SENTIMENT ANALYSIS ON AMAZON PRODUCT REVIEWS BY USING LONG SHORT-TERM MEMORY NEURAL NETWORKS AND FASTTEXT CLASSIFIERS

ANLY-580 COURSE PROJECT

YI LI

PROBLEM STATEMENT

Sentimental analysis on Amazon product reviews by using two kinds of models:

- long short-term memory (LSTM) neural networks
- fastText classifiers (efficient text classifiers developed by Faceback AI Research)

Both kinds of models take a sequence of words as an input and produces a probability distribution over the predefined classes.

DESCRIPTION OF THE DATA USED

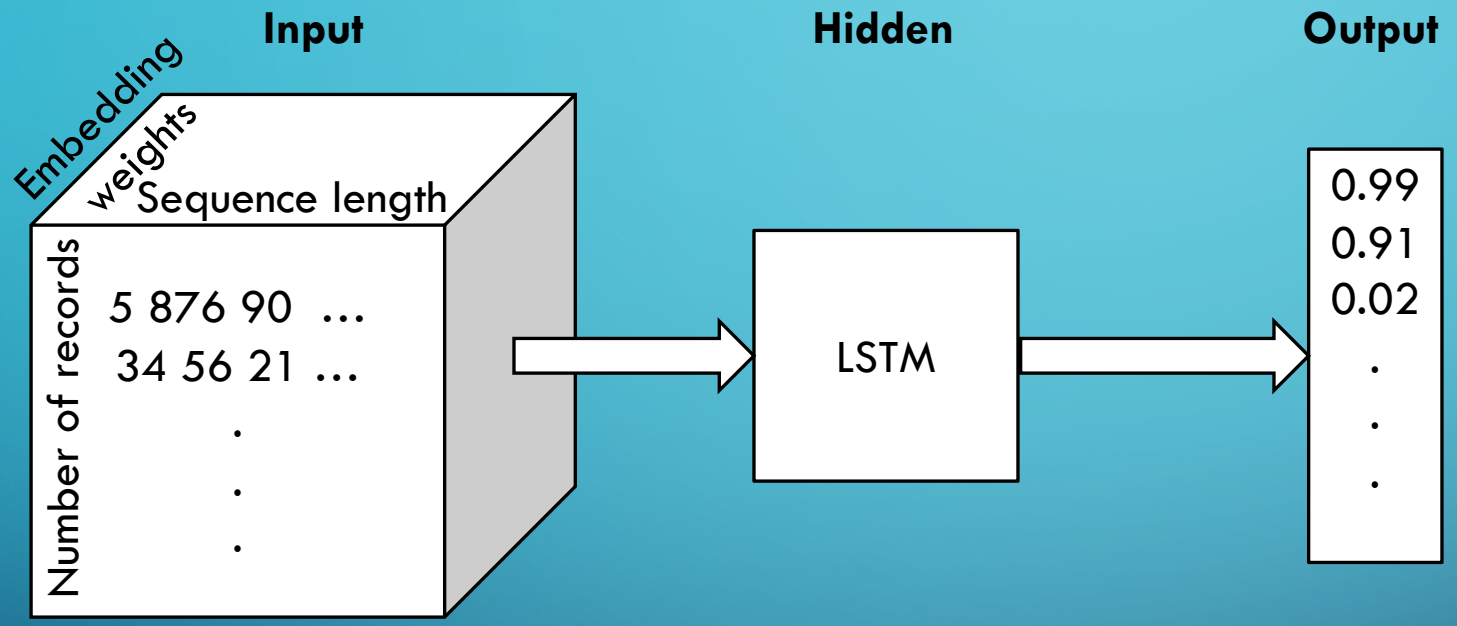
Amazon product review samples:

- __label__1 Batteries died within a year ...: I bought this charger in Jul 2003 and it worked OK for a while. The design is nice and convenient. However, after about a year, the batteries would not hold a charge. Might as well just get alkaline disposables, or look elsewhere for a charger that comes with batteries that have better staying power.

Source: https://www.kaggle.com/bittlingmayer/amazonreviews/data

METHODOLOGY

LSTM



1) Load pre-trained google news word vectors to create an ID's matrix of the text data

2) Build, train, and test LSTM models

Tools: Keras, Tensorflow, NLTK (tokenization)

FastText

Model: FastText

(an efficient text classifier that uses linear algorithms)

Tools: FastText

https://pypi.python.org/pypi/fasttext

RESULTS

Method	Total records	Training data size	Test data size	Accuracy of test data
LSTM	10k	8k	2k	0.82
LSTM	50k	40k	10k	0.85
LSTM	100k	80k	20k	0.87
LSTM	200k	160k	40k	0.88
fastText	4000k	3600k	400k	0.91

```
for i, result in enumerate(results):
    if result > 0.8:
        print("Posi", result, sample_sents[i])
    elif result < 0.2:
        print("neg", result, sample_sents[i])
    else:
        print("neutral", result, sample_sents[i])
```

Manual Test Results

```
neg [ 0.00744833] This is a joke
neg [ 0.00443599] The worst product I have ever bought
Posi [ 0.97385174] I like it, and I want to buy it again
neutral [ 0.52989006] I am going to give a presentation.
neutral [ 0.71282566] I am going to have an exam.
Posi [ 0.99185044] The new tea table looks amazing.
Posi [ 0.99524188] This is the best class ever!
Posi [ 0.99224257] My mom loves it.
Posi [ 0.99034542] nice weather
Posi [ 0.9875195] This is the best 30 bucks that I have ever spent
neg [ 0.00449477] Complete waste of money.
```

SCREEN SHOTS OF CODES

LSTM

```
# build the LSTM model
model = Sequential()
model.add(Embedding(vocab_size, embed_size,
                    name="word_vec",
                    weights=[embeddings_with_unk,]))

model.add(LSTM(64, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss="binary_crossentropy",
              optimizer = "adam",
              metrics=["accuracy",])
```

```
# use the first 80% records to train the model
model.fit(X_matrix[:int(0.8*total_records),], y[:int(0.8*total_records)],
          epochs=3,
          batch_size=1024, # The larger the batch, the better the approximation;
          verbose=1) # Verbosity mode. 0 = silent, 1 = progress bar, 2 = one line per epoch.
```

```
Epoch 1/3
160000/160000 [==============================] - 199s 1ms/step - loss: 0.4200 - acc: 0.8058
Epoch 2/3
160000/160000 [==============================] - 193s 1ms/step - loss: 0.3028 - acc: 0.8734
Epoch 3/3
160000/160000 [==============================] - 193s 1ms/step - loss: 0.2698 - acc: 0.8867

<keras.callbacks.History at 0x13c749c88>
```

```
# use the last 20% records to test the model
score = model.evaluate(X_matrix[int(0.8*total_records):,], y[int(0.8*total_records):],
                       batch_size=1024, verbose=1)
print(score)
```

```
40000/40000 [==============================] - 12s 294us/step
[0.28687794504165648, 0.87809999999999999]
```

FastText

References: https://pypi.python.org/pypi/fasttext

```
import fasttext
```

```
classifier = fasttext.supervised('train.ft.txt', 'model',
                                 lr=0.1, dim=300,
                                 label_prefix='__label__')
```

```
result = classifier.test('test.ft.txt')
print('P@1:', result.precision)
print('R@1:', result.recall)
print('Number of examples:', result.nexamples)
```

```
P@1: 0.91594
R@1: 0.91594
Number of examples: 400000
```