# Quality Assessment for Text Simplification (QATS)

## Workshop Programme

**Saturday, May 28, 2016**

<table>
<thead>
<tr>
<th>Time</th>
<th>Session/Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:00 – 09:20</td>
<td>Introduction by Sanja Štajner</td>
</tr>
<tr>
<td>09:20 – 10:00</td>
<td>Invited Talk by Advaith Siddharthan</td>
</tr>
</tbody>
</table>

### Session: General Track

<table>
<thead>
<tr>
<th>Time</th>
<th>Speaker(s) / Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>10:00 – 10:30</td>
<td>Gustavo H. Paetzold and Lucia Specia: <em>PLUMBErr: An Automatic Error Identification Framework for Lexical Simplification</em></td>
</tr>
<tr>
<td>10:30 – 11:00</td>
<td>Coffee break</td>
</tr>
<tr>
<td>11:00 – 11:30</td>
<td>Sandeep Mathias and Pushpak Bhattacharyya: <em>How Hard Can it Be? The E-Score - A Scoring Metric to Assess the Complexity of Text</em></td>
</tr>
<tr>
<td>11:30 – 12:00</td>
<td>Sanja Štajner, Maja Popović and Hanna Béchara: <em>Quality Estimation for Text Simplification</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>Session: Shared Task Introduction/Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:00 – 12:15</td>
<td>Shared Task: Introduction by Maja Popović</td>
</tr>
</tbody>
</table>

### Session: Shared Task 1

<table>
<thead>
<tr>
<th>Time</th>
<th>Speaker(s) / Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:15 - 12:45</td>
<td>Maja Popović and Sanja Štajner: <em>Machine Translation Evaluation Metrics for Quality Assessment of Automatically Simplified Sentences</em></td>
</tr>
<tr>
<td>13:15 - 14:30</td>
<td>Lunch break</td>
</tr>
</tbody>
</table>
### Session: Shared Task 2

<table>
<thead>
<tr>
<th>Time</th>
<th>Speaker(s)</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>14:30 – 15:00</td>
<td>Gustavo H. Paetzold and Lucia Specia</td>
<td><em>SimpleNets: Evaluating Simplifiers with Resource-Light Neural Networks</em></td>
</tr>
<tr>
<td>15:00 – 15:30</td>
<td>Sergiu Nisioi and Fabrice Nauze</td>
<td><em>An Ensemble Method for Quality Assessment of Text Simplification</em></td>
</tr>
<tr>
<td>15:30 – 16:00</td>
<td>Elnaz Davoodi and Leila Kosseim</td>
<td><em>CLaC @ QATS: Quality Assessment for Text Simplification</em></td>
</tr>
<tr>
<td>16:00 – 16:30</td>
<td></td>
<td>Coffee break</td>
</tr>
<tr>
<td>16:30 – 17:30</td>
<td></td>
<td><strong>Round Table</strong></td>
</tr>
<tr>
<td>17:30 – 17:45</td>
<td></td>
<td><strong>Closing</strong></td>
</tr>
</tbody>
</table>
Editors
Sanja Štajner
Maja Popović
Horacio Saggion
Lucia Specia
Mark Fishel

University of Mannheim, Germany
Humboldt University of Berlin, Germany
Universitat Pompeu Fabra, Spain
University of Sheffield, UK
University of Tartu, Estonia

Organizing Committee
Sanja Štajner
Maja Popović
Horacio Saggion
Lucia Specia
Mark Fishel

University of Mannheim, Germany
Humboldt University of Berlin, Germany
Universitat Pompeu Fabra, Spain
University of Sheffield, UK
University of Tartu, Estonia

Programme Committee
Sandra Aluisio
Eleftherios Avramidis
Susana Bautista
Stefan Bott
Richard Evans
Mark Fishel
Sujay Kumar Jahuar
David Kauchak
Elena Lloret
Ruslan Mitkov
Gustavo Paetzold
Maja Popović
Miguel Rios
Horacio Saggion
Carolina Scarton
Matthew Shardlow
Advaith Siddharthan
Lucia Specia
Miloš Stanojević
Sanja Štajner
Irina Temnikova
Sowmya Vajjala
Victoria Yaneva

University of São Paolo
DFKI Berlin
Federal University of Rio Grande do Sul
University of Stuttgart
University of Wolverhampton
University of Tartu
Carnegie Mellon University
Pomona College
Universidad de Alicante
University of Wolverhampton
University of Sheffield
Humboldt University of Berlin
University of Leeds
Universitat Pompeu Fabra
University of Sheffield
University of Manchester
University of Aberdeen
University of Sheffield, UK
University of Amsterdam
University of Mannheim
Qatar Computing Research Institute
Iowa State University
University of Wolverhampton
# Table of Contents

