AmritaCEN at SemEval-2016 Task 11: Complex Word Identification using Word Embedding

Sanjay S.P and **Anand Kumar M** and **Soman K P**
Centre for Computational Engineering & Networking (CEN)
Amrita School of Engineering, Coimbatore
Amrita Vishwa Vidyapeetham
Amrita University
India
sanjay.poongs@gmail.com, m_anandkumar@cb.amrita.edu, kp_soman@amrita.edu

Abstract

Complex word identification task focuses on identifying the difficult word from English sentence for a Non-Native speakers. Non-Native speakers are those who don't have English as their native language. It is a sub-task for lexical simplification. We have experimented with word embedding features, orthographic word features, similarity features and POS tag features which improves the performance of the classification. In addition to the SemEval 2016 results we have evaluated the training data by varying the vector dimension size and obtained the best possible size for producing better performance. The SVM learning algorithm will attains constant and maximum accuracy through linear classifier. We achieve a G-score of 0.43 and 0.54 on computing complex words for two systems.

1 Introduction

Complex Word Identification (CWI) is the task of identifying difficult words for a Non-Native speaker. The main objective of Complex Word Identification is to simplify and enhance the perplexing words in the sentences for Non-Native speakers. Complex Word Identification is used in Lexical Simplification (LS). The Lexical Simplification is important for all Text Simplification (Gustavo Henrique Paetzold, 2006). The perspective of each user is different therefore the task Complex Word Identification have been demanding one. For distinguish complex words for individual user will be a tedious task, therefore the CWI task is a difficult one, so we are identifying the complex words for a group of users.

CWI could be used in summarization of stories or generate abstract of a book. Shortening a paragraph would be difficult without removing the complex words, since the shortened text with complex words would be very difficult or confusing for non-native users to read and understand. We can make a story more simple for kids to read by replacing complex words with most common words (Tomoyuki Kajiwara et al., 2013). The main ways to identify the complex words are Threshold-based and Classification-based approches (Gustavo Henrique Paetzold, 2006). In Threshold-based approach, length, number of syllables, count of ambiguity and word frequency are considered as features. If the values are above the threshold, then the word is said to be a complex word. In the Classifier-based approach, the word features are trained and the word features which are similar to the trained word features are considered to be a complex word.

2 Complex Word Identification (CWI) - Task Description

The SemEval-2016 organizers have released three files *cwi_training, cwi_training_allannotations, and cwi_testing.* The first two files are training files and the third file is a testing file. Our task is to find the words in the testing file and label each one as 1 if the given word is a complex word, or 0 if the word is not a complex word.

The *training_allannotations* file contains tags of 20 authors for given sentences. The training file contains a single tag for the sentences representing the group of 20 authors in allannotations file. Each word is tagged complex if any of the author tagged it as 1.

If the all authors tagged it as non-complex then it is tagged as 0. We trained a binary classifier on the *cwi_training* data to identify the complex words in the testing data.

3 Neural word embedding

From the given sentences, unique words are converted into a more meaningful mathematical vector representation called a Neural word embedding (Tomas Mikolov et al., 2013). For a given word, the vector formed by the word is related to the semantic and syntactic contexts of the word, and also related to the the neighboring occurrence of the word. To produce the maximum vector of given size. The w_0 is the given word where the conditional probability is given in the Figure 1, where j is the index of actual output in the output layer.

3.1 Gensim - Implementation

The Gensim word2vec uses skip-gram and CBOW models. The word2vec run two passes to the training data. The first pass will collect the required unique words and build the tree structure. The second pass will train the neural model (Tomas Mikolov et al., 2013). We can vary the dimensions of the vector space model.

4 Methodology

The input data to our system is in the form of sentences but the output requires only word to tagged, so we cannot train the classifier with sentence features. The sentence with the word to be identified are tokenized. We have given the tokenized data into a word embedding models, it produced vectors for each words. we have acquired similarity features with nearby words. The other features like POS tagging and orthographic word features are obtained from the word and these features does not depend on the sentence. All the features are incorporated and saved in a text file. The text file is given to a SVM classifier which compares with the trained model and generates the result. The working process is given in Figure 1.

5 Feature Extraction

The features, are extracted from the given word in the testing data. We have extracted four types of features, namely orthographic features, similarity features, word embedding features and POS tag features. In these four types of features the orthographic feature based on the word itself, it provides the shape features of the word. The other two features namely similarity features and word embedding features are extracted from the word2vec model which is mentioned above. The Part Of Speech (POS) tag features is obtained with the help of NLTK tool kit and all the features are incorporated for training.

