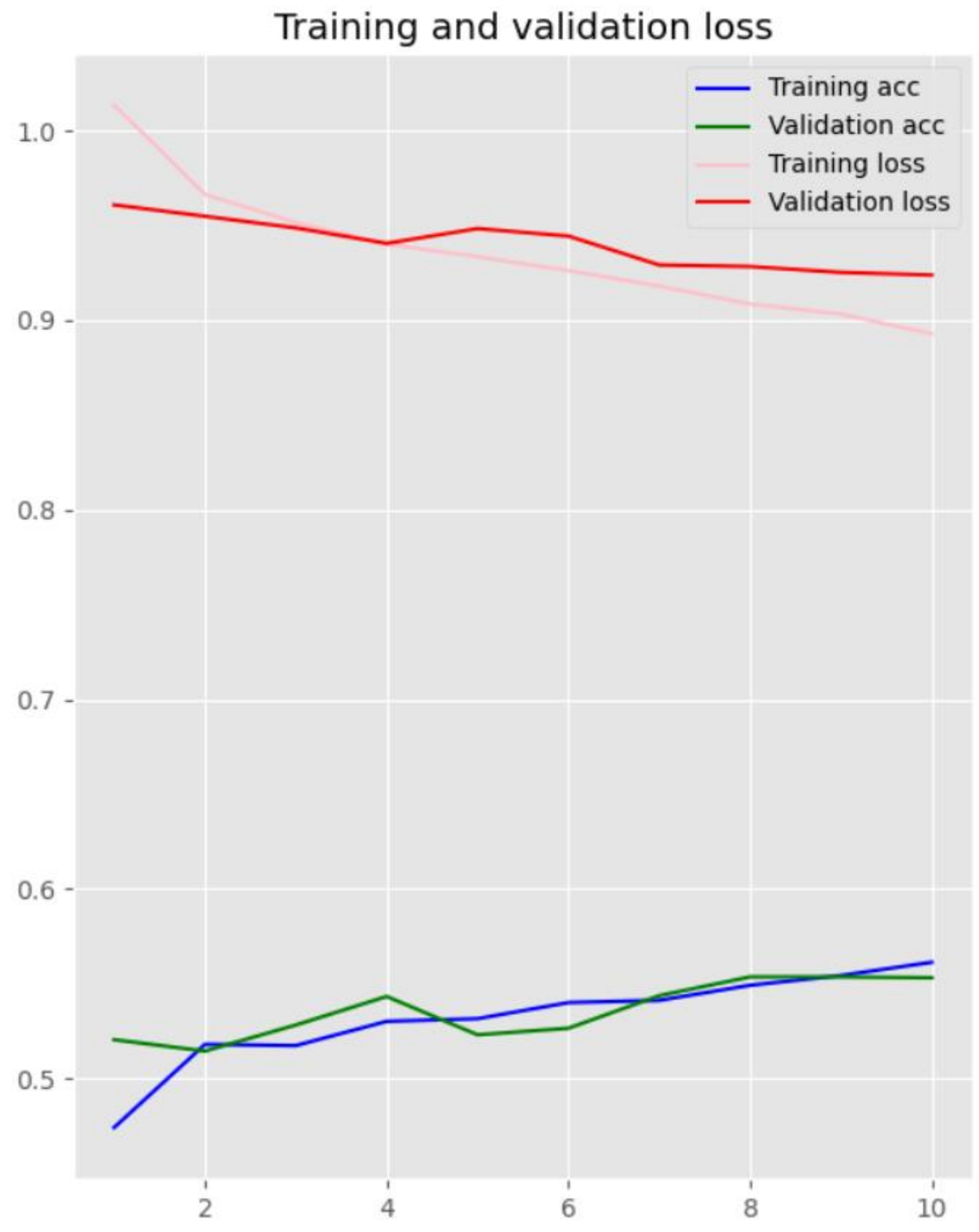


Figure 3 Best performance obtained – Embedded model. max_len: 100, embedding_length : 300, filter_n : 100, filter_high : 3, strides : 1, batch_size : 32, epochs : 10, skipgram, hierarchical softmax

References

Kim, Yoon. 2014. “Convolutional Neural Networks for Sentence Classification.” *ArXiv:1408.5882 [Cs]*.

Mhatre, Mayuri, Dakshata Phondekar, Pranali Kadam, Anushka Chawathe, and Kranti Ghag. 2017. “Dimensionality Reduction for Sentiment Analysis Using Pre-Processing Techniques.” Pp. 16–

21 in *2017 International Conference on Computing Methodologies and Communication (ICCMC)*.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. “Efficient Estimation of Word Representations in Vector Space.” *ArXiv:1301.3781 [Cs]*.

Resyanto, Fero, Yuliant Sibaroni, and Ade Romadhony. 2019. “Choosing The Most Optimum Text Preprocessing Method for Sentiment Analysis: Case:IPhone Tweets.” Pp. 1–5 in *2019 Fourth International Conference on Informatics and Computing (ICIC)*.