Project Report for Assignment 3

Logistic Regression

Approach 1 - Bag of words and TF-IDF

In this approach, I had taken difference between TFIDF of sentence1 and sentence 2 as feature for logistic regression and 'golden_label' as target.
This was giving an accuracy of 60.8%.

Approach 2 – Word2vec and TF-IDF

The different features we computed are as follows:

- *Feature set 1*: First create word embedding using word2vec library in genism package in python. Calculate pair-wise distance between words in both sentence 1 and sentence 2. This is done by computing similarity between words. Then multiply it by TF-IDF of corresponding words in sentence 1 and sentence2. This is because TF_IDF gives significance of a word in the sentence. This is done for every pair of words in sentence 1 and sentence2 and its sum is taken.
- *Feature set 2*: A set of features are computed based on differences between sentences based on number of single character edits. This is done by python library named fuzzy-wuzzy. The measures considered are:
    - Simple ratio
    - Partial ratio
    - Token sort ratio
    - Token set ratio
- *Feature set 3*: First words in a sentence is converted to word-vectors using word2vec library. Then take weighted sum of word embeddings where weights are TF-IDF of corresponding words. This will give rise to 300 odd features corresponding to sentence1 and another 300 features corresponding to sentence 2

Accuracy obtained by this method = 60.1%

I did hyper parameter tuning for C value, which is inverse of regularization strength.
Best value of C = 0.01
I didn't tune multiple parameters simultaneously because it was taking several days in my system to complete.
I had improved the pre-trained word2vec model with training data using transfer learning.
Also, I have used snli_1.0_test.csv, because I have verified using code that snli_1.0_test.jsonl and snli_1.0_test.csv are exactly same.

Code is in log_regression.ipynb
Prediction result is in logistic_regresson.txt

Extensions

If I get more memory bandwidth, I will try to improve this model by adding more features. This will be done by taking k top words for each sentence based on TF-IDF. Then determine

vector embedding of each selected word and add it to feature set. This will be done for sentence1 and sentence2 prior to training.

RNN

I had done data preprocessing for train and test data for sentence 1 and sentence 2. Data preprocessing consists of

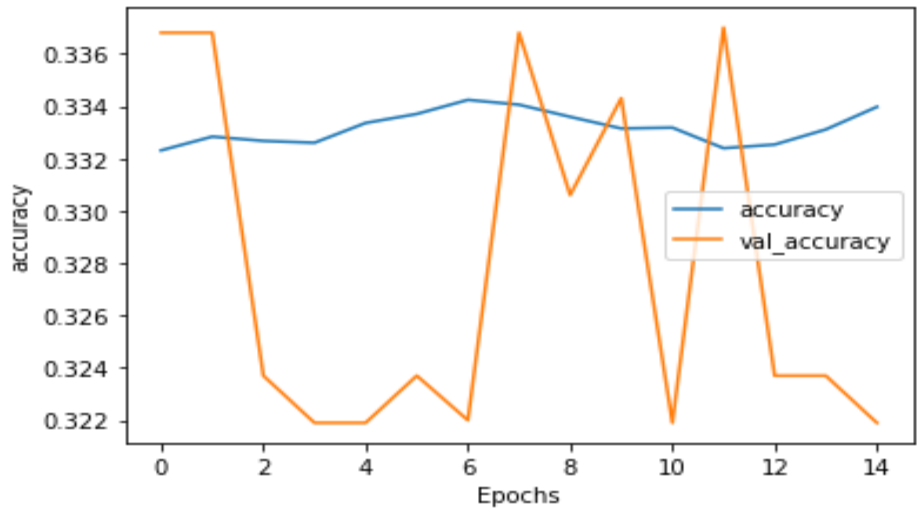
- Strip tags
- Removing punctuation
- Remove numeric
- Remove multiple white spaces
- Remove stop words
- Text stemming

Then, I converted words in a sentences to indices, where each word is indicated by a unique number. I did pad of 2 sentences to contain number of words in that of maximum word sentence.

Eg
sentence1: how are you
sentence1: how old are you
padded sentence1: 20 22 34 0 0 0
padded sentence2: 20 21 22 34 0 0
Each sentence has 6 words

Then two sentences are combined them to a single feature.
Eg.
20 22 34 0 0 0 20 21 22 34 0 0

I had done padding for train and test set in same manner. This feature is given to rnn. The target, which is in golden_label is also converted to numeric format. The code is in rnn1.ipnyb notebook



Accuracy for rnn model: 33%

Challenge:

I think accuracy increases if we train for a greater number of epochs. However, it is challenging to train for large number of epochs in our laptop.