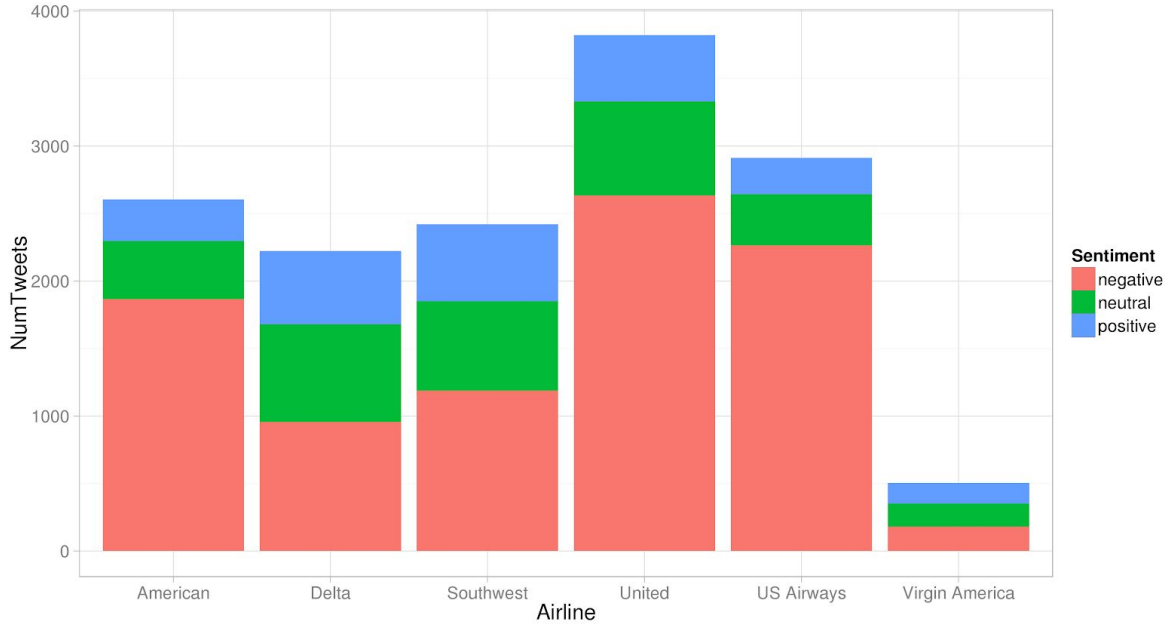
Sentiment prediction from Twitter text messages using word embeddings

Introduction:

Automatic sentiment analysis is a process of capturing human emotions from text. The texts can be usually classified with positive, negative or neutral sentiment. Applications of sentiment analysis are prediction of price of stocks and commodities, automatic customer feedback analysis or market competitor monitoring. It's main advantage is a cost reduction in comparison with human text analysing.

Dataset:

As a dataset we used text messages from famous social network Twitter called tweets. More specifically, we used the dataset of tweets about customer experience with major US airlines, which is freely available on Kaggle. This dataset contains 14485 tweets about services of these airlines and as we can see on the image below, they are mostly negative, since people are much more likely to express negative emotion than positive. We must be aware of the distribution, because the classifier might be biased to classify all the tweets with negative sentiment.



Tweet sentiment distribution for various US airlines in our dataset

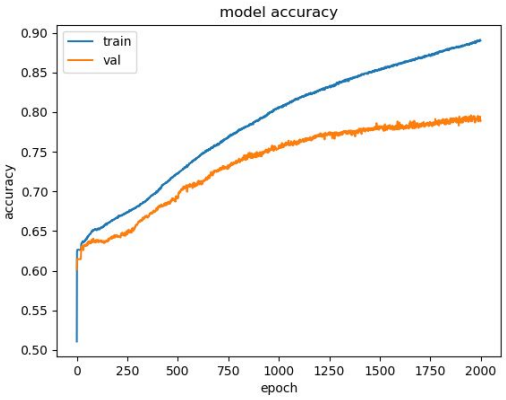
Preprocessing:

We need to split the text into sequences before processing them by the embedding layer. First we used keras Tokenizer to split our texts to lists of tokens. Subsequently, they are transformed into the sequences of integers with equal size. Before we got to actual training we split a test sample of 10%.