**PLUMBErr: An Automatic Error Identification Framework for Lexical Simplification**  
Gustavo H. Paetzold and Lucia Specia ................................................................. 1

**How Hard Can it Be? The E-Score - A Scoring Metric to Assess the Complexity of Text**  
Sandeep Mathias and Pushpak Bhattacharyya ...................................................... 10

**Quality Estimation for Text Simplification**  
Sanja Štajner, Maja Popović and Hanna Bèchara .................................................. 15

**Shared Task on Quality Assessment for Text Simplification**  
Sanja Štajner, Maja Popović, Horacio Saggion, Lucia Specia and Mark Fishel .......... 22

**Machine Translation Evaluation Metrics for Quality Assessment of Automatically Simplified Sentences**  
Maja Popović and Sanja Štajner ........................................................................... 32

**Using Machine Translation Evaluation Techniques to Evaluate Text Simplification Systems**  
Sandeep Mathias and Pushpak Bhattacharyya ...................................................... 38

**SimpleNets: Evaluating Simplifiers with Resource-Light Neural Networks**  
Gustavo H. Paetzold and Lucia Specia ................................................................... 42

**An Ensemble Method for Quality Assessment of Text Simplification**  
Sergiu Nisioi and Fabrice Nauze .................................................................... 47

**CLaC @ QATS: Quality Assessment for Text Simplification**  
Elnaz Davoodi and Leila Kosseim ..................................................................... 53
Author Index

Bhattacharyya, Pushpak ................................................................. 10,38

Davoodi, Elnaz ................................................................. 53

Fishel, Mark ................................................................. 22

Kosseim, Leila ................................................................. 53

Mathias, Sandeep ................................................................. 10,38

Nauze, Fabrice ................................................................. 47

Nisioi, Sergiu ................................................................. 47

Paetzold, Gustavo H. ................................................................. 1,42

Popović, Maja ................................................................. 15, 22, 32

Saggion, Horacio ................................................................. 22

Specia, Lucia ................................................................. 1, 22, 42

Štajner, Sanja ................................................................. 15, 22, 32
Preface

In recent years, there has been an increasing interest in automatic text simplification (ATS) and text adaptation to various target populations. However, studies concerning evaluation of ATS systems are still very scarce and there are no methods proposed for directly comparing performances of different systems. This workshop addresses this problem and provides an opportunity to establish some metrics for automatic evaluation of ATS systems.

Given the close relatedness of the problem of automatic evaluation of ATS system to the well-studied problems of automatic evaluation and quality estimation in machine translation (MT), the workshop also features a shared task on automatic evaluation (quality assessment) of ATS systems.

We accepted three papers in the general track and five papers describing the systems which participated in the shared task. The papers describe a variety of interesting approaches to this task.

We wish to thank all people who helped in making this workshop a success. Our special thanks go to Advaith Siddharthan for accepting to give the invited presentation, to the members of the program committee who did an excellent job in reviewing the submitted papers and to the LREC organisers, as well as all authors and participants of the workshop.

Sanja Štajner, Maja Popović, Horacio Saggion, Lucia Specia and Mark Fishel

May 2016
PLUMBErr: An Automatic Error Identification Framework for Lexical Simplification

Gustavo H. Paetzold and Lucia Specia
Department of Computer Science, University of Sheffield
Western Bank, Sheffield, UK
ghpaetzold1@sheffield.ac.uk, l.specia@sheffield.ac.uk

Abstract
Lexical Simplification is the task of replacing complex words with simpler alternatives. Using human evaluation to identify errors made by simplifiers throughout the simplification process can help to highlight their weaknesses, but is a costly process. To address this problem, we introduce PLUMBErr: an automatic alternative. Using PLUMBErr, we analyze over 40 systems, and find out the best combination to be the one between the winner of the Complex Word Identification task of SemEval 2016 and a modern simplifier. Comparing PLUMBErr to human judgments we find that, although reliable, PLUMBErr could benefit from resources annotated in a different way.

Keywords: Error Analysis, Lexical Simplification, Evaluation

1. Introduction
Lexical Simplification is perhaps the most self-contained form of Text Simplification among all. It consists in replacing complex words in sentences with simpler alternatives. Unlike Syntactic Simplification, it does not involve performing any deep modifications to the sentence’s syntactic structure. Despite its simplicity, it is still a very challenging task. In order to be reliable and effective, a lexical simplifier must be able to:

1. Predict which words challenge a reader.
2. Avoid introducing any grammatical errors to the sentence.
3. Avoid omitting and/or changing any piece of relevant information in the sentence.
4. Make the sentence as simple as possible.

