Lexical Text Simplification

Paula Gombar, Ivan Katanić

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{paula.gombar, ivan.katanic}@fer.hr

Abstract

In this paper, we consider the task of lexical text simplification by building a system to replace complex and hard-to-comprehend words with less complex semantically-matching words. We use several different approaches, combining both supervised and unsupervised methods relying on word vector representations, regular and simplified corpora, context-dependent and context-independent features. One supervised approach makes use of ranking SVM and different kernel functions, while the other obtains the best linear combination of features. Unsupervised approaches make use of linguistic features and external lexico-semantic resources such as WordNet. The system was tuned, trained and evaluated on the SemEval-2012 Task 1 dataset and outperforms two of three given baselines.

1. Introduction

Lexical simplification is the task of finding less complex semantically-appropriate words or phrases and replacing those that are difficult to comprehend, especially for certain groups of people such as non-native speakers, people with low literacy or intellectual disabilities, and language-impaired people.

Common steps in a pipeline for a lexical simplification system include the following:

1. complexity analysis: finding out which words or phrases are considered complex,
2. substitute lookup: retrieving adequate replacements, simpler than the original word or phrase, from a thesaurus or lexico-semantic resources,
3. context-based ranking: ranking of substitutes to produce the final replacement.

To give an example, let us consider the following sentence: "The incisions will feel constricted for the first 24-48 hours." The system would first identify complex words, e.g. *constricted*, then search for substitutes that might adequately replace it. Suppose the retrieved candidates were "uncomfortable", "tight", "stretched", "compressed" and "constricted". Finally, the system would score each of the candidates on simplicity and context-adequacy, rank them and determine the simplest one, e.g. "tight", and use it to replace the complex word, yielding the sentence "The incisions will feel tight for the first 24-48 hours."

This paper mainly focuses on the third step, context-based ranking, as complexity analysis and substitute lookup have already been given in the SemEval-2012 Task 1 resources this paper is based on. The definition of SemEval-2012 Task 1 is, given a short context, a target word in English, and several substitutes for the target word that are deemed adequate for that context, rank these substitutes according to how "simple" they are, allowing ties.

We decided to use several different approaches to solving this task, mixing both supervised and unsupervised methods, using external resources such as word vector representations, Simple Wikipedia, WordNet, linguistic features, as well as context-dependent and context-independent features.

2. Related work

There have been multiple different approaches to lexical text simplification. Before simplified corpora, the approach was based on substituting longer words with shorter and more frequent alternatives, such as in (Carroll et al., 1998; De Belder and Moens, 2010) and referred to lexico-semantic resources like WordNet (Fellbaum, 1998).

After the emergence of simplified corpora, most notably Simple Wikipedia, the focus was shifted to using this type of resource, either using context-aware unsupervised methods (Biran et al., 2011), or supervised methods to learn substitutions from the sentence-aligned corpora (Horn et al., 2014).

Recent advantages in word vector representations (Pennington et al., 2014) have paved the path for unsupervised approaches that do not use simplified corpora, but regular corpora and vector representations (Glavaš and Štajner, 2015).

3. Dataset description

The dataset used in SemEval-2012 Task 1 originates from SemEval-2007 and the Lexical Substitution task. It is extracted from the English Internet Corpus of English (Sharoff, 2006), and is a selection of sentences, or contexts. In total there are 2010 contexts which are divided into Trial and Test sets, consisting of 300 and 1710 contexts respectively. The datasets cover a total of 201 (mostly polysemous) target words, including nouns, verbs, adjectives and adverbs, and each of the target words is shown in 10 different contexts.

Given the list of contexts and each respective list of substitutes, the annotators ranked substitutes for each individual context in ascending order of complexity. The Trial dataset was annotated by four people while the Test dataset was annotated by five people. In both cases each annotator tagged the complete dataset, and the inter-annotator agreement was computed using the kappa index with pairwise rank comparisons, the same metric being used for system evaluation. On the Trial dataset, a kappa index of 0.386 was

found, while for the Test dataset, a kappa index of 0.398 was found. Regarding the low kappa scores, the highly subjective nature of the annotation task must be taken into account. It is also worth noticing that this agreement metric is highly sensitive to small differences in annotation, thus leading to overly pessimistic scores.

Finally, here is an example of a sentence, or context, from the Trial dataset, as well as the gold-standard ranking of substitutions.

Sentence: "The Governor *took* the big sheets of imitation parchment, glanced over them, signed his name to each laid down the pen, and handed the papers across the table to Dreier."

Gold rankings: {get} {take} {pick up} {collect, grasp} {gather}.

4. Feature extraction

The role of the features is to capture information about the applicability of the word in the context of the sentence as well as the simplicity of the word. We rank the simplification candidates according to several features, both context-dependent and context-independent.

Inverse word length. It is a valid assumption that a word is more complex if it longer than another candidate. We use the inverse of the candidate's length, because in our system a higher score means the word or phrase is simpler.