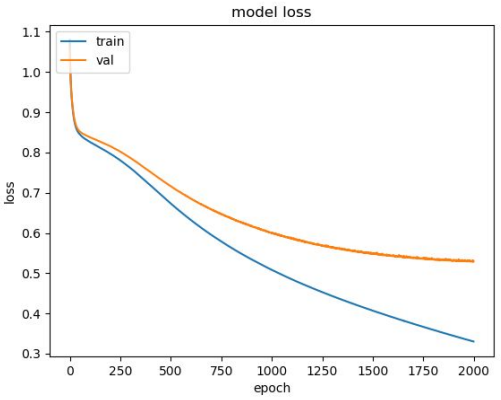
Training an embedding layer:

First we trained the embedding layer on our dataset data. We need to specify the number of number of words, number of dimensions and the length of sequences. We started with a very simple network:

1. Embedding layer with 10000 words, 300 dimensions and the sequence length of 24 words.
2. Softmax classifier.
3. Stochastic gradient descent for network parameter optimization.
4. Batch size is 512 and training lasts for 2000 epochs.



Model accuracy on train and validation dataset



Model loss on train and validation dataset

Test loss: 0.5036909026526363
Test accuracy: 0.799863387978142
Analysis of results

	precision	recall	f1-score	support
Negative	**0.86**	**0.90**	**0.88**	**940**
Neutral	**0.65**	**0.63**	**0.64**	**292**
Positive	**0.73**	**0.62**	**0.67**	**232**
avg / total	**0.80**	**0.80**	**0.80**	**1464**

[[843 69 28]
[83 183 26]
[57 30 145]]

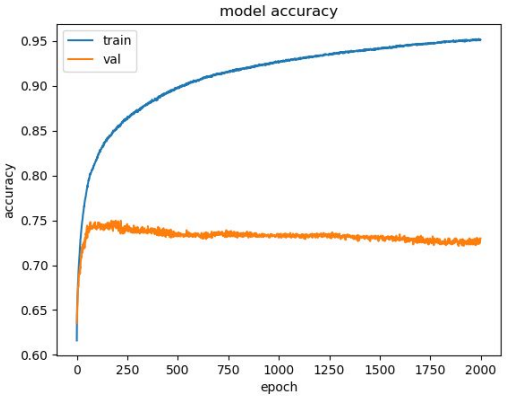
Our model has reached an accuracy of about 79% and we can identify the beginning of overfitting on epoch 500 and almost no accuracy improvement since epoch 1500. The high number of epochs required to reach it's best possible accuracy is caused by 3 000 000 trainable parameters in embedding layer and high batch size - 512.

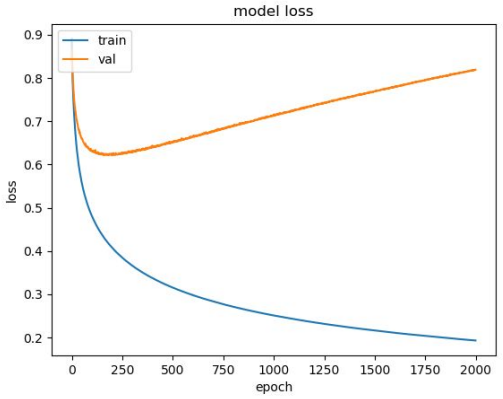
We can notice a sharp increase in the first epochs, which is caused by high number of tweets with negative sentiment and the network will start with classing all the tweets as negative. In addition we can also notice a high recall on negative test samples in confusion matrix, what supports this theory. This is also affecting the accuracy on individual classes, since negative class has by far highest accuracy.

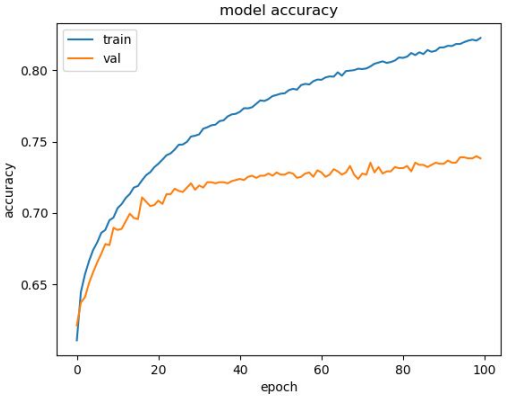
Using pre trained GloVe embeddings in embedding layer:

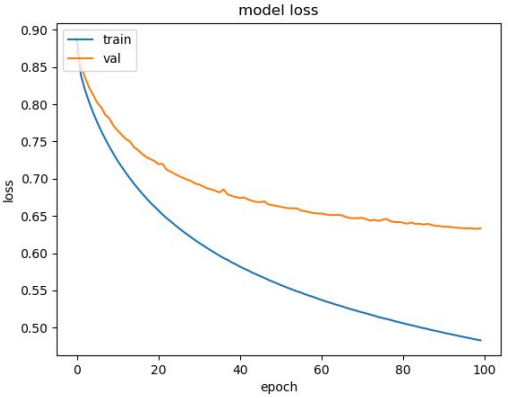
Instead of training our own embeddings we can load pretrained embeddings. We experimented with very popular pretrained embeddings - Global vector for word representation (GloVe).