In previous attempts to meet all of these requirements, various solutions have been devised for the task, all of which follow roughly the same pipeline of Figure 1.

![Figure 1: Lexical Simplification Pipeline](image)

The first lexical simplifier in literature is quite simple: given a complex word, it extracts its synonyms from WordNet, and then replaces the complex word with the synonym with the highest word frequency in the Brown corpus (Francis and Kucera, 1979). Since then, the strategies used have become much more elaborate. Take, for example, the approach of (Kajiwara et al., 2013), which simplifies sentences in Japanese. They first generate candidate replacements for complex words by extracting words with the same Part-of-Speech tags as them from dictionary definitions, then rank them by using a very sophisticated ensemble of metrics, which include frequency in corpora, co-occurrence model similarity and semantic distance. Another modern example is the approach of (Horn et al., 2014). They learn candidate replacements from word alignments between complex and simple equivalent sentences, then rank them using a sophisticated supervised ranking strategy. Other notable examples are the unsupervised solutions introduced by (Glavaš and Štajner, 2015) and (Paetzold and Specia, 2016d) and the tree transduction strategy of (Paetzold and Specia, 2013).

In order to assess and compare the performance of such varied strategies, previous work has resorted to both manual and automatic evaluation methods. The most widely used, and arguably the most reliable, is human evaluation. A very similar approach is used in various papers (Biran et al., 2011; Paetzold and Specia, 2013; Glavaš and Štajner, 2015): a human judge is presented with various simplifications produced by systems and asked to make judgments with respect to Grammaticality, Meaning Preservation and Simplicity. Automatic evaluation approaches are very different. The most widely used method is the one introduced by (Horn et al., 2014), in which the simplifications produced by a system for a set of problems are compared to a gold-standard produced by hundreds of humans through various metrics.

Neither of these two approaches, however, provide detailed insight on the strengths and limitations of the simplifiers. Although human evaluation can highlight the errors a simplifier makes that lead to ungrammatical replacements, for example, it is often hard for a human to outline the reason why the simplifier makes such mistakes. (Shardlow, 2014) introduces a solution to this problem. Their approach uses human evaluation not to assess the quality of simplifications, but rather to verify the correctness of each decision.
made by a simplifier with respect to the usual pipeline. Although innovative, their error categorization approach is subject to the same limitations of other human evaluation strategies: human judgments are costly to acquire. Such costs make the process of obtaining evaluation results prohibitive.

In this paper we introduce PLUMBErr: an error analysis framework for Lexical Simplification that introduces an accessible automatic solution for error categorization. In the Sections that follow, we discuss the approach of (Shardlow, 2014) in more detail, and present the resources and methods used in PLUMBErr.

2. Error Analysis in Lexical Simplification

Shardlow (2014) describes a pioneer effort in Lexical Simplification evaluation. Their study introduces an error analysis methodology that allows to outline in detail the intricacies of a simplifier. Taking the usual Lexical Simplification pipeline as a basis, they first outline all possible types of errors that a system can make when simplifying a word in a sentence:

- **Type 1**: No error. The system did not make any mistakes while simplifying this word.
- **Type 2A**: The system mistook a complex word for simple.
- **Type 2B**: The system mistook a simple word for complex.
- **Type 3A**: The system did not produce any candidate substitutions for the word.
- **Type 3B**: The system did not produce any simpler candidate substitutions for the word.
- **Type 4**: The system replaced the word with a candidate that compromises the sentence’s grammaticality or meaning.
- **Type 5**: The system replaced the word with a candidate that does not simplify the sentence.

Finally, they establish a methodology for error identification that uses human assessments to judge the output produced by the simplifier after each step of the pipeline. Their methodology, which is illustrated in Figure 2, is very intuitive and sensible. But as previously discussed, acquiring human judgments is often costly, which can consequently limit the number of systems that could be compared in a benchmark. In their work (Shardlow, 2014), for example, they are only able to assess the performance of one simplifier. Although they were able to gain interesting insight on error types and their frequencies for their simplifier, they did not cover comparisons among various Complex Word Identification, Substitution Generation, Selection and Ranking strategies. PLUMBErr offers a solution to this problem.

3. PLUMBErr: An Automatic Alternative

PLUMBErr is a framework for the automatic identification of errors made by pipelined Lexical Simplification systems. To produce a full report on the types of errors made by a lexical simplifier, PLUMBErr employs the same overall error categorization methodology introduced by (Shardlow, 2014). However, in order to bypass the need for human judgments, it resorts to a set of pre-computed gold-standards and a list of complex words produced by English speakers (NNSVocab).

