TensorFlow & NLP

Junwei Chen

CS 410

Fall 201

Introduction

TensorFlow is an open-source end to end platform for machine learning, deep learning and scientific computing that provides you with rich collection of tools for building models. These include load & preprocess data, build – train = reuse models and finally deploy models using Python. You could run TensorFlow on a variety of devices and platforms.

- ### But why TensorFlow in the first place?

  Not only TensorFlow can be beginner friendly and handy tool for experts. It allows for powerful experimentation. It powers your research into new technique to solve novel problems. Such as predicting extreme weather conditions, diagnosis of diabetic retinopathy or even helping farers spot diseased plants. It's also driven by a worldwide community with a vision to provide a machine learning platform for everyone, in anywhere.

NLP

In today's world, I could say majority of daily things are digitalized, if not, they will be. According to IBM, 1000 million gigabytes of data were generated in 2017 in a daily basis. Of all this data, a large fraction is unstructured text and speech as there are millions of emails and social media content created and phone calls made every day. To put that into perspective, if all the human beings in the world were to process that data, it would be roughly 300 MB for each of us every day to process. The goal for NLP is to make our computers understand our spoken and written languages.

The Main Concepts of NLP But Cut to the Chase

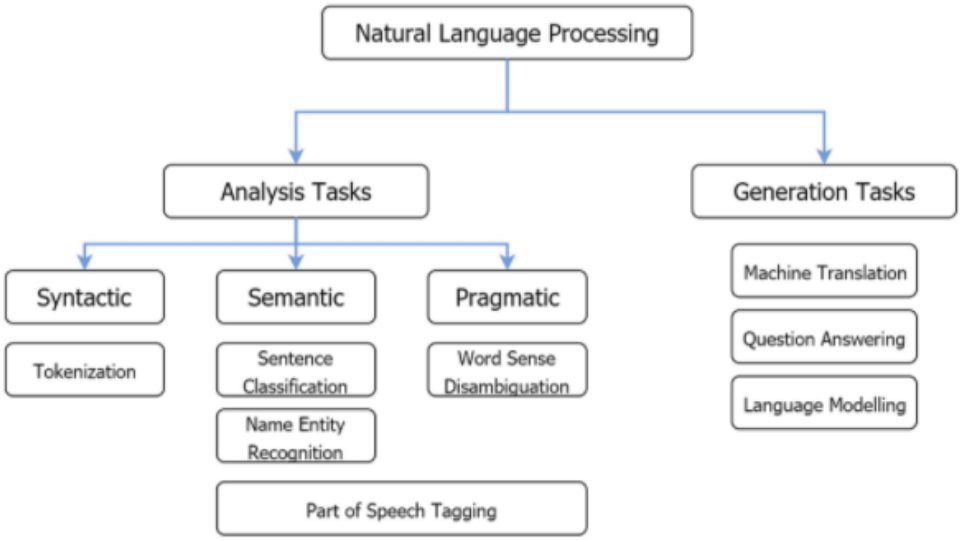
Tokenization – it is the process of splitting up sentences into various "tokens" like individual words and get rid of unimportant characters.

Stop Words Removal – Remove irrelevant words like "and", "the", "or", etc.

Stemming - Convert words to their word stem. Like computing & computed to compute.

Lemmatization – Turning words to its base form and standardizing synonyms to their roots.

Topic Modeling – Unsupervised learning that group different texts under certain subjects.



TensorFlow with Keras For NLP

This TensorFlow example is probably the coolest knowledge that I've gain within this week, which tackled a classic NLP problem with **detecting the emotion of text**. For instance, a sentence can be "When I achieved my long-term goal that I made five years ago, I almost jumped five feet high". Something like that and the output would be "excite".

