SV000gg at SemEval-2016 Task 11: Heavy Gauge Complex Word Identification with System Voting

Gustavo Henrique Paetzold and **Lucia Specia**
Department of Computer Science
University of Sheffield, UK
{ghpaetzold1,l.specia}@sheffield.ac.uk

Abstract

We introduce the SV000gg systems: two Ensemble Methods for the Complex Word Identification task of SemEval 2016. While the SV000gg-Hard system exploits basic Hard Voting, the SV000gg-Soft system employs Performance-Oriented Soft Voting, which weights votes according to the voter's performance rather than its prediction confidence, allowing for completely heterogeneous systems to be combined. Our performance comparison shows that our voting techniques outperform traditional Soft Voting, as well as other systems submitted to the shared task, ranking first and second overall.

1 Introduction

In Complex Word Identification (CWI), the goal is to find which words in a given text may challenge the members of a given target audience. It is part of the usual Lexical Simplification pipeline, which is illustrated in Figure 1. As shown by the results obtained by (Paetzold and Specia, 2013) and (Shardlow, 2014), ignoring the step of Complex Word Identification in Lexical Simplification can lead simplifiers to neglect challenging words, as well as to replace simple words with inappropriate alternatives.

Various strategies have been devised to address CWI and most of them are very simple in nature. For example, to identify complex words, the lexical simplifier for the medical domain in (Elhadad and Sutaria, 2007) uses a Lexicon-Based approach that exploits the UMLS (Bodenreider, 2004) database: if a medical expression is among the technical terms registered in UMLS, then it is complex.

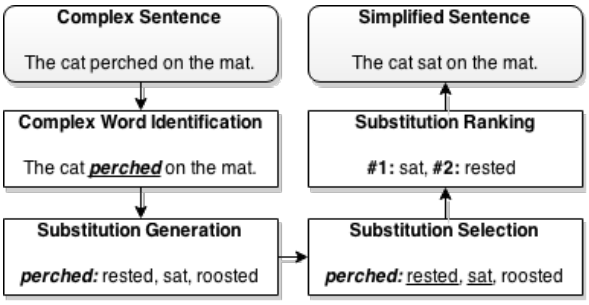

Figure 1: Lexical Simplification pipeline

The complexity identifier for the lexical simplifier in (Keskisärkkä, 2012), for Swedish, uses a threshold over word frequencies to distinguish complex from simple words. Recently, however, more sophisticated approaches have been used. (Shardlow, 2013) presents a CWI benchmarking that compares the performance of a Threshold-Based strategy, a Support Vector Machine (SVM) model trained over various features, and a "simplify everything" baseline.

(Shardlow, 2013)'s SVM model has shown promising results, but CWI approaches do not tend to explore Machine Learning techniques and, in particular, their combination. As an effort to fill this gap, in this paper we describe our contributions to the Complex Word Identification task of SemEval 2016. We introduce two systems, SV000gg-Hard and SV000gg-Soft, both of which use straightforward Ensemble Methods to combine different predictions for CWI. These come from a variety of models, ranging from simple Lexicon-Based approaches to more elaborate Machine Learning classifiers.

Proceedings of SemEval-2016, pages 969–974,
San Diego, California, June 16-17, 2016.

2 Dataset and Evaluation

In the CWI task of SemEval 2016, participants were asked to submit predictions on the complexity of words based on the needs of non-native English speakers. The setup of the task is as follows: given a target word in a sentence, predict whether or not a non-native English speaker would be able to understand it. For training, a **joint** and a **decomposed** dataset were provided. Both datasets consist in $2,237$ instances containing a sentence, a target word, its position in the sentence, and complexity label(s). The **decomposed** dataset contains 20 binary complexity labels, provided by 20 annotators, while the **joint** dataset contains only one label: 1 if at least one of the 20 annotators did not understand it (complex), and 0 otherwise (simple). Participants were allowed to train their systems over either, both or none of the datasets, as well as use any external resources.

