Complex Word Identification: Convolutional Neural Network vs. Feature Engineering

Segun Taofeek Aroyehun
CIC, Instituto Politécnico Nacional
Mexico City, Mexico
aroyehun.segun@gmail.com

Jason Angel
CIC, Instituto Politécnico Nacional
Mexico City, Mexico
ajason08@gmail.com

Daniel Alejandro Pérez Alvarez
CIC, Instituto Politécnico Nacional
Mexico City, Mexico
daperezalvarez@gmail.com

Alexander Gelbukh
CIC, Instituto Politécnico Nacional
Mexico City, Mexico
www.gelbukh.com

Abstract

We describe the systems of NLP-CIC team that participated in the Complex Word Identification (CWI) 2018 shared task. The shared task aimed to benchmark approaches for identifying complex words in English and other languages from the perspective of non-native speakers. Our goal is to compare two approaches: feature engineering and a deep neural network. Both approaches achieved comparable performance on the English test set. We demonstrated the flexibility of the deep-learning approach by using the same deep neural network setup in the Spanish track. Our systems achieved competitive results: all our systems were within 0.01 of the system with the best macro-F1 score on the test sets except on Wikipedia test set, on which our best system is 0.04 below the best macro-F1 score.

1 Introduction

Complex Word Identification (CWI) is an important step in text simplification. The ability to accurately identify word(s) as complex or not in a given context for a given target population significantly impacts subsequent processing steps such as lexical substitution in the simplification pipeline.

The organizers of the 2018 CWI shared task (Yimam et al., 2018) provided participants with multilingual human-annotated datasets (Yimam et al., 2017a,b) for the identification of complex words. Training and development data were provided for three languages: English, Spanish, and German. In the case of English, three genres were covered: news, Wikinews, and Wikipedia.

Some of the participants of the previous CWI shared task used neural network-based approaches. For instance, Bingel et al. (2016) used a simple feed-forward neural network, while Nat (2016) used an ensemble of recurrent neural networks (RNN). The performance of the neural network approaches was not impressive. The RNN achieved the best result among all the submissions that used neural networks (Paetzold and Specia, 2016b).

In this paper, we report further experiments with the efficacy of deep neural networks for CWI, using another deep neural network architecture—Convolutional Neural Network (CNN). Namely, we compare two approaches for the task of CWI: one based on an extensive feature engineering and the tree ensembles classifier, and another one based on deep neural network using the word embedding representation. The latter approach is, to the best of our knowledge, the first attempt to apply CNNs to the task of CWI. Apart from comparing the performance of the two approaches on the classification subtask of CWI on English, we demonstrate the flexibility of the CNN-based approach by applying it to another language—Spanish in our case.

The remainder of the paper is organized as follows. Section 2 outlines relevant work. Sections 3 and 4 present our two approaches. Section 5 gives some details on the datasets used. Results of our experiments are in Section 6. Section 7 presents error analysis. Finally, Section 8 concludes the paper and outlines future work directions.

2 Related Work

The majority of works on CWI are related to feature engineering at various linguistic levels. Section 2.1 below discusses existing approaches to feature engineering for machine-learning models used for CWI. On the other hand, Section 2.2 men-

Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications, pages 322–327
New Orleans, Louisiana, June 5, 2018. ©2018 Association for Computational Linguistics

tions some relevant applications of CNNs to natural language processing (NLP).

2.1 Feature Engineering for he CWI Task

Participants of the first edition of CWI shared task have experimented with various linguistic features. These linguistic features span various linguistic levels: morphological, syntactic, semantic, and psycholinguistic. Paetzold and Specia (2016c) used morphological, lexical, and semantic features to train frequency-based, lexicon-based, and machine-learning models for CWI. Konkol (2016) used only frequency of occurrence of a word in Wikipedia as the only feature to train a Max entropy classifier. Davoodi and Kosseim (2016) experimented with the degree of abstractness of a word as a psycholinguistic feature for CWI.

In this work, we used some of these features and experimented with some new features, such as contextual and entity information and additional psycholinguistic scores.

