Complex Word Identification

Mario Alejandro Hevia Fajardo
Department of Computer Science
University of Sheffield, UK
maheviafajardo1@sheffield.ac.uk

1 Introduction

The task of this class project is to identify whether a word is considered complicated or not, especially to a group of people with difficulty in reading the language such as non-native readers or impaired readers. (Stajner et al., 2018)

This task is important because it is the start of the pipeline in a lexical simplification algorithm (Paetzold and Specia, 2016a), and since it is the first step it can affect greatly the outcome of the whole algorithm. Moreover the lexical simplification algorithm can be used to help certain population to have better reading comprehension.

2 Baseline system

2.1 Motivation

The idea behind the baseline is to use embeddings of words to represent them and make several standard classification algorithms learn what is a "difficult word" with these embeddings as features. This would give a certain semantic idea of what a "difficult word" is, without necessarilly defining the features.

This idea is suitable for two reasons; since the task is relatively new, there are no defined "best features" in the literature and as it has been mentioned before, the embeddings bring to the scene a semantic way what a word is. The second reason is the amount of training data given, it could be considered as small therefore any way to increment the amount of data is benefitial.

2.2 Description

Since the system uses embeddings of words as features, pretrained embeddings were used. For English the pre-trained word vector model from Google News corpus was used (Mikolov, 2013), and for Spanish the pre-trained word vector model from Spanish Billion Word Corpus and Embeddings was used (Cardellino, 2016), they are composed of 3 million 300-dimension English word vectors and 1 million 300-dimension Spanish word vectors respectively.

Since the embeddings need a word seen in the training a preprocessing of the target words were made, in the preprocessing the following symbols were taken away. *(Some symbols are not the exact ones taken away but representations and one was not represented here)*

Symbols: . , ' " ( ) ≪ ≫

Another assumption made was that if a word wasn't in the embeddings it was a difficult word.

Using the word vectors of each preprocessed word ten classifiers were trained. Nine of these classifiers were taken from the scikit classification example (Varoquaux et al., 2016) without altering the arguments of the classifiers, and the last one was a logistic regression. A list of the ten algorithms is below.

1. Nearest Neighbors
2. Linear SVM
3. RBF SVM
4. Decision Tree
5. Random Forest
6. AdaBoost
7. Logistic Regression
8. Naive Bayes
9. QDA
10. Neural Net

For the training some of the target words were sentences therefore every word inside them was considered difficult and in the prediction if any of the words was considered difficult the whole sentence was considered difficult. After training and testing them on the developer set with all of the algorithms the best one was chosen as a baseline.

3 Improved system

3.1 Motivation

As an improved system a mix of the conventional features used in the SemEval 2016 Task 11 (Paetzold and Specia, 2016b) and the baseline was used. The motivation for this is that a combination of results from different types of features could become better than using only one of them.

3.2 Description

A methodology similar to the one used in the baseline was used here. Given a set of features the ten algorithms were tested and the best ones with the development set were used to combine with the baseline for the final test. The description of the features will be given first and at the end the algorithm for combining the results of the two types of features will be given.

Some of the features used were copied from two of the participants from SemEval 2016 Task 11 (Francesco et al., 2016) (Paetzold and Specia, 2016c) and the other ones were implied using common sense and reading some examples from the training set. The same preprocessing was used and some features used special preprocessing. In the end the following 18 features were used.

