NAME : DEEPTHI N RAO & SRAVANI KAMISETTY
SID : 104406331, 304414410
COURSE : CS 263A - LANGUAGE AND THOUGHT
TOPIC : CREEPY TWIN, WORD PREDICTION SYSTEM

PROJECT REPORT

OVERALL TASK DESCRIPTION

- Word prediction is an important NLP problem in which we want to predict the correct word in a given context.
- Word completion utilities, predictive text entry systems, writing aids, and language translation are some of common word prediction applications.
- The goal of the project is to build q utility which can read the user's mind and predict the next word in a partially typed sentence.
- The options for the next word will be displayed on a pop-up section near the text area, next to the cursor.
- Input : Any partially constructed sentence given by the user.
- Output : List of words that could be the next word in the sentence in the given context.

MOTIVATION

- Increases the quality of spelling and grammar.
- Improves flow of thought.
- Reduces the number of key strokes required to type the document.
- Helps people with not so rich vocabulary.

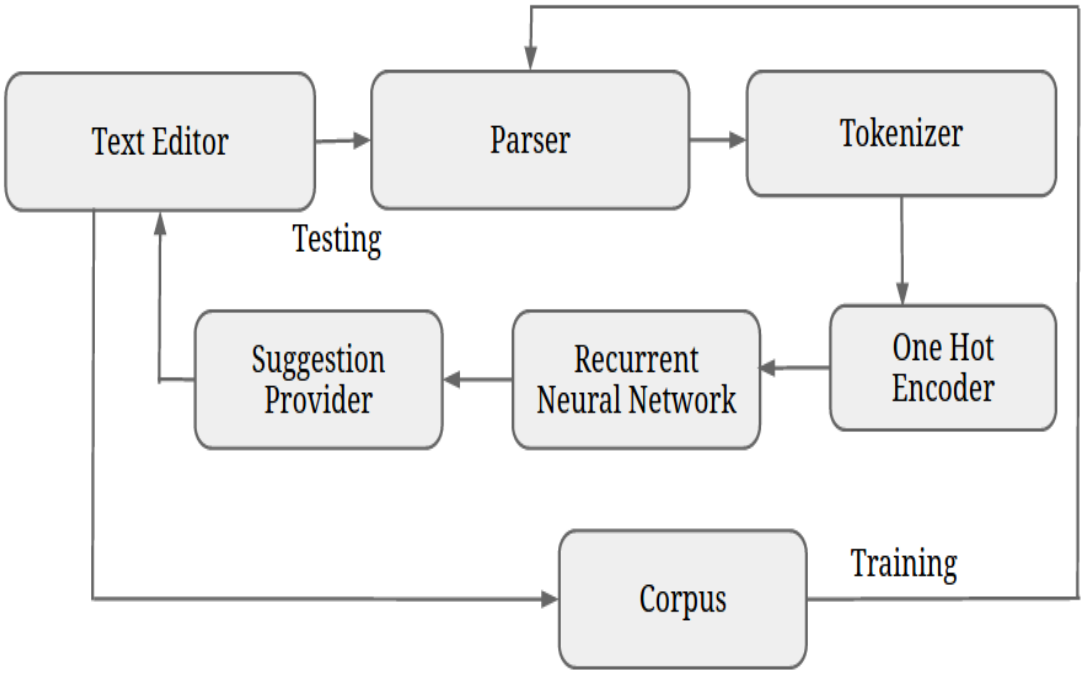
EMPHASIS AND METHODOLOGY

- A recurrent neural network based model is used.
- The other options for this word prediction model is n-gram model
- In a simple N-gram model a count of number of times a word appears in a context is used to calculate probability and make predictions.
- N-grams however have problems with representing patterns over more than a few words.
- Also with increasing order of n-gram model the number of parameters exponentially increase exponentially. This limits the model.
- Hence in this project we use recurrent neural network model over n-gram model as it does not face such extreme limitations.

- Even the recurrent neural network model becomes very slow converge with a very big vocabulary. Hence the vocabulary in this project is limited to 10000 words.

