News category classification from headlines using recurrent neural networks

Github repository url:

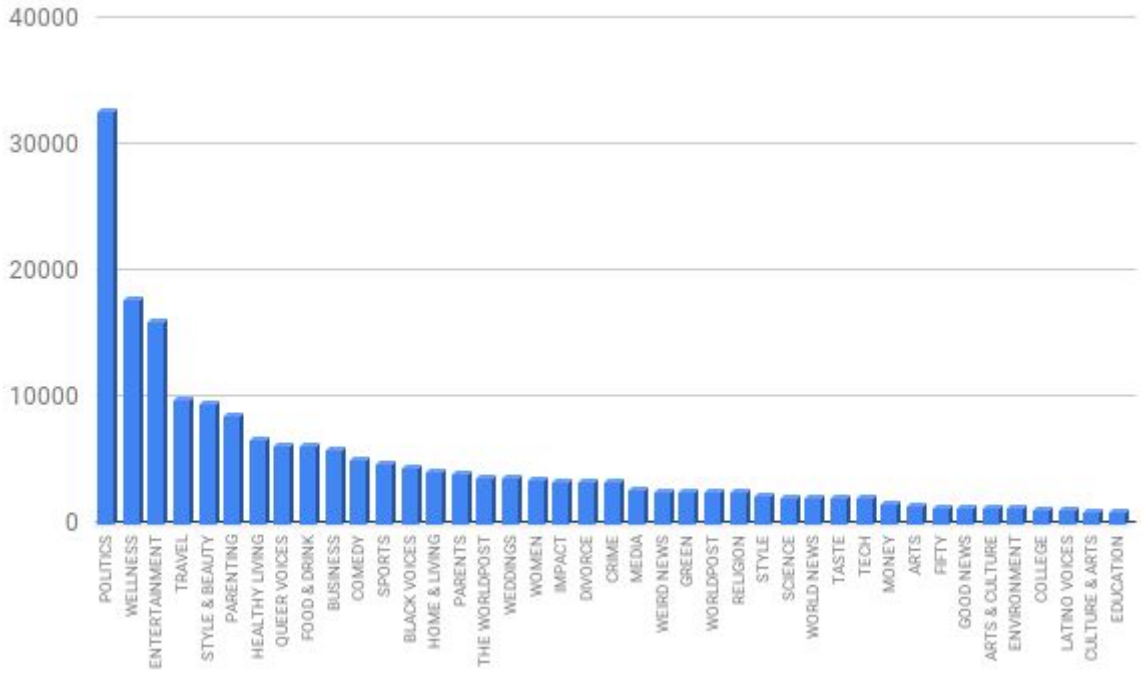
https://github.com/xdolnak/UPC-DL

Introduction:

An expansion of the internet has considerably increased the amount of information that is available to the public. A major part of this information is nowadays published in the form of short internet news. These news are usually tagged with a specific category in order to ease the search for the user. This categorization is usually done manually by an employee of the news. We experimented with news categorization using recurrent neural networks to automate this process.

Dataset:

As a dataset we used news headlines obtained from major american online news HuffPost, which is freely available at Kaggle. This dataset contains over 200000 news classified into 41 classes, which were published between 2012 and 2018. The distribution of news (which can be seen below) into different classes is not uniform and it varies between 1004 and 32739 samples. We must be aware of the distribution, because the classifier might be biased to classify all news with the most common classes.



Distribution of news in the dataset to the 41 categories

Preprocessing:

Our solution is based on the example for tweet sentiment classification provided on the guided lab for recurrent neural networks. This solution transforms raw text into sequence of words by removing all non letter characters from the text. In addition, since this solution manually splits the dataset into 80% data used for training, 10% data used for validation and 10% data used for testing, we had to sort the data by category and then shuffle it in order to guarantee a fair distribution of classes among all 3 data splits.

