COEN 296 Winter 2018

Term Project

Sentence Completion Using Text Prediction

Submitted By :

Team 6

Team Members:

Aparna Gangwar,

Pinak Ghate,

Vaibhaw Raj

Instructor :

Prof. Ming Hwa Wang

Santa Clara University

PREFACE

The report has been made in fulfillment of the requirement for the subject: Natural Language Processing in March 2018 under the supervision of Prof Dr. Ming-Hwa Wang. For making this project we have studied various concepts related to the semantic analysis and word sense. We also studied about various statistical language processing algorithms and methodology that can be used to solve the problem. The project aims at applying N-Gram linguistic model, Latent Semantic Analysis methodology and Pointwise Mutual Information methodology in order to answer SAT type single word sentence completion challenge posted by Microsoft. The end goal is to evaluate narrow and general purpose language processing capabilities which has been wildly discussed in the research community.

ACKNOWLEDGEMENTS

The success of any project depends on contribution of team and guidance of others. We take this as an opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project, especially Dr. Ming-Hwa Wang, for his guidance throughout the term.

We would also like to thank and acknowledge the researchers, as their research provided us the basis to study the problem and derive consequential solutions to it.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

1.	GTD	Good Turing Discounting
2.	LSA/LSI	Latent Semantic Analysis
3.	MSR	Microsoft Research
4.	PMI	Pointwise Mutual Information
5.	PPMI	Positive PMI

LIST OF TABLES

ABSTRACT

The project aims to study, analyze and perform sentence completion task based on SAT style questions. In this project, we plan to apply and compare various approaches for automated sentence completion which include n-gram modelling, Latent Semantic Indexing and PointWise Mutual Information(PMI). Results from the past research demonstrate that the PMI model outperforms the conventional n-gram model models on the Microsoft Research Sentence Completion Challenge. The LSA model outperforms other non-neural network models. Hence, we will compare the results of the PMI model with LSA model to decide which of them performs better considering the scale of the data.

CHAPTER 1

INTRODUCTION

1.1 Objective

The objective of this project was to be able to apply techniques and methods learned in Natural Language Processing course to a rather famous real-world problem, the task of sentence completion using text prediction. The project aims at implementing and analyzing techniques like n-grams, Latent Semantic Analysis and Pointwise Mutual Information and compare their accuracy and efficiency on the datasets provided by Microsoft Research Group.

1.2 Overview of the Problem

Text prediction algorithms for automatic or semi-automatic sentence completion have been vastly discussed in literature. They are widely used to enhance the speed of communication as well as reducing the total time taken to compose text. They have applications in search engines and mobile devices so as to reduce the efforts of the end user.

One of the oldest software applications was Profet in 1987. Profet allowed to predict a partially typed word by means of a unigram frequency list: the nine best matches were shown to the user. It also suggested the next word by using a bigram frequency list; however, upon partially typing of the next word, Profet reverted to unigrams-based suggestions.
Word prediction generally relies on n-grams occurrence statistics, which may have huge data storage requirements and does not take into account the general meaning of the text.

1.3 Relation to the Class

The main purpose behind picking up this problem is that it allows us to work on and implement a big array of topics we've learnt as a part of the curriculum for COEN 296. All the approaches that we have used we're based on the material we have read and understood in the class and we wanted to be able to visualize the practical applications and implications of them.

1.4 Problem statement

This problem statement is a part of the Microsoft Sentence Completion Challenge and we plan to use the dataset provided by the challenge. The task is to complete the sentence from the given 5 options. The dataset is a part of Project Gutenberg, which contains around 50000 free ebooks. Challenge of our problem is to achieve Word prediction for sentence completion. For which we targeted Microsoft Research Sentence Completion Challenge dataset which consist of 1040 questions. Each question consists in a sentence having a missing word and five possible words as valid answers. This challenge includes a training dataset composed by 522 books from Project Gutenberg.