1. Number of characters: Count the number of characters in the target word.
2. Number of tokens: Count the number of words in the target word.
3. Low frecuency characters: Adding the values of letters with $< 1\%$ relative frecuency (RF). ($val = \frac{1-RF}{Numberofcharacters}$) (Letter frequency, 2018)
4. High frecuency characters: Adding the values for (5) letters with highest relative frecuancy (HF). ($val = \frac{HF/10}{Numberofcharacters}$) (Letter frequency, 2018)
5. Number of syllables: Count the number of syllables divided by the number of tokens.
6. Number of vowels: Count the number of vowels divided by the number of tokens.
7. Number of consonants: Count the number of consonants divided by the number of tokens.
8. Number of multiple vowels: Count the number of groups of three or more vowels divided by the number of tokens.
9. Number of multiple consonants: Count the number of groups of three or more consonants divided by the number of tokens.
10. Number of double characters: Count the number of characters repeated two or more times divided by the number of tokens.
11. Number of transitions: Count the number of transitions between vowels and consonants and viceversa divided by the number of characters.
12. Target frequency: Word frequency in the Leipzig Corpora Collection (Goldhahn et al., 2012) divided by the number of tokens. Only words with more than 50 word frequency were used, for all others the value was 0.05 ($val = \frac{(sigmoid(\frac{word\_count}{1000})-0.5)*2}{Numberofcharacters}$)
13. Number of senses: Number of senses in WordNet (Princeton University, 2010) divided by the number of tokens.
14. Number of synonyms: Number of synonyms in WordNet (Princeton University, 2010) divided by the number of tokens.
15. Number of hypernyms: Number of hypernyms in WordNet (Princeton University, 2010) divided by the number of tokens.
16. Number of hyponyms: Number of hyponyms in WordNet (Princeton University, 2010) divided by the number of tokens.
17. Number of POS: Number of POS in WordNet (Princeton University, 2010) divided by the number of tokens.
18. Length of the definition: Length of the definition in WordNet (Princeton University, 2010) divided by the number of tokens.

To combine the algorithms 5 of them were used the best two from the baseline and the best three from the previous part. They were chosen different for each language and the reasoning behind that is that there were too much differences between the algorithms used in every language. The final algorithms used were:

English:

- QDA with w2v
- Neural Net with w2v
- Nearest Neighbors
- RBF SVM
- Decision Tree

Spanish:

- QDA with w2v
- Neural Net with w2v
- AdaBoost
- Logistic Regression
- Neural Net

To combine them the prediction of all of them was made (1 or 0), and then the values were averaged and rounded to decide the final prediction. The number of algorithms used was needed to be odd since the algorithms from each type of features (word vectors and normal features) made similar predictions. If they were even when the algorithms wouldn't agree almost always the average prediction would be 0.5 and rounded always up to 1. Finally the algorithms with normal features were chosen to be more since they were better at predictions giving them more weight.

4 Experiments on development set

4.1 Baseline

When experimenting on the development set for both languages the baseline worked better than a logistic regression with two features (Number of characters and tokens) and that was what was expected. And another interesting thing is that the neural network was the best classifier for both of them. You could say that the reason for this is that the embeddings are created with a neural network therefore it is well adjusted to work with one, but there need to be more testing to get to that conclusion. The Scores for both are below:

English:

Macro-F1 for Neural Net with w2v: 0.7688

Spanish:

Macro-F1 for Neural Net with w2v: 0.7145

4.2 Improved

With the improved algorithm two things happened. First, the hard voting between the five algorithms had a better score than any of the algorithms alone, as expected. And second, the algorithms using specified features was better than the word-vectors; eventhough this wasn't expected it wasn't a surprise either, since generally we as humans have an understanding of what makes a word difficult and we can encode that in good features. The scores for the five algorithms and the final hard-voting are given below for both languages.

English:

- Macro-F1 for Nearest Neighbors: 0.7915
- Macro-F1 for RBF SVM: 0.8227
- Macro-F1 for Decision Tree: 0.7819
- Macro-F1 for QDA with w2v: 0.7584
- Macro-F1 for Neural Net with w2v: 0.7688
- Macro-F1 for hard voting with all models: 0.8328

Spanish:

- Macro-F1 for AdaBoost: 0.7538
- Macro-F1 for Logistic Regression: 0.7529
- Macro-F1 for Neural Net: 0.7499
- Macro-F1 for QDA with w2v: 0.7033
- Macro-F1 for Neural Net with w2v: 0.7145
- Macro-F1 for hard voting with all models: 0.7722

Within the development data a comparison with the target words that were wrong predicted between the two models was made, and we find that the ratio between words predicted wrong by the baseline and correct by the improved against the words predicted wrong by the improved and correct by the baseline is around 2 for both languages giving us a clear sign that it has improved a lot eventhough there is still a lot of room improvement. Some examples were the baseline predicted incorrectly but the improved didn't were:

English	Spanish
control	rusa
faced	mago
similar	copa
loans	rayas
angry	lazo

In all the examples shown the baseline gave a false positiv i.e. saying they were difficult when their labels were easy. I have chosen these examples because they ejemplify clearly the problem with the baseline.

