EMPIRICAL TEST DESIGN STRATEGIES USING NATURAL LANGUAGE PROCESSING

T S Abishek 312216104001
Adithya Viswanathan 312216104002
Akash Kumar Pujari 312216104007
BE CSE, Semester 8

Dr. V S Felix Enigo, Kaushik Raghavan
Supervisor

Project Review: 1 (8 January 2020)
Department of Computer Science and Engineering
SSN College of Engineering

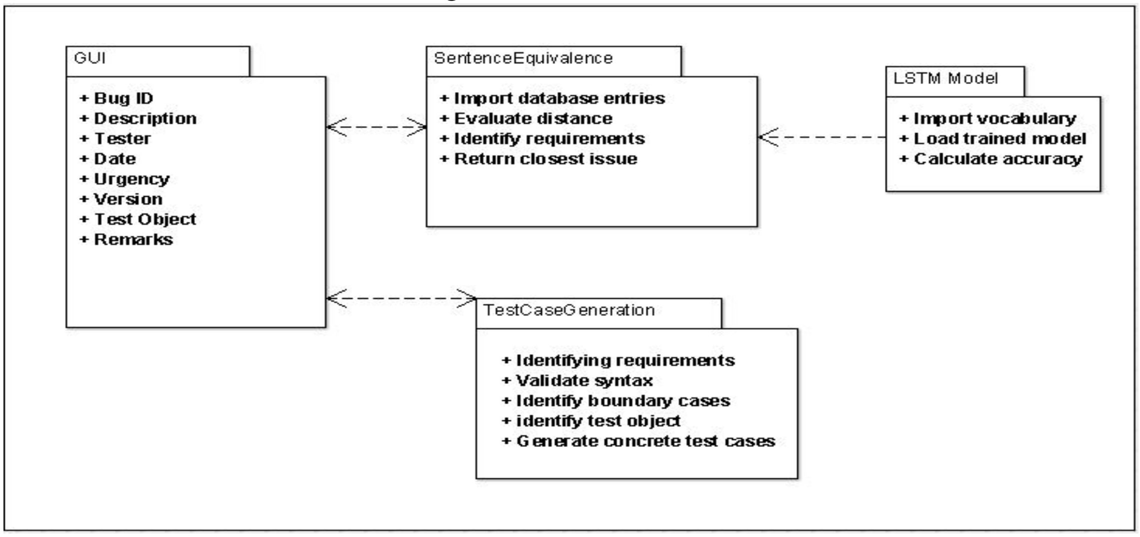
Abstract

Software testing is defined as an activity to check whether the actual results match the expected results and to ensure that the software system is defect free. It involves generating test cases from requirements and execution of a software component or system component to evaluate one or more properties of interest. But as the complexity of the product increases, the generation of test cases becomes tedious and time consuming. Our proposed system aims at automating the generation of test cases from requirements using Natural Language Processing and hence reducing the time consumed. Our system seeks to provide a comprehensive environment, to be accessed by both end users and developers. End users input the bugs/issues that they have identified, where the system prompts similar pre-existing issues and the requirement not satisfied as well. There exists a sentence similarity LSTM model at the core of the system, that identifies the relationship between the input and stored data. The data used to develop the model, was obtained from the Stanford NLP snli project. Thus the three tasks to be performed by the system is 1. Identify similar issues and intimate the user, 2. Identify the requirement not satisfied by identified issue, 3. Generate Test Case that must be run by developer before closing the issue.

1 Architectural Design

As the primary representation of the structural components of the system, the architectural design is described in the form of a package diagram.

Figure 1: Architecture



2 Data Flow Diagram and Use Case Diagram

To understand the complete working of the system, the Data-Flow diagram and the Use Case Diagram have been given.

