Pavan sai K
18BCB0034
E2

NATURAL LANGUAGE PROCESSING J COMPONENT REVIEW-3:
TEXTUAL ENTAILMENT USING TENSORFLOW

Abstract:

Understanding entailment and contradiction is fundamental to understanding natural language processing, and inference about entailment and contradiction is a valuable testing ground for the development of semantic representations. However, machine learning research in this area has been dramatically limited by the lack of large-scale resources. So in this paper we will be building a small Deep learning model that is not computationally too costly and a model which can be run in any type of PC.The increase in the amount of data by the introduction of SNLI corpus by stanford has provided leeway for lexicalized classifiers to outperform some sophisticated existing entailment models, and it allows a neural network-based model to perform competitively on natural language inference benchmarks.

Textual entailment is useful as a component in much larger applications. For example, question-answering systems may use textual entailment to verify the integrity of an answer from the information stored in the database. Textual entailment may also enhance document summarization by filtering out sentences that are repetitive in nature and sentences that don’t introduce anything new(redundant sentences).

Technology used:

1. **Python**
2. **Tensorflow**
3. **Numpy**
4. **Jupyter notebook(for ease of execution)**

Dataset used:

SNLI(The Stanford Natural Language Inference(SNLI) Corpus)

Pretrained model used:

GloVe(Global Vectors for Word Representation) by Stanford

Hardware requirements:

- Python 3.5–3.8
- pip 19.0 or later (requires manylinux2010 support)

- Ubuntu 16.04 or later (64-bit)

OR

- Windows 7 or later (64-bit)
- Microsoft Visual C++ Redistributable for Visual Studio 2015, 2017 and 2019

Introduction:

Textual entailment is a core NLP process where two sentences are compared with each other, and the output can be either positive,neutral or negative entailment. Positive entailment is where the second sentence can be a direct implication of the first statement. Negative entailment is where the first statement can be used to disprove the second statement and neutral entailment is where both the sentences have no correlation between them. In this article we have built a nlp model using tensorflow that will be able to output the

text entailment given any two sentences.
In this paper we have used an LSTM with a pretrainned Glove model

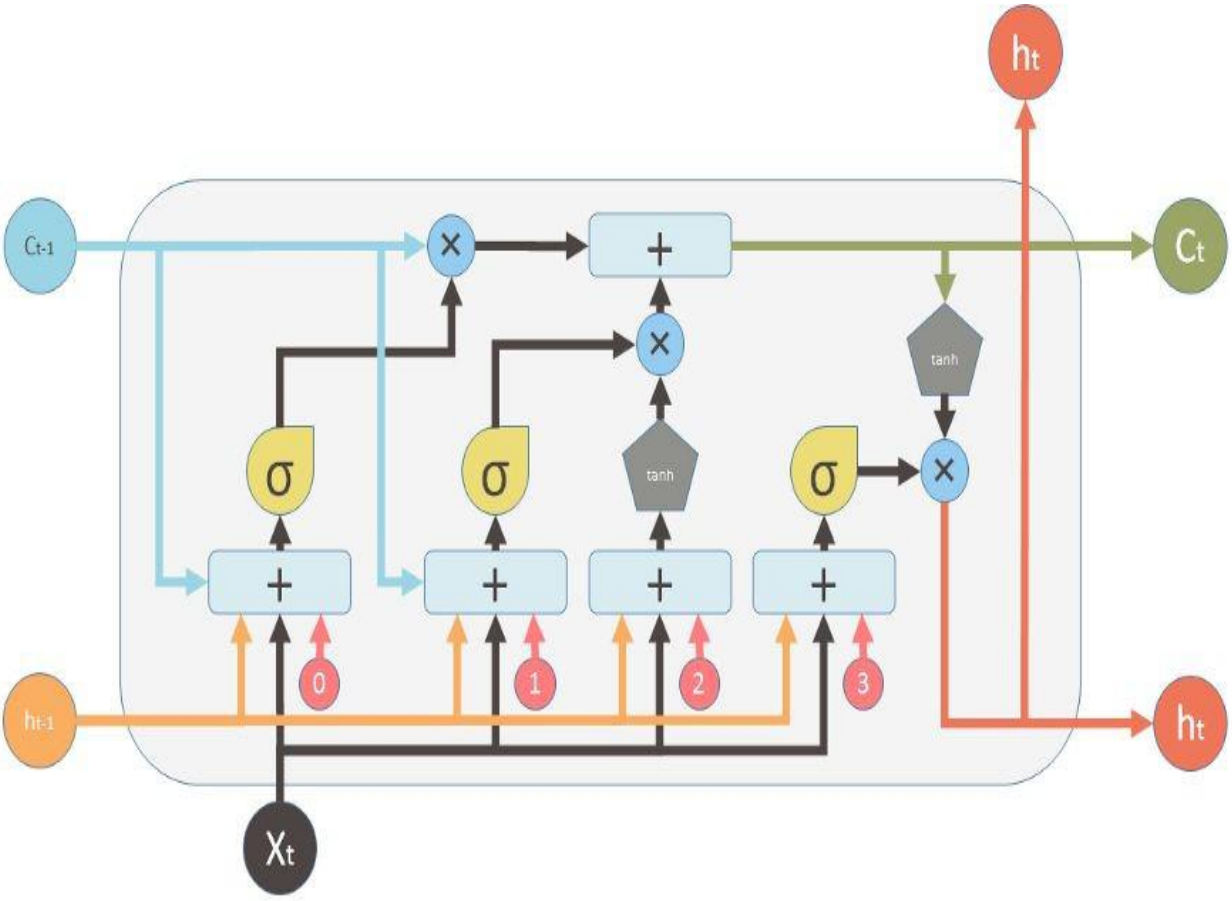
A characteristic of natural language is that there are many different ways to state what one wants to say: several meanings can be contained in a single text and that the same meaning can be expressed by different texts. This variability of semantic expression can be seen as the dual problem of language ambiguity. Together, they result in a many-to-many mapping between language expressions and meanings. The task of paraphrasing involves recognizing when two texts have the same meaning and creating a similar or shorter text that conveys almost the same information.