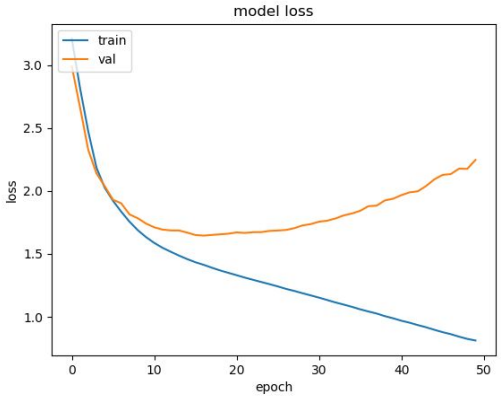
Our solution:

We dealt with a classification problem into 41 classes and used categorical cross entropy to compute the loss of our model. This is a standard loss function for multiclass classification. As an optimizer we used a stochastic gradient descent. There might be an argument for experimenting with adam, rmsprop or other optimizers, but our task was focused on the specific aspects of recurrent neural networks, so we kept all other parameters of the network consistent during all experiments in order to be able to evaluate our results.

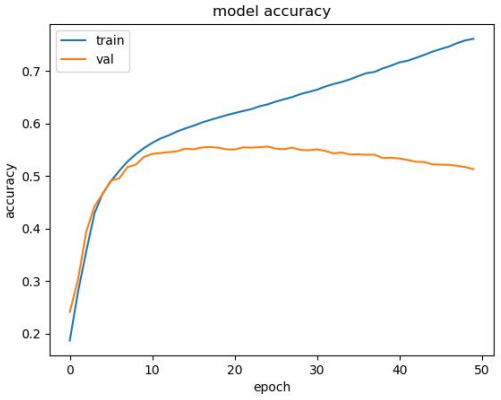
The first layer of the neural network is word embedding layer consisting of 5000 words and 50 dimensions. It is followed by one or several recurrent unit cell layers depending on the experiment setup. At the end of the network is dense softmax classifier with 41 neurons, which is responsible for the classification.

Initial setup:

The initial experiment ran with a single LSTM 128 neuron layer between embedding and classification layers. The experiment ran for 50 epochs with a batch size of 100 samples. Since training set included over 160000 samples, the network needed almost 1 minute to process 1 epoch. The total time of this experiment was 46 minutes and 54 seconds.



Model loss on train and validation dataset



Model accuracy on train and validation dataset

Classification Report:

class	precision	recall	f1-score	support
0	0.19	0.19	0.19	153
1	0.17	0.14	0.15	154
2	0.36	0.33	0.35	465

3	0.36	0.33	0.35	562
4	0.34	0.27	0.30	130
5	0.45	0.42	0.44	562
6	0.49	0.47	0.48	359
7	0.39	0.29	0.34	106
8	0.64	0.65	0.65	305
9	0.30	0.24	0.27	96
10	0.57	0.57	0.57	1614
11	0.29	0.23	0.26	138
12	0.24	0.12	0.16	153
13	0.61	0.58	0.59	599
14	0.26	0.20	0.23	142
15	0.31	0.28	0.30	291
16	0.27	0.20	0.23	703
17	0.68	0.65	0.66	404
18	0.20	0.18	0.19	316
19	0.39	0.28	0.33	120
20	0.36	0.33	0.34	264
21	0.32	0.31	0.31	162
22	0.44	0.54	0.48	846
23	0.36	0.22	0.27	413
24	0.69	0.76	0.72	3280
25	0.61	0.61	0.61	622
26	0.45	0.35	0.39	268
27	0.36	0.40	0.38	207
28	0.52	0.54	0.53	472
29	0.31	0.24	0.27	234
30	0.71	0.69	0.70	954
31	0.30	0.20	0.24	206
32	0.32	0.31	0.31	209
33	0.43	0.38	0.40	370
34	0.61	0.68	0.64	989
35	0.75	0.70	0.72	374
36	0.21	0.21	0.21	250
37	0.48	0.60	0.53	1775
38	0.27	0.31	0.29	315
39	0.25	0.21	0.23	225
40	0.31	0.35	0.33	241
avg / total	**0.50**	**0.51**	**0.50**	**20048**

Test accuracy: 0.5096767620441444

We noticed that model started to overfit at the 8th epoch and it stopped decreasing the loss between 10th and 20th epoch. Moreover, the model had significantly higher accuracy for the most common classes and this shows how the distribution of the number of classes affects the results.

In conclusion, we decided to lower the number of epochs for our next experiments to 20 and we will increase it only if the model does not overfit after 20 epochs.