Figure 2: DFD

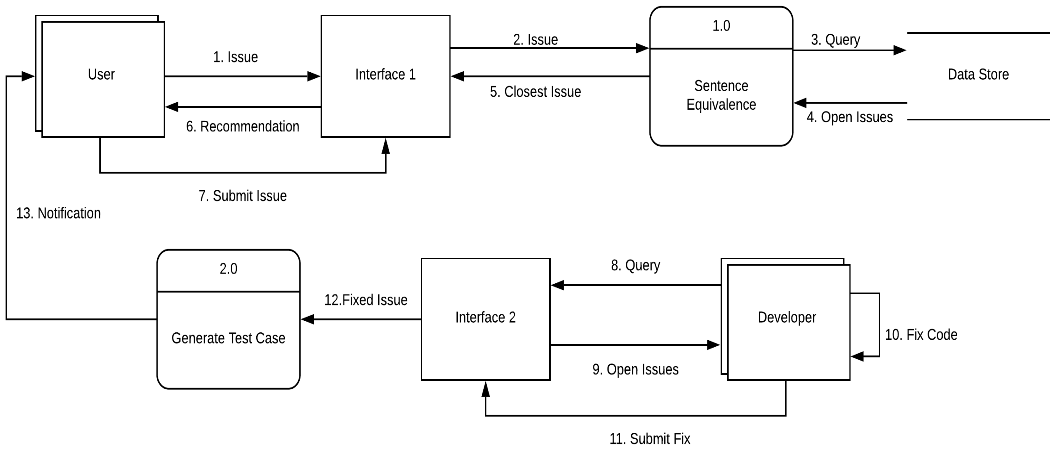
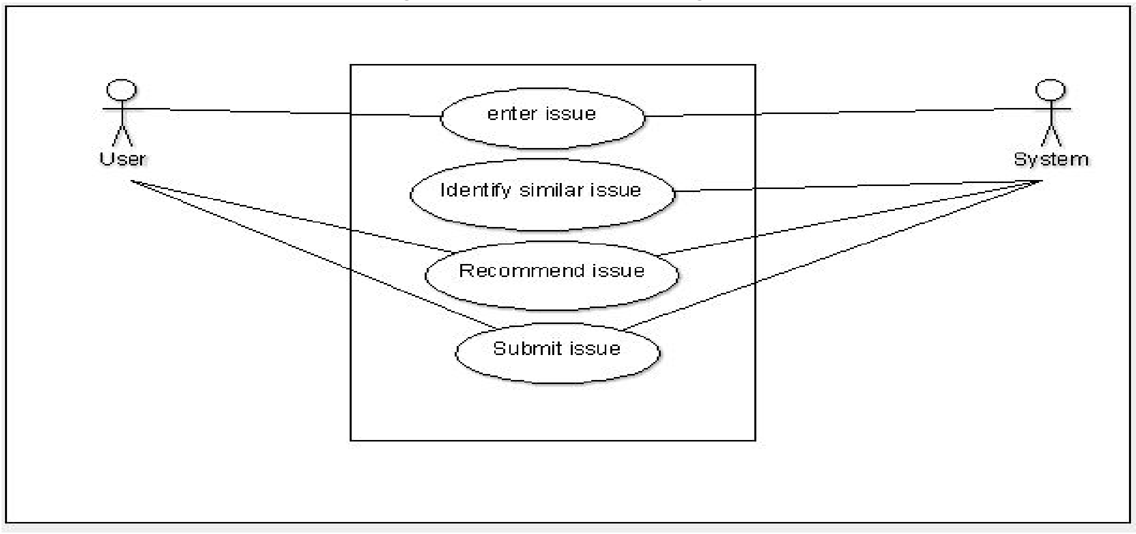


Figure 3: Use Case Diagram



3 Algorithm

3.1 Cosine Distance

The most fundamental method used in sentence similarity is the concept of Cosine Similarity. This method involves the prepossessing of input sentences and using these to obtain the 'distance' between them. The various operations that are employed in the pre-processing stage are similar to the work done by Per Runeson et al. [3]