1.5 Hypothesis

Initially, we plan to predict the words in the sentences using n-gram models. To improve on the n-gram model, we will implement smoothing techniques like Good-Turing Smoothing. Then we propose to implement an alternative methodology, based on Latent Semantic Analysis, to address the problem of text completion.

LSA/LSI improves the classic n-grams based approach by considering also terms distant from the word to predict. We also plan to apply Pointwise Mutual Information(PMI) to measure the degree of association between answer options and other sentence tokens for improving the sentence completion challenge by Microsoft.

1.7 Area of Investigation

The project on a broader dimension is based on core application of Natural Language Processing with the focus on the sub-domain of semantic analysis.

CHAPTER 2

THEORETICAL BASIS AND LITERATURE REVIEW

2.1 Definition of problem:

Our project focuses on the problem of sentence level semantic coherence by answering SAT style sentence completion questions. We tackle the problem with three approaches: methods that use local lexical information, such as the n-grams of a classical language model; methods that evaluate global coherence, such as latent semantic analysis and methods that fuses various types of input provided to other classes of language models based on point wise mutual information. We evaluate these methods on a suite of practice SAT questions, and on a recently released sentence completion task based on data taken from five Conan Doyle novels.

2.2 Theoretical background of the problem:

In recent years, standardized examinations have proved a fertile source of evaluation data for language processing tasks. They are valuable for many reasons: they represent facets of language understanding recognized as important by educational experts; they are organized in various formats designed to evaluate specific capabilities; they are yardsticks by which society measures educational progress; and they affect a large number of people.

Narrow and general language processing capabilities has been wildely discussed in previous researches. Among the narrower tasks, the identification of synonyms and antonyms has been studied by (Landauer and Dumais, 1997; Mohammed et al., 2008; Mohammed et al., 2011; Turney et al., 2003; Turney, 2008), who used questions from the Test of English as a Foreign Language (TOEFL), Graduate Record Exams (GRE) and English as a Second Language (ESL) exams. Tasks requiring broader competencies include logic puzzles and reading comprehension. Logic puzzles drawn from the Law School Administration Test (LSAT) and the GRE were studied in (Lev et al., 2004), which combined an extensive array of techniques to solve the problems. The DeepRead system (Hirschman et al., 1999) initiated a long line of research into reading comprehension based on test prep material (Charniak et al., 2000; Riloff and Thelen, 2000).

In this project, we study a new class of problems intermediate in difficulty between the extremes of synonym detection and general question answering - the sentence completion questions found on the Scholastic Aptitude Test (SAT). These questions present a sentence with one or two blanks that need to be filled in. Five possible words (or short phrases) are given as options for each blank. All possible answers except one result in a nonsense sentence.

The questions are highly constrained in the sense that all the information necessary is present in the sentence itself without any other context. Nevertheless, they vary widely in difficulty. The first of these examples is relatively simple: the second half of the sentence is a clear description of the type of behavior characterized by the desired adjective. The second example is more sophisticated; one must infer from the contrast between medicine and poison that the correct answer involves a contrast, either useless vs. effective or curative vs. toxic. Moreover, the first, incorrect, possibility is perfectly acceptable in the context of the second clause alone; only irrelevance to the contrast between medicine and poison eliminates it. In general, the questions require a combination of semantic and world knowledge as well as occasional logical reasoning. We study the sentence completion task because we believe it is complex enough to pose a significant challenge, yet structured enough that progress may be possible.

2.3 Related research to solve the problem:

The past work which is most similar to ours is derived from the lexical substitution track of SemEval 2007 (McCarthy and Navigli, 2007). In this task, the challenge is to find a replacement for a word or phrase removed from a sentence. In contrast to our SAT-inspired task, the original answer is indicated. For example, one might be asked to find alternates for match in “After the match, replace any remaining fluid deficit to prevent problems of chronic dehydration throughout the tournament.” Two consistently high-performing systems for this task are the KU (Yuret, 2007) and UNT (Hassan et al., 2007) systems. These operate in two phases: first they find a set of potential replacement words, and then they rank them. The KU system uses just an N-gram language model to do this ranking. The UNT system uses a large variety of information sources, and a language model score receives the highest weight.