The test set contains $88,221$ instances and follows the same format of the joint dataset, but was generated using only one word complexity label. The difference between the training and test sets is that while each instance in the training set was annotated by 20 people, each instance in the test set was annotated by only one person. The goal with this setup was that of replicating a realistic scenario in Text Simplification, where systems must predict the individual preferences of a target audience based on the overall needs of a population sample.

For evaluation, common metrics – Accuracy, Precision, Recall and F-score – are used, along with a new metric designed specifically for CWI: the **G-score**. The G-score consists of the harmonic mean between Accuracy and Recall, and aims at capturing the performance of a CWI approach to be used within a Lexical Simplification system. The reasoning behind the metric is that an ideal CWI system should avoid both false negatives and false positives, which is measured through Accuracy, and at the same time capture as many complex words as possible, which is measured through Recall. High values on these two metrics would prevent a lexical simplifier from making unnecessary and possibly erroneous word replacements and from neglecting words which should be simplified.

3 System Overview

Our strategy explores the idea behind the popular saying "*two heads are better than one*" for the CWI problem. We believe that combining the "opinion" of various distinct approaches to a given task can yield better results than any of the individual approaches. This idea is not new for classification tasks like ours, and have been thoroughly explored in several ways. Strategies that combine multiple Machine Learning classifiers are often referred to as Ensemble Methods. Such methods range from very simple solutions, such as Hard Voting, in which labels are determined based on how many times they were predicted by the classifiers, to very elaborate approaches, such as Random Forests (Breiman, 2001) and Gradient Boosting (Friedman, 2001).

The strategy we employ consists of a variant of Soft Voting, in which the class of a given instance is determined as in Equation 1.

$$c_f = \arg\max_c \sum_{s \in S} T(s, c) \tag{1}$$

In traditional Soft Voting, c_f is the selected class, c is one of the possible classes in a classification problem, S the collection of systems considered, and T a confidence estimate, i.e. a function that expresses how confident system s is that c is the correct class. Its goal is to increment Hard Voting by incorporating the systems' classification confidence in the decision process, hopefully making for a more reliable way of exploiting their strengths and weaknesses.

Although sensible in principle, Soft Voting might not be able to effectively combine systems if they do not have a reasonably uniform way of determining the confidence on their predictions. The presence of over-optimistic or over-pessimistic systems may skew the results severely, and hence make the resulting classifier have worse performance than that of the best system among those considered in the voting. Another clear limitation of traditional Soft Voting is that it cannot include systems which simply cannot estimate the confidence level of their prediction. Lexicon-Based CWI approaches such as the ones of (Elhadad and Ph, 2006) and (Elhadad and Sutaria, 2007), for example, predict that a word is simple if it is present in a certain vocabulary. These

approaches tend to be very effective in certain contexts, but can only produce binary confidence estimates: if the word is in the vocabulary, then it is 100% sure the word is simple, if not, it is 100% sure the word is complex.

In order to address these limitations, we exploit Performance-Oriented Soft Voting (Georgiou and Mavroforakis, 2013). Instead of using the systems' summed confidence to predict a label, it uses their performance score over a certain validation dataset. Formally, we decompose function T from Equation 1 into the two functions illustrated in Equation 2.

$$c_f = \arg\max_c \sum_{s \in S} P(s, d) * D(s, c) \qquad (2)$$

In Equation 2, P represents the score of system s over a certain dataset d given a certain performance metric, such as Precision, Recall, F1, Accuracy, etc. Function D, on the other hand, outputs value 1 if system s has predicted c for the classification problem in question, and 0 otherwise.

This setup works under the assumption that the systems' performance under a validation dataset is a reliable surrogate for confidence predictions, and allows for any type of systems to be combined, whether or not they are homogeneous in their way of predicting classes.

In what follows, we described the features and settings used in the creation of our two CWI systems: SV000gg-Hard and SV000gg-Soft. While SV000gg-Hard uses basic Hard Voting, SV000gg-Soft uses Performance-Oriented Soft Voting. Since both of them combine a series of sub-systems, to avoid confusion, we henceforth refer to these subsystems as "voters".