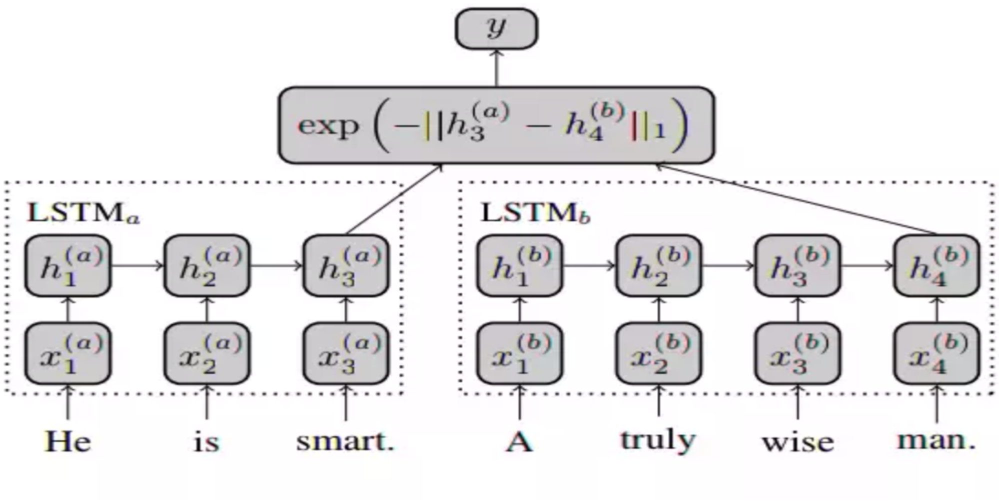
- Stemming - This involves the crude chopping off of prefixes and post-fixes of words to obtain the root word. This method does not guarantee that the correct root word is always obtained
  e.g. eating - eat
  e.g. Laziness - Lazi

- Lemmatization - This method performs the same function as that above, however, this takes into consideration the 'POS'(part of speech) to which the word belongs to, thus providing considerably better results than its counter part. Usually, if the part of speech of the word is not provided, the default is taken as noun.
  e.g better,none - better
  better,verb - good

- Case folding - It is the process of converting the case of all letters to either uppercase or lowercase so that the words 'Stop' and 'stop' do not appear as different words. However it is important to take into consideration words like 'Black' as they should not be converted to lower case if they represent a person's name, but should be converted to lower case if they represent the color

- Stop words - These are words which convey no real 'meaning' to the computer and can safely be removed to increase the accuracy of the system. Usually, python packages such as the 'Natural Language Toolkit' or nltk, contain a dictionary of stop words which can be removed from the input sentences.
- Punctuation - It is vitally important to remove the punctuation from sentences. However removal of punctuation may result in the loss of much needed information, as words like 'can't' will be converted to 'can t', thus completely changing the meaning of the sentence. Thus before removing punctuations, it is necessary to remove concatenations as well

Figure 4: Siamese LSTM



3.2 Siamese Sentence Similarity

This system uses a Siamese network to identify the similarity between sentences. A Siamese network is one which has two identical sub-networks in it. The model used in this project uses a ManhattanLSTM. To completely understand this system, it is important to be familiar with the concept of LSTM itself. LSTM, which represents Long Short Term Memory, is a type of Recurrent Neural Network, which has the capability to overcome the 'Long-Term Dependency' problem where there is a considerable gap between relevant terms. More on this can be found in the work done by Hochreiter and Schmidhuber[5].

This system converts variable length sentences to fixed size vectors, to be precise, vectors with 15 dimensions. The use of vector embeddings is to help the computer understand the 'meaning' of the words in some abstract sense. This also employs 'word embeddings' which are used to give the words semantic meanings in vector representations. The system is termed ManhattanLSTM, as it simply uses the Manhattan distance between the two LSTMs to measure the similarity between the sentences.