Before we can use these embeddings in our network we must transform them to a matrix of number of words used in our network times the number of dimensions of our chosen embeddings. We prohibit the network from modifying these values by setting trainable flag to false.









Model accuracy on train and validation dataset (experiment on the top is trained with 2000 epochs and the bottom one with 100 epochs)

Model loss on train and validation dataset (experiment on the top is trained with 2000 epochs and the bottom one with 100 epochs)

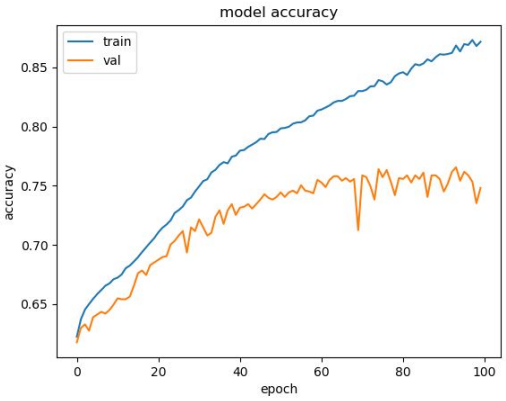
The plots show us that using GloVe embeddings shortens the time required to achieve a good accuracy (40 epochs to reach 73% vs almost 1000 epochs in the previous solution), but it is not able to reach as good as results as custom trained embeddings. The peak accuracy of this model is a little bit less than 74%, while the previous model reached 79% accuracy.

Adding CNN to the solution using GloVe embeddings:

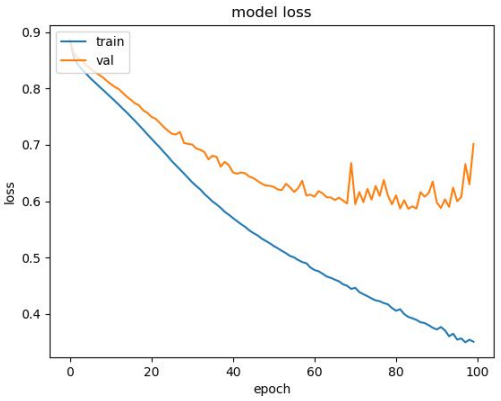
During the training of an embedding layer the network optimizes about 3 000 000 parameters, but in the case we use GloVe embeddings then network can only optimize 3 neurons in the softmax classifier layer. We tried to increase the power of the classifier by adding a convolutional or recurrent neural network between the embedding layer and the classifier layer.

We experimented with one dimensional convolutional neural network, which is subsequently flattened into a dense fully connected layer. Here is our current network architecture:

1. Embedding layer with the sequence length 24 and 300 dimensions.
2. One dimensional convolutional neural network with 128 neurons and relu activation function.
3. One dimensional max pooling with pool size 5.
4. A dense fully connected layer with 1024 neurons.
5. Softmax classifier.



Model accuracy on train and validation dataset



Model loss on train and validation dataset

The 1D CNN networks are not particularly effective for sentiment analysis. We have been able to achieve an improvement in models accuracy in the range of 1-1.5% over a model with single embedding layer with GloVe weights. On the other hand, this improvement has been achieved at the expense of a negligible increase of training time.

Adding GRU RNN to the solution using GloVe embeddings:

An alternative to using 1D convolutional neural networks is to use recurrent neural networks, which are able to capture patterns in sequences of variable length. We experimented with recurrent neural network composed of Gated rectified units (GRU), which have similar performance as Long short-term memory (LSTM) units, but are faster to train.

We removed the CNN layer and replaced it with GRU RNN layer with 128 neurons. Since the time to train of GRU layer is much longer than training CNN layer, we modified the batch size to 64 in order to faster see the effect of training. All other parameters remained the same as in the previous experiment.