Number of synsets in WordNet. WordNet[1] is a lexico-semantic resource that groups together words based on their meanings, or synsets. The number of candidate's synsets correlates with interchangeability in several contexts, so a candidate with a higher number of synsets is considered to be simpler.

Frequency in Simple Wikipedia. We obtained a list of word frequencies from an up-to-date Simple Wikipedia dump[2]. Simple Wikipedia is primarily written using basic English, making it a useful resource for this task.

Frequency in Wikipedia. We obtained a list of word frequencies from an English Wikipedia dump[3] from 2014. The assumption is that an often-used word must be simpler than words that are not so often used.

Corpus complexity. As seen in (Biran et al., 2011), corpus complexity is defined as the ratio of a candidate's frequency in Wikipedia and Simple Wikipedia:

$$C_w = \frac{f_{w,English}}{f_{w,Simple}}$$

where $f_{w,c}$ is the frequency of candidate w in corpus c. We use the inverse score, because we rank higher the simpler candidates.

Context similarity. As seen in (Glavaš and Štajner, 2015), the idea is that the simplification candidates that are synonyms of the correct sense of the original word should be more semantically similar to the context of the original word. This feature is obtained by computing the semantic similarity of the simplification candidate and each content word from the original context. The semantic similarity of two words is computed as the cosine of the angle between their corresponding GloVe[4] vectors. In all experiments, we used 200-dimensional GloVe vectors pretrained on the English Wikipedia corpus. Formally defined:

$$csim(w, c) = \sum_{w' \in C(w)} cos(\mathbf{v_w}, \mathbf{v_{w'}})$$

where $C(w)$ is the set of context words of the original word w and $\mathbf{v_w}$ is the GloVe vector of the word w.

Semantic similarity. Similar to context similarity, this feature is obtained by computing the semantic similarity of the original word marked for replacement and the simplification candidate:

$$ssim(w, c) = cos(\mathbf{v_w}, \mathbf{v_{w'}})$$

where $\mathbf{v_w}$ is the GloVe vector of the original word w, and $\mathbf{v'_w}$ is the GloVe vector of the replacement candidate.

5. Methods used

We used a total of four different methods, three supervised and one unsupervised. When using a supervised approach, we optimized the algorithm's parameters using grid search and cross-validation. Cross-validation is performed by splitting the Trial dataset into a training set and validation set, the ratio being 70:30. The systems are then evaluated on the Test dataset.

We initially used SVM^{rank} described in (Joachims, 2006), but found the implementation too rigid to perform hyperparameter optimization. We ended up implementing our own version of a ranking SVM algorithm, with the help of SVM implementation in scikit-learn (Pedregosa et al., 2011).

Ranking SVM with RBF kernel. Grid search and cross-validation produced the optimal hyperparameters shown in Table 1. The scaler parameter defines the method used to scale the feature array. Standard scaler first translates the data so the mean value is zero, then scales is so that the standard deviation is one, MinMax scaler scales and translates each feature between zero and one, and the Imputer is usually used to replace missing values, but in our case it is used as an identity transformation. PolynomialFeatures is used to add complexity to the model by considering nonlinear features. If the degree is 2, and we have features x_1 and x_2, the resulting feature array is $(1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$.

Ranking SVM with linear kernel. Similar to the previous method, the only difference is the kernel used in the SVM algorithm. Grid search and cross-validation produced the optimal hyperparameters shown in Table 2. What is interesting is that the linear kernel outputs the coefficients belonging to each of the features. The following optimal linear coefficients were obtained: $weights = (-0.355, -0.109, -0.296, 0.273, -0.041, -0.835, 0.004)$, whereas the order of features is as described beforehand.

Linear combination of features. We used grid search to find the optimal combination of linear coefficients, similar

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

[2] https://dumps.wikimedia.org/simplewiki/

[3] https://dumps.wikimedia.org/enwiki/

[4] http://www-nlp.stanford.edu/data/glove.6B.200d.txt.gz

Table 1: Optimal hyperparameters for Ranking SVM with RBF kernel.

Hyperparameter	Optimal value	Possible values
Scaler	Standard	StandardScaler, MinMaxScaler, Imputer
PolynomialFeatures degree	1	1, 2
C	2^5	$[2^{-15}, ..., 2^8]$
γ	0.00098	$[2^{-15}, ..., 2^8]$

Table 2: Optimal hyperparameters for Ranking SVM with linear kernel.

Hyperparameter	Optimal value	Possible values
Scaler	Standard	StandardScaler, MinMaxScaler, Imputer
PolynomialFeatures degree	1	1, 2
C	2^4	$[2^{-15}, ..., 2^8]$

to the output of ranking SVM with linear kernel. The optimal hyperparameters can be seen in Table 3. Additionally, the optimal scaler used is MinMax scaler.

Unsupervised approach. The unsupervised approach was more popular in SemEval-2012, eight out of twelve teams using it instead of a supervised one, as well as the winning team. The reason might be due to the limited number of training examples given. In this approach, we first scale the data using the MinMax scaler and then declare all coefficients as 1, so the resulting function is equivalent to obtaining the average of all features. We do not use cross validation in this approach, so after tuning the model on the Trial dataset, we evaluate it on the Test dataset.