4 Expected Outcomes

1. Use NLP and tensorflow to compare the equivalence of two sentences, by generating vectors for both sentences and computing distance between them. In case the distance is smaller than a predefined threshold, then conclude that the sentences are similar.

2. Successfully compare the description of issue entered by user with the ones stored in the Django Framework database using the sentence equivalence algorithm. Notify the user in case of similarity of issues.

3. To generate test cases from functional requirements of Software Requirement Specification document based on keywords in context. Hence the end goal of the proposed system is to reduce effort and time consumed by software tester to test a product.

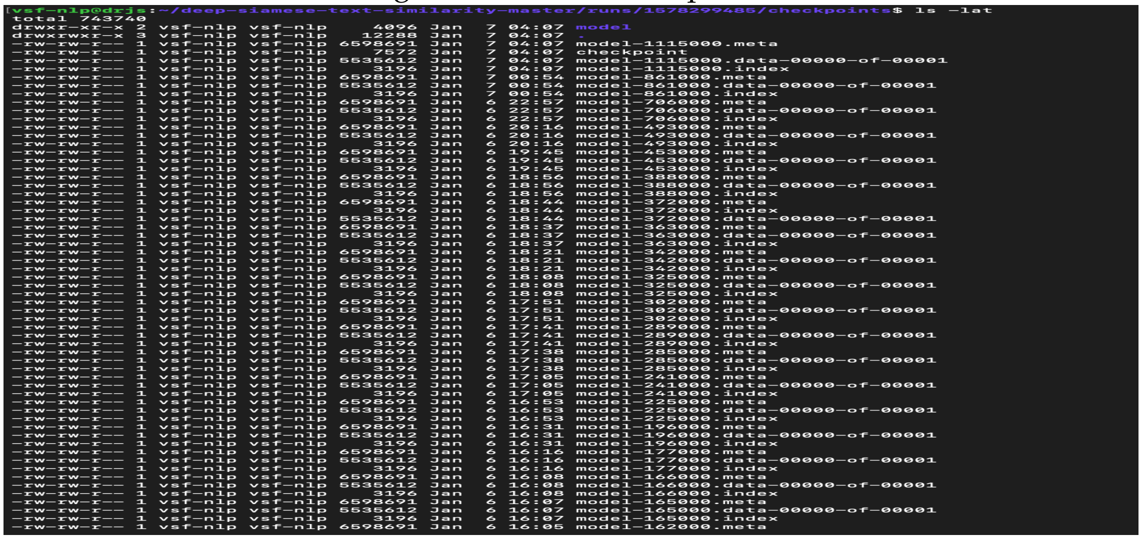
5 Phase 1 Workflow

There are two major programs that constitute the working of phase 1,

1. train.py - This contains the code for training the model and takes the file containing the training data as argument.

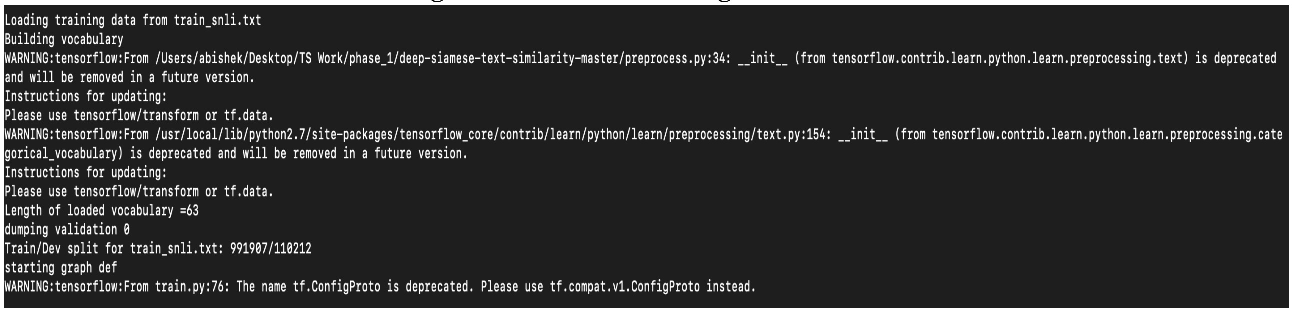
2. test.py - This contains the code for evaluating the trained model, and is the core of the project itself. It must be modified to accept the user input and use this to compare with the data already stored in the database. This takes the model checkpoint to be imported as argument.