2.4 Advantage/Disadvantage of those research:

N-gram statistics were also very effective in (Giuliano et al., 2007). That paper also explores the use of Latent Semantic Analysis to measure the degree of similarity between a potential replacement and its context, but the results are poorer than others. Since the original word provides a strong hint as to the possible meanings of the replacements, we hypothesize that N-gram statistics are largely able to resolve the remaining ambiguities. The SAT sentence completion sentences do not have this property and thus are more challenging.

2.5 Our Solution to this Problem:

This project proposes to analyze LSA (Mikolov, 2013) and Pointwise Mutual Inclusion (Church and Hanks, 1990; Mnih and Kavukcuoglu. 2013) methods which has been explored by researchers in similar context on different problem of linguistic models. We will be measuring accuracy of these methods on the N-gram statistics discussed in previous sections.

We propose an alternative methodology, based on Latent Semantic Analysis, to address these issues. An asymmetric Word-Word frequency matrix is employed to achieve higher scalability with large training datasets than the classic Word-Document approach. We propose a function for scoring candidate terms for the missing word in a sentence. We show how this function approximates the probability of occurrence of a given candidate word.

Additionally, We would be evaluating a novel approach to automated sentence completion based on Pointwise mutual information (PMI). Feature sets are created by fusing the various types of input provided to other classes of language models, ultimately allowing multiple sources of both local and distant information to be considered. Furthermore, it is shown that additional precision gains may be achieved by incorporating feature sets of higher-order n-grams.

2.6 Where our solution different from others:

Our approaches instead of relying only on N-gram occurrence also consider the historical context of the sentence. Additionally, our approach uses fast computation technique discussed in this paper (A. Mnih, Y. W. Teh, 2012) to address training and computation issue.

2.7 Where our solution is better:

Our solution differs from previous approaches at multiple points. We believe that these new novel approach which has outperformed N-gram models in similar context, will results in improvement in our project as well. Experiments discussed in previously researched paper had shown state-of-the-art performance and accuracy improvement over non-neural linguistic model. We assume the same improvement in our project over previously researched methods.

CHAPTER 3

METHODOLOGY

3.1 Dataset

The dataset being used is the Microsoft Research Sentence Completion Challenge dataset. This dataset was developed by Microsoft Research and publicly made available in 2012 (G. Zweig, C. Burges, 2012) in an effort to advance the research in the field of sematic analysis.

The dataset is based on five of Conan Doyle's Sherlock Holmes books that were a part of Project Gutenberg. It has 1040 sentence completion tasks that are based on the SAT question format. Each task has five options of which one is correct but all five fit well into the sentence.

The training set contains 522 books from Project Gutenberg open corpus, each of them having adequate headers. The test set had 1040 sentence completion tasks with five syntactically correct answers to each but not semantically.

3.2 Data Pre-Processing

At initial stages of data pre-processing, we read the files into the memory and removed the uninformative header texts followed by converting them to lower case. Next, we tokenized the sentences and then the words using NLTK's built in methods. The next step was to remove the stop words. Doing this for every file gave us our corpus.

While we were able to run our models with this initial pre-processing, the testing and training were taking a lot of time due to the size of the dataset. To resolve this, we stored the pre-computed data as our model and saved a significant amount of time for running tests and analysis.

3.3 N-gram Model

This is the most basic and intuitive approach for text prediction. So, for creating a baseline, we will implement the unsmoothed N-gram model. For each sentence, we check if the n-gram, for n = 2,3 and 4, occurs in the training set or not. If it does, we score the sentence with one as a score for bigram, two as a score for trigram and three as a score for 4-gram.

