NILC at CWI 2018: Exploring Feature Engineering and Feature Learning

Nathan Siegle Hartmann[1,2] **and Leandro Borges dos Santos**[1,2]
[1]Data Science Team, Itaú-Unibanco, São Paulo, Brazil*
[2]Institute of Mathematics and Computer Science, University of São Paulo, São Carlos, Brazil
nathansh@icmc.usp.com.br, leandrobs@usp.br

Abstract

This paper describes the results of NILC team at CWI 2018. We developed solutions following three approaches: (i) a feature engineering method using lexical, n-gram and psycholinguistic features, (ii) a shallow neural network method using only word embeddings, and (iii) a Long Short-Term Memory (LSTM) language model, which is pre-trained on a large text corpus to produce a contextualized word vector. The feature engineering method obtained our best results for the classification task and the LSTM model achieved the best results for the probabilistic classification task. Our results show that deep neural networks are able to perform as well as traditional machine learning methods using manually engineered features for the task of complex word identification in English.

1 Introduction

Research efforts on text simplification have mostly focused on either lexical (Devlin and Tait, 1998; Biran et al., 2011; Glavaš and Štajner, 2015; Paetzold and Specia, 2016b) or syntactic simplification (Siddharthan, 2006; Kauchak, 2013). Lexical simplification involves replacing specific words in order to reduce lexical complexity. Lexical simplification is an open problem, as identifying and simplifying complex words in a given context is not straightforward. Although very intuitive, this is a challenging task since the substitutions must preserve both the original meaning and the grammaticality of the sentence being simplified. Complex word identification is part of the usual lexical simplification pipeline (Paetzold and Specia, 2015), which is illustrated in Figure 1.

For the challenge, we focused on the English monolingual CWI track. We implemented three approaches using machine learning: the first one uses feature engineering; the second one takes the average embedding of target words as input to a neural network; and the third approach models the context of the target words using an LSTM (Gers et al., 1999). Our code is publicly available at github[1].

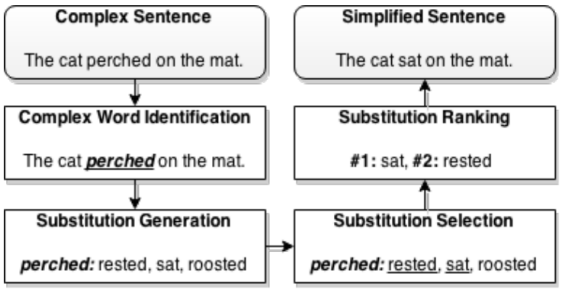

Figure 1: Lexical Simplification pipeline.

2 Task Description

The setup of the CWI Shared Task 2018 is as follows: given a target word (or a chunk of words) in a sentence, predict whether or not a non-native English speaker would be able to understand it. These predictions are based on annotations collected from a mixture of 10 native and 10 non-native speakers. The labels in the binary classification task were assigned "1" if at least one of the 20 annotators did not understand it (complex), and "0" otherwise (simple). The labels in the probabilistic classification task were assigned as the number of annotators who marked the word as difficult divided by the total number of annotators.

In this edition a multilingual dataset was available and participants could choose to participate in one or more of the following tracks: English monolingual CWI, German monolingual

* The opinions expressed in this article are those of the authors and do not necessarily reflect the official policy or position of the Itaú-Unibanco.

[1] https://github.com/nathanshartmann/NILC-at-CWI-2018

Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications, pages 335–340
New Orleans, Louisiana, June 5, 2018.

CWI, Spanish monolingual CWI, Multilingual CWI shared task with French test set. Also, the participants could choose between binary classification or probabilistic classification task. We chose to participate in the English monolingual track and in both classification tasks (see in Table 1 the dataset distribution for the track).

Dataset	Train	Dev	Test
News	14,002	1,764	2,095
WikiNews	7,746	870	1,287
Wikipedia	5,551	694	870
Total	27,299	3,328	4,252

Table 1: CWI 2018 english dataset distribution.

More relevant task description, data and results are available in Yimam et al. (2018).

3 Datasets

In this work, we used two extra corpora to train language models, one of these to train a neural language model:

- BookCorpus dataset: which has 11,038 free books written by yet unpublished authors (Zhu et al., 2015);
- One Billion Word dataset: which is the largest public benchmark for language modeling (Chelba et al., 2013).