Figure 5: Model Checkpoints



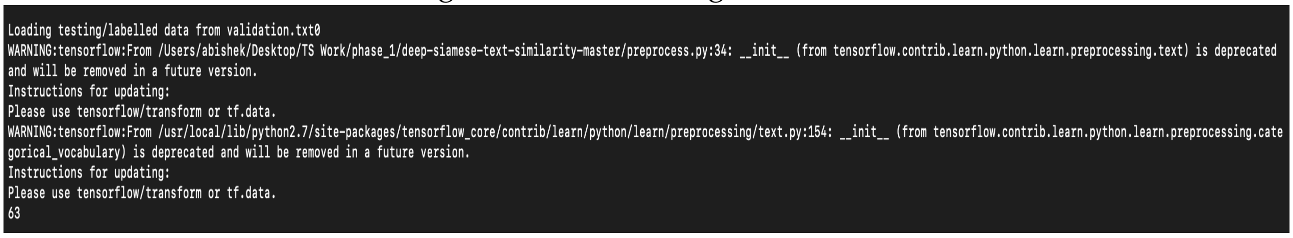
```
[vsf-nlp@drjs:~/deep-siamese-text-similarity-master/runs/1578299485/checkpoints$ ls -lat
 total 743740
 drwxr-xr-x 2 vsf-nlp vsf-nlp    4096 Jan  7 04:07 model
 drwxrwxr-x 3 vsf-nlp vsf-nlp   12288 Jan  7 04:07 .
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 6598691 Jan  7 04:07 model-1115000.meta
 -rw-rw-r-- 1 vsf-nlp vsf-nlp    7572 Jan  7 04:07 checkpoint
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 5535612 Jan  7 04:07 model-1115000.data-00000-of-00001
 -rw-rw-r-- 1 vsf-nlp vsf-nlp    3196 Jan  7 04:07 model-1115000.index
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 6598691 Jan  7 00:54 model-861000.meta
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 5535612 Jan  7 00:54 model-861000.data-00000-of-00001
 -rw-rw-r-- 1 vsf-nlp vsf-nlp    3196 Jan  7 00:54 model-861000.index
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 6598691 Jan  6 22:57 model-706000.meta
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 5535612 Jan  6 22:57 model-706000.data-00000-of-00001
 -rw-rw-r-- 1 vsf-nlp vsf-nlp    3196 Jan  6 22:57 model-706000.index
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 6598691 Jan  6 20:16 model-493000.meta
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 5535612 Jan  6 20:16 model-493000.data-00000-of-00001
 -rw-rw-r-- 1 vsf-nlp vsf-nlp    3196 Jan  6 20:16 model-493000.index
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 6598691 Jan  6 19:45 model-453000.meta
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 5535612 Jan  6 19:45 model-453000.data-00000-of-00001
 -rw-rw-r-- 1 vsf-nlp vsf-nlp    3196 Jan  6 19:45 model-453000.index
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 6598691 Jan  6 18:56 model-388000.meta
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 5535612 Jan  6 18:56 model-388000.data-00000-of-00001
 -rw-rw-r-- 1 vsf-nlp vsf-nlp    3196 Jan  6 18:56 model-388000.index
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 6598691 Jan  6 18:44 model-372000.meta
 -rw-rw-r-- 1 vsf-nlp vsf-nlp    3196 Jan  6 18:44 model-372000.index
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 5535612 Jan  6 18:44 model-372000.data-00000-of-00001
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 6598691 Jan  6 18:37 model-363000.meta
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 5535612 Jan  6 18:37 model-363000.data-00000-of-00001
 -rw-rw-r-- 1 vsf-nlp vsf-nlp    3196 Jan  6 18:37 model-363000.index
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 6598691 Jan  6 18:21 model-342000.meta
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 5535612 Jan  6 18:21 model-342000.data-00000-of-00001
 -rw-rw-r-- 1 vsf-nlp vsf-nlp    3196 Jan  6 18:21 model-342000.index