Next, to analyze the improvement in the system we will introduce smoothed n-gram model using Good Turing Smoothing. Smoothing is done to account for the "zero probability N-grams" (A.M. Woods, 2016) which are effectively the N-grams that haven't appeared in the training corpus but may be perfectly legible natural language phrase. The simplest way to smooth the probabilities is to add one to all counts of the N-grams, known as the Laplace Smoothing, formulated as follows for unigrams:

$$P_{Laplace}(w_i) = \frac{c_i + 1}{N + V}$$

where, V are all the words in the vocabulary and since one is added to all counts, we add V to the denominator to account for all the words. This can be extended to N-grams in general, as :

$$P^*_{Laplace}(w_n | w^{n-1}_{n-N+1}) = \frac{C(w^{n-1}_{n-N+1} w_n) + 1}{C(w^{n-1}_{n-N+1}) + V}$$

A little more sophisticated approach for smoothing is the Good Turning Discounting method. The intuition behind GTD is to use the count of things you've seen once to account for things that haven't occurred in the training set yet (D.Jurafsky, J.Martin, 2008). The GTD is formalized as (Jurafsky, Martin, 2008):

$$P^*_{GT}(things\ with\ zero\ frequency\ in\ training) = \frac{N_1}{N}$$

3.4 Latent Semantic Analysis

Stepping into semantic analysis, we propose to implement Latent Semantic Analysis(LSA) as the next model. Each sentence is considered as a document and a word-vector is formed for each sentence. Similarity between two words is calculated as follows:

$$(q, w) = \frac{\sum_{i=1}^{N} w_{i,q} \times w_{i,j}}{\sqrt{\sum_{i=1}^{N} w_{i,q}^2} \times \sqrt{\sum_{i=1}^{N} w_{i,j}^2}}$$

Latent Semantic Analysis(LSA) helps in determining hidden features in documents. Thus the technique is used to extract the contextual-usage meaning of words from the documents.

The LSI framework consists of 3 steps which are mentioned below.

1. Preprocessing of input data: In the initial step, the input text is tokenized and stopwords are removed from every document of the corpus1. Each term in the text is then represented as a tuple containing term-id and term frequency. A matrix is created in which the rows denote the unique terms and the columns denote the documents. Every cell denotes the term count in the corresponding document. The matrix of term-frequency counts obtained from the term document matrix is then modified using the TF-IDF technique so as to give more weight to rare terms compared to common terms across documents and also to frequently occurring terms in a particular document. The formula for weighing each term can be represented as,

$$DocumentTermWeight = f_{t,d} \times ln\left(\frac{N}{n_t}\right)$$

   Where:

   $f_{t,d}$: count of term t in document d
   N : the total count of documents
   n_t : the count of documents having term t

The term-document matrix gets modified to contain weights of each term in a given document. The dimensionality reduction of this matrix is done using Singular Value Decomposition (SVD).

2. Singular Value Decomposition: LSA uses SVD for generating the vectors of a particular text. The matrix X(term-document) is used to calculate two matrices. These are,

$$Y = X_T X$$
$$Z = X X_T$$

Where:

X : term - document matrix
Y : document - document matrix
Z : term - term matrix

After finding eigenvectors of B and C matrices, we get left singular matrix, L and right singular matrix, R respectively.

Thus, term - document matrix, X, is divided into unique combination of three matrices as follows -

$$X = LSR_T$$

Where:

L : Term - Concept weight matrix
R_T : Concept - Document weight matrix
S : Diagonal matrix representing concept weights

S is calculated by taking the square root of the eigenvalues of matrix Y .

