Deep Learning with Jon Krohn - My Take

Zach McCormick
Senior Software Engineer

Course Experience

5 Saturdays, 40 hours, 10 units, 1 presentation.

Tons of Hands-on Material

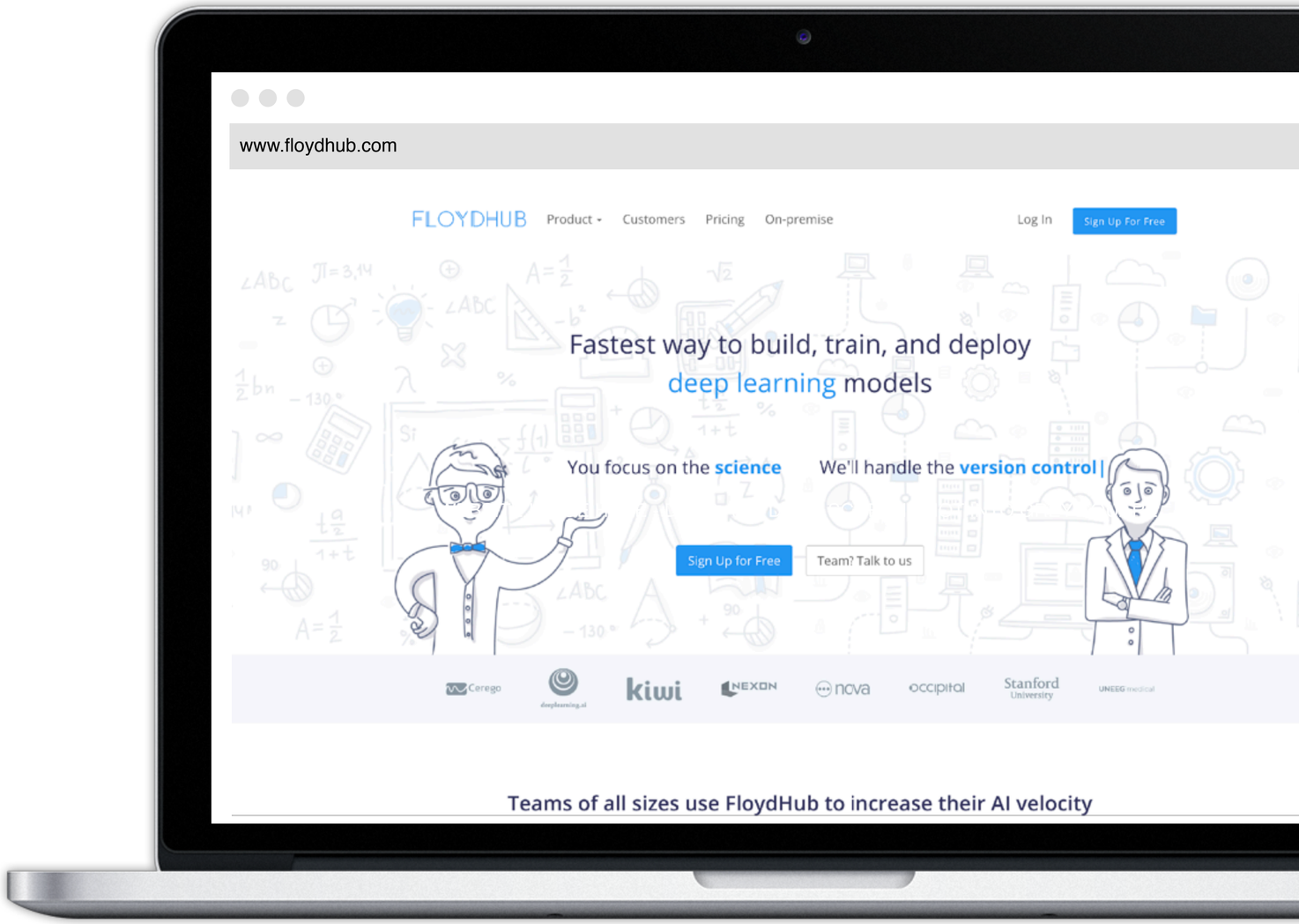
- Walking through concepts like Gradient Descent
- Various models in Jupyter Notebooks