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 6598691 Jan  6 18:08 model-325000.meta
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 5535612 Jan  6 18:08 model-325000.data-00000-of-00001
 -rw-rw-r-- 1 vsf-nlp vsf-nlp    3196 Jan  6 18:08 model-325000.index
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 6598691 Jan  6 17:51 model-302000.meta
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 5535612 Jan  6 17:51 model-302000.data-00000-of-00001
 -rw-rw-r-- 1 vsf-nlp vsf-nlp    3196 Jan  6 17:51 model-302000.index
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 6598691 Jan  6 17:41 model-289000.meta
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 5535612 Jan  6 17:41 model-289000.data-00000-of-00001
 -rw-rw-r-- 1 vsf-nlp vsf-nlp    3196 Jan  6 17:41 model-289000.index
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 6598691 Jan  6 17:38 model-285000.meta
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 5535612 Jan  6 17:38 model-285000.data-00000-of-00001
 -rw-rw-r-- 1 vsf-nlp vsf-nlp    3196 Jan  6 17:38 model-285000.index
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 6598691 Jan  6 17:05 model-241000.meta
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 5535612 Jan  6 17:05 model-241000.data-00000-of-00001
 -rw-rw-r-- 1 vsf-nlp vsf-nlp    3196 Jan  6 17:05 model-241000.index
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 6598691 Jan  6 16:53 model-225000.meta
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 5535612 Jan  6 16:53 model-225000.data-00000-of-00001
 -rw-rw-r-- 1 vsf-nlp vsf-nlp    3196 Jan  6 16:53 model-225000.index
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 6598691 Jan  6 16:31 model-196000.meta
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 5535612 Jan  6 16:31 model-196000.data-00000-of-00001
 -rw-rw-r-- 1 vsf-nlp vsf-nlp    3196 Jan  6 16:31 model-196000.index
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 6598691 Jan  6 16:16 model-177000.meta
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 5535612 Jan  6 16:16 model-177000.data-00000-of-00001
 -rw-rw-r-- 1 vsf-nlp vsf-nlp    3196 Jan  6 16:16 model-177000.index
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 6598691 Jan  6 16:08 model-166000.meta
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 5535612 Jan  6 16:08 model-166000.data-00000-of-00001
 -rw-rw-r-- 1 vsf-nlp vsf-nlp    3196 Jan  6 16:08 model-166000.index
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 6598691 Jan  6 16:07 model-165000.meta
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 5535612 Jan  6 16:07 model-165000.data-00000-of-00001
 -rw-rw-r-- 1 vsf-nlp vsf-nlp    3196 Jan  6 16:07 model-165000.index
 -rw-rw-r-- 1 vsf-nlp vsf-nlp 6598691 Jan  6 16:05 model-162000.meta
```