5.1 Orthographic features

Orthographic word features do not depend on the sentence. They are generic features of a given word like length of the word, number of syllables, ambiguity count and frequency. The count of the characters in the corresponding target word which provides the length of the word. The syllable feature gives the number of syllables present in the word, using the pyhyphen package we have retrieved the features. We used NLTK WordNet synsets (Steven Bird, 2006) for obtaining the ambiguity of the word. WordNet has an built-in dictionary from which number of ambiguity present for the word is obtained (George A. Miller, 1995; **?**). The ambiguity is defined as the number of words that have similar meaning. A word is said to be more familiar if the word has more meaning. In the frequency feature, we count occurrences of the word in both training and testing data. These word features improve the accuracy.

5.2 Similarity features

These features represent how similar the word occurs to the neighboring words. We have derived four similarity values. Let us assume that the given word is represented as w_0 then the similarity values are attained as $[w_0 \& w_{-2}][w_0 \& w_{-1}][w_0 \& w_1][w_0 \& w_2]$. The similarity values are acquired from the word2vec model using Gensim.

5.3 Word embedding features

All the training and testing data were combined and passed into word2vec model. In word2vec, the unique words are obtained and its corresponding vector representations are created for all the words. We can vary the size of the vector while creating the

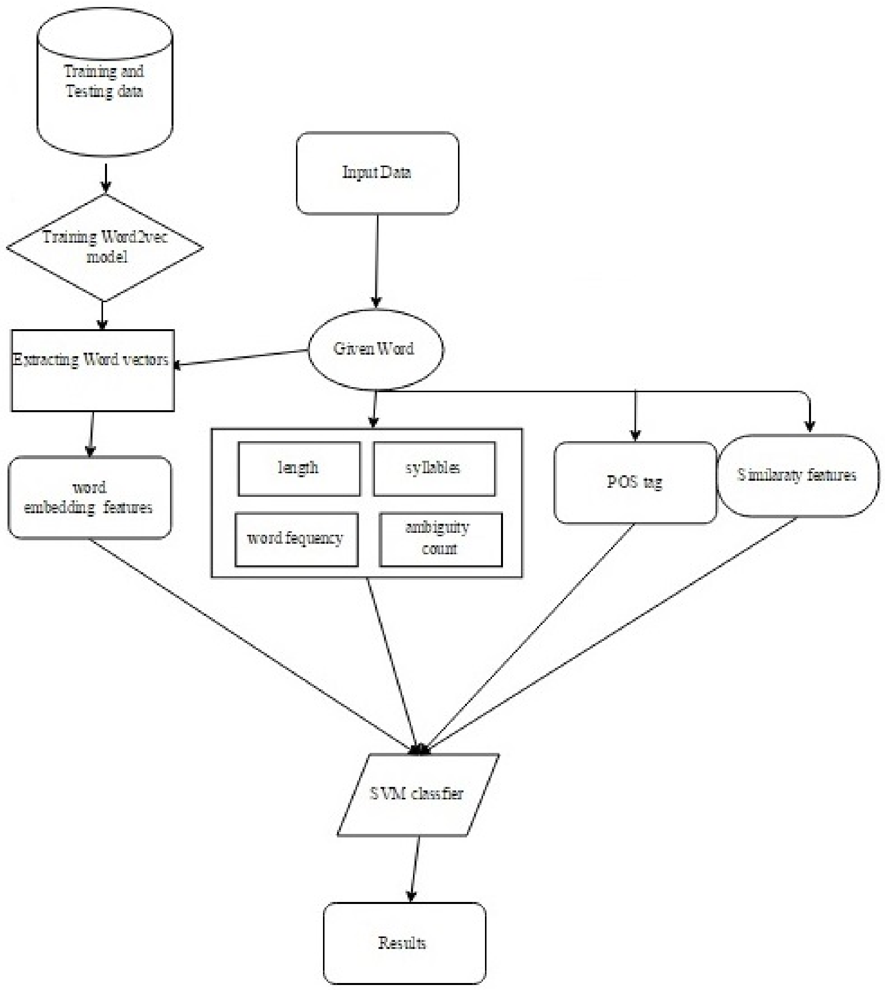

Figure 1: Working model

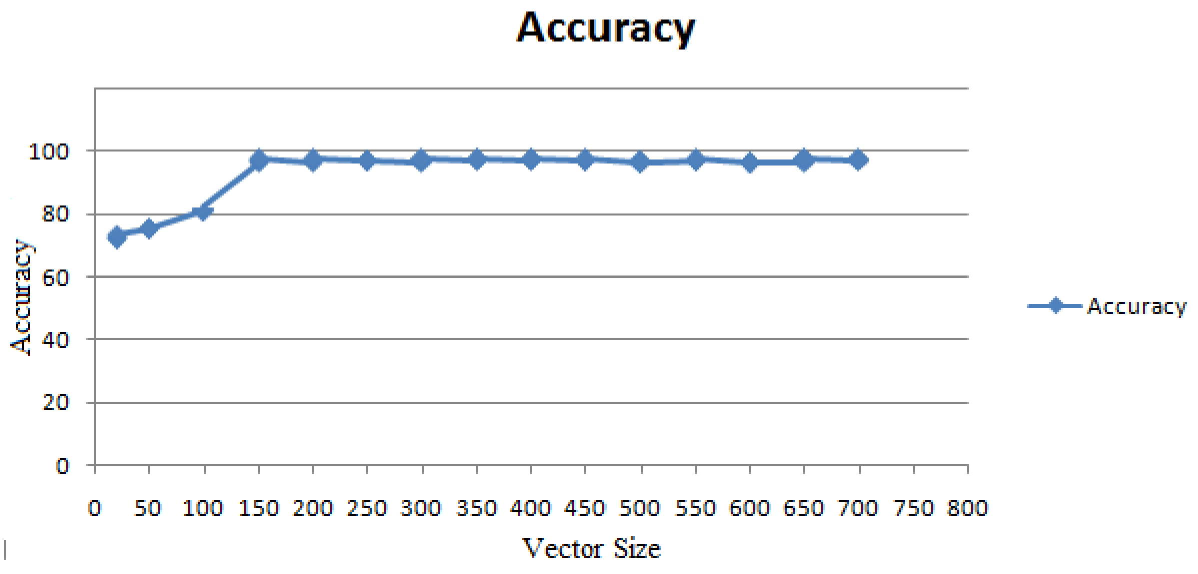

Figure 2: Training accuracy

vector representation model. We have obtained various results by varying the vector size. We varied the vector size, getting good accuracy after a vector size of 150, as shown in Figure 3. Finally concluded the best vector size as 500 at which we have obtained reasonable results for all other features. The curve tends to decrease after vector size of 500.

5.4 POS tag features

The Part Of Speech tag is taken for five words, by assuming the current word as w_0, then the POS tagging is taken for $w_{-2}, w_{-1}, w_0, w_1, w_2$. We are using NLTK (Natural Language Tool Kit) for POS tagging. The NLTK is an open source tool kit for python[3]. We implemented 2 systems. The first system trained an SVM without including the POS tagging. In the second system, POS tagging with other features has been included. The POS tag increases the accuracy of the data but it might reduce the recall.