To be evaluated by PLUMBErr, the Lexical Simplification system is first required to solve a series of pre-determined simplification problems present in the BenchLS dataset (Paetzold and Specia, 2016b). Through the PLUMBErr workflow, the judgments and resources produced by the system after each step of the pipeline are then compared to the gold-standards present in BenchLS, as well as the set of complex words present in NNSVocab, which then allow for errors to be found and categorized.

3.1. BenchLS

BenchLS is a dataset introduced by (Paetzold and Specia, 2016b), which was created with the intent of facilitating the benchmarking of Lexical Simplification systems. It is composed of 929 instances. Each instance contains a sentence, a target word, and various gold replacements suggested by English speakers from the U.S. with a variety of backgrounds. Although these replacements are not guaranteed to make the sentence simpler, they do ensure that the sentences are grammatical and meaning preserving. The instances of BenchLS are automatically corrected versions of the instances from two previously created datasets:

- **LexMTurk**: Composed of 500 instances with sentences extracted from Wikipedia. The target word of each instance was selected based on word alignments between the sentence in Wikipedia and its equivalent simplified version in Simple Wikipedia. Candidate substitutions were produced by English speakers through Amazon Mechanical Turk. Each instance contains 50 candidate substitutions for the target word, each produced by a different annotator.
- **LSeval**: Composed of 439 instances with sentences extracted from the English Internet Corpus of English. The target word of each instance was selected at random. Candidate substitutions were produced by English speakers through Amazon Mechanical Turk, and then validated by PhD students.

The automatic correction steps used for BenchLS are two: spelling and inflection correction. For spelling, Norvig’s algorithm is used to fix any words with typos in them. For inflection, the Text Adorning module of LEXenstein (Burns, 2013; Paetzold and Specia, 2015) is used to inflect any substitution candidates that are verbs, nouns, adjectives and adverbs to the same tense as the target word.

---

2. [http://corpus.leeds.ac.uk/internet.html](http://corpus.leeds.ac.uk/internet.html)
3.2. NNSVocab

NNSVocab is a vocabulary of 3,854 words deemed complex by non-native English speakers. The words in NNSVocab were extracted from the datasets used in the Complex Word Identification task of SemEval 2016. They were produced through a user study with sentences whose words were annotated with respect to their complexity (Paetzold and Specia, 2016a).

In the user study, 400 non-native English speakers were presented with 80 sentences each, and then asked to judge the complexity of all content words. Annotators were instructed to select all content words that they did not understand individually, even if the context allowed them to comprehend them. NNSVocab contains all words which were deemed complex by at least one annotator in (Paetzold and Specia, 2016a)’s user study.

3.3. Workflow

The workflow of PLUMBErr, which is illustrated in Figure 3, combines BenchLS and NNSVocab in a manner that allows for all error types described in Section 2. to be identified.

The system being evaluated first takes as input the target word from a simplification problem in BenchLS. The target word is then checked for complexity: is it in NNSVocab? i.e. has it been deemed complex by a non-native? If not, then it does not need to be simplified, otherwise, it must be. The system then predicts the complexity of the word, which is again cross-checked in NNSVocab. If there is a disagreement between the system’s prediction and the judgment of non-natives, then an error of Type 2 is identified. Otherwise, the system then goes through the steps of Substitution Generation and Selection, and hopefully produces a set of candidate substitutions for the target complex word.
word. The candidates produced are then checked for errors of Type 3. If there is at least one candidate available, and it is not a complex word in NNSVocab, then no errors are identified and the system moves on to ranking the candidates. After Substitution Ranking, the best candidate among all is checked for errors of type 4 and 5: if the best candidate is among the replacements suggested by annotators in BenchLS, and it is not in NNSVocab, then it has successfully simplified the sentence.

Finally, PLUMBErr produces a full report of the errors made in each of the problems present in BenchLS.

4. Experimental Settings
As previously mentioned, the work of (Shardlow, 2014) features an error analysis of a single simplifier. In addition, this simplifier does not perform any form of Complex Word Identification or Substitution Selection. In order to showcase the potential of PLUMBErr, we have conducted an error categorization benchmark with several Lexical Simplification systems. The systems we chose have in common that they do not employ any explicit Complex Word Identification steps, i.e. they simplify all words in a sentence. In order to make our experiments more meaningful and informative, we paired these lexical simplifiers with various Complex Word Identification strategies:

- **Simplify Everything (SE):** Deems all words to be complex. This strategy is the most commonly used in literature.
- **Support Vector Machines (SV):** Using various features, it learns a word complexity model from training data using Support Vector Machines. As features, it uses the words’ frequency and movie count in SUBTLEX (Brysbaert and New, 2009), length, syllable, sense and synonym count. Syllables were obtained with the help of Morph Adorner (Burns, 2013). Sense and synonym counts were extracted from WordNet (Fellbaum, 1998). This is the first English language Complex Word Identification approach in literature that uses Machine Learning (Shardlow, 2013).
- **Threshold-Based (TB):** Given a certain complexity metric, it learns the threshold t through exhaustive search from the training data such as to best separate complex from simple words. As metric, it uses raw word frequencies from Simple Wikipedia. This strategy achieved the highest F-score in the Complex Word Identification task of SemEval 2016 (Paetzold and Specia, 2016a). Threshold-Based (TB) can be considered as the first step in the pipeline.
- **Performance-Oriented Soft Voting (PV):** Combines several Complex Word Identification strategies by weighting their predictions according to their overall performance in a validation dataset. We use the same systems and settings described in (Paetzold and Specia, 2016c). This approach obtained the highest G-score (harmonic mean between Accuracy and Recall) in the Complex Word Identification task of SemEval 2016.

To train the supervised complex word identifiers, we use the training set provided in the SemEval 2016 task. In the Sections that follow, we describe each of the lexical simplifier used in our experiments.

4.1. The Devlin Simpltifier
The first lexical simplifier found in literature (Devlin and Tait, 1998). Its approaches to each step of the pipeline are:

- **Substitution Generation:** Extracts synonyms from WordNet.
- **Substitution Selection:** Does not perform Substitution Selection.
- **Substitution Ranking:** Uses the words’ Kucera-Francis coefficient (Rudell, 1993).

4.2. The Horn Simpltifier
One of the most effective supervised lexical simplifiers in literature (Horn et al., 2014). Its approaches to each step of the pipeline are:

- **Substitution Generation:** Extracts complex-to-simple word correspondences from word alignments between Wikipedia and Simple Wikipedia.
- **Substitution Selection:** Does not perform Substitution Selection.
- **Substitution Ranking:** Learns a ranking model using Support Vector Machines (Joachims, 2002) from the examples in the LexMTurk dataset.

We use the same resources and parameters described in (Horn et al., 2014).

4.3. The Glavas Simplifier
An entirely unsupervised system that performs similarly to the Horn simplifier (Glavaš and Štajner, 2015). It approaches each step of the pipeline as follows:

- **Substitution Generation:** Extracts the 10 words closest to a given target complex word using a word embeddings model.
- **Substitution Selection:** Does not perform Substitution Selection.
- **Substitution Ranking:** Ranks candidates using the average ranking obtained for various semantic and locational metrics.

We use the same resources and parameters described in (Glavaš and Štajner, 2015).

4.4. The Unsupervised Paetzold Simpltifier
An entirely unsupervised simplification system that focuses on the needs of non-native English speakers (Paetzold and Specia, 2016d). Its solutions to each step of the pipeline are:

- **Substitution Generation:** Extracts the 10 words closest to a given target complex word using a context-aware word embeddings model.
• **Substitution Selection:** Learns an unsupervised Boundary Ranking model from automatically produced training data, then discards the 25% candidates with the lowest ranks.

• **Substitution Ranking:** Ranks the remaining candidates according to their 5-gram language model probability in the context from which the target complex word was found, with a two-token window to the left and right.

We use the same resources and parameters described in (Paetzold and Specia, 2016d).

### 4.5. The Supervised Paetzold Simplifier

This approach improves over the Unsupervised Paetzold Simplifier with supervised learning. Its solutions to Substitution Generation and Substitution Selection are the same used by the Unsupervised Paetzold Simplifier, but instead of an unsupervised frequency-based ranker, they use a supervised Boundary Ranking approach for Substitution Ranking. Their supervised approach was introduced in (Paetzold and Specia, 2015), and learns a ranking model from a binary classification setup inferred from ranking examples.

For Substitution Generation and Substitution Selection, we use the same resources and parameters described in (Paetzold and Specia, 2016d). For Substitution Ranking, we use the same features and feature selection strategy described in (Paetzold and Specia, 2015), but train the model over the LexMTurk corpus (Horn et al., 2014) as opposed to the one used in (Paetzold and Specia, 2015), which is the training set from the English Lexical Simplification task of SemEval 2012 (Specia et al., 2012). We do this to make our results more comparable, given that LexMTurk is the dataset used to train the supervised SVM ranker of the Horn Simplifier. This decision also makes our comparison more reliable: because LexMTurk was annotated by English speakers from the U.S. as opposed to non-native English speakers, the supervised rankers used by the Horn and Supervised Paetzold Simplifiers may not have any explicit domain advantage over the unsupervised rankers used by the other simplifiers. All aforementioned simplifiers were implemented with the help of LEXenstein (Paetzold and Specia, 2015).