Comparison of various recurrent unit cells in our solution:

The first natural question to answer is which recurrent unit cells should we use and how deep should be our neural network. We decided to experiment with 3 different recurrent unit cells implementations, that are supported by Keras - long short term memory cells (LSTM), gated recurrent units (GRU) and simple recurrent unit cells (vanilla RNN or simple RNN). We compared these recurrent unit cells in various configurations and the results are presented in the table below.

Recurrent unit cell	Network configuration	Accuracy	Experiment time
LSTM	1 x LSTM - 128 neuron layer	0.5533220140294656	0:24:56
LSTM	2 x LSTM - 128 neuron layer	0.5456903315515682	0:37:56
LSTM	3 x LSTM - 128	0.5348164279902	1:11:05

	neuron layer	828	
GRU	1 x GRU - 128 neuron layer	0.5492318294651015	0:19:55
GRU	2 x GRU - 128 neuron layer	0.5606045350670243	0:38:27
GRU	3 x GRU - 128 neuron layer	0.5567138742028193	0:57:40
SimpleRNN	1 x SimpleRNN - 128 neuron layer	0.5233938440276162	0:06:43
SimpleRNN	2 x SimpleRNN - 128 neuron layer	0.4820430835222874	0:12:33

Comparison table between simple LSTM, GRU and simple recurrent unit cells

In spite of the shortest training time simple recurrent unit cells we decided to discontinue their usage, because they achieve the lowest accuracy. It might be caused by the fact their layer has less than third of trainable parameters(32896) in comparison with either LSTM(131584) or GRU(98688) layers and thus have lower information capacity.

Experiments with GRU networks take about 10-20% less time than LSTM experiments. This can be attributed to the fact, that GRU performs only 3 matrix multiplications during one pass through the network, but LSTM needs 4.

Somewhat surprising are results with multiple recurrent layers. Experiments with LSTM networks showed a consistent decrease in accuracy with every added layer. On the other hand, GRU networks achieved approximately similar results irrespective on the number of layers.

In conclusion, we decided to continue experiments with the best performing GRU network, which had two 128 neuron layers.

Optimization of the number of neurons and batch size:

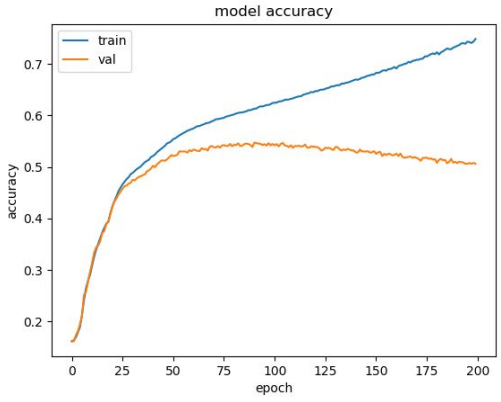
Previous GRU network experiments showed a prospect of improvement in the accuracy of the model with an increase in the number of layers Since we fixed the type and number of layers of our network we decided to switch our attention to the size of these layers. We also decided to optimize the training time of our network by experimenting with various batch sizes and by adjusting the number of epochs accordingly. In general, an increase of the batch size should be accompanied with an increase of the number of epochs in order to allow the network to train under comparable training conditions.

Modified parameter	Network configuration	Accuracy	Experiment time

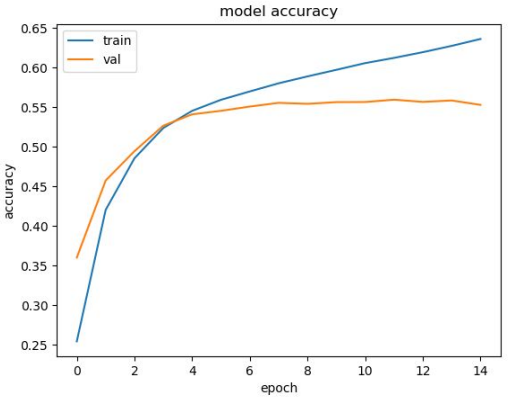
Number of neurons	2 x GRU - 256 neuron layer	0.5525738093987143	0:42:57
Number of neurons	2 x GRU - 512 neuron layer	0.5508280005558719	1:06:55
Number of neurons	2 x GRU - 64 neuron layer	0.5549680638674822	0:27:48
Batch size	100 - batch size, 200 epochs	0.5113727269130808	1:04:56
Batch size	50 - batch size, 15 epochs	0.558958483269737	0:55:03

Comparison table between networks with different configurations

Experiments with the number of neurons in the networks layers did not have any influence on the results. In spite of an increase in the information capacity of the network the accuracy is fluctuating around 55%.



