IMDB Review Classification With Deep Learning and GloVe Word Embeddings

Olga Pichuzhkina,
Higher School of Economics

Dataset

Source: http://ai.stanford.edu/~amaas/data/sentiment/index.html

The core dataset contains 50,000 rated movie reviews from IMDB split evenly into 25k train and 25k test sets. The overall distribution of labels is balanced: 25k positive (rating >= 7) and 25k neg (rating <= 4). No more than 30 reviews are allowed for any given movie.

Goal: a model that predict the rating of a given review.

Preprocessing

- All reviews were tokenized using NLTK tweet tokenizer (because it includes some non-word tokens that may be sentiment markers -- i. e., ‘:-)’) trained on training set
- No stop words removal, because some stop words (i. e., negation words) may also be sentiment markers
- Reviews were then transformed to sequences and padded to length 100

Approach 1: neural network without embeddings

Layer (type)	Output Shape	Param #
dense_4 (Dense)	(None, 512)	51712
activation_4 (Activation)	(None, 512)	0
batch_normalization_3 (Batch	(None, 512)	2048
dense_5 (Dense)	(None, 256)	131328
activation_5 (Activation)	(None, 256)	0
batch_normalization_4 (Batch	(None, 256)	1024
dense_6 (Dense)	(None, 8)	2056
activation_6 (Activation)	(None, 8)	0

Total params: 188,168
Trainable params: 186,632
Non-trainable params: 1,536

Train on 22500 samples, validate on 2500 samples

1000 epochs, batch size 100

Train accuracy: 0.5533

Validation accuracy: 0.0824

Test accuracy: 0.1613

High enough accuracy on training set, but VERY low on validation and test sets. :(

Approach 2: neural network with GloVe word embeddings

Source: https://nlp.stanford.edu/projects/glove/

Pre-trained on a dataset of 6 billion tokens 50-dimensional word vectors with a vocabulary of 400 000 words.

Approach 2: neural network with GloVe word embeddings

Layer (type)	Output Shape	Param #
embedding_7 (Embedding)	(None, 100, 50)	6149300
flatten_3 (Flatten)	(None, 5000)	0
dense_21 (Dense)	(None, 512)	2560512
activation_15 (Activation)	(None, 512)	0
dropout_3 (Dropout)	(None, 512)	0
dense_22 (Dense)	(None, 256)	131328
activation_16 (Activation)	(None, 256)	0
dropout_4 (Dropout)	(None, 256)	0
dense_23 (Dense)	(None, 8)	2056
activation_17 (Activation)	(None, 8)	0

Total params: 8,843,196
Trainable params: 2,693,896
Non-trainable params: 6,149,300