Ongoing Concept Map/Glossary

- Highlighted all of the important terms
- Easy to do more research offline

Capstone Project

- Great way to tie everything together

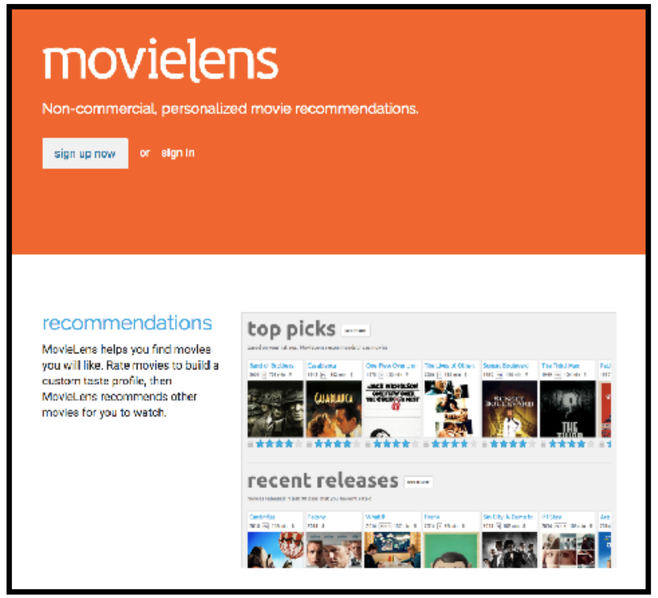


You might be asking...

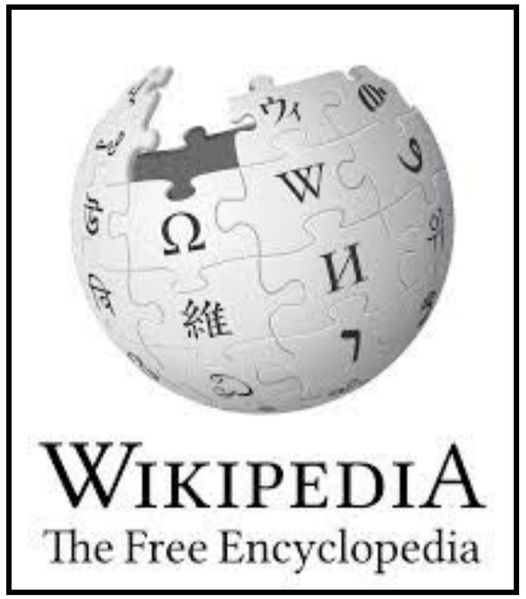
What was my capstone project?

Can we predict movie ratings by using plot summaries?

Can we predict ratings by the plot?



+



=

What if we take movie ratings from MovieLens' dataset...

... and combine them with plot summaries scraped from Wikipedia?

Can we predict the rating of our awesome screenplay???

Load dependencies and manipulate data

Our `plots` CSV has the movie year in the title field and a bunch of extra columns. Our `movie` and `rating` CSVs are normalized and have extra columns as well, not to mention that the ratings are all per-user and need to be averaged. Let's use `pandas` to manipulate our data into a single dataframe that has ratings and plots together.

[1]:
```
import pandas as pd
import numpy as np
import re

# load CSVs
movies_df = pd.read_csv('movie.csv')
rating_df = pd.read_csv('rating.csv')
plots_df = pd.read_csv('wiki_movie_plots_deduped.csv')

# average ratings per film
grouped_ratings_df = rating_df.drop(columns=['userId']).groupby(['movieId']).mean()

# merge movies and ratings
joined_df = pd.merge(movies_df, grouped_ratings_df, left_on="movieId", right_on="movieId").drop(columns=['movieId', 'genres'])

# remove year from title
joined_df['title'] = joined_df['title'].apply(lambda x: re.sub("\(\d\d\d\d\)", "", x).strip())

# remove extraneous columns from plots
clean_plots_df = plots_df.drop(columns=['Release Year', "Origin/Ethnicity", 'Director', 'Cast', 'Genre', 'Wiki Page'])

# merge ratings and plots
df = pd.merge(joined_df, clean_plots_df, left_on='title', right_on='Title', how='inner').drop(columns=['Title']).rename(index=str, columns={"Plot": "plot"})

# confirm that our data looks good!
print(df.head())
```

Load dependencies for using Keras

[2]:
```
import keras
from keras.datasets import imdb
from keras.preprocessing.sequence import pad_sequences
from keras.preprocessing.text import Tokenizer
from keras.models import Sequential, load_model
from keras.layers import Dense, Dropout, Embedding, SpatialDropout1D, LSTM
from keras.layers.wrappers import Bidirectional
from keras.callbacks import ModelCheckpoint
import os
import os.path
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
%matplotlib inline
```

```
Using TensorFlow backend.
```