Accuracy evolution of the model with 1000 batch size and 200 epochs



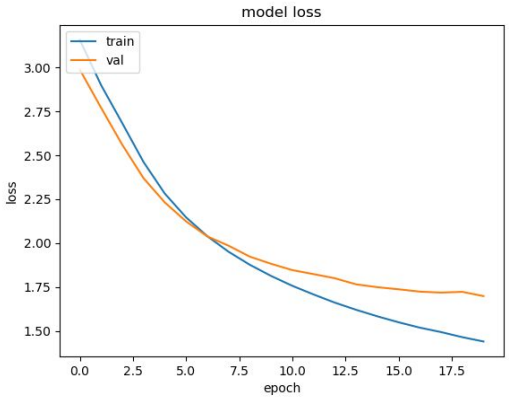
Accuracy evolution of the model with 50 batch size and 15 epochs

Experiments with batch size did not have any influence on the results either, but we noticed a more consistent accuracy between training and validation, when we increased batch size and number of epochs. This is caused by the smaller steps this network makes during convergence to the maximum.

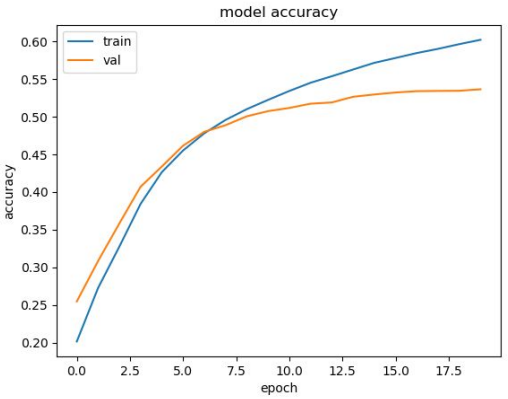
Experiments with the embedding layer:

We noticed in the previous experiments that modifications in the recurrent layers have relatively little impact on the overall results of our model. This led us to a hypothesis that a major part of the important information for the classification is obtained from the first embedding layer and not from the following recurrent layers.

We ran several experiments without recurrent layers and replaced them with fully connected or 1 dimensional convolutional layers. Surprisingly, the best model is the one with single embedding layer and without any fully connected layers or convolutional layers. We present the results of this model below.



Model loss on train and validation dataset



Model accuracy on train and validation dataset

Test accuracy: 0.5416001478196713
Training time: 0:01:49

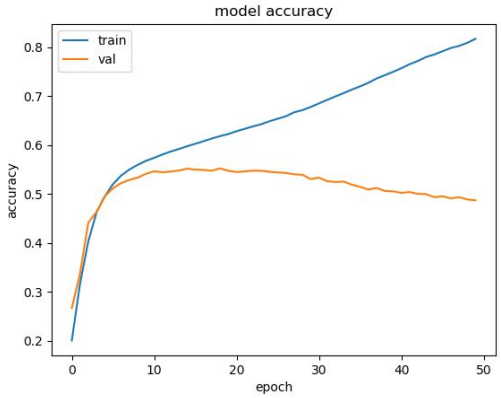
The single embedding layer of the model is responsible for most of the accuracy of the model and can be trained in less than 2 minutes. This casts a doubt on the usage of recurrent layers in our network as they drastically increase the training time of the network, while providing only 1-2% increase of accuracy.

We tried to prove the previous findings by contradiction. There can be two main reasons, why recurrent layers so far did not improve our model. Either they are ovefitting or they do not have enough information capacity and we need to add more fully connected or dense layers. If neither of them would turn out correct, then we proved our findings from the previous paragraph.