## 5. Cumulative Analysis

In our cumulative analysis, we use the same error propagation technique by (Sharldow, 2014), in which the errors made in a given step are carried onto to the next. If a simplifier makes a mistake in 90% of the instances during Complex Word Identification, for example, then it will only have 10% of the instances left during Substitution Generation. Table 1 shows the count and proportion (in parenthesis) of instances in which each type of error was made for all combinations of Complex Word Identification methods and simplifiers. In Table 1, Paetzold-U refers to the Unsupervised Paetzold Simplifier, while Paetzold-S refers to the Supervised Paetzold Simplifier.

In Complex Word Identification, it becomes clear that Machine Learning approaches are much more reliable than the other two alternatives. The Performance-Oriented System Voting (PV) method makes the fewest Type 2 errors. When it comes to the rest of the pipeline, however, the Supervised Paetzold Simplifier (Paetzold-S) is the clear winner. It performs the smallest amount of Type 3 errors during Substitution Generation and Selection, but most importantly, it offers the smallest total proportion of Type 4 and 5 Substitution Ranking errors. It is also the most consistent: it correctly simplifies the largest number of instances across all Complex Word Identification strategies.

Our results diverge from the findings reported in (Glavas and Stanton, 2015), in which no statistically significant difference was found between the Horn and Glavas Simplifiers. Inspecting their output, we have found that, while the Substitution Generation strategy used by the Horn Simplifier offers higher precision, the one used by the Glavas Simplifier offers higher recall. We believe that the main limitation of the Horn Simplifier lies in Simple Wikipedia’s reduced vocabulary (190,432 distinct words), which is nearly seven times smaller than the vocabulary from the word embeddings model used by the Glavas Simplifier (1,274,545 distinct words). The relatively low Precision achieved by the Glavas Simplifier’s generator, on the other hand, is due mostly to the fact that traditional embedding models do not account for word ambiguity, consequently grouping substitution candidates for all of the words’ meanings together. The context-aware embeddings model used by the Paetzold Simplifiers seem to be a good compromise between the two: it offers the high recall, inherent to employing word embedding models for Substitution Generation, at the same time that it increases precision by accounting for the words’ context during generation.

For Substitution Ranking, the domain-agnostic unsupervised ranker from the Glavas Simplifier seems to be more reliable than the supervised SVM ranker from the Horn Simplifier. Nevertheless, its performance is either equal or inferior to the unsupervised rankers used by the Devlin and Unsupervised Paetzold Simplifiers, which are much simpler in nature. Perhaps most revealing finding is the contrast between the Supervised (Paetzold-S) and Unsupervised (Paetzold-U) Paetzold simplifiers. The Substitution Ranking strategy employed by Paetzold-U has shown to be considerably less reliable than the more sophisticated supervised Boundary Ranker of Paetzold-S. Effective supervised rankers can offer superior flexibility and reliability by not only automatically learning how to combine multiple features, but also by being adaptable to different domains through the use of datasets annotated by different target audiences. These results are in accordance with the findings of (Horn et al., 2014), who also reveal the potential of supervised ranking strategies.

In practice, the combination between Performance-Oriented System Voting (PV) and the Supervised Paetzold Simplifier (Paetzold-S) yields the highest proportion of correctly simplified problems, which is in line with the previously mentioned observations.

### 6. Non-Cumulative Analysis

Our second experiment features a non-cumulative analysis of simplifiers. Unlike in our cumulative analysis, this experiment pairs the Lexical Simplification systems being
evaluated with a “perfect” Complex Word Identification strategy, i.e. an identifier that predicts word complexity with 100% accuracy. This analysis allows to better isolate the pipeline components that compose our simplifiers, and hence obtain more detailed insight on their performance.

Table 2 illustrates the results for Substitution Generation and Substitution Selection. Notice that, since the Supervised and Unsupervised Paetzold Selectors use the same solutions to these tasks, we represent both of them under the “Paetzold” alias. In accordance to our previous experiment, the strategies used by the Paetzold Simplifiers have managed to produce the smallest amount of Type 3 errors, outperforming all other strategies.

The results in Table 3 report on the number of errors of Type 4 and 5, and correct simplifications made by each of the rankers evaluated. These are very revealing: they clearly outline the difference between using a supervised and an unsupervised Substitution Ranking. The rankers of both Paetzold-S and Paetzold-U have made the exact same number of Type 3 errors, which happen when the highest ranking candidate compromises the sentence’s grammaticality or meaning.

