**Questions Predictor**

**Overview:** The objective of this document is to give a brief overview of underlying task, algorithm and tools chosen and details about the training and test data sets. For a more detailed explanation of tasks and algorithms please refer to the notebook ( .ipynb ) files. All the code in this project is developed with an objective to develop a better classifier, but not with an objective to develop most resource efficient code.

**Objective:** Develop a Deep Learning Classifier such that, given a sentence predict whether that sentence is a question or not.

**Tools Used:**

* Programming Language: Python 3.6.7
* Algorithm: Stacked / Multiple LSTM Cells - Recurrent Neural Network (to train on vectors),

Document2Vector (to create vectors),

TSNE (for visualization) and PCA (for dimensionality reduction).

* IDE: Jupyter notebook, Keras frame work with tensor flow as back end.

**Files:**

* **Data:** The training data was chosen from multiple sources / contributors, the training data is a set of text sentences which is mix of Amazon, imdb reviews and a set of sentences which are question phrases taken from <http://cogcomp.org/Data/QA/QC/> , these files are stored in a nested sub folder “Questions” with in datasets\_train folder.

The training data / text sentences contains a mix of questions and non-questions. There are about 12,600 training samples with 8000 samples for questions and 4000 odd for non-questions and 24,295 testing samples.

All the documents which are used to train and test this algorithm can be found inside the “datasets\_train” and “datasets\_test” sub-folder of this repository.

* **Code:** There are two separate code files in this application:

1. preprocess\_and\_vectorize: Used to collect data from different sources, clean them and then train a Doc2Vec model to create vector representation to each of these text documents. At various stages in this application, all the relevant files are saved (ex: a csv file with sentences and relevant labels for each sentence, a text file which contains all the vector representations of each sentence for later training etc) with in the same folder.
2. Train\_and\_Predict Sentences: Used to collect the vector representation of each document, and
3. Normalize all the samples, for better accuracy
4. visualize how samples are separated across different labels in higher dimensions
5. Reduce dimensions using PCA
6. Re-visualize reduced dimensions
7. Visualize any named entities in the training documents
8. Train an LSTM based network, and save the weights and other parameters
9. Follow steps 1 to 3 on test documents
10. Load the saved parameters from Step 6, and predict on the test documents
11. Save the predicted labels to a text file, separated by a new line

* **Other Files:**

1. QNQ.csv: This file contains two columns, ‘Text’ with a mix of training documents, and ‘label’ with 1 depicting the sentence as a question and 0 depicting not a question.
2. Vectors\_QNQ\_Data.txt: A text file containing vector representation of each text document. This file is used to train an LSTM network.
3. Ques\_Classify\_vectors.doc2vec: File containing required hyper parameters of the trained doc2Vec file.
4. Lstm\_model.json: File that stores parameters of a trained keras model.
5. Lstm\_model\_weights: Contains weights of the trained lstm model.
6. Predicted\_Labels\_QNQ.txt: Text file containing labels for each sentence in the test file separated by a new line.

**Approach:**

The text documents are pre-processed such that,

1. Sentence is split in to individual words,
2. Each word is lower cased and transformed in to its based form using lemmatization,
3. Any special characters, names of a person, dates and symbols (‘,’, ‘.’, ‘?’, ‘!’) from the text are removed.

Stop words such as is, a, the and names of the locations are preserved; because we felt removing those words will hamper the meanings of sentences which are already short which will eventually hamper the learning of the neural network.