2.2 CNNs in NLP

Convolutional neural networks have shown notable results in the fields of computer vision, speech recognition and recently in NLP.

CNN models achieve state-of-the-art results in NLP tasks such as clause coherence (Yin and Schütze, 2015b), paraphrase identification (Yin and Schütze, 2015b,a) and Twitter sentiment analysis (Severyn and Moschitti, 2015).

Kim (2014) presents a CNN fed with word2vec word embedding vectors (Mikolov et al., 2013) used for detection of positive and negative reviews, as well as sentence classification. Their results suggest that pre-trained vectors encode generic semantic features, which can benefit various NLP classification tasks. In our work, we used a similar model with slight additions to the architecture of the network and a different preprocessing step.

3 Feature-Engineering Approach

In this section, we present the set of features used to build one of our CWI systems.

Morphological Features Most of the morphological features we used consist of frequency count of target text on large corpora such as Wikipedia and Simple Wikipedia. We computed term frequency, inverse term frequency, document frequency and term document entropy. Also, the tf-idf values were calculated. In addition, we used characteristics of each target text such as number of characters, vowels, and syllables.

Syntactic and Lexical Features We used OpenNLP[1] part-of-speech (POS) tagger to determine the target word's POS in the context. We used the POS as a parameter to filter the possible meanings of the target word. With this, we obtained the number of senses, lemmas, hyponyms, and hypernyms given by WordNet.[2]

Psycholinguistic and Entity Features We included some psycholinguistic scores provided by the improved MRC psycholinguistics database (Paetzold and Specia, 2016a) as features. The database provides familiarity, age of acquisition, concreteness, and imagery scores for each word. We hypothesized that these scores would be useful to identify complex word. Unfortunately, many target words were absent in this database. We used OpenNLP and Stanford CoreNLP[3] to tag target words as organization, person, location, date, and time. The resulting tag was used as an *entity* feature.

Word Embedding Distances as Features Beyond these classic linguistic features, we used word embeddings. Namely, we downloaded the pre-trained Word2vec (Mikolov et al., 2013) vectors of 300 dimensions to measure the distance between the sentence and the target word. The distance was computed using cosine similarity and Euclidean similarity over the average of the vector representation of the words in the sentence and the target text.

Classical Machine Learning Models We noticed that for this task (with our features), the tree learner offered better performance than other models. Thus, we tried several settings for the tree learner model provided by KNIME (Berthold et al., 2009), as well as more complex variations such as random forest, gradient boosted, and tree ensembles. The best obtained result was given by the tree ensembles with 600 models.

4 Deep-Learning Approach

In this section, we present our deep-learning approach. It is based on 2D convolution and word-