This phenomenon is in line with the expected limitations of a simple frequency-based ranker. In a context where both the Substitution Generation solution used produces every single suitable alternative for a complex word, and the Substitution Selection strategy employed is capable of perfectly filtering all ungrammatical and meaning compromising candidate substitutions, a frequency-based ranker might very well suffice. As it has been shown in previous experiments (Carroll et al., 1998; Specia et al., 2012; Rello et al., 2013; Paetzold and Specia, 2016d), word frequencies are one of the strongest indicators of simplicity for many target audiences, and can often outperform even sophisticated supervised solutions in the task of Substitution Ranking alone. In a realistic scenario, however, generators and selectors are not perfect. As demonstrated in the benchmarking of (Paetzold and Specia, 2016b), even the state-of-the-art Substitution Generation and Selection strategies evaluated in this contribution offer very unsatisfactory precision, which means that rankers still have to be wary of spurious candidates. By entirely disregarding the context in which a target complex word is found, a simple frequency-based ranker is entirely incapable of dealing with this problem. Rankers which exploit context-aware features, on the other hand,
can more effectively do so.

7. Combinatory Analysis

One of the most outstanding advantages of pipelined approaches to Lexical Simplification is the flexibility that comes with their modularity. In order to increase the performance of a simplifier as a whole, one can simply attempt to improve on the solution being used for each step of the pipeline individually, which often disregards the need of rethinking the model in its entirety. Perhaps the biggest advantage of pipelined simplifiers is the ease with which one can evaluate the effectiveness of the solutions employed by them. By simply replacing, say, the Substitution Generation approach of a consolidated simplifier from literature with a new solution, one can quickly experiment and acquire valuable insight on how promising the new strategy is.

In our third experiment, we exploit this advantage to enrich our findings. Table 4 shows a comprehensive error identification benchmark of all possible combinations of Substitution Generation, Selection and Ranking strategies used by our simplifiers. To simulate a realistic scenario, this analysis was conducted in cumulative fashion. The Complex Word Identification approach used for all 40 system combinations is Performance-Oriented System Voting, for having performed best in the cumulative analysis of Section 5.

The scores obtained are very thought-provoking. One of the first things we have noticed is that neither of the winners of our benchmark, be it with respect to the raw count or proportion of correctly simplified words, are among the systems evaluated in our previous experiments. This phenomenon reveals just how important it is for one to try multiple combinations of generators, selectors and rankers before deciding on the final architecture of a simplifier to be used in practice.

Another interesting finding refers to the performance of supervised rankers. The reason behind the unimpressive performance of the SVM ranker used by the Horn Simplifier in our first cumulative analysis becomes clear: the Substitution Generation solution with which it was paired is the one to yield the lowest average amount of correctly simplified words across all system combinations. When paired with other generators and selectors, the Horn ranker often outperforms most or all other rankers. Surprisingly, however, the unsupervised Paetzold-U ranker performs outstandingly well when paired with any combination of generators and selectors except for the ones introduced in (Paetzold and Specia, 2016d), which are ironically the ones it was originally paired with. These findings highlight, once more, the effectiveness of supervised rankers, as well as the proficiency with which word frequencies from the SubIMDB corpus, which is the one by the Paetzold-U ranker, in capturing simplicity.

Nonetheless, we strongly believe that the most important finding from our benchmark refers to the role of Substitution Selection in Lexical Simplification. In the majority of scenarios, adding the unsupervised Boundary Ranking selector used by the Paetzold Simplifiers increases the number of correctly simplified words achieved by the generator/ranker pair. More interesting yet is how their selector affects the simplification process. It can be noticed that, in almost all cases, incorporating the Paetzold selector leads the simplifier to make a higher number of Type 3 errors, but a lower number of errors of Type 4 and 5. By discarding candidates which it is not confident about, the selector lowers the Recall of the generator, at the same time that it increases its Precision. The higher Precision allows for the ranker to consequently make less mistakes, often leading to an increase in the number of correctly simplified words. Generators with inherently high precision such as the Horn generator, however, have shown not to benefit from Substitution Selection.

8. Manual vs. Automatic

Although the results in our previous experiment look very much revealing, it is crucial that we assess the reliability of PLUMBErr. To do so, we compare our results with the ones reported by (Shardlow, 2014), who also analyze the performance of the Devlin simplifier when paired with a Simplify Everything identifier. The proportion of errors reported in the manual approach of (Shardlow, 2014) and the automatic approach of PLUMBErr are reported in Figure 4.

While errors of Type 2 and 3 have very similar proportions, an interesting contrast was found for errors that occur during Substitution Ranking: the gold replacements present in BenchLS are more restrictive than the human judgments of (Shardlow, 2014). Nonetheless, this phenomena is expected, given that annotators of BenchLS were able to suggest only a single candidate substitution for each problem. This annotation approach compels them to suggest what they believe to be most appropriate replacement for the target word in question, consequently leading to a lot of repeated suggestions, and hence a lower coverage.