Set hyperparameters

[3]:
```
# training:
epochs = 2  # seem to keep overfitting after 2 epochs
batch_size = 128

# vector-space embedding:
n_dim = 64
n_unique_words = 10000
max_review_length = 1000
pad_type = trunc_type = 'pre'
drop_embed = 0.2

# LSTM layer architecture:
n_lstm_1 = 64 # lower
n_lstm_2 = 64 # new!
drop_lstm = 0.2
```

Tokenize plot data, split train/test, and pad sequences to max length

[4]:
```
tokenized_df = df.copy()
t = Tokenizer(num_words=n_unique_words)
t.fit_on_texts(tokenized_df['plot'])
tokenized_df['plot'] = t.texts_to_sequences(tokenized_df['plot'])
print(tokenized_df.head())

msk = np.random.rand(len(df)) < 0.8
train_x = tokenized_df[msk]['plot'].values
train_y = tokenized_df[msk]['rating'].values

test_x = tokenized_df[~msk]['plot'].values
test_y = tokenized_df[~msk]['rating'].values


train_x = pad_sequences(train_x, maxlen=max_review_length, padding=pad_type, truncating=trunc_type, value=0)
test_x = pad_sequences(test_x, maxlen=max_review_length, padding=pad_type, truncating=trunc_type, value=0)
```

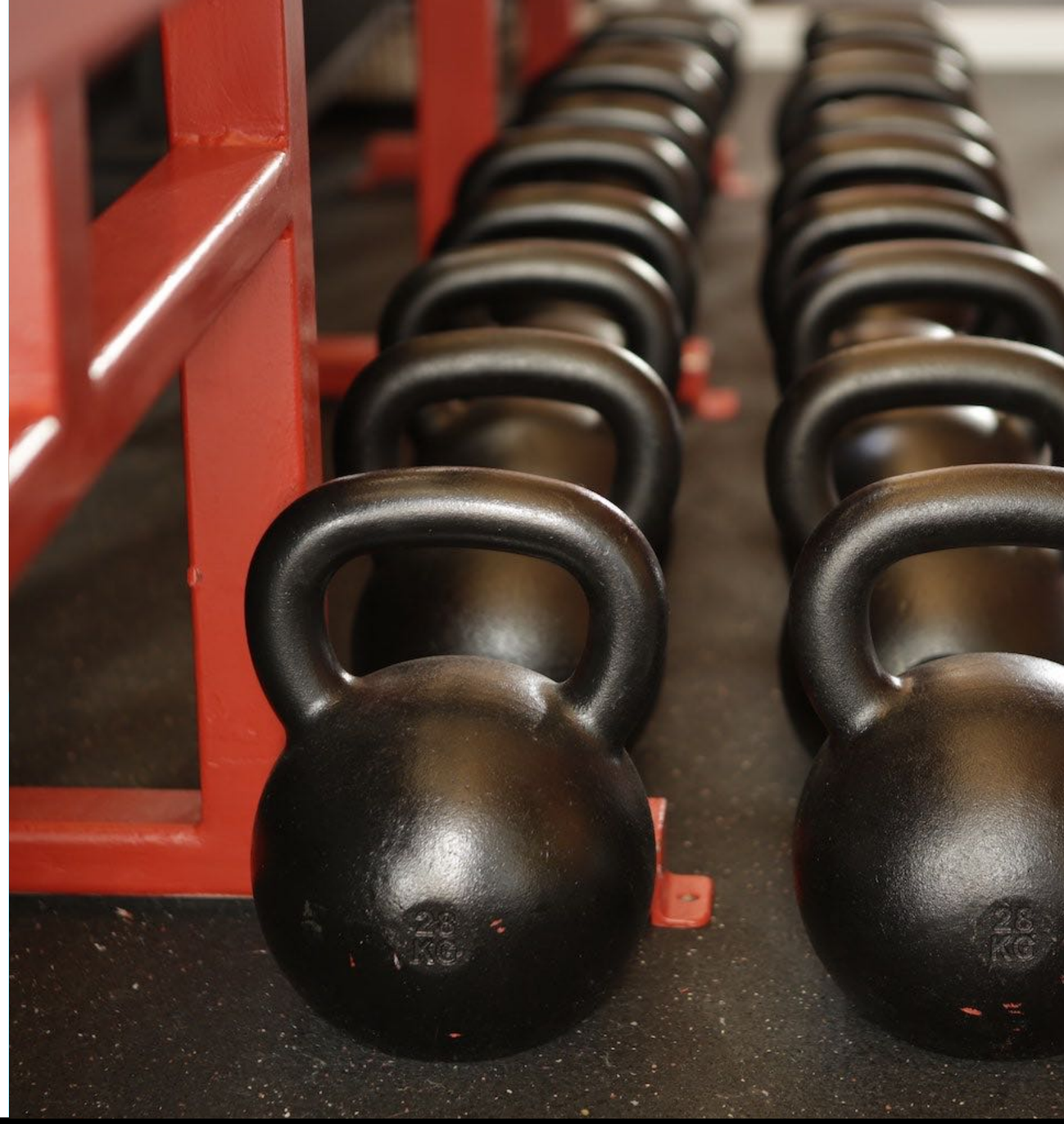
```
                        title    rating  \
0                   Toy Story  3.921240
1                     Jumanji  3.211977
2            Grumpier Old Men  3.151040
3           Waiting to Exhale  2.861393
4  Father of the Bride Part II  3.064592

                                                plot
0  [7, 4, 212, 45, 3357, 30, 324, 496, 23, 3914, ...
1  [7, 514, 52, 49, 802, 4354, 4, 1780, 3, 1030, ...
2  [1, 5342, 209, 370, 706, 3, 127, 139, 25, 3, 1...
3  [138, 30, 1, 211, 23, 438, 436, 36, 229, 438, ...
4  [1, 100, 116, 691, 129, 29, 1, 742, 5, 1, 91, ...
```