[1] http://opennlp.apache.org/

[2] https://wordnet.princeton.edu/

[3] https://stanfordnlp.github.io/CoreNLP

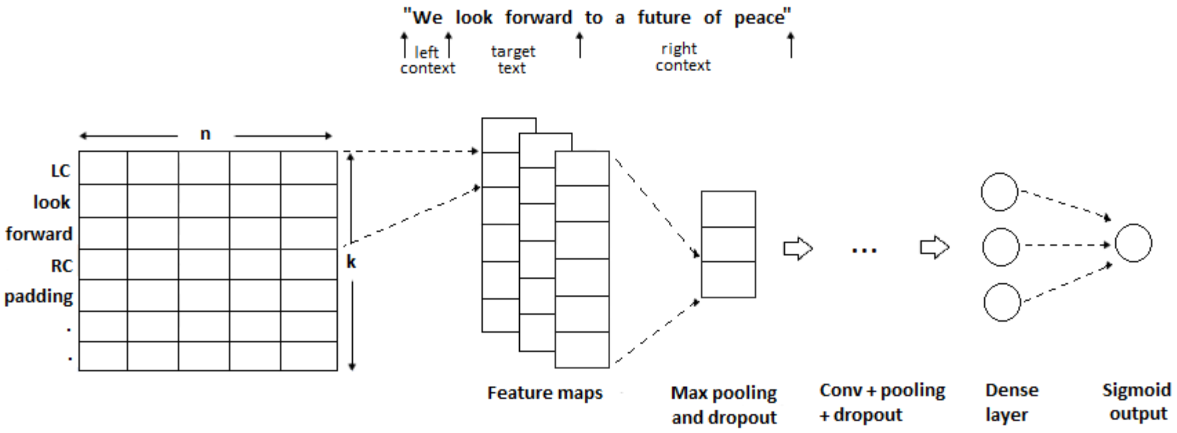

Figure 1: The architecture of our network

embedding representation of the target text fragment and its context.

Since text is one-dimensional, we applied a preprocessing step described in Section 4.1 prior to the application of convolution layer. Section 4.2 describes our network architecture, and Section 4.3 presents the training procedure.

4.1 Preprocessing

As a first step, we removed punctuation marks, digits, and special characters. Each word was then replaced by its vector representation using the pre-trained word vectors from the Word2vec model (Mikolov et al., 2013) for English and fastText model (Grave et al., 2018) for Spanish. A min-max normalization was applied to every vector to convert the values from the range $[-1, 1]$ to $[0, 1]$. We assigned a zero vector to the words missing in the pre-trained embeddings.

We defined the left context (LC) and the right context (RC) as those words that appear to the left and the right of the target text, respectively. As a compact representation of the left or right context, we used one 300-dimensional vector calculated as the average of the vectors of all the words in the LC and RC, respectively (if the target text was located at the beginning or the end of the sentence, we used a zero vector as the respective context representation). Next, we generated a matrix where the first row corresponds to the LC vector, the next k rows are given by the embedding vectors of the words contained in the target text, where k is the number of words in the target text, and the last row corresponds to the RC vector. In order to have a regular representation, we padded the matrix with $p = m - k$ zero vectors, where m is the maximum value of k in the training set.

Figure 1 illustrates the preprocessing step on the sentence of an example in the English training set. The output of the preprocessing step is the input of the network.

We believe that the averaging operation on the words in the contexts allowed differentiating between cases where the same sentence has distinct target texts. Those words included or excluded in the context will slightly modify the representation of the context, which will help the model to learn some relationships between the target text and the rest of the sentence. We could have compressed the representation matrix by combining the vector representation of the words in the target text instead of stacking them. However, this could dramatically reduce the valuable information pertaining to the target text.

4.2 Architecture of our Network

In our architecture, we used an input, convolution, pooling, and fully-connected layers; see Figure 1. Below we describe each of these layers except the input layer, which was described in Section 4.1.

Convolution The number of filters in this layer varied from 16 to 256 with a convolution stride of 1 and kernel size in the range of 2 to 4. We applied the rectified linear unit activation function to the output of this layer in order to introduce non-linearity. This layer is central to the idea of CNN, which enables the network to identify the most important features in the input. The output of this layer is often referred to as feature maps. Our net-

Source	Training Set				Test Set		
	Avg. Length Target Text	Avg. Sentence Length	Examples	Positive	Avg. Length Target Text	Avg. Sentence Length	Examples
Wikinews	8	168	7746	42%	8	166	1287
News	8	174	14002	40%	8	153	2095
Wikipedia	8	175	5551	45%	9	194	870
Spanish	9	193	13750	40%	9	190	2233

Table 1: English and Spanish datasets

work included four convolutional layers.

Pooling Max pooling was applied to the output of the convolution layer to downsample the feature maps. The feature maps of the last pooling layer were flattened.

Fully-Connected Layer We used three fully-connected layers (FC). The first FC took as input the flattened output of the last pooling layer. The first two FCs used a linear activation function and the third applied the sigmoid activation function. The last FC gave a number in the range $[0, 1]$, which was the final output of the network. By a threshold (we found 0.5 to be optimal), we determined whether the output on a given example implied a label of 0 (simple) or 1 (complex).