ARCHITECTURE & MODULES

- The utility has 7 important parts.
- Text editor is the front end part of the utility where the text will typed by the user. It will have a drop down of suggestions for the next word
- Parser and Tenderizer is used to divide the text into words which are later fed into the recurrent neural network.
- Before the data is passed into the recurrent neural network, one hot encoding is performed on it.
  (Refer the section Knowledge and Representation for more details).
- A recurrent neural network is used for learning the context.
- The neural network has 3 layers- input, hidden and output.
- Input layer is linear, hidden layer is Sigmoid layer, output layer is soft-max layer.
- The number of neurons in the input and output layer is equal to size of the vocabulary.
- The number of neurons in the hidden layer is comparatively less than the other layers ~ 100 neurons.
- The suggestions refer to the words that are suggested as next words to the user.
- The corpus is a collection of documents. It is elaborated in a one of the sections below.



FIXED vs LEARNED COMPONENTS

- The learning in the word prediction utility happens both during training and testing.
- During training recurrent neural network is trained by a supervised dataset constructed as follow:

  Eg: If the document used for training contains a sentence like: 'After one year's work experience '

Tuple	One hot encoding
{'after','one'}	{10000,01000}
{'one','year'}	{01000,00100}

- During the training phase, the recurrent neural network is trained by a supervised dataset that is a collection of bi-grams of consecutive words in the each sentence.
- During the testing phase, recurrent neural network is used to suggest next words in the text editor. At the end of user's session a snapshot of the text entered by the user is captured and this data is used to train the recurrent neural network further.
- Rest of the components — text editor, parser, tokenizer, suggestion provider, one-hot encoder are fixed components in the utility

KNOWLEDGE AND REPRESENTATIONS

- The corpus is broken down into a ordered list of consecutive words.
- The ordered list is traversed and bi-grams are picked from the list.

  Eg: If the document used for training contains a sentence like: 'After one year's work experience'

  Ordered list: {'after','one','year','work','experience'}
  Bi-grams: {{'after','one'},{'one','year'},{'year','work'}, {'work','experience'}}

- The first value in every supervised dataset tuple corresponds to input word to be given to the neural network

- The second value in every supervised dataset tuple corresponds to the output word expected from the neural network at the end of that iteration.
- Each of the bi-grams are later one hot encoded before feeding them into the recurrent neural network.
- One hot encoding is a simple way of encoding discrete concepts such as words.

  Ex: If the document used for training contains a sentence like: 'After one year's work experience'

  The size of the vocabulary is 5. Then each of the words in the vocabulary can are encoded as:

  after – 10000
  one – 01000
  year – 00100
  work – 00010
  experience – 00001

- Using the one hot encoded words the supervised dataset is constructed.

CORPORA

- Collection of Statement of Purpose documents written by students while applying to universities for both undergraduate and graduate positions.
- Students who have written these Statement of purpose documents belong to different origins.
- Students who have written these Statement of purpose documents are applying ti different majors
- Also these students have applied to different universities in different countries.
- The size of corpora used for training is about 10000 words.

EVALUATION & INSTRUMENTATION

- The utility is evaluated by the use of front-end text editor.
- Text from a already written Statement of purpose is typed in the text editor.
- A hit occurs when the next word suggestion in the utility matches the next word in the document used for testing. Else it is counted as a miss.
- The evaluation was done with 3 types of testing data- in-domain, out of domain, mixed domain.

- In in-domain both training and testing data overlap i.e part of training data is used as testing.
- Out of domain testing involves different data for training and testing.
- Mixed domain testing is a combination of in-domain and out of domain. In this testing part of training data and a part of completely independent set of data is used for testing.
- Keystroke savings is another metric used during evaluation. It is defined as the amount of savings made in terms of key board strokes when a suggested word in chosen instead of typing the word.
- Keystroke savings percentage can be calculated as :
  (total characters in the typed document – keystrokes)/ total characters in the typed document.