Problem statement:

1) What are the inputs and the outputs:

The whole agenda of this is project is to output the entailment between the given sentences that are given as input by the user.

2)BLOCK DIAGRAM of the LSTM Cell:



Inputs to the LSTM Cell:
Xt => Input vector
Ct-1 => Memory from the previous block
Ht-1 => Output from the previous block

Outputs:
Ct => Memory of the current block
Ht => Output of the current block

Explanation for Pseudo code:

(N: The number of elements in each of our batches, which we use for training
l_h: The maximum length of a hypothesis, or the second sentence. This is
used because training an RNN is extraordinarily difficult without
rolling it out to a fixed length.
l_e: The maximum length of evidence, the first sentence. This is used
because training an RNN is extraordinarily difficult without
rolling it out to a fixed length.
D: The size of GloVe or other vectors.)

hyp = tf.placeholder(tf.float32, [N, l_h, D], 'hypothesis')
evi = tf.placeholder(tf.float32, [N, l_e, D], 'evidence')
y = tf.placeholder(tf.float32, [N, 3], 'label')

hyp: Where the hypotheses will be stored during training.
evi: Where the evidence will be stored during training.
y: Where correct scores will be stored during training.

lstm_size: the size of the gates in the LSTM,as in the first LSTM layer's initialization.

lstm_back = tf.contrib.rnn.BasicLSTMCell(lstm_size)

lstm_back: The LSTM used for looking backwards through the sentences, similar to a regular lstm.

input_p: the probability that inputs to the LSTM will be retained at each

iteration of dropout.
output_p: the probability that outputs from the LSTM will be retained at each iteration of dropout.
lstm_drop_back = tf.contrib.rnn.DropoutWrapper(lstm_back, input_p, output_p)
lstm_drop_back: A dropout wrapper for lstm_back, like lstm_drop.

fc_initializer = tf.random_normal_initializer(stddev=0.1)
fc_initializer: initial values for the fully connected layer's weights.
hidden_size: the size of the outputs from each lstm layer.
Multiplied by 2 to account for the two LSTMs.
fc_weight = tf.get_variable('fc_weight', [2*hidden_size, 3],
initializer = fc_initializer)
fc_weight: Storage for the fully connected layer's weights.
fc_bias = tf.get_variable('bias', [3])
(fc_bias: Storage for the fully connected layer's bias.

tf.GraphKeys.REGULARIZATION_LOSSES: A key to a collection in the graph designated for losses due to regularization.
In this case, this portion of loss is regularization on the weights for the fully connected layer.)
tf.add_to_collection(tf.GraphKeys.REGULARIZATION_LOSSES,
tf.nn.l2_loss(fc_weight))

x = tf.concat([hyp, evi], 1) # N, (Lh+Le), d
(Permuting batch_size and n_steps)
x = tf.transpose(x, [1, 0, 2]) # (Le+Lh), N, d
(Reshaping to (n_steps*batch_size, n_input))
x = tf.reshape(x, [-1, vector_size]) # (Le+Lh)*N, d
(Split to get a list of 'n_steps' tensors of shape (batch_size, n_input))
x = tf.split(x, l_seq,)