The process starts with import libraries such tensorflow, nump, nlp. Tokenizer and pad_sequences from "tensorflow.keras.preprocessing". Dataset is imported from package of "nlp" and separates it into training, validation and testing sets.

```
def get_tweet(data):

tweets = [x['text'] for x in data]

labels = [x['label'] for x in data]

return tweets, labels

tweets, labels = get_tweet(train)
```

Above code will separates training data into two arrays: “tweets” and “labels”.

```
tweets[0], labels[0]
```

```
('i didnt feel humiliated', 'sadness')
```

Then using Tokenization which represent words in a way that a computer can process them.

```
tokenizer = Tokenizer(num_words=10000, oov_token='<UNK>')

tokenizer.fit_on_texts(tweets)
```

```
tweets[0]
```

```
'i didnt feel humiliated'
```

```
tokenizer.texts_to_sequences([tweets[0]])
```

```
[[2, 139, 3, 679]]
```

Next, make all sequences the same shape as the ML model expects the input to be a fixed shape and length.

```
maxlen=50

def get_sequences(tokenizer, tweets):

sequences = tokenizer.texts_to_sequences(tweets)

padded = pad_sequences(sequences, truncating = 'post',
padding='post', maxlen=maxlen)

return padded
```

Preparing the model

This step will create a set of all of our labels, and a dictionary that we can use when converting our classes to their indexes and the indexes to classes. Classes will be keys in the dictionary and they are assigned a number from 0 to 5.

```
classes = set(labels)

class_to_index = dict((c,i) for i, c in enumerate(classes))

index_to_class = dict((v,k) for k, v in class_to_index.items())

names_to_ids = lambda labels: np.array([class_to_index.get(x) for x
in labels])

train_labels = names_to_ids(labels)
```

```
[ ] class_to_index

    {'anger': 3, 'fear': 5, 'joy': 2, 'love': 0, 'sadness': 1, 'surprise': 4}
```

Creating, training, evaluating, and testing model

Although TensorFlow made it easy for developers to do their research. It requires some diggings in LSTM layers, RNN architecture and its dependencies, Adam optimizer and sparse

categorical crossentropy. I will link those sources in the citation pages. Here I will only display TensorFlow.

```
model = tf.keras.models.Sequential([

tf.keras.layers.Embedding(10000,16,input_length=maxlen),

tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(20,
return_sequences=True)),

tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(20)),

tf.keras.layers.Dense(6, activation='softmax')

])

model.compile(

     loss='sparse_categorical_crossentropy',

     optimizer='adam',

     metrics=['accuracy']

)
```

```
test_tweets, test_labels=get_tweet(test)

test_seq = get_sequences(tokenizer, test_tweets)

test_labels=names_to_ids(test_labels)

model.evaluate(test_seq, test_labels)
```

```
i = random.randint(0,len(test_labels)-1)

print('Sentence:', test_tweets[i])

print('Emotion:', index_to_class[test_labels[i]])

p = model.predict(np.expand_dims(test_seq[i], axis=0))[0]

print(test_seq[i])

pred_class=index_to_class[np.argmax(p).astype('uint8')]

print('Predicted Emotion: ', pred_class)
```

The code generates random tweet and has the model predict what class it belongs to. It also predicts the tweet and its label so you can compare the prediction and the correct answer.

```
sentence = 'i am not sure what to do'

sequence = tokenizer.texts_to_sequences([sentence])

paddedSequence = pad_sequences(sequence, truncating = 'post',
padding='post', maxlen=maxlen)

p = model.predict(np.expand_dims(paddedSequence[0], axis=0))[0]

pred_class=index_to_class[np.argmax(p).astype('uint8')]

print('Sentence:', sentence)

print('Predicted Emotion: ', pred_class)
```

```
Sentence: i am not sure what to do
Predicted Emotion:  fear
```

Conclusion

Although TensorFlow is often reprimanded over its incomprehensive API, but it does its job perfectly after digging into necessary prerequisite knowledge to TensorFlow. Keras will be the

high-level API for TensorFlow and you can use advanced features of TensorFlow directly from keras packages. With integration between TensorBoard and Keras, now complicated NLP algorithms now can be one-liners. In the future, TensorFlow will consolidate its position as go-to framework for most deep learning tasks, including NLP.

References

https://www.youtube.com/watch?v=yjprpOoH5c8

https://www.packtpub.com/product/natural-language-processing-with-tensorflow/9781788478311

https://medium.com/geekculture/nlp-with-tensorflow-keras-explanation-and-tutorial-cae3554b1290

https://medium.com/geekculture/nlp-with-tensorflow-keras-explanation-and-tutorial-cae3554b1290

https://www.youtube.com/watch?v=fNxaJsNG3-s

https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

https://towardsdatascience.com/tensorflow-vs-pytorch-vs-keras-for-nlp-exxact-8e51dd13c3f5