4 Proposed Methods

In this section we show our developed methods, following three approaches: feature engineering, feature learning and ensembles.

4.1 Methods based on Feature Engineering

We used linguistic, psycholinguistic and language model features to train several classification and probabilistic classification methods. Our feature set consists of three groups of features:

- LEX: includes word length, number of syllables, number of senses, hypernyms and hyponyms in WordNet (Fellbaum, 1998);
- N-gram: includes log probabilities of an n-gram containing target words in two language models trained on BookCorpus and One Billion Word datasets using SRILM (Stolcke, 2002);
- PSY: contains word-level psycholinguistic features such as familiarity, age of acquisition, concreteness and imagery values for every target word (Paetzold and Specia, 2016a).

Because an instance can contain more than a target word, mean, standard deviation, min and max values were calculated for each feature. A total of 38 features are extracted for each instance. We also normalized features using Z-score.

We trained Linear Regression, Logistic Regression, Decision Trees, Gradient Boosting, Extra Trees, AdaBoost and XGBoost methods for both classification and probabilistic classification tasks.

4.2 Methods based on Feature Learning and Transfer Learning

An alternative approach to feature engineering is to make the machine learning model itself create a data representation. This is the principle of feature learning. In this scenario, all elements of the vector contain an independent value, which has some meaning for the model (LeCun et al., 2015).

Most importantly, we can reuse this representation in another tasks, which is called transfer learning or domain adaptation. This strategy is already used with success in Computer Vision, where deep neural networks are pre-trained in large supervised training sets like ImageNet (Girshick et al., 2014; Esteva et al., 2017).

It is common in Natural Language Processing (NLP) tasks to use pre-trained word embeddings with models like Word2Vec (Mikolov et al., 2013) or GloVe (Pennington et al., 2014). However, more recently some studies have used distributed sentences to produce contextualized embeddings, from a language model, machine translation model, or auto-encoder (Dai and Le, 2015; Kiros et al., 2015; Yuan et al., 2016; Le et al., 2017; Peters et al., 2017, 2018; McCann et al., 2017; Howard and Ruder, 2018).

In the next section we will explain how we used both strategies.

4.2.1 Average Embedding Method

Word embedding is a technique to represent words into dense real vectors, that helps NLP tasks and improves neural networks models (Collobert et al., 2011; Kim, 2014; Bowman et al., 2015), because this dense representation captures semantic and morphological information of the words. In this work, we obtained word vector representations for

complex words. When a complex word was a chunk of words, we took the average of their vectors. We used word vectors from GloVe (6B tokens (Pennington et al., 2014)).

The resulting vector was passed on to a neural network with two ReLU layers (Nair and Hinton, 2010) followed by a Sigmoid layer, which predicted the probability of whether or not the word was complex (Figure 2).

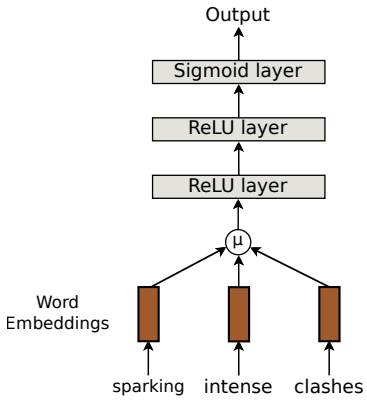

Figure 2: Example of average embedding method processing target words.

4.2.2 LSTM Method

LSTM is a powerful tool for modeling sequential data. This type of neural network architecture can learn to map an input sentence of variable length into a fixed-dimensional vector representation. For this reason, a lot of state-of-the-art systems in several NLP tasks incorporate an LSTM, for example, language modeling (Jozefowicz et al., 2016; Melis et al., 2017), machine translation (Di Gangi et al., 2017), textual inference (Tay et al., 2017), and others.

Some studies used a pre-trained LSTM language model (Dai and Le, 2015; Yuan et al., 2016; Le et al., 2017; Peters et al., 2017, 2018) to represent a sentence/document and used this representation to improve their results.