3.1 Features

Our voters use a total of 69 features. They can be divided in four categories:

- **Binary:** If a target word is part of a certain vocabulary, then it receives label 1, otherwise, 0. We extract vocabularies from Simple Wikipedia (Kauchak, 2013), Ogden's Basic English (Ogden, 1968) and SubIMDB (Paetzold, 2015).
- **Lexical:** Includes word length, number of syllables, number of senses, synonyms, hypernyms and hyponyms in WordNet (Fellbaum, 1998), and language model probability in Wikipedia (Kauchak and Barzilay, 2006), Simple Wikipedia and SubIMDB.
- **Collocational:** Language model probabilities of all n-gram combinations with windows $w < 3$ to the left and right of the target complex word in Wikipedia, SUBTLEX (Brysbaert and New, 2009), Simple Wikipedia and SubIMDB.
- **Nominal:** Includes the word itself, its POS tag, both word and POS tag n-gram combinations with windows $w < 3$ to the left and right, and the word's language model backoff behavior (Uhrik and Ward, 1997) according to a 5-gram language model trained over Simple Wikipedia with SRILM (Stolcke and others, 2002).

In order for language model probabilities to be calculated, we train a 5-gram language model for each of the aforementioned corpora using SRILM (Stolcke and others, 2002). Nominal features were obtained with the help of LEXenstein (Paetzold and Specia, 2015).

3.2 Voters

We train a total of 21 voters which we have grouped in three categories:

- **Lexicon-Based (LB):** If a word is present in a given vocabulary of simple words, then it is simple, otherwise, it is complex. We train one Lexicon-Based voter for each binary feature described in the previous Section.
- **Threshold-Based (TB):** Given a certain feature, learns the threshold t which best separates complex and simple words. In order to learn t, it first calculates the feature value for all instances in the training data and obtains its minimum and maximum. It then divides the interval into $10,000$ equally sized parts, and performs a brute force search over all $10,000$ values to find the one which yields the highest G-score over the training data. We train one Threshold-Based voter for each lexical feature described in the previous Section.