To reduce the dimensionality of the matrices in equation 4, top k concepts are selected and thus matrix X is approximated as,

$X_k = L_k S_k R_T$

In the testing phase, after generating weight matrix using the Term Frequency - Inverse Document Frequency (TFIDF) model, input text gets converted into a query matrix, Q. This matrix Q is then multiplied with matrices LK and Sk to generate new query vectors calculated as follows:

$$NewQueryVectors = QL_k S_k$$

The probability of a word given its history is computed by the LSA model in the following way. Let h be the sum of all the LSA word vectors in the history. Let m be the smallest cosine similarity between h and any word in the vocabulary V : m = minw∈V sim(h, w). The probability of a word w in the context of history h is given by

$$P_{LSA}(w|h) = \frac{(h,w) - m}{\sum_{q \in V}\big((h,q) - m\big)}$$

3.5 Pointwise Mutual Information

To apply Pointwise Mutual Information(PMI) the dataset will be preprocessed. We will convert all words to lowercase followed by removal of stop words. This is followed by POS tagging of the words using NLTK. After this process, the tokens with uninformative tags will be removed. Uninformative tags include tags like determiners as they add no value to the contextual analysis. The PMI technique defines three feature sets and the tokens are added to each matric per relevance. The following are the three feature sets (A.M.Woods, 2016):

1. Reduced Context: The words remaining after pre-processing ,i.e., removal of uninformative tags were included in this set.
2. Dependencies: Sentence words that are semantically dependent on the candidate were included in this set.
3. Keywords: A list of tokens that have more information than other tokens such as the common nouns are kept in this set.

Once, this is done, we construct a word-context matrix, X where each element *x(i,j)* represents the number of times a word *i* occurs in context *j*. Then, we apply PMI model to create the matrix Y where each element *y(i,j)* is defined as:

$$y(i,j) = \frac{x(i,j)}{\sum_{i=1}^{n}\sum_{j=1}^{m} x(i,j)}$$

We calculate the PMI as follows:

$$P(i,j) = \frac{x(i,j)}{\sum_{i=1}^{n}\sum_{j=1}^{m} x(i,j)}$$
$$P(i\,*) = \frac{\sum_{j=1}^{m} x(i,j)}{\sum_{i=1}^{n}\sum_{j=1}^{m} x(i,j)}$$
$$P(*\,j) = \frac{\sum_{i=1}^{n} x(i,j)}{\sum_{i=1}^{n}\sum_{j=1}^{m} x(i,j)}$$
$$pmi(i,j) = \log\left(\frac{P(i,j)}{P(i\,*)P(*\,j)}\right)$$

To avoid negative values, we replace them by zeros, defined as the metric Positive Pointwise Mutual Information(PPMI) as follows:

$$ppmi(i,j) = \max\left\{0, \log\left(\frac{P(i,j)}{P(i\,*)P(*\,j)}\right)\right\}$$

The score for each candidate, i, replaced with blank in a sentence is found out and the best answer is returned. The score is calculated as follows:

$$similarity(i,S) = \sum_{j\in S} \in dpmi(i,j)\,.\,\gamma$$

where, $dpmi(i,j) = pmi(i.j).\delta(i,j)$, and γ is the number of feature sets containing *j.* Further, *δ(i,j)* is found as follows:

$$mincontext = min\left(\sum_{k=1}^{n} x(k,j) \sum_{k=1}^{m} x(i,k)\right)$$
$$\delta(i,j) = \frac{x(i,j)}{x(i,j)+1}\cdot\frac{mincontext}{mincontext+1}$$

As an improvement to PMI, we also performed the smoothing to account for the zero values by implementing add-one or Laplace smoothing by adding one to all the counts in the co-occurrence matrix.

3.6 Performance Metrics

The performance of each method will be tested based on the accuracy metric, which is measured as follows:

$$Accuracy = \frac{no.of correctpredictions}{totalpredictions}$$

3.7 Testing

Microsoft Research has provided the correct answers to each of the sentence completion tasks. We test the output from our model against the correct answers to compute the overall accuracy of our models.

CHAPTER 4

IMPLEMENTATION AND RESULTS

4.1 Implementation Details

All the models have been implemented on Python3 using Jupyter Notebooks and Colaboratory by Google Research. We’ve used Python libraries like NLTK, Gensim, SciPy and Numpy.