Therefore, we trained a language model in the One Billion Word dataset using similar parameters from Le et al. (2017): one-layer LSTM with 512 units, 128 embedding size, and sampled softmax loss (Jean et al., 2015). However, we used weight tying, which means the weights between the embedding and softmax layer are shared, consequently reducing the total parameters of the model (Melis et al., 2017). For the CWI task, the LSTM read five words before the complex word, then the complex word itself (or the chunk of words). We took the last hidden vector from the LSTM and passed it through a Sigmoid layer.

In Figure 3 we show the pipeline where the blue boxes represent the context words and red boxes represents the complex word, which is a chunk in this example.

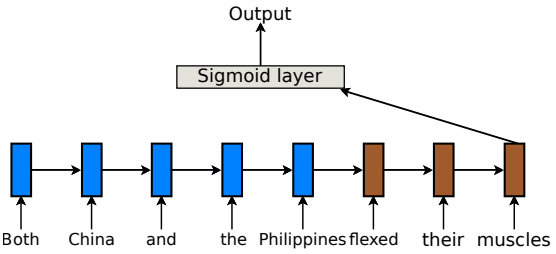

Figure 3: Example of a LSTM processing target words.

4.2.3 Sent2Vec

We also used sentence embeddings generated by Skip-thought (Kiros et al., 2015). This model produces a sentence representation of 2,400 dimensions. We trained two models using sentence embedding. In the first, we passed the embedding through a sigmoid layer and in the second, we used two layers with ReLUs of 1,200 and 600 dimensions respectively, followed by a Sigmoid layer. In the last model we employed a dropout layer (0.7) between all of the layers. Both models obtained good results in the training set, however, the models had poor results in the development set.

4.3 Ensembles

We combined our three best systems: Feature Engineering, MLP Average Embedding and LSTM Transfer Learning. For the binary classification task, we combined the system by majority voting rule. For the probabilistic classification task we used stacking with Linear Regression as a base learner, which took the probabilities from our three best system as features.

5 Results

For the binary classification task, we first evaluated the ROC-AUC in the development set for all methods. For the Feature Engineering method, we decided to use an XGBoost classifier which achieved the best AUC in development set (0.91). Although we selected the threshold which maximizes the F1 in the training set for each Feature Engineering, Shallow and Deep Neural network method, it is important to mention that these

	News			WikiNews			Wikipedia		
	F1	#Subm.	#Author	F1	#Subm.	#Author	F1	#Subm.	#Author
XGBoost Linguistics	0.8606	9th		**0.8277**	7th	3rd	0.7918	4th	
MLP Avg. Embeddings	0.8467	15th		0.7977	16th		0.7360	26th	
LSTM Transfer Learning	0.8173	27th		0.7961	17th		0.7528	20th	
Voting	**0.8636**	5th	4th	0.8270	8th		**0.7965**	2nd	2nd
Best competition results	0.8736			0.8400			0.8115		

Table 2: F1 (macro) for English monolingual classification task.

	News			WikiNews			Wikipedia		
	MAE	#Subm.	#Author	MAE	#Subm.	#Author	MAE	#Subm.	#Author
XGBoost Linguistics	0.2978	14th		0.3203	15th		0.3819	7th	
MLP Avg. Embeddings	0.2958	13th		0.3240	16th		0.3578	7th	
LSTM Transfer Learning	**0.0588**	7th	4th	0.0742	7th		0.0822	7th	
Stacking	0.0590	8th		**0.0733**	6th	4th	**0.0819**	6th	3rd
Best competition results	0.0510			0.0674			0.0739		

Table 3: MAE for English monolingual probabilistic classification task.

thresholds were found for the whole training set and not for each subset. This guarantees that we are not overfitting our method to test data or to a specific dataset. Our results for the English monolingual classification task are described in Table 2. The Feature Engineering method itself achieved by far our best results for the three test sets. In order to achieve better results, we submitted a fourth system which calculated the majority voting of our three methods. This voting system surpassed our individual methods in two test sets, but was inferior compared to the Feature Engineering method by less than 1^{-3} F1 in the WikiNews dataset. Majority voting was our best method for the classification task.