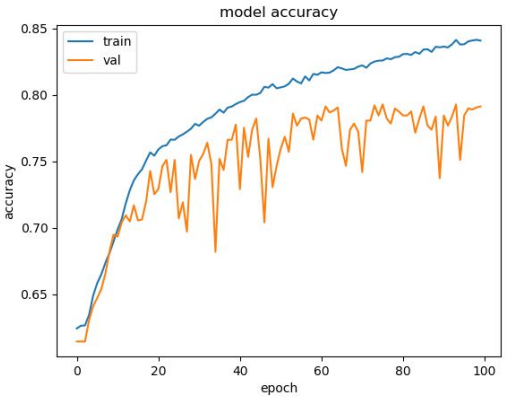
Test loss: 0.5427099072216638

Test accuracy: 0.7950819678645317

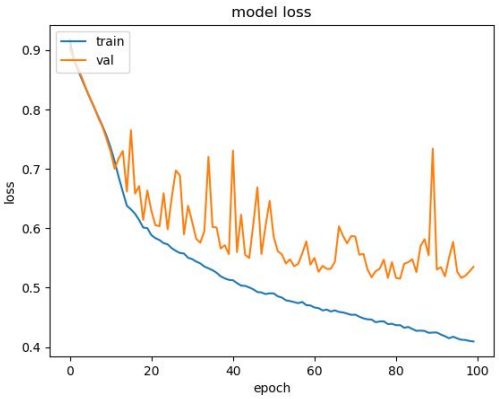
Analysis of results

	precision	recall	f1-score	support
Negative	0.81	0.96	0.88	940
Neutral	0.75	0.39	0.51	292
Positive	0.75	0.66	0.70	232
avg / total	0.79	0.80	0.78	1464

```
[[899  21  20]
 [147 113  32]
 [ 64  16 152]]
```



Model accuracy on train and validation dataset



Model loss on train and validation dataset

The GRU RNN layer increased the accuracy of the model to approximately 80%, thus reaching the accuracy of the single custom trained embedding layer. On the other hand, the results on the test dataset show us a network bias to classify all tweets with negative sentiment (high recall on negative sentiment). We can conclude that training of an embedding layer improves the network ability to get more even recall per class and is in general more robust solution, although in the end both solutions achieve similar accuracy.

The spikes are most probably caused by smaller batch size in comparison with previous experiments.

Conclusion of the Glove embeddings experiments:

Despite running dozens experiments with both CNN and RNN networks with embedding layer using GloVe weights we were unable to get a notable and consistent improvement over training of a single embedding layer.

The maximum achieved accuracy using CNN 1D network was almost 77%. This is just a small improvement over 75% accuracy, that we can achieve using a single one dense 1024 neuron layer between glove embedding layer and softmax classifier.

On the other hand, the GRU RNN network showed a promising improvement of 5%. This improvement can be contributed to the fact, that some of the tweets are longer than our

sequence length of 24 words allowing RNN to use its ability to remember previous computations.

In conclusion, GloVe embeddings provide a quickly trainable solution to tweet sentiment analysis. However, the accuracy of this solution is lower than training a custom embedding layer from the dataset. We can match this accuracy by adding a GRU RNN network to the model, but it will make training time comparable with our first model.

Experiments with the embedding layer:

As model with custom trained embedding layer achieved the best results and RNN network considerably improved GloVe model, it is natural to combine them into a new model and expect an improvement. We have added one GRU 128 neuron layer and one dense 1024 neuron layer to the first model. The batch size remained at 64.

Test loss: 0.6669543916410435
Test accuracy: 0.784398907755242
Analysis of results

	precision	recall	f1-score	support
Negative	**0.83**	**0.92**	**0.87**	**940**
Neutral	**0.69**	**0.48**	**0.56**	**292**
Positive	**0.73**	**0.67**	**0.70**	**232**
avg / total	**0.78**	**0.79**	**0.78**	**1464**

[[867 47 26]
[121 140 31]
[59 17 156]]

However, this combined solution actually achieves a little bit worse accuracy then both models - only about 78%. This might be caused by the huge number of parameters the network is optimizing at the same time. We can try a little bit different approach and that is to train our embedding layer first and than use its weights in RNN network.

Before we will get to the implementation of this solution we will first try to experiment with the parameters of the training of the embedded layer, so we save the weights from the model with the best accuracy.