Train on 22500 samples, validate on 2500 samples

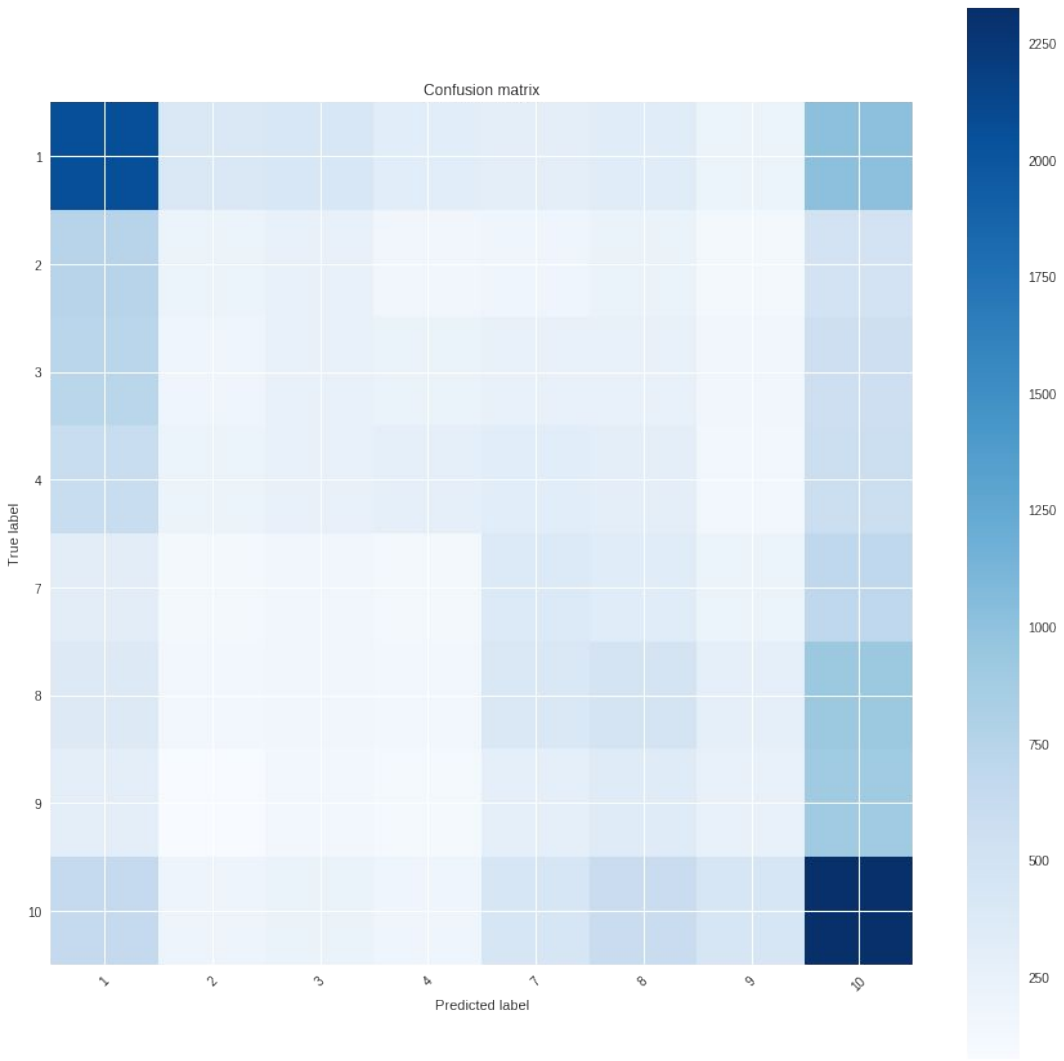
1000 epochs, batch size 100

Train accuracy: 0.9966

Validation accuracy: 0.2344

Test accuracy: 0.2479

Still low, but better. But then, it's almost impossible to predict the exact rating of a movie review by its text alone.



A matrix that shows which labels are more often confused with others. As you can see, ratings 1 and 10 are almost always predicted correctly.

Examples of classification

```
new_text = """What a unique production! Netflix definitely got something (very) right here. In fact, I wish there would be more quality content like this.
Every single episode is intriguing and spectacular; it can be very violent at times, funny and sad. The art is just breathtaking. Writing is on point.
So far I liked "Sonnie's Edge", "The Witness" and "Beyond the Aquila Rift, "Good Hunting" and "Zima Blue" the best. More, please, Netflix!"""
#source -- https://www.imdb.com/review/rw4723901 , рейтинг -- 10
x_new = pad_sequences([text_to_sequence(tokenizer.tokenize(new_text), vocab, vocab_size)], maxlen=100, padding='post')
cl_new = model.predict_classes(x_new)[0]
print(classes[cl_new])
```

```
9
```

```
new_text = """If nobody does, I will. After the critically panned Date Movie, Epic Movie, and Meet the Spartans, these two assholes decide to make another. It's t
I saw it with a few friends, hoping maybe I could get maybe one or two laughs out of it (I didn't expect anything). When I came out, I was seriously stunned at ho
I'm not even going to bother explaining the plot, since there really isn't a plot at all. All I can say is, don't waste your money watching the film. It contains
#source -- https://www.imdb.com/review/rw1938122 , рейтинг -- 1
x_new = pad_sequences([text_to_sequence(tokenizer.tokenize(new_text), vocab, vocab_size)], maxlen=100, padding='post')
cl_new = model.predict_classes(x_new)[0]
print(classes[cl_new])
```

```
1
```

Thank you for attention!

The code is available here: https://github.com/vyhuholl/imdb_review_classification