PROGRAMMING LANGUAGES

- The entire utility is coded in Python 2.7- https://www.python.org/download/releases/2.7/
- Python library The Natural language toolkit (NLTK)- http://www.nltk.org/ is used to implement the parser and tokenizer. It is a suite of text processing libraries.
- Python library PyBrain- http://pybrain.org/ is used to implement the recurrent neural network and the supervised data set. PyBrain is a modualar Machine learning library for python that offers easy to use algorithms for machine learning tasks.
- Python library Tkinter- https://wiki.python.org/moin/TkInter is used to make the front-end text editor. Tkinter is a Graphical user interface package for Python
- Python library NumPy- http://www.numpy.org/ is used for matrix and array manipulation. NumPy is a fundamental package for scientific computing in python.

PHASES

- The project was divided into 2 phases.

PHASE 1

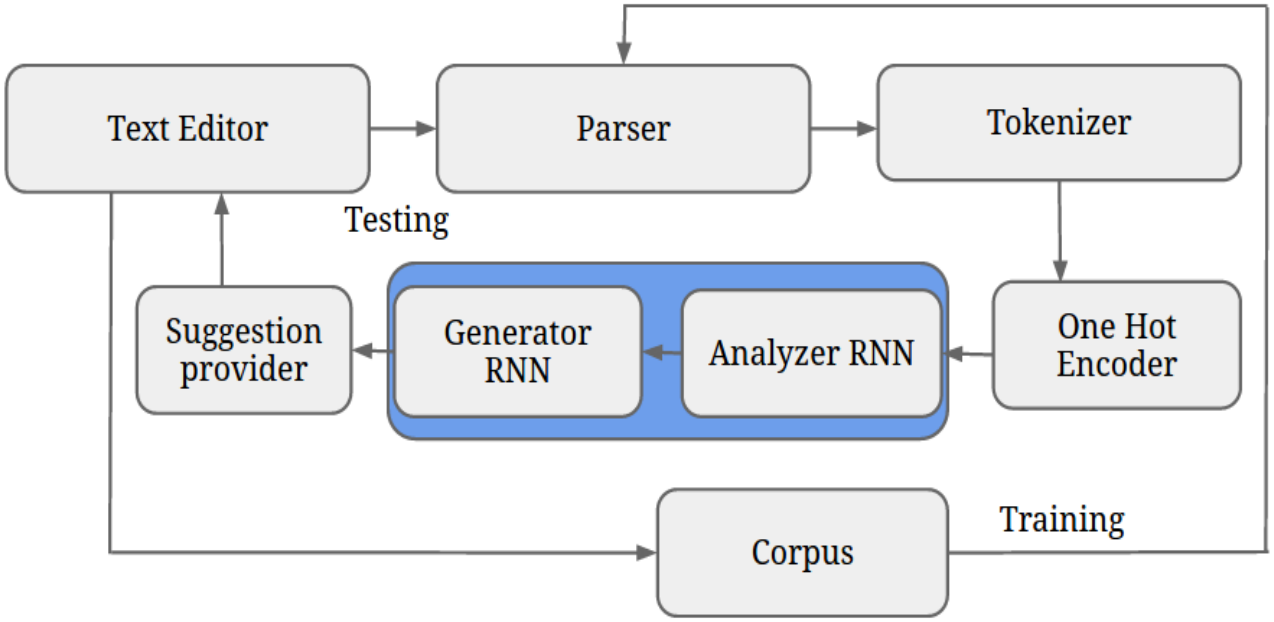
- The first phase involved constructing and training the recurrent neural network with the Statement of purpose documents.
- The training phase was followed by building of the front end text editor and connecting the neural network with it to provide suggestions.

PHASE 2

- In the second phase the fixed parts remain the same as the first

phase. However the learned parts I.e the recurrent neural network was split into 2 recurrent neural networks.

- The two recurrent neural networks are the analyzer RNN and generator RNN.
- The analyzer recurrent neural network takes a sequence of words and is trained to form a vector that represents the conceptual context of the sentence.
- The generator recurrent neural network takes a semantic vector (the output of the analyzer RNN) along with the current word in the text editor as input.
- The generator RNN then generates a set of words that act as the suggestions for the next word.