(x: the inputs to the bidirectional_rnn

tf.contrib.rnn.static_bidirectional_rnn: Runs the input through

two recurrent networks, one that runs the inputs forward and one
that runs the inputs in reversed order, combining the outputs.)
rnn_outputs, _, _ = tf.contrib.rnn.static_bidirectional_rnn(lstm, lstm_back,
x, dtype=tf.float32)
(rnn_outputs: the list of LSTM outputs, as a list.
What we want is the latest output, rnn_outputs[-1])

classification_scores = tf.matmul(rnn_outputs[-1], fc_weight) + fc_bias
(The scores are relative certainties for how likely the output matches
a certain entailment:
0: Positive entailment
1: Neutral entailment
2: Negative entailment)

4)Experiment and results:

Dataset:

The pre-trained data for Stanford's GloVe word vectorization is used which consist of six-billion-token Wikipedia 2014 + Gigaword 5 vectors, At the same time, we'll also be picking up our data set for textual entailment: Stanford's SNLI dataset. We'll be using the development set in the interest of speed (it has only 10,000 sentence pairs).

The SNLI corpus (version 1.0) is a collection of 570k human-written English sentence pairs manually labeled for balanced classification with the labels entailment, contradiction, and neutral, supporting the task of natural language inference (NLI), also known as recognizing textual entailment (RTE).

Dataset sample:

Text	Judgments	Hypothesis
A man inspects the uniform of a figure in some East Asian country.	contradiction C C C C C	The man is sleeping
An older and younger man smiling.	neutral N N E N N	Two men are smiling and laughing at the cats playing on the floor.
A black race car starts up in front of a crowd of people.	contradiction C C C C C	A man is driving down a lonely road.
A soccer game with multiple males playing.	entailment E E E E E	Some men are playing a sport.
A smiling costumed woman is holding an umbrella.	neutral N N E C N	A happy woman in a fairy costume holds an umbrella.

Output:

In [23]:
```
evidences = ["Maurita and Jade both were at the scene of the car crash."]

hypotheses = ["Multiple people saw the accident."]

sentence1 = [fit_to_size(np.vstack(sentence2sequence(evidence)[0]),
                         (30, 50)) for evidence in evidences]

sentence2 = [fit_to_size(np.vstack(sentence2sequence(hypothesis)[0]),
                         (30,50)) for hypothesis in hypotheses]

prediction = sess.run(classification_scores, feed_dict={hyp: (sentence1 * N),
                                                        evi: (sentence2 * N),
                                                        y: [[0,0,0]]*N})
print(["Positive", "Neutral", "Negative"][np.argmax(prediction[0])]+
      " entailment")
```

```
/home/pavan/.virtualenvs/tf1.15/lib/python3.5/site-packages/ipykernel_launcher.py:32: FutureWarning: Using a non-t
uple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the fut
ure this will be interpreted as an array index, `arr[np.array(seq)]`, which will result either in an error or a di
fferent result.
```

```
Positive entailment
```

In [24]:
```
sess.close()
```

Conclusion:

The light weight LSTM model gives an accuracy of 75%+ which is a very good accuracy considering how light the model is, this model is specifically aimed at people who want to implement text entailment while using average or below average hardware.This model is especially useful during the pandemic times because developers do not have access to their Office PC, and hence are forced to work from home, without good hardware working with Deep learning models with over 10 million parameters will be close to impossible.This lightweight text entailment model will serve as viable alternative for anyone who is looking for an easily executable model.

References:

1) **Mehdad, Y., Carenini, G., Ng, R. and Joty, S., 2013, June. Towards topic labeling with phrase entailment and aggregation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 179-189).**

2) **Dziri, N., Kamalloo, E., Mathewson, K.W. and Zaiane, O., 2019. Evaluating coherence in dialogue systems using entailment. *arXiv preprint arXiv:1904.03371*.**

3) **Pennington, J., Socher, R. and Manning, C.D., 2014, October. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532-1543).**

4) **Bowman, S.R., Angeli, G., Potts, C. and Manning, C.D., 2015. The SNLI Corpus.**

5) **Bowman, S.R., Angeli, G., Potts, C. and Manning, C.D., 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.**

6) **Liu, Y., Sun, C., Lin, L. and Wang, X., 2016. Learning natural language inference using bidirectional LSTM model and inner-attention. *arXiv preprint arXiv:1605.09090*.**

7) **Chen, Q., Zhu, X., Ling, Z., Wei, S., Jiang, H. and Inkpen, D., 2016. Enhanced lstm for natural language inference. *arXiv preprint arXiv:1609.06038*.**

8) **Rocktäschel, T., Grefenstette, E., Hermann, K.M., Kočiský, T. and Blunsom, P., 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.**

9) **Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N., Peters, M., Schmitz, M. and Zettlemoyer, L., 2018. Allennlp: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*.**

10) **Kang, D., Khot, T., Sabharwal, A. and Hovy, E., 2018. Adventure: Adversarial training for textual entailment with knowledge-guided examples. *arXiv preprint arXiv:1805.04680*.**