Changed parameter	Value	Model loss (epochs)	Model accuracy (epochs)
None	**-**	0.503690902652636 (2000)	0.799863387978142 (2000)
Number of words	20000	0.503394031785224 (2000)	0.785519125683060 (2000)
Number of words	5000	0.507857027275314 (2000)	0.78483606622518 (2000)
Batch size	256	0.502342971947675 (1000)	0.797131147540983 (1000)

Batch size	128	0.505341870537221 (500)	0.794398907103825 (500)
Batch size	**64**	0.501468679944022 (250)	0.796448087431694 (250)
Max sequence length	**30**	0.504374781267239	0.803961748633879
Max sequence length	18	0.523687624866193	0.786202185792349
Number of dimensions	500	0.523687624866193	0.786202185792349
Number of dimensions	**100**	0.497227505256569	0.797131147540983

Table with models modifications and model results for single custom trained embedding layer model

As we can see adjusting the number of the most common embedded words does not affect the model as the most important words for sentiment analysis are quite common.

Decreasing the batch size also does not affect the model's precision, but slightly decreases the learning time of the model.

A maximum sequence length lower than 20 words tends to increase the loss of the model. On the other hand increasing the maximum sequence length does not change the models loss and the optimal sequence length is somewhere between 20-35 words.

Surprisingly a reduction of dimensions of the embedded layer decreases the models loss by a notable margin, although the loss start to increase when the number of dimensions drops below 50.

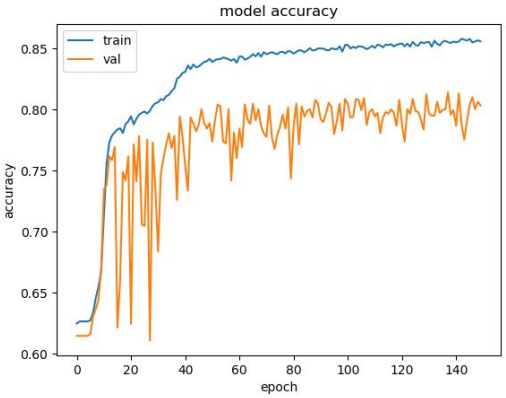
Eventually we used the parameters in bold for our final model and reached following accuracy and loss:

Test loss: 0.49922750525656945

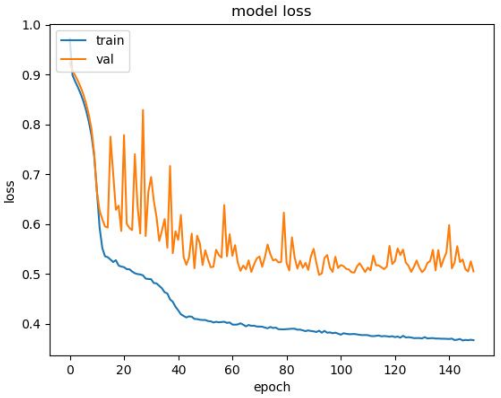
Test accuracy: 0.7957650273224044

Saved trained embedding layer weights and RNN network:

We have trained a model with single embedding layer and saved its weights. In the next step, we loaded these weights into our new RNN network with similar parameters as our last GloVe embedding experiment. We only changed following parameters number of dimensions to 100, max sequence length to 30, batch size to 64. We decided to train model for 150 epochs, because it is still able to learn on the epoch 100.



Model accuracy on train and validation dataset



Model loss on train and validation dataset

Test loss: 0.4969428062764673
Test accuracy: 0.8019125686317193
Analysis of results

	precision	recall	f1-score	support
Negative	**0.88**	**0.88**	**0.88**	**940**
Neutral	**0.64**	**0.64**	**0.64**	**292**
Positive	**0.68**	**0.69**	**0.69**	**232**
avg / total	**0.80**	**0.80**	**0.80**	**1464**

[[828 74 38]
[69 186 37]
[43 29 160]]

The model has achieved a slight improvement over previous models and is the only one, which has consistently scored above 80% accuracy and below 0.5 loss. It is also the fastest model to achieve accuracy of over 75%. Our model achieves this accuracy in approximately 40 epochs.

Conclusion of the paper:

The sentiment analysis is a challenging problem and convolutional or recurrent neural networks did not achieved any serious improvement over single trained embedding layer. Therefore we can conclude that word embedding is a powerful tool in tackling sentiment analysis problem and are usually responsible for the most of the models performance.

Precomputed GloVe embeddings achieved lower accuracy than custom trained embeddings. However, in combination with RNN network they were able to close the gap between them.

In conclusion, we would probably need more data to further improve our model and the maximum accuracy of the model is approximately 80%. We have also noticed, that simple models often works better than complex models with multiple layers and one of the most accurate model used only single custom trained embedding layer.