System	Accuracy	Precision	Recall	F-score	G-score
All Complex	0.047	0.047	1.000	0.089	0.089
All Simple	0.953	0.000	0.000	0.000	0.000
(LB) SubIMDB	0.913	0.217	0.332	0.262	0.487
(LB) Ogden's	0.248	0.056	0.947	0.105	0.393
(LB) Wikipedia	0.047	0.047	1.000	0.089	0.090
(LB) Simple Wikipedia	0.953	0.241	0.002	0.003	0.003
(TB) Probability: Wikipedia	0.536	0.084	0.901	0.154	0.672
(TB) Probability: Simple Wiki	0.513	0.081	0.902	0.148	0.654
(TB) Number of Hypernyms	0.572	0.076	0.728	0.137	0.641
(TB) Probability: SUBTLEX	0.492	0.077	0.896	0.142	0.636
(TB) Probability: SubIMDB	0.445	0.072	0.912	0.133	0.598
(TB) Number of Senses	0.436	0.068	0.861	0.125	0.579
(TB) Number of Hyponyms	0.384	0.065	0.906	0.121	0.539
(TB) Length	0.332	0.057	0.852	0.107	0.478
(ML) Decision Trees	0.805	0.158	0.733	0.260	0.767
(ML) Adaptive Boosting	0.799	0.153	0.728	0.253	0.762
(ML) Random Forests	0.826	0.170	0.698	0.273	0.756
(ML) Gradient Boosting	0.802	0.147	0.672	0.241	0.731
(ML) Multi-Layer Perceptron	0.691	0.105	0.741	0.183	0.715
(ML) Passive Aggressive Learning	0.852	0.171	0.562	0.262	0.677
(ML) Conditional Random Fields	0.534	0.076	0.808	0.140	0.643
(ML) Stochastic Gradient Descent	0.648	0.057	0.423	0.101	0.512
(ML) Support Vector Machines	0.715	0.061	0.357	0.105	0.476
TALN-RandomForest_WEI	0.812	0.164	0.736	0.268	0.772
UWB-All	0.803	0.157	0.734	0.258	0.767
PLUJAGH-SEWDF	0.795	0.152	0.741	0.252	0.767
JUNLP-NaiveBayes	0.767	0.139	0.767	0.236	0.767
HMC-RegressionTree05	0.838	0.182	0.705	0.290	0.766
HMC-DecisionTree25	0.846	0.189	0.698	0.298	0.765
JUNLP-RandomForest	0.795	0.151	0.730	0.250	0.761
MACSAAR-RFC	0.825	0.168	0.694	0.270	0.754
TALN-RandomForest_SIM	0.847	0.186	0.673	0.292	0.750
MACSAAR-NNC	0.804	0.146	0.660	0.240	0.725
Pomona-NormalBag	0.604	0.095	0.872	0.171	0.714
Melbourne-runw15	0.586	0.091	0.870	0.165	0.701
UWB-Agg	0.569	0.089	0.885	0.161	0.693
Pomona-GoogleBag	0.568	0.088	0.881	0.160	0.691
IIIT-NCC	0.546	0.084	0.880	0.154	0.674
LTG-System2	0.889	0.220	0.541	0.312	0.672
MAZA-A	0.773	0.115	0.578	0.192	0.661
Melbourne-runw3	0.513	0.080	0.895	0.147	0.652
Sensible-Baseline	0.591	0.078	0.713	0.140	0.646
ClacEDLK-ClacEDLK-RF_0.6	0.688	0.081	0.548	0.141	0.610
PLUJAGH-SEWDFF	0.922	0.289	0.453	**0.353**	0.608
IIIT-NCC2	0.465	0.071	0.860	0.131	0.604
ClacEDLK-ClacEDLK-RF_0.5	0.751	0.090	0.475	0.152	0.582
MAZA-B	0.912	0.243	0.420	0.308	0.575
AmritaCEN-w2vecSim	0.627	0.061	0.486	0.109	0.547
Soft Voting	0.780	0.125	0.615	0.207	0.688
SV000g-Soft	0.779	0.147	0.769	0.246	**0.774**
SV000g-Hard	0.761	0.138	0.787	0.235	0.773

Table 1: Performance scores. Separated by double horizontal lines are three system groups: our voters, other systems submitted to the SemEval task, and our Ensemble solutions.

- **Machine-Learning-Assisted (ML):** Learn a binary classification model from the training data using a Machine Learning algorithm. We build models using the following seven algorithms in the scikit-learn toolkit (Pedregosa et al., 2011):

  1. Support Vector Machines
  2. Passive Aggressive Learning
  3. Stochastic Gradient Descent
  4. Decision Trees
  5. Ada Boosting
  6. Gradient Boosting
  7. Random Forests

  Additionally, we use Keras[1] to otrain a Multi-Layer Perceptron voter. Its architecture, including number and size of hidden-layers, was decided through 5-fold cross-validation over the training set. The aforementioned models use as input all binary, lexical and collocational features. Finally, we also train a Conditional Random Field model using CRFSuite (Okazaki, 2007). It uses as input all nominal features described in the previous Section. The hyper-parameters of all Machine Learning-assisted voters are determined through 5-fold cross-validation over the G-score.

We select the number of the top G-score systems to be considered through 5-fold cross-validation over the joint dataset. For completion, we also include a traditional Soft Voting system that combines Machine Learning approaches only, given that the others do not have well-established ways of calculating prediction probability estimates.

4 Results

Table 1 illustrates the performance scores of all individual voters, along with the 25 best performing systems in the CWI task, a standard Soft Voting approach, and our two SV000gg systems. Despite their simplicity, our system voting strategies are the two most effective CWI solutions submitted to SemEval 2016, having both obtained considerably higher G-scores than traditional Soft Voting. These results show the importance of finding clever ways to combine distinct strategies for a task, since, by not considering Lexicon and Threshold-Based voters, the traditional soft voter suffered a considerable loss in G-score.