Design LSTM neural network architecture

[5]:
```
model = Sequential()
model.add(Embedding(n_unique_words, n_dim, input_length=max_review_length))
model.add(SpatialDropout1D(drop_embed))
model.add(Bidirectional(LSTM(n_lstm_1, dropout=drop_lstm, return_sequences=True))) # retain temporal dimension
model.add(Bidirectional(LSTM(n_lstm_2, dropout=drop_lstm)))
model.add(Dense(1, activation='linear'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 1000, 64)          640000
_________________________________________________________________
spatial_dropout1d_1 (Spatial (None, 1000, 64)          0
_________________________________________________________________
bidirectional_1 (Bidirection (None, 1000, 128)         66048
_________________________________________________________________
bidirectional_2 (Bidirection (None, 128)               98816
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 129
=================================================================
Total params: 804,993
Trainable params: 804,993
Non-trainable params: 0
_________________________________________________________________
```

this is the part where we wait a really long time for model training...



Let's see how "wtf is Star Wars" scores!

[8]:
```
# plot from: https://noisey.vice.com/en_us/article/65z4zb/wtf-is-star-wars
plot = """
Eventually Princess Leia gets rescued but Obi Wan dies and becomes a blue ghost.
So Luke Skywalker's like, "Oh hell no, I'm gonna go fight this a-hole, to whom I
bear no familial relation that I know of... yet." But Yoda's like, "A training sequence,
there must be." So Luke gets trained in the ways of the Jedi Knights. At the end of
the training, Yoda says, "May the force be with you and I hope annoying people don't post
this on Facebook every May for all eternity." He goes to the Death Star where the
Jedis are fighting the Storm Troopers (this is what's known as "a star war"). Boba Fett
is there riding around on a jet pack, shooting everyone in the star war.

Luke finds Darth and breaks out his light saber. Luke hears Yoda in his head: "Use
The Force." There's a big light saber fight between the blue light saber and the red
one. Vader gets a good shot in and chops Luke's hand off and drops the bomb on him that
he's his father all Maury Povich-like. And Luke is like, "Noooooooo!" Han Solo comes along
and saves them and blows up the Death Star and then gets frozen in carbonite.
"""
my_plot = t.texts_to_sequences([plot])
my_plot = pad_sequences(my_plot, maxlen=max_review_length, padding=pad_type, truncating=trunc_type, value=0)
print(my_plot[0])
model.predict(my_plot)
```

How does it stack up?

It probably didn't deserve this rating...

```
[8]: array([[3.136698]], dtype=float32)
```

In my amateur model, most reviewed movie plots ended up scoring around the median/mean score, which makes sense, since that's the safest bet for a reward function when you don't really have a clue.

Overall, I learned a lot about developing neural networks for things like this, but I don't think plot summaries alone are a good indicator of how well people are going to like a movie!

Thanks!