6 Classifier

The SVM classifier is used for classification. The data is classified based on the features of the test data. For a given training point (x^i, y^i) it finds out the hyper plane $w^t(x) + b = 0$. Here we have used LIBSVM (Chang and Lin, 2011), which is integrated tool for Support Vector Classification (SVC) with regression and distributed estimation [1]. Python LIBSVM supports cross-validation. It is built on the C/C++ core SVM performance.

7 Data set:

We were provided with 2,237 training words and 88,221 testing words. In the training data there are 706 (31.56%) complex words and 1531 (68.4%) non complex words. The training file is in the format of <sentences> <word> <position> <1 or 0>. 1 indicates if the word is complex and 0 indicates if the word is not complex. The test data is in the format of <sentences> <word> <position>.

8 Experimental results

The Figure 3 gives our system's precision, recall, and F-score on the CWI training data. We tried varying the vector size in the word2vec model. Word features include length, number of syllables, ambiguity count, and frequency. We have four possible combinations and the test results are been tabled in Table 1. The first combination includes word2vec features. The second combination includes word2vec and word features. The third combination includes word2vec and similarity features. The fourth combination includes word2vec, similarity and word features. By varying the size and features the precision and recall tend to combine at a point called break though point. After the break though point the F-score becomes steady. The break through point for data is 150. Above which the accuracy tends to be linear, After the break through point the curve tends to decrease after 500. Therefore, we have concluded at a vector size of 500 and including all the features we have achieved the maximum accuracy. From the Figure 3 we can clearly see the F-score gradually decreasing at 550.

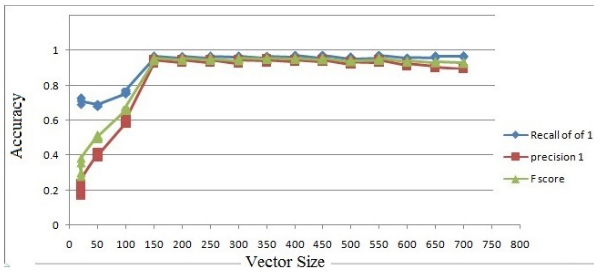

Figure 3: F-score accuracy

The accuracy graph steadily increases and reaches a maximum at 500 then accuracy tends to steadily decreases. Table 1, provides the additional results experimented by us.