The results of the individual voters reveal that Decision Trees and Ensemble Methods achieve noticeably higher performance than the Multi-Layer Perceptron, which have been used as state-of-the-art solutions to various tasks. Another surprise comes with the scores of Threshold-Based voters, which offer competitive performance in comparison to Machine Learning techniques. The performance of our Conditional Random Field voter suggest that nominal features are not as reliable as numeric features in predicting word complexity.

The effectiveness of Ensemble Methods is further highlighted by the scores of ours' and others' solutions for the SemEval task: precisely 50% of the top 10 systems use some type of Ensemble.

5 Conclusions

We have presented our contributions to the Complex Word Identification task of SemEval 2016: the SV000gg systems, which exploit two types of system Ensemble voting schemes. Along with the typical Hard Voting, we employ Performance-Oriented Soft Voting, which diverges from traditional Soft Voting by weighting votes not by their prediction confidence, but rather by overall system performance.

Our performance comparison shows how effective our voting strategies can be: they top the rankings in the SemEval task, outperforming even elaborate Ensemble strategies. We hope that our approach will serve as a reliable alternative to other problems in Natural Language Processing and beyond.

In the future, we also intend to explore the use of Gaussian Processes and Multi-Task Learning for Complex Word Identification.

References

O. Bodenreider. 2004. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32.

Leo Breiman. 2001. Random forests. *Machine Learning*, 45:5–32.

[1] http://keras.io

Marc Brysbaert and Boris New. 2009. Moving beyond kucera and francis: a critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for american english. *Behavior research methods*, 41(4):977–90, December.

Noemie Elhadad and D Ph. 2006. Comprehending technical texts : Predicting and defining unfamiliar terms. pages 239–243.

Noemie Elhadad and Komal Sutaria. 2007. Mining a lexicon of technical terms and lay equivalents. pages 49–56.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

Harris V Georgiou and Michael E Mavroforakis. 2013. A game-theoretic framework for classifier ensembles using weighted majority voting with local accuracy estimates. *arXiv preprint arXiv:1302.0540*.

David Kauchak and Regina Barzilay. 2006. Paraphrasing for automatic evaluation. In *Proceedings of the 2006 NAACL*, pages 455–462.

David Kauchak. 2013. Improving text simplification language modeling using unsimplified text data. In *Proceedings of the 51st ACL*, pages 1537–1546.

R Keskisärkkä. 2012. Automatic text simplification via synonym replacement.

Charles Kay Ogden. 1968. *Basic English: international second language*. Harcourt, Brace & World.

Naoaki Okazaki. 2007. CRFsuite: a fast implementation of conditional random fields. http://www.chokkan.org/software/crfsuite/.

Gustavo H. Paetzold and Lucia Specia. 2013. Text simplification as tree transduction. In *Proceedings of the 9th STIL*.

Gustavo Henrique Paetzold and Lucia Specia. 2015. Lexenstein: A framework for lexical simplification. In *Proceedings of The 53rd ACL*.

Gustavo Henrique Paetzold. 2015. Reliable lexical simplification for non-native speakers. In *Proceedings of the 2015 NAACL Student Research Workshop*.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Matthew Shardlow. 2013. A comparison of techniques to automatically identify complex words. In *Proceedings of the 51st ACL Student Research Workshop*, pages 103–109.

Matthew Shardlow. 2014. Out in the open: Finding and categorising errors in the lexical simplification pipeline. In *Proceedings of the 9th LREC*.

Andreas Stolcke et al. 2002. Srilm - an extensible language modeling toolkit. In *Interspeech*.

C Uhrik and W Ward. 1997. Confidence metrics based on n-gram language model backoff behaviors. In *Proceedings of EUROSPEECH*.