4.3 Training

We used the binary cross-entropy as our objective function for training the network. We experimented with various types of optimizers. We chose optimizers with static learning rate and those with adaptive learning rate schedules. Based on the performance of the model on the validation set, we found RMSprop to be the best on updating the network parameters and minimizing the loss function while using 100 epochs.

The dataset is imbalanced: it contains unequal proportion of examples by class labels, roughly 60% negative examples and 40% positive examples. So, we introduced class weights in our training procedure, which resulted in performance improvement. We computed class weights using scikit-learn (Pedregosa et al., 2011).

To mitigate overfitting, we tried several regularization alternatives (Goodfellow et al., 2016) including kernel and weight regularization, batch normalization, dropout, and early stopping. We found dropout and early stopping useful. Our final model included dropout (Srivastava et al., 2014) after every layer with dropout probability of 0.25.

5 Datasets

Table 1 shows some statistics on the corpora we used: the average length of the target text and sentences and the number of examples in the training and test sets, with the percentage of positive examples (target texts labeled as complex) in the training set. The table shows that the datasets are skewed towards negative examples: the percentage of positive examples on the datasets did not exceed 45%. The Wikipedia dataset has the smallest number of training examples, 5551. The average length of target text in the training examples and test examples are comparable. One can see some variations in the average length of sentences in the training and test sets. These variations are remarkable for the Wikipedia and News datasets.

6 Results

This section presents the performance of both models on the English test set and that of the CNN model on the Spanish test set.

Table 2 shows the macro-F1 and accuracy scores as well as the respective ranks of both CNN and TreeE models on the English test set. The performance measures are given per genre in the English test set. Out of 11 teams, our best model places fifth on News; second on Wikinews, and seventh on Wikipedia. All our systems were within 0.01 of the system with the best macro-F1 score on the test sets except on Wikipedia test set. On the Wikipedia test set, our best system was 0.04 below the best macro-F1 score.

On the Spanish test set, we submitted only the CNN-based system. Table 3 shows its macro-precision, macro-recall, macro-F1, and accuracy scores. Our best submission ranks third among seven teams that participated in the Spanish track.

The main advantage of the CNN model is that it can be applied to any language for which an embedding can be easily created given the availability of sufficient electronic textual resources.

Models	News			Wikinews			Wikipedia		
	Macro-F1	Accuracy	Rank	Macro-F1	Accuracy	Rank	Macro-F1	Accuracy	Rank
NLP-CIC-TreeE	0.851	0.859	9	**0.831**	**0.837**	3	0.772	**0.774**	11
NLP-CIC-CNN	**0.855**	**0.863**	8	0.824	0.828	7	0.772	0.772	12

Table 2: Accuracy and macro-F1 scores by genres on the English test set

Model	Macro-Recall	Macro-Precision	Macro-F1	Accuracy	Rank
NLP-CIC-CNN	0.765	0.772	0.767	0.772	3

Table 3: CNN performance scores on the Spanish test set

Source	NLP-CIC-TreeE Model		NLP-CIC-CNN Model	
	Correct	Wrong	Correct	Wrong
Wikinews	0.94 ± 0.53	1.10 ± 0.65	0.94 ± 0.51	1.12 ± 0.72
News	0.97 ± 0.55	1.21 ± 0.75	0.97 ± 0.55	1.17 ± 0.75
Wikipedia	1.05 ± 0.65	1.04 ± 0.68	1.04 ± 0.66	1.08 ± 0.65

Table 4: Target text Normalized character count BY model performance on English test set

7 Discussion

We observed a relationship between the length of the target text—character count—and the performance of our models.

On the News genre dataset of the English test set, our CNN model tends to show better performance on target texts with fewer words compared to Tree Ensembles. When the target text contains more than three words, Tree ensembles perform better than CNN. Similarly, both models tend to make mistakes when the average character count in the target text is higher. Table 4 shows the normalized mean character count of the target text in the English test set when each of our models made correct and wrong predictions.

We believe that this behavior is a reflection of the training examples: there are fewer examples with longer target texts.