We can see that these words are small and simple words but since the baseline uses a context approach to the problem some of them might be similar to more complex words giving this false positive; furthermore the improved system implements simple features that help the baseline differentiate between these words and other that are complex.

5 Final test

When checking on the test set the same results as in the devset were seen, indicating that the model really works for unseen data and was not just biased to best fit the development set. The final results are below.

English:

- Macro-F1 for Neural Net with w2v: 0.7795
- Macro-F1 for hard voting with all models: 0.8388

Spanish:

- Macro-F1 for Neural Net with w2v: 0.7465
- Macro-F1 for hard voting with all models: 0.7730

After checking on the test set, several tests were made with the same test set on the reliability of the systems with less training data. The amount of training data and the result of these test is shown below.

English:

Data %	*Baseline*	*Improved*
100	0.7795	0.8388
90	0.7788	0.8393
80	0.7779	0.8356
70	0.7741	0.8265
60	0.7687	0.8260
50	0.7608	0.8133
40	0.7633	0.8122
30	0.7686	0.8100
20	0.7604	0.8039
10	0.7537	0.7717
5	0.7075	0.6446
1	0.6674	0.6296

Spanish:

Data %	*Baseline*	*Improved*
100	0.7465	0.7730
90	0.7363	0.7657
80	0.7487	0.7699
70	0.7393	0.7666
60	0.7461	0.7628
50	0.7404	0.7544
40	0.7400	0.7650
30	0.7303	0.7596
20	0.6972	0.7646
10	0.6585	0.7519
5	0.6479	0.7241
1	0.6204	0.7050

We can see that the differences between the Spanish language and English language with small training data for the improved system are great, and this is because of the models used in the hard-voting in english which is RBF SVM which needs a certain amount of training data to work well, whilst the algorithms used in Spanish are not that vulnerable to this problem.

6 Possible improvements

There are several possible improvements to the system above, the simplest one would be modifying the parameters in the algorithms since there were parameters from a sci-kit tutorial therefore there is more room of improvement there.

Another improvement could come with the features used in the improvement, since I notice some of them seemed to decrease the efectiveness of the algorithms; because of this a possible improvement could come verifying these features and their contribution to the outcome.

Another improvement could be to use more and different features, on the top of my mind some features are the next three. Use n-gram features but instead of using words using characters, using a translation algorithm such as used by langdetect and using the amount of languages that it detect as a feature and using lexical features.

At last the voting could be changed into a soft-voting were each algorithm gets a weight based on their F1-score on the training data.

7 Conclusions

From the experiments I learnt that most of the time is better to relay on several algorithms voting than on only one, even when using the same features, the only limitation I found is that the F1-score from all of them need to be similar in order to see the most improvement.

Further examination on the features used is needed to improve the algorithms, some ideas are given above but even with them we need to extend our knowledge of what makes a word difficult for certain groups of people.

In the end what I think could help a lot is having a better anotated and better quality datasets, since I found a lot of words in the three datasets that were words from other languages and even in other alphabets.

References

Cristian Cardellino. 2016. Spanish billion word corpus and embeddings.

Ronzano Francesco, Ahmed Aburaed, Luis Espinosa-Anke, and Horacio Saggion. 2016. Taln at semeval-2016 task 11: Modelling complex words by contextual, lexical and semantic features.

D. Goldhahn, T. Eckart, and U. Quasthoff. 2012. Building large monolingual dictionaries at the leipzig corpora collection. In *Proceedings of the 8th International Language Ressources and Evaluation (LREC'12)*.

Letter frequency. 2018. Letter frequency.

Tomas Mikolov. 2013. word2vec.

Gustavo Paetzold and Lucia Specia. 2016a. Semeval-2016 task 11.

Gustavo Paetzold and Lucia Specia. 2016b. Semeval 2016 task 11: Complex word identification.

Gustavo Paetzold and Lucia Specia. 2016c. Sv000gg at semeval-2016 task 11: Heavy gauge complex word identification with system voting.

Princeton University. 2010. "about wordnet." wordnet.

Sanja Stajner, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Lucia Specia, Anais Tack, Seid Muhie Yimam, and Marcos Zampieri. 2018. Complex word identification (cwi) shared task 2018.

Gael Varoquaux, Andreas Muller, and Jaques Grobler. 2016. Classifier comparison.