Document categorization with deep learning using pre-trained word embeddings

Guillem Escriba (u188331), David Pérez (u188332) & Eduardo Ruiz (u162345)

Comparative table of results:

#	Model type (layers)	Parameter values	Accuracy	F1-Score
1	MLP	Layers: Dense| 512 units, 0.5 dropout | Dense | 256 units, 0.5 dropout batch_size = 64, epochs = 10, optimizer = ‘adam’, No word embeddings, max_words = 10000	81.07	62.00
2	LSTM	{'layer_type': 'LSTM', 'num_units': 256, 'dropout_rate': 0.0, 'return_sequences': True, 'second_layer': True, 'embedding_dim': 300, 'max_sequence_length': 50, 'batch_size': 32, 'epochs': 10, 'optimizer': 'adam', 'padding': 'post', 'truncating': 'post', 'embedding_type': 'glove',’max_words’: 30000}	74.58	40.16
3	Bidirectional LSTM (pre)	{'layer_type': 'Bidirectional LSTM', 'num_units': 256, 'dropout_rate': 0.0, 'return_sequences': True, 'second_layer': True, 'embedding_dim': 300, 'max_sequence_length': 50, 'batch_size': 32, 'epochs': 10, 'optimizer': 'adam', 'padding': 'pre'', 'truncating': 'pre', 'embedding_type': 'glove','max_words': 30000}	67.63	23.18
4	Bidirectional LSTM (post)	{'layer_type': 'Bidirectional LSTM', 'num_units': 256, 'dropout_rate': 0.0, 'return_sequences': True, 'second_layer': True, 'embedding_dim': 300, 'max_sequence_length': 50, 'batch_size': 32, 'epochs': 10, 'optimizer': 'adam', 'padding': 'post', 'truncating': 'post', 'embedding_type': 'glove',’max_words’: 30000}	78.27	53.07
5	Stacked GRU	Layers: GRU 300 | GRU 300 embedding_dim: 300, max_sequence_length: 300, dropout_rate: 0.0, batch_size: 64, epochs: 15, optimizer: adam, padding: post, truncating: post, embedding_type: fast_text, max_words: 10000	70.88	32.0
6	Bidirectional Stacked GRU	Layers: Bidirectional GRU 128 | GRU 64 embedding_dim: 300, max_sequence_length: 100, dropout_rate: 0.4, batch_size: 64, epochs: 10, optimizer: adam, padding: post, truncating: post, embedding_type: glove, max_words: 20000	72.31	24.91
7	Stacked LSTM	'layer_type': 'LSTM', 'num_units': 256, 'dropout_rate': 0.0, 'return_sequences': True, 'second_layer': True, 'embedding_dim': 300, 'max_sequence_length': 50, 'batch_size': 32, 'epochs': 10, 'optimizer': 'adam', 'padding': 'post', 'truncating': 'post', 'embedding_type': 'glove', 'stack_layers': 2}	73.55	39.92

8	LSTM + GRU	Layers: LSTM 128 | GRU 64 embedding_dim: 300, max_sequence_length: 100, dropout_rate: 0.4, batch_size: 64, epochs: 10, optimizer: adam, padding: post, truncating: post, embedding_type: glove, max_words: 20000	66.52	17.87
9	CNN + BiLSTM	Layers: Conv1D 64x64, 3x3 | MaxPooling1D | Bidirectional LSTM 128 embedding_dim: 300, max_sequence_length: 100, dropout_rate: 0.4, batch_size: 64, epochs: 10, optimizer: adam, padding: post, truncating: post, embedding_type: glove, max_words: 20000	73.02	39.43
10	Deep CNN	Layers: Conv1D 128x128, 3x3 | MaxPooling1D | Conv1D 128x128, 3x3 | MaxPooling 1D | LSTM 128 embedding_dim: 300, max_sequence_length: 100, dropout_rate: 0.4, batch_size: 64, epochs: 10, optimizer: adam, padding: post, truncating: post, embedding_type: glove, max_words: 30979	73.59	28.29

CONCLUSIONS

After trying multiple model architectures with various hyper-parameter configurations, we have found a wide range of well-performing models, despite none of them performing better than the simple MLP, which we believe it's reasonable since the dense architecture we have chosen for it makes it easier to capture the complex patterns of the data that can be useful for the classification task. To be precise, our MLP is based on two dense layers, each one with a dropout threshold of 0.5 and with ReLU activation, the first one of them with 512 hidden units and the second with 256.

When it comes to the complex networks, we have found multiple models that reach a test accuracy around 75% and a F1 score achieving around 50%. This makes sense because we are talking about the macro LSTM, and the classes distribution may be very unbalanced, which can bias the macro F1 score even though reaching a good micro score, which in our case overpassed the 80%.

Also it is important to mention that in almost every model we used glove instead of fasttext because it gives better results and we could focus on changing other parameters that could affect the accuracy of the different models. We also tried multiple configurations with no embeddings, but it always returned worse results, so the chosen models include the usage of pre-trained embeddings in all cases but the MLP.

Some general conclusions that we have found is that using "pre" in padding and truncating leads in average to worse results in most, if not all, of the models. Related to the max_words parameter (Dictionary size) it usually provides higher accuracy, apparently having a positive correlation. However, we must consider the risk of overfitting if we use the whole dictionary.

We also noticed that increasing the batch size usually led to lower accuracy. Larger batch sizes often result in less varied gradients, which can hinder optimization. Additionally, using a maximum sequence length of 50 in the LSTM and Bidirectional LSTM models generally yielded better results than a higher maximum sequence length. Moreover, adding more units to the LSTM and Bidirectional LSTM layers had little effect on performance, suggesting that having more units doesn't necessarily lead to better results.

Bidirectional LSTM tends to outperform standard LSTM because it captures information from both directions in a sequence. It processes input both forwards and backwards, allowing the network to consider the entire context surrounding each word or token. This dual-directional processing enables the model to better understand the data, which leads to improved accuracy.

On the other hand, the bidirectional stacked GRU has some good results, the problem is the execution time is slower than the others like in all the stacked neural networks. We have tried with just a bidirectional but did not work as well as the normal GRU, however, adding an additional GRU layer after the bidirectional resulted in a slight improvement of the performance.

One of the last stacked architectures that we have tried is the LSTM + GRU and GRU + LSTM, both of them had worse performance than the others, it may make sense if the hyperparameters are not correctly chosen. Therefore, we consider that it would not be the best option for this problem.

Regarding the Convolutional Neural Networks, from now CNN, we have tried several variations. Firstly, we were not sure if it was a good approach but after some consideration we have decided to try them even if they are more common for volumes or images. The reason behind this approach was essentially how CNN handles relationships between data thanks to their kernels. Our main architectures are essentially a Deep CNN with a final LSTM and a CNN with a Bidirectional LSTM. Even if they achieved good results in the training set, some of them with accuracies near the 90%, in the validation or test set they were outperformed by other models, for this reason we do not delve deeper in those models.

In conclusion, after trying a wide range of different models, the MLP demonstrated the best performance among various tested architectures highlighting the effectiveness of simpler designs. However, the Bidirectional LSTM models showed promising results but might require more precise fine-tuning. On the other hand stacked and hybrid models underperformed, likely due to lack of training or even overfitting. CNN models did not perform as well in validation as in training most likely to overfitting. Overall, these results show how simplicity and accurate fine-tuning could surpass more complex models in some situations.