6. Evaluation

Evaluation for all methods was done on the Test dataset, consisting of 1710 contexts. In the SemEval-2012 task description, three baselines were provided.

L-Sub Gold: This baseline uses the gold-standard annotations from the Lexical Substitution corpus of SemEval-2007 as is. In other words, the ranking is based on the goodness of fit of substitutes for a context, as judged by human annotators. This method also serves to show that the Lexical Substitution and Lexical Simplification tasks are indeed different.

Random: This baseline provides a randomized order of the substitutes for every context. The process of randomization is such that is allows the occurrence of ties.

Simple Freq.: This simple frequency baseline uses the frequency of the substitutes as extracted from the Google Web 1T Corpus (Brants and Franz, 2006) to rank candidate substitutes within each context.

The evaluation metric used is the same measure used for inter-annotator agreement, the metric based on the kappa index (Kohen, 1960). It is used for both contrasting two human annotators and contrasting a system output to the average of human annotations that together forms the gold-standard and is defined as the following:

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)}$$

where $P(A)$ denotes the proportion of times the system output corresponds to the gold-standard, and $P(E)$ denotes the probability of agreement by chance between the two. Kappa is calculated for every pair of ranked items for a given context, and then averaged to get an overall kappa score:

$$\kappa = \frac{\sum_{n=1}^{|N|} \frac{P_n(A) - P_n(E)}{1 - P_n a(E)}}{|N|}$$

where N is the total number of contexts, and $P_n(A)$ and $P_n(E)$ are calculated based on counts extracted from the data on the particular context n.

Table 4: Baseline kappa scores on Trial and Test datasets.

	Trial	Test
L-Sub Gold	0.050	0.106
Random	0.016	0.012
Simple Freq.	0.397	0.471

Results in table 4 show that the "Simple Freq." baseline performs very strongly, despite being simple. In fact, it surpasses the average inter-annotator agreement on both Trial and Test datasets.

Results in table 5 show that the ranking SVM with RBF kernel performs best, although all supervised methods perform similarly on the Test set, the unsupervised method being somewhat weaker. All four methods outperform two of the three given baselines. It is possible that the supervised approaches would outperform the "Simple Freq." baseline if there were more training data available.

7. Conclusion

We presented four different methods in solving the lexical simplification task, using both context-dependent and

Table 3: Optimal hyperparameters for linear combination of features.

Feature	Weight
Inverse word length	1
WordNet synsets	0
Simple Wikipedia frequency	10
English Wikipedia frequency	0
Corpus complexity	0
Context similarity	9
Semantic similarity	0

Table 5: Implemented methods kappa scores on the Test dataset.

Method name	Test score
Ranking SVM with RBF kernel	0.461
Ranking SVM with linear kernel	0.443
Linear combination of features	0.459
Unsupervised approach	0.313

context-independent features, as well as external resources such as state-of-the-art word vector representations and simplified corpora.

The top-performing method in our approach was the ranking SVM with RBF kernel, although it is followed closely by linear combination of features and the ranking SVM with linear kernel. The unsupervised approach does not perform as well, but it outperforms two out of three given baselines.

We believe that the performance of supervised approaches is likely to improve with larger training sets, and that the scarcity of training data is the main reason why the context-independent "Simple Freq." baseline outperforms these methods. Additionally, the evaluation metric is very susceptible to penalize slight changes, making it overly pessimistic about the system's performance.

In the linear combination of features, the largest weight was given to Simple Wikipedia frequency, followed by context similarity. This shows that there is a very strong relation between distributional frequency of words and their perceived simplicity, as well as the importance context-dependent features and word vector representations. On the other hand, it appears that relying on the entire English Wikipedia corpus and lexico-semantic resources like WordNet do not have any impact on the system's performance.

References

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

Thorsten Brants and Alex Franz. 2006. {Web 1T 5-gram Version 1}.

John Carroll, Guido Minnen, Yvonne Canning, Siobhan Devlin, and John Tait. 1998. Practical simplification of english newspaper text to assist aphasic readers. In *Proceedings of the AAAI-98 Workshop on Integrating Artificial Intelligence and Assistive Technology*, pages 7–10. Citeseer.

Jan De Belder and Marie-Francine Moens. 2010. Text simplification for children. In *Prroceedings of the SIGIR workshop on accessible search systems*, pages 19–26. ACM.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Goran Glavaš and Sanja Štajner. 2015. Simplifying lexical simplification: Do we need simplified corpora? *Volume 2: Short Papers*, page 63.

Colby Horn, Cathryn Manduca, and David Kauchak. 2014. Learning a lexical simplifier using wikipedia. In *ACL (2)*, pages 458–463.

Thorsten Joachims. 2006. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226. ACM.

Jacob Kohen. 1960. A coefficient of agreement for nominal scale. *Educ Psychol Meas*, 20:37–46.

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. *The Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Serge Sharoff. 2006. Creating general-purpose corpora using automated search engine queries. *WaCky*, pages 63–98.