Théo Szymkowiak Follow
@McGillU CS Grad. Founder of the McGill AI Society (@AiMcgill, https://mcgillai.com). Full Stack Dev at Clay (https://clay.run). Entrepreneur.
Feb 17, 2017 · 3 min read

How to implement Sentiment Analysis using word embedding and Convolutional Neural Networks on Keras.

On the Imdb movie reviews dataset.

Imdb has released a database of 50,000 movie reviews classified in two categories: Negative and Positive. This is a typical sequence binary classification problem.

In this article, I will show how to implement a Deep Learning system for such sentiment analysis with ~87% accuracy. (State of the art is at 88.89% accuracy).

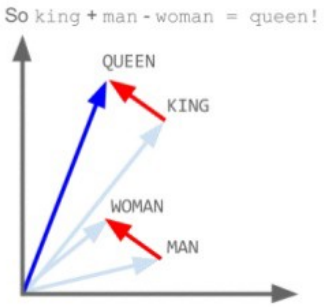
Keras

Keras is an abstraction layer for Theano and TensorFlow. Meaning that we don't have to deal with computing the input/output dimensions of the tensors between layers.

How to represent the words

Movie reviews are sequences of words. So first we need to encode them.

We map movie reviews to sequences of word embeddings. Word embeddings are just **vectors that represent multiple features of a word**. In Word2Vec, vectors represent relative position between words. One simple way to understand this is to look at the following image:



This comes from the deck of slides: http://www.slideshare.net/ChristopherMoody3/word2vec-lda-and-introducing-a-new-hybrid-algorithm-lda2vec-57135994

```
king - man + women = queen
```

After mapping every movie review to sequences of word embeddings, we need to pad the sequences to get the same length on all of them. i.e. we add zeroes to the small sequences and truncate the larger ones.

```
1  from keras.preprocessing import sequence
2  max_review_length = 1600
3
4  X_train = sequence.pad_sequences(X_train, maxlen=max_review_
```

The model

Here we used a 3-layered convolution neural network with 2 dense layers.

Why Convolutional? Because it works. Convolutional layers are really powerful to extract higher level feature in images. And quite amazingly, they actually work in most 2D problems. Another big reason that should convince you is the training time, CNN train 50% to 60% faster than LSTMs on this problem.

The Keras model code:

```
from keras.layers import LSTM, Convolution1D, Flatten, Drop
from keras.layers.embeddings import Embedding
from keras.models import Sequential

max_review_length = 1600
embedding_vecor_length = 300
model = Sequential()
model.add(Embedding(top_words, embedding_vecor_length, inpu
model.add(Convolution1D(64, 3, border_mode='same'))
model.add(Convolution1D(32, 3, border_mode='same'))
model.add(Convolution1D(16, 3, border_mode='same'))
model.add(Flatten())
```

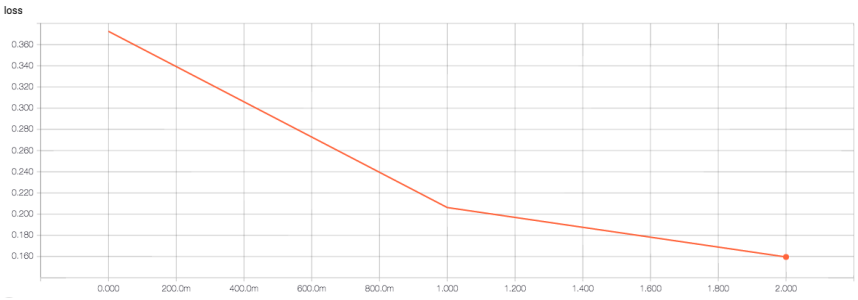
This model has ~7M trainable parameters, which takes around 15min on a MacBook Pro (We used binary cross entropy loss here because it is a binary classification problem).

In this case the Convolutional layers extract features horizontally from multiple words. Allowing the network to extract higher level writing style.

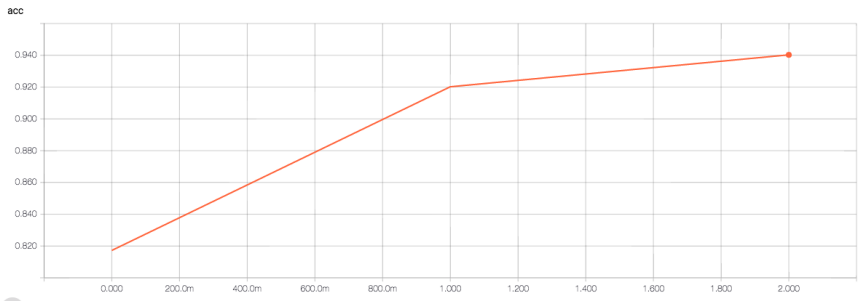
Dropout was necessary because otherwise, the model was overfitting to the training data (96% accuracy on training data, 84% on test data). Crippling the network with holes during training reinforces the generalization power; it forces the network to build new paths and extract new patterns.

Results

After 20 short minutes of training, we get **86.6%** accuracy (87% if you are lucky). This is another advantage of CNN, they are **extremely fast** to train compared to LSTM for the same result (in this case at least).



Cumulative loss (Training set)



Accuracy (Training set)

State of the art is 88.89%.

GITHUB REPOSITORY: https://github.com/Theo-/sentiment-analysis-keras-conv