Figure 6: Load Training Data



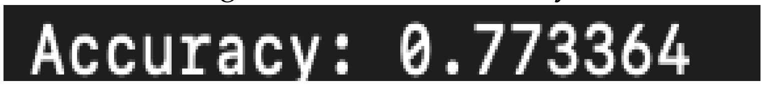
```
Loading training data from train_snli.txt
Building vocabulary
WARNING:tensorflow:From /Users/abishek/Desktop/TS Work/phase_1/deep-siamese-text-similarity-master/preprocess.py:34: __init__ (from tensorflow.contrib.learn.python.learn.preprocessing.text) is deprecated
and will be removed in a future version.
Instructions for updating:
Please use tensorflow/transform or tf.data.
WARNING:tensorflow:From /usr/local/lib/python2.7/site-packages/tensorflow_core/contrib/learn/python/learn/preprocessing/text.py:154: __init__ (from tensorflow.contrib.learn.python.learn.preprocessing.cate
gorical_vocabulary) is deprecated and will be removed in a future version.
Instructions for updating:
Please use tensorflow/transform or tf.data.
Length of loaded vocabulary =63
dumping validation 0
Train/Dev split for train_snli.txt: 991907/110212
starting graph def
WARNING:tensorflow:From train.py:76: The name tf.ConfigProto is deprecated. Please use tf.compat.v1.ConfigProto instead.
```

Figure 7: Load Testing Data



```
Loading testing/labelled data from validation.txt0
WARNING:tensorflow:From /Users/abishek/Desktop/TS Work/phase_1/deep-siamese-text-similarity-master/preprocess.py:34: __init__ (from tensorflow.contrib.learn.python.learn.preprocessing.text) is deprecated
and will be removed in a future version.
Instructions for updating:
Please use tensorflow/transform or tf.data.
WARNING:tensorflow:From /usr/local/lib/python2.7/site-packages/tensorflow_core/contrib/learn/python/learn/preprocessing/text.py:154: __init__ (from tensorflow.contrib.learn.python.learn.preprocessing.cate
gorical_vocabulary) is deprecated and will be removed in a future version.
Instructions for updating:
Please use tensorflow/transform or tf.data.
63
```

Figure 8: Overall Accuracy



```
Accuracy: 0.773364
```

6 Literature Survey and Future Work

The work done by Sepp Hochreiter and Jürgen Schmidhuber[5], provide enhancements to standard RNNs, and help overcome the 'Long-Term dependency' problem. This core concept has been used in the work done by Jonas Mueller and Aditya Thyagarajan [7], which uses 2 LSTM networks to create a 'Siamese Network' and thus identify similarity between sentences. This architecture has been imported to create a ManhattanLSTM model in the experimental project present in, https:/github.com/dhwajraj/deep-siamese-text-similarity. This claims an accuracy of 81 percent. The work done by Per Runeson et al. [3] contains the various pre-processing methods mentioned in section 3.1. Thus, there is a high chance of improvement in accuracy if these are used in the training set as well as the input given by the user, as they highlight the root words in the sentences.

References

[1] Bowman, Samuel R. and Angeli, Gabor and Potts, Christopher, and Manning, Christopher D., *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, A large annotated corpus for learning natural language inference, 2015

[2] *Proceedings of the 1st Workshop on Representation Learning for NLP*, Blunsom, Phil and Cho, Kyunghyun and Cohen, Shay and Grefenstette, Edward and Hermann, Karl Moritz and Rimell, Laura and Weston, Jason and Yih, Scott Wentau aug, 2016, Berlin, Germany, Association for Computational Linguistics, https://www.aclweb.org/anthology/W16-1600,

[3] *Detection of Duplicate Defect Reports Using Natural Language Processing*, Per Runeson ; Magnus Alexandersson ; Oskar Nyholm, Minneapolis, MN, USA, 04 June 2007

[4] *Natural Language Processing (NLP) Applied on Issue Trackers*, Mathias Ellmann, NL4SE 2018: Proceedings of the 4th ACM SIGSOFT International Workshop on NLP for Software Engineering, November 2018, Pages 38–41

[5] *Long Short-Term Memory*,Sepp Hochreiter, Jürgen Schmidhuber, Neural Computation, November 1997

[6] David Croft ; Simon Coupland ; Jethro Shell ; Stephen Brown *A fast and efficient semantic short text similarity metric*, 2013 13th UK Workshop on Computational Intelligence (UKCI)

[7] Jonas Mueller and Aditya Thyagarajan, *Siamese Recurrent Architectures for Learning Sentence Similarity*, Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, 2016