For the probabilistic classification task, our Feature Engineering method used also an XGBoost classifier which achieved the best MAE in development set (0.28). Our results for English monolingual probabilistic classification task are described in Table 3. While both Feature Engineering and Average Embedding did not perform well, our best individual system by a large margin was the LSTM method. In order to achieve better results, we used stacking of our three models. The stacking performed better than individual methods in two datasets, but was not better than LSTM for the News test set (2^{-4} gap).

6 Conclusion

For the binary classification task, majority voting achieved our best results, although only slightly better than the standalone Feature Engineering model.

For the probabilistic classification task, LSTM had better results in one data set, but the stacking method performed slightly better in the other data sets. The deep learning method showed its potential when contrasted with the feature engineering method.

In the future, we intend to explore more powerful neural language models, such as encoding characters embeddings (Jozefowicz et al., 2016), bidirectional language model (Peters et al., 2017, 2018), and other transfer learning methods (Howard and Ruder, 2018).

Acknowledgments

This work was supported by CAPES, CNPq and Google Research Awards in Latin America. We would like to thank NVIDIA for their donation of Titan X. Research carried out using the computational resources of the Center for Mathematical Sciences Applied to Industry (CeMEAI) funded by FAPESP (grant 2013/07375-0).

References

Or Biran, Samuel Brody, and Noémie Elhadad. 2011. Putting it simply: a context-aware ap-

proach to lexical simplification. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2, pages 496–501. Association for Computational Linguistics.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. arXiv preprint arXiv:1508.05326.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. Technical report, Google.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. Journal of Machine Learning Research, 12(Aug):2493–2537.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In Advances in Neural Information Processing Systems, pages 3079–3087.

Siobhan Devlin and John Tait. 1998. The use of a psycholinguistic database in the simplification of text for aphasic readers. Linguistic databases, pages 161–173.

Mattia Antonino Di Gangi, Nicola Bertoldi, and Marcello Federico. 2017. Fbk's participation to the english-to-german news translation task of wmt 2017. In Proceedings of the Second Conference on Machine Translation, pages 271–275.

Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. 2017. Dermatologist-level classification of skin cancer with deep neural networks. Nature, 542(7639):115.

Christiane Fellbaum. 1998. WordNet. Wiley Online Library.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 1999. Learning to forget: Continual prediction with lstm.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, pages 580–587. IEEE Computer Society.

Goran Glavaš and Sanja Štajner. 2015. Simplifying Lexical Simplification: Do We Need Simplified Corpora? In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP-2015), volume 2, pages 63–68.

Jeremy Howard and Sebastian Ruder. 2018. Fine-tuned language models for text classification. arXiv preprint arXiv:1801.06146.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. 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), volume 1, pages 1–10.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. arXiv preprint arXiv:1602.02410.

David Kauchak. 2013. Improving Text Simplification Language Modeling Using Unsimplified Text Data. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL-2013), pages 1537–1546.

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.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In Advances in neural information processing systems, pages 3294–3302.

Minh Le, Marten Postma, and Jacopo Urbani. 2017. Word sense disambiguation with lstm: Do we really need 100 billion words? arXiv preprint arXiv:1712.03376.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. nature, 521(7553):436.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In Advances in Neural Information Processing Systems, pages 6297–6308.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2017. On the state of the art of evaluation in neural language models. arXiv preprint arXiv:1707.05589.

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

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on International Conference on Machine Learning, pages 807–814. Omnipress.

Gustavo Paetzold and Lucia Specia. 2015. Lexenstein: A framework for lexical simplification. Proceedings of ACL-IJCNLP 2015 System Demonstrations, pages 85–90.

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.

Gustavo Henrique Paetzold and Lucia Specia. 2016b. Benchmarking lexical simplification systems. Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC-2016), pages 3074–3080.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pages 1532–1543.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), volume 1, pages 1756–1765.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. arXiv preprint arXiv:1802.05365.

Advaith Siddharthan. 2006. Syntactic simplification and text cohesion. Ph.D. thesis, University of Cambridge, Inglaterra.

Andreas Stolcke. 2002. Srilm-an extensible language modeling toolkit. In Seventh international conference on spoken language processing.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2017. A compare-propagate architecture with alignment factorization for natural language inference. arXiv preprint arXiv:1801.00102.

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.

Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, and Eric Altendorf. 2016. Semi-supervised word sense disambiguation with neural models. In Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1374–1385.

Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. arXiv preprint arXiv:1506.06724.