8 Conclusion and Future Work

We have described two approaches for the classification subtask of the CWI 2018 shared task: one using feature engineering with Tree Ensembles and one using CNN. We compared them on the test set provided for the CWI 2018 shared task. On the English test set, the two approaches showed comparable performance: the difference between the performance scores was within 0.01. On the English test set, our best model placed fifth on News, second on Wikinews, and seventh on Wikipedia. On the Spanish test set, the CNN model ranked third. This result demonstrates the flexibility of applying CNN to CWI on any language for which pre-trained embeddings are available.

Our models behaved differently depending on the length of the target text: they tend to make mistakes on longer target texts. We attribute this behavior to the skewness of the training set.

In the future, it would be interesting to evaluate the impact of domain-specific features, as well as of different vector operations used to generate context vectors, on the performance of our models.

Acknowledgements

Thanks to the CWI 2018 task organizers. We appreciate the anonymous reviewers for their useful comments on this paper. The last author acknowledges the support of the Mexican government via CONACYT (SNI) and the Instituto Politécnico Nacional grant SIP-20181792.

References

Michael R Berthold, Nicolas Cebron, Fabian Dill, Thomas R Gabriel, Tobias Kötter, Thorsten Meinl, Peter Ohl, Kilian Thiel, and Bernd Wiswedel. 2009. KNIME-the Konstanz information miner: version 2.0 and beyond. *ACM SIGKDD explorations Newsletter*, 11(1):26–31.

Joachim Bingel, Natalie Schluter, and Héctor Martínez Alonso. 2016. CoastalCPH at SemEval-2016 Task 11: The importance of designing your Neural Networks right. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1028–1033, San Diego, California. Association for Computational Linguistics.

Elnaz Davoodi and Leila Kosseim. 2016. CLaC at SemEval-2016 Task 11: Exploring linguistic and psycho-linguistic Features for Complex Word Identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 982–985, San Diego, California. Association for Computational Linguistics.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press, Cambridge, Massachusetts. http://www.deeplearningbook.org.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar.

Michal Konkol. 2016. UWB at SemEval-2016 Task 11: Exploring Features for Complex Word Identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1038–1041, San Diego, California. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Gillin Nat. 2016. Sensible at SemEval-2016 Task 11: Neural Nonsense Mangled in Ensemble Mess. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 963–968, San Diego, California. Association for Computational Linguistics.

Gustavo Paetzold and Lucia Specia. 2016a. Inferring psycholinguistic properties of words. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 435–440, San Diego, California. Association for Computational Linguistics.

Gustavo Paetzold and Lucia Specia. 2016b. SemEval 2016 Task 11: Complex Word Identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 560–569, San Diego, California. Association for Computational Linguistics.

Gustavo Paetzold and Lucia Specia. 2016c. SV000gg at SemEval-2016 Task 11: Heavy Gauge Complex Word Identification with System Voting. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 969–974, San Diego, California. Association for Computational Linguistics.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Aliaksei Severyn and Alessandro Moschitti. 2015. Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 959–962, Santiago, Chile. ACM.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. 2018. A Report on the Complex Word Identification Shared Task 2018. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*, New Orleans, United States. Association for Computational Linguistics.

Seid Muhie Yimam, Sanja Štajner, Martin Riedl, and Chris Biemann. 2017a. CWIG3G2-Complex Word Identification Task across Three Text Genres and Two User Groups. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 401–407, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Seid Muhie Yimam, Sanja Štajner, Martin Riedl, and Chris Biemann. 2017b. Multilingual and cross-lingual complex word identification. In *Proceedings of RANLP*, pages 813–822, Varna, Bulgaria. INCOMA Ltd.

Wenpeng Yin and Hinrich Schütze. 2015a. Convolutional neural network for paraphrase identification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 901–911, Denver, Colorado. Association for Computational Linguistics.

Wenpeng Yin and Hinrich Schütze. 2015b. Multigrancnn: An architecture for general matching of text chunks on multiple levels of granularity. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 63–73, Beijing, China. Association for Computational Linguistics.