8.1 SemEval Results:

We have submitted 2 files w2vecSimPos and w2vecSim. The first file w2vecSimPos is extracted by including all the four features mentioned namely word embedding, similarity, special word features, and POS tag features. The second file consist of word embedding, similarity and special word features. In the second file we have not included POS tag features. From the table, we can clearly see by including POS tag features there is in an improvement in the accuracy of the system. F-score denotes harmonic mean between precision and recall. G-score denotes harmonic mean between accuracy and recall. By including the POS tag as a feature the ac-

Table 1: Additional Results

Embedding Size	**Word2vec**	**Word2vec + Orthographic**	**Word2vec + Similarity**	**Word2vec + Orthographic+ Similarity**
20	0.27721	0.35542	0.30112	0.38263
50	0.511586453	0.513227513	0.495495495	0.505829596
100	0.657097289	0.657643312	0.669316375	0.679904686
150	**0.941852118**	**0.951833214**	**0.949466192**	**0.954903364**
200	0.943287868	0.94751977	0.950537634	0.955587393
250	0.944723618	0.954154728	0.946543122	0.953471725
300	0.943042538	0.951149425	0.95149786	0.955523673
350	0.948497854	0.949567723	0.955777461	0.955587393
400	0.948424069	0.950395399	0.955714286	0.95764537
450	0.946236559	0.951079137	0.951428571	0.95764537
500	**0.946599424**	**0.951516245**	**0.954960686**	**0.952101501**
550	0.944723618	0.949640288	0.953604568	0.957706093
600	0.93410572	0.93487699	0.935158501	0.940836941
650	0.932016353	0.933913358	0.930656314	0.934688976
700	0.927825678	0.928478303	0.927098887	0.928424859

curacy is improved but the recall is decreased therefore G-score is also decreased.

Table 2: SemEval Results.

System	**W2VecSimPos**	**W2VecSim**
Accuracy	0.743	0.627
Precision	0.060	0.061
Recall	0.306	0.486
F-Score	0.100	0.109
G-Score	0.434	0.547

9 Conclusion

The proposed model with features specially derived for identifying complex words produced good results. The word embedding features played a major role compared with other features. By increasing the size of vectors our accuracy increased constantly. We tried all possible combinations with the four features mentioned in the Table 1 and conclude that the word embedding features alone also have a good accuracy. Our systems have generated a good precision compared with the baseline. Among 45 systems, we ranked 27th position for G-score evaluation. Although, the frequency counts and embedding features are reasonable for a given resource-constraint task, if we have provided for English language in general it might have provide with better accuracy.

References

Chih-Chung Chang and Chih-Jen Lin. (2011). *LIBSVM: a library for support vector machines.* ACM Transactions on Intelligent Systems and Technology (TIST).2(3), p.27.

Christiane Fellbaum. (1998). *WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press.*

Daniel Ferres, Montserrat Marimon, Horacio Saggion. (2015). *A Web-based Text Simplification System for English.* Procesamiento del Lenguaje Natural, 55, 191-194.

Dany Amiot. (2004). *Between compounding and derivation Elements of word-formation corresponding.* Morphology and its demarcations: Selected papers from the 11th Morphology meeting, Vienna, Vol. 264.

George A. Miller. (1995). *WordNet: A Lexical Database for English.* Communications of the ACM Vol. 38, No. 11: 39-41.

Gustavo Henrique Paetzold. (2015). *Reliable Lexical Simplification for Non-Native Speakers.* NAACL-HLT 2015 Student Research Workshop (SRW).

La Pointe, Linda B and Randall W. Engle. (1990). *Simple and complex word spans as measures of working memory capacity.* Journal of Experimental Psychology: Learning, Memory, and Cognition 16.6 (1990): 1118.

Lucia Specia, Dhwaj Raj, Marco Turchi. (2010). *Machine translation evaluation versus quality estimation.* Machine translation, 24(1), pp.39-50.

Or Biran, Samuel Brody, Noemie Elhadad. (2011). *Putting it simply: a context-aware approach to lexical* *simplification.* Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2. Association for Computational Linguistics.

Radim Rehurek and Petr Sojra. (2010). *Software framework for topic modelling with large corpora.* Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, ELRA, p.45-50

Edward Loper and Steven Bird. (2006). *NLTK: the natural language toolkit.* Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics-Volume 1. Association for Computational Linguistics, 2002.

Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean. (2010). *Efficient estimation of word representations in vector space.* rXiv preprint arXiv:1301.3781.

Tomoyuki Kajiwara, Hiroshi Matsumoto, Kazuhide Yamamoto. (2013). *Selecting Proper Lexical Paraphrase for Children.* Proceedings of the Twenty-Fifth Conference on Computational Linguistics and Speech Processing (ROCLING 2013).

William Snyder. (2001). *On the nature of syntactic variation: Evidence from complex predicates and complex word-formation.* Language Vol. 77, No. 2, pp. 324-342.