RESULTS

EXPERIMENT 1

- The results of first phase which consists of only one recurrent neural network is as follows:

Domain Type	**Percentage of training to testing**	**Accuracy**	**Keystroke savings**
In-Domain	100 : 20	49.67%	59.82%
Out of domain	80 : 20	47.30%	44.43%

Mixed domain	90 : 20	48.28%	52.78%

- The second which is an improvement of the first phase involves splitting of the recurrent neural network into 2 parts – analyzer RNN and generator RNN.
- The generator recurrent neural network in the second phase not just considers the previous word for training but the context of the entire sentence. Hence prediction accuracy becomes considerably better in the second phase.
- The results of the second phase are as follows:

Domain Type	**Percentage of training to testing**	**Accuracy**	**Keystroke savings**
In-Domain	100 : 20	53.98%	60.29%
Out of domain	80 : 20	49.98%	47.65%
Mixed domain	90 : 20	53.01%	56.22%

EXPERIMENT 2

- The only difference between experiment 1 and 2 is the percentage of data used for training and that for testing.
- The results below clearly show that the higher percentage of training data results in better predictions and higher accuracy.

PHASE 1

Domain Type	**Percentage of training to testing**	**Accuracy**	**Keystroke savings**
In-Domain	100 : 40	44.53%	49.99%
Out of domain	60 : 40	44.09%	42.43%
Mixed domain	80 : 40	46.28%	47.77%

PHASE 2

Domain Type	**Percentage of training to testing**	**Accuracy**	**Keystroke savings**
In-Domain	100 : 40	48.37%	54.98%
Out of domain	60 : 40	46.87%	46.66%

Mixed domain	80 : 40	50.89%	53.64%

REFERENCES

- T. Mikolov, M. Karafiat, L.vBurget, J.vCernocky, S. Khudanpur. Recurrent neural network ´ based language model, In: Proceedings of Interspeech, 2010. http://www.fit.vutbr.cz/research/groups/speech/publi/2010/mikolov_interspeech2010_IS100722.pdf

- T. Mikolov, S. Kombrink, L. Burget, J. Cernocky, S. Khudanpur. Extensions of recurrent neural network language model, In: Proceedings of ICASSP 2011. http://www.fit.vutbr.cz/research/groups/speech/publi/2011/mikolov_icassp2011_presentation_rnnlm-extension.pdf

- T. Mikolov, A. Deoras, S. Kombrink, L. Burget, J. Cernocky. Empirical Evaluation and Combination of Advanced Language Modeling Techniques, In: Proceedings of Interspeech, 2011. http://www.fit.vutbr.cz/~imikolov/rnnlm/is2011_emp.pdf

- Ghayoomi, Masood and S. M. Assi (2005) "Word prediction in a running text: A statistical language modeling for the Persian language" In Proceeding of the Australasian Language Technology Workshop, University of Sydney, Australia. http://www.aclweb.org/anthology/U05-1010

- Mikael Boden. A Guide to Recurrent Neural Networks and Back-propagation. In the Dallas project, 2002. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.3.9311

DIVISION OF WORK

Item	**Name**	**Phase**
Recurrent Neural network	Sravani Kamisetty	1
Parser and Tokenizer	Deepthi N Rao	1
Text Editor	Both	1
Suggestion Provider	Sravani Kamisetty	1
One Hot encoder	Deepthi N Rao	1
Supervised dataset construction	Deepthi N Rao	1
Generator RNN	Deepthi N Rao	2

Analyzer RNN	Sravani Kamisetty	2
Supervised dataset construction 2	Sravani Kamisetty	2
Project Presentation	Both	2
Project Report	Sravani Kamisetty	2
Appendix	Deepthi N Rao	2

CURRENT STATUS

- Both the phases 1 and 2 are complete and in full working condition.
- Both the phases are evaluated using multiple domain data as explained in the evaluation section.
- For a full demo please view the vedio in the next section.

VIDEO

- Please find the complete video at the location: https://drive.google.com/file/d/0BxqIUDEfQIsnUktvXzY2TDh6Nms/view?usp=sharing