AI LAB Evaluation

Nov 17, 2017, 1:30 pm to 5:30 pm

Objective

In this exercise we will build a Encoder-Decoder based model that corrects tweets. Given a labelled dataset that consists of tweets and their corrected form, build and train a classifier that can perform corrections.

Steps

(a) Dataset

i. The dataset is a CSV file "consolidated.csv" where the column "tweet" has the uncorrected tweet and the column "corrected" contains the corrected form. The last column "valid" should be 1 that indicates that the tweet is valid. For this lab you can ignore the "sentiment" column.

ii. You are required to select 10000 samples for training and 500 for testing using a random selection process. You can increase this number gradually and observe the performance after you have built the basic model.

(b) Processing the dataset and creating (inputs, targets)

i. You will be using the code examples as discussed in https://machinelearningmastery.com/encoder-decoder-attention-sequence-to-sequence-prediction-keras/

ii. The dataset needs to be prepared as per the model's requirement. Our model will take sequences of vectors as inputs and will predict the output using a Softmax layer. The target for softmax is a one hot vector per time step that is predicted.

iii. First, extract the tweet column, each row is a tweet. You should treat a tweet as a sentence and tokenize it. This will give a sequence of words for each tweet. You are required to use genism Word2Vec.

iv. Create a corpus as per the requirements of genism Word2Vec and train the Word to Vector model. See https://rare-technologies.com/word2vec-tutorial/ and https://radimrehurek.com/gensim/models/word2vec.html

v. Create the input feature vectors for 10000 rows using the word2vec model you created and trained

vi. Read the column "corrected", which is the labelled tweet. For each corrected tweet, do word tokenization. Let us call the vocabulary formed out of these tokens to be output_vocab

vii. Restrict the output_vocab to be maximum of 3500 words. Replace any token that is omitted from the vocabulary with a special word "__unk__" to represent "unknown" words. Thus, the size of output_vocab will be 3500 + 1 in my example. You should choose the words as per their occurrence counts in the corpus. This can be done by setting some minimum count parameter, where this parameter specifies the minimum number of occurrences of a word in order to get included in our output_vocab

viii. For each tweet that you tokenized you will get a sequence of tokens. If the tweet is of length 12 (say), you will get 12 tokens. You need to form 12 (in this example) one hot vectors to represent the output layer. Create a one hot vector, whose length is the size of vocabulary where the elements are all 0's except at the index that corresponds to the word which is the expected token. For example, if your expected token (as seen in the dataset) is "hello" and the index of this word in the output_vocab is 1217, you will create a vector of 3501 with all elements as zero and you will set the element at 1217 to be 1.

ix. Now, for 10K rows you will have 10K sequences constituting the inputs and 10K target sequences, where each target sequence represents the tokens that are predicted.

x. As the length of each sequence is a variable, you may pad the sequences as needed. See the references https://machinelearningmastery.com/encoder-decoder-attention-sequence-to-sequence-prediction-keras/

(c) Building RNN

i. Look at Keras code example and use the encoder decoder model without attention.

ii. The model looks something like this:

```
# define model

model = Sequential()

model.add(LSTM(150, input_shape=(n_timesteps_in, n_features)))

model.add(RepeatVector(n_timesteps_in))

model.add(LSTM(150, return_sequences=True))
```

```
model.add(TimeDistributed(Dense(n_features, activation='softmax')))

model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['acc'])
```

iii. You need to modify the sample code and train this system

(d) Experiment with different sizes of dataset rows, different hyper parameter settings and report resulys

Deliverables

Submit the following by 5:30 pm, 17th Nov 2017:

1. Source code of your classifier
2. Source of the test program

Do the following by 10 pm 19th Nov 2017:

1. Post your results and analysis on the Facebook AI group. Regardless of the accuracy you got, explain in the document what went right and where you could have taken a better approach to getting better results.
2. Optionally you can include any graphics, visualization etc

Best wishes from the faculty, enjoy AI development!

Anantharaman Palacode Narayana Iyer,Shruti Kaivalya