For n-grams, we created the bigrams, trigrams and 4-grams. Because of a very large number of n-grams the computation capabilities required were a lot. The training time went upto 3 hours. For scoring, we added a score of one for a bigram match, two for trigram match and three for 4-gram match, where match implies atleast one occurrence in the background text.

For LSA, we first implemented a bag of words model to the corpus followed by finding the tf-idf weights for the word vectors. Next, we performed LSA on the tf-idf vectors, for 100 features.Then, we computed the cosine similarity between the candidate answer and the question statement. Finally, we computed average cosine similarity for all the features and returned the option with the highest average cosine similarity.

For PMI, we set a minimum threshold at 5. This was to disregard the words with lower frequency as they do not give significant improvement over accuracy but can end up using a lot of computation and storage capacity. This helped to reduce the words from 2M to 45,000. Then, we computed the co-occurrence matrix for all the words in the reduced context. Next, we computed PMI for each candidate answer with other question statement tokens and returned the answer with the best PMI score.

4.2 Results

The results coincided with the expectations derived by our theoretical basis and knowledge as summarized in Table 1. We expected to get the lowest accuracy with n-grams as the task has specifically been designed to not be solved by the sole use of n-grams. We achieved the benchmark accuracy set of 38% (G. Zweig, C. Burges, 2012).

Latent Semantic Analysis being a more sophisticated model for semantic analysis gave an accuracy of UNK which improved upon n-grams. The penultimate model that we ran was the Pointwise Mutual Information, which calculates the score for a word with its context and thus was expected to give an accuracy surpassing the above methods. We achieved an accuracy of 49.61%.

Next, to analyze another approach, we implemented the add one smoothing to PPMI. While we expected PPMI to give an improved performance, we got an accuracy of 28% with PPMI.

Approach	Accuracy	Expected Accuracy	Reference
N-grams	38%	31%	Zweig and Burges, 2012
LSA	36%	44%	Zweig, Platt, Meek, 2012
PMI	61.09%	61.44%	A.M. Woods, 2016
PPMI	28%	> 61.09%	-

Table 1: Summary of different methods

4.3 Result Analysis

While the results of n-grams and PMI were as per our expectations, the results of PPMI were not. We had expected PPMI to give an improvement in accuracy over PMI but we received a fairly low accuracy for it. It's our speculation that this happened because PPMI takes max of the PMI measure and 0 and factors in the co-occurrence of the candidate answer and the words in the vocabulary. While, the independent occurrence frequency is present, the co-occurrence is sparse which leads to negative values of PMI. Thus, when we take max we get the score of zero for all the candidate answers and hence the choice becomes ambiguous, resulting in a very low accuracy. Also, we observed that we received an accuracy lower than expected for LSA

4.4 Final Note

While going about the implementation, one thing that was considerably disturbing to us was the computation time and efficiency required to process the dataset, as the dataset we had chosen was considerably a large one. So, as a work around for it, we looked for other methods. As a result of our search, we came across Google Colaboratory, which is an open source version of Jupyter Notebooks running on Google's servers. It has recently launched a GPU support which helped us to run the computations quickly. So, we have completed the pre-processing on Colaboratory and then ran the base program on our system.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this project, we took up the Microsoft Research Sentence Completion Challenge and implemented sentence completion using text prediction on it. We have tried to solve the problem by implementing the approaches learnt as a part of part of coursework, namely, n-grams, Latent Semantic Analysis, Pointwise Mutual Information and Positive PMI. The accuracy achieved by PMI was considerably more than LSA and n-grams. While we saw a decline in accuracy for PPMI, we suspect it is because we took the max of zero and PMI score, thus, undermining the penalized words.

5.2 Future Work

The future work for the project could involve improvement through the conjunction of Neural Networks models, mainly, LSTM along with the statistical approaches we've used and learnt throughout the project.

BIBLIOGRAPHY

[1] A Challenge Set for Advancing Language Modeling, G. Zweig and C. Burges, Microsoft Research, 2012.

