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

Introduction

- **Lexical simplification** involves replacing specific words in order to reduce lexical complexity. **Complex word identification** is part of the usual lexical simplification pipeline
- For the challenge, we focused on the English monolingual CWI track
- We implemented three approaches using machine learning:
  1. **Feature engineering**
  2. **Average embedding** of target words as input to a neural network
  3. **LSTM** to model the context of the target words

Methods: Feature Engineering

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

- **LEX**: includes word length, number of syllables, number of senses, hypernyms and hyponyms in WordNet
- **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
- **PSY**: contains word-level psycholinguistic features such as familiarity, age of acquisition, concreteness and imagery values for every target word [4]

Methods: Average Embedding

- 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)
- The resulting vector was passed on to a neural network with two ReLU layers followed by a Sigmoid layer, which predicted the probability of whether or not the word was complex

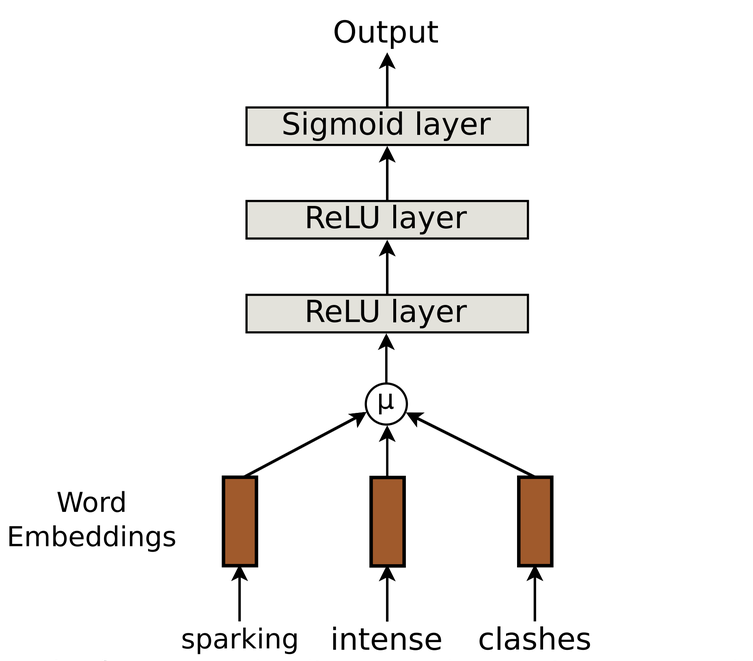

Figure 1. Example of average embedding method processing target words

Methods: LSTM

- We trained a language model in the One Billion Word dataset using similar parameters from [3]: one-layer LSTM with 512 units, 128 embedding size, and sampled softmax loss
- We used weight tying, which means the weights between the embedding and softmax layer are shared, consequently reducing the total parameters of the model
- 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

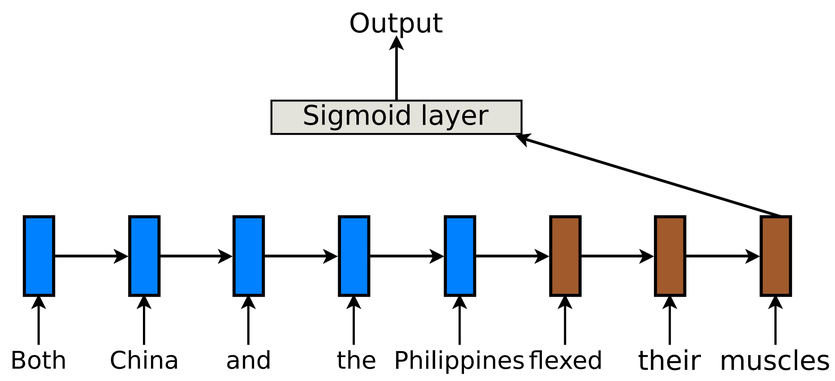

Figure 2. Example of an LSTM processing target words

Methods: Ensemble/Stacking

- **Binary classification**: We combined the systems by majority voting rule
- **Probabilistic classification**: We used stacking with Linear Regression as a base learner, which took the probabilities from our three system as features

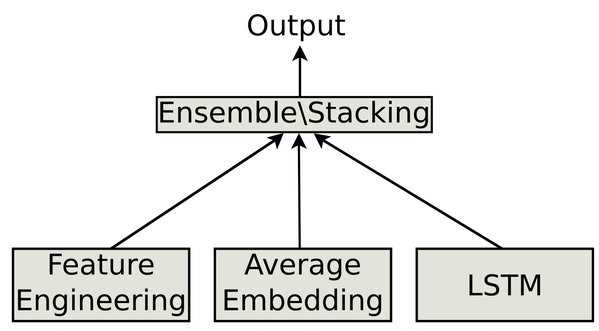

Figure 3. Example of ensemble methods

Datasets

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

- **BookCorpus dataset**: It has 11,038 free books written by yet unpublished authors
- **One Billion Word dataset**: It is the largest public benchmark for language modeling

Results

	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 1. 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 2. MAE for English monolingual probabilistic classification task

- For binary classification, thresholds which maximize F1 in the whole training set were used

Conclusions and Future Work

- **Binary classification:** majority voting achieved our best results, although only slightly better than the Feature Engineering model
- **Probabilistic classification:** 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
- **Future work:** to explore more powerful neural language models, such as encoding characters embeddings [2], bidirectional language model [5], and other transfer learning methods [1]

References

[1] Jeremy Howard and Sebastian Ruder.
Universal Language Model Fine-tuning for Text Classification.
arXiv preprint arXiv:1801.06146, 2018.

[2] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu.
Exploring the Limits of Language Modeling.
arXiv preprint arXiv:1602.02410, 2016.

[3] Minh Le, Marten Postma, and Jacopo Urbani.
Word Sense Disambiguation with LSTM: Do We Really Need 100 Billion Words?
arXiv preprint arXiv:1712.03376, 2017.

[4] Gustavo Paetzold and Lucia Specia.
Inferring Psycholinguistic Properties of Words.
In *Proceedings of the 2016 NAACL HLT*, pages 435–440, 2016.

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