Added dropout to the recurrent layers:

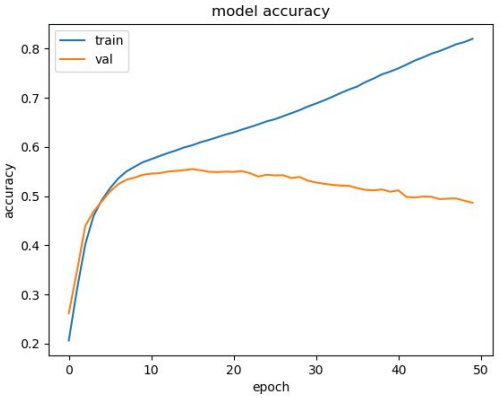
We decided to apply dropout to decrease overfitting. The dropout was only applied in the recurrent layers and not in the embedding layer, because we try to test the hypothesis from previous experiment.

In general, it is recommended to define the dropout value between 0.2 and 0.5. We ran several experiments with various dropout values for GRU and LSTM networks and present results for two experiments with 2x GRU 128 neuron layers and training time of 50 epochs. We provide results for the border values (0.2 and 0.5) of the dropout for our base network.



Model with 0.2 dropout accuracy on train and validation dataset

Test accuracy: 0.49695729882904555



Model with 0.5 dropout accuracy on train and validation dataset

Test accuracy: 0.4895750183823198

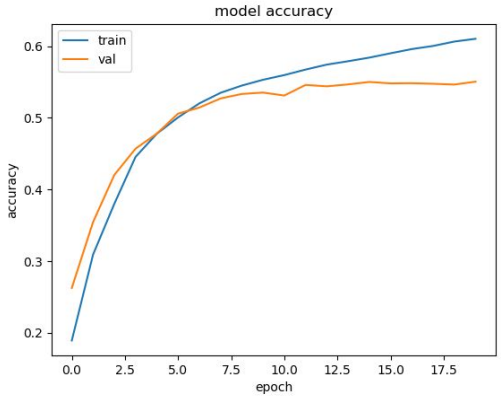
The increase in the dropout rate led to a decrease in the accuracy of the model. This shows us that overfitting did not happen in the recurrent layers, but in the embedding layer. Unfortunately, Keras does not support dropout in the embedding layer and therefore we cannot confirm it by an experiment.

In conclusion of this part, we refuted that our recurrent layers are overfitting and therefore not improving our model.

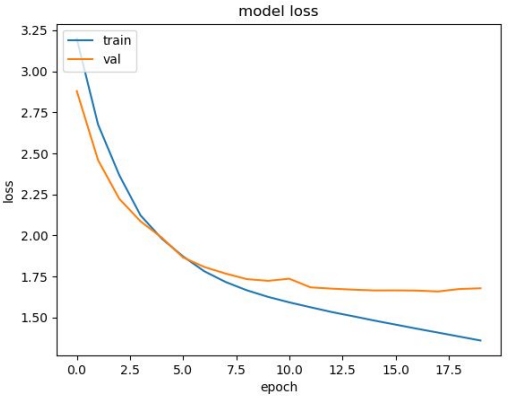
Increase in the number of layers of our model:

We tried to add fully connected layers between recurrent layers and softmax classifier as their addition helped in our experiments with image classification using convolutional networks.

We ran several experiments with various setup of fully connected layers. We experimented with 2 x 256 neuron layers, 3 x 128 neuron layers, 1 x 128 neuron layer and 1 x 1024 neuron layer, which results we present below.



Model accuracy on train and validation dataset



Model loss on train and validation dataset

Test accuracy: 0.5500299253479443

The additional layers decreased the accuracy of model, rather than increasing it. This results proved, that it is the first embedding layer, which has most of the information used for the classification.

Conclusion:

In conclusion, we proved that most of the accuracy of text classification neural network is in the embedding layer of the network. The position of words in the headline is not as important for the classification as their meaning. Adding recurrent layers increased the accuracy of our model by cca. 2% at the expense of 20 time higher training time. Therefore, we recommend that a good news classification model can be trained in very short time using single embedding layer with enough dimensions.

Simple recurrent unit cells have the shorter training time than GRU or LSTM, but do not provide any improvement in the accuracy. GRU provided higher accuracy as well shorter training time than LSTM, so we recommend to use them as default recurrent unit cells in the recurrent neural networks.