[2] Speech and Language Processing, D. Jurafsky and J. Martin, Second Edition, Prentice Hall, 2008.

[3] Exploiting Linguistic Features for Sentence Completion, A.M. Woods, Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing, 2016.

[4] Word association norms, mutual information, and lexicography. K. Chruch and P. Hanks, Computational Linguistics, 1990.

[5] Thomas Landauer and Susan Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge. Psychological Review, 104(2), pages 211–240.

[6] Saif Mohammed, Bonnie Dorr, and Graeme Hirst. 2008. Computing word pair antonymy. In Empirical Methods in Natural Language Processing (EMNLP).

[7] Saif M. Mohammed, Bonnie J. Dorr, Graeme Hirst, and Peter D. Turney. 2011. Measuring degrees of semantic opposition. Technical report, National Research Council Canada.

[8] Peter Turney. 2008. A uniform approach to analogies, synonyms, antonyms, and associations. In International Conference on Computational Linguistics (COLING).

[9] Peter D. Turney, Michael L. Littman, Jeffrey Bigham, and Victor Shnayder. 2003. Combining independent modules to solve multiple-choice synonym and analogy problems. In Recent Advances in Natural Language Processing (RANLP).

[10] Iddo Lev, Bill MacCartney, Christopher D. Manning, and Roger Levy. 2004. Solving logic puzzles: from robust processing to precise semantics. In Proceeding of the 2nd Workshop on Text Meaning and Interpretation, pages 9–16. Association for Computational Linguistics.

[11] Lynette Hirschman, Mark Light, Eric Breck, and John D. Burger. 1999. Deep read: A reading comprehension system. In Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics.

[12] Eugene Charniak, Yasemin Altun, Rodrigo de Salvo Braz, Benjamin Garrett, Margaret Kosmala, Tomer Moscovich, Lixin Pang, Changhee Pyo, Ye Sun, Wei Wy, Zhongfa Yang, Shawn Zeller, and Lisa Zorn. 2000. Reading comprehension programs in a statistical-language-processing class. In Proceedings of the 2000 ANLP/NAACL Workshop on Reading comprehension tests as evaluation for computer-based language understanding sytems - Volume 6, ANLP/NAACL-ReadingComp '00, pages 1–5. Association for Computational Linguistics.

[13] Ellen Riloff and Michael Thelen. 2000. A rule-based question answering system for reading comprehension tests. In Proceedings of the 2000 ANLP/NAACL Workshop on Reading comprehension

tests as evaluation for computer-based language understanding sytems - Volume 6, ANLP/NAACL-ReadingComp ’00, pages 13–19.

[14] Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. Computational Linguistics, 16(1):22–29.

[15] Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In Advances in Neural Information Processing Systems 26, pages 2265–2273. Curran Associates, Inc.

[16] Mikolov T., Chen K., Corrado G., Dean J., "Efficient estimation of word representations in vector space, " arXiv preprint arXiv: 130 1.3781, 2013

[17] Deniz Yuret. 2007. Ku: word sense disambiguation by substitution. In Proceedings of the 4th International Workshop on Semantic Evaluations, SemEval ’07, pages 207–213, Stroudsburg, PA, USA. Association for Computational Linguistics.

[18] Diana McCarthy and Roberto Navigli. 2007. Semeval 2007 task 10: English lexical substitution task. In Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007), pages 48–53.

[19] Samer Hassan, Andras Csomai, Carmen Banea, Ravi Sinha, and Rada Mihalcea. 2007. Unt: Subfinder: Combining knowledge sources for automatic lexical substitution. In Proceedings of the 4th International Workshop on Semantic Evaluations, SemEval ’07, pages 410–413, Stroudsburg, PA, USA. Association for Computational Linguistics.

[20] A. Mnih, Y. W. Teh, "A fast and simple algorithm for training neural probabilistic language models, " arXiv preprint arXiv:1206.6426, 2012.