

# 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 gensim 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\_regression.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. Then, I did padding of data 2 sentences and combined them to a single text feature. The text feature is converted to numpy array and fed to rnn.

The code is in rnn1 notebook