9. Conclusions

In this contribution, we have introduced PLUMBErr: an automatic error identification framework for Lexical Simplification. PLUMBErr uses the same workflow for error identification proposed by (Shardlow, 2014), but replaces human judgments with the data from two pre-produced datasets: BenchLS and NNSVocab. We have used PLUMBErr to analyze the performance of numerous combinations of Complex Word Identification strategies and Lexical Simplification systems in literature.
### Table 4: Combinatory analysis results

<table>
<thead>
<tr>
<th>Generator</th>
<th>Selector</th>
<th>Ranker</th>
<th>3A (%)</th>
<th>3B (%)</th>
<th>4 (%)</th>
<th>5 (%)</th>
<th>No Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Devlin</td>
<td>None</td>
<td>Devlin</td>
<td>146 (61%)</td>
<td>22 (9%)</td>
<td>35 (49%)</td>
<td>8 (11%)</td>
<td>29 (40%)</td>
</tr>
<tr>
<td>Devlin</td>
<td>None</td>
<td>Horn</td>
<td>146 (61%)</td>
<td>22 (9%)</td>
<td>28 (39%)</td>
<td>7 (10%)</td>
<td>37 (51%)</td>
</tr>
<tr>
<td>Devlin</td>
<td>None</td>
<td>Glavas</td>
<td>146 (61%)</td>
<td>22 (9%)</td>
<td>40 (56%)</td>
<td>9 (12%)</td>
<td>23 (32%)</td>
</tr>
<tr>
<td>Devlin</td>
<td>None</td>
<td>Paetzold-U</td>
<td>146 (61%)</td>
<td>22 (9%)</td>
<td>31 (43%)</td>
<td>8 (11%)</td>
<td>33 (46%)</td>
</tr>
<tr>
<td>Devlin</td>
<td>None</td>
<td>Paetzold-S</td>
<td>146 (61%)</td>
<td>22 (9%)</td>
<td>43 (60%)</td>
<td>6 (8%)</td>
<td>23 (32%)</td>
</tr>
<tr>
<td>Devlin</td>
<td>Paetzold</td>
<td>Devlin</td>
<td>154 (64%)</td>
<td>30 (12%)</td>
<td>36 (55%)</td>
<td>10 (15%)</td>
<td>44 (52%)</td>
</tr>
<tr>
<td>Devlin</td>
<td>Paetzold</td>
<td>Horn</td>
<td>154 (64%)</td>
<td>30 (12%)</td>
<td>32 (49%)</td>
<td>12 (18%)</td>
<td>52 (54%)</td>
</tr>
<tr>
<td>Devlin</td>
<td>Paetzold</td>
<td>Glavas</td>
<td>154 (64%)</td>
<td>30 (12%)</td>
<td>32 (49%)</td>
<td>17 (25%)</td>
<td>44 (50%)</td>
</tr>
<tr>
<td>Devlin</td>
<td>Paetzold</td>
<td>Paetzold-U</td>
<td>154 (64%)</td>
<td>30 (12%)</td>
<td>30 (43%)</td>
<td>12 (14%)</td>
<td>38 (44%)</td>
</tr>
<tr>
<td>Devlin</td>
<td>Paetzold</td>
<td>Paetzold-S</td>
<td>154 (64%)</td>
<td>30 (12%)</td>
<td>57 (69%)</td>
<td>8 (9%)</td>
<td>22 (25%)</td>
</tr>
<tr>
<td>Horn</td>
<td>None</td>
<td>Devlin</td>
<td>123 (51%)</td>
<td>30 (12%)</td>
<td>51 (59%)</td>
<td>11 (13%)</td>
<td>25 (29%)</td>
</tr>
<tr>
<td>Horn</td>
<td>None</td>
<td>Horn</td>
<td>123 (51%)</td>
<td>30 (12%)</td>
<td>50 (57%)</td>
<td>13 (15%)</td>
<td>28 (32%)</td>
</tr>
<tr>
<td>Horn</td>
<td>None</td>
<td>Glavas</td>
<td>123 (51%)</td>
<td>30 (12%)</td>
<td>49 (59%)</td>
<td>11 (13%)</td>
<td>38 (38%)</td>
</tr>
<tr>
<td>Horn</td>
<td>None</td>
<td>Paetzold-U</td>
<td>123 (51%)</td>
<td>30 (12%)</td>
<td>37 (43%)</td>
<td>12 (14%)</td>
<td>38 (44%)</td>
</tr>
<tr>
<td>Horn</td>
<td>None</td>
<td>Paetzold-S</td>
<td>123 (51%)</td>
<td>30 (12%)</td>
<td>57 (66%)</td>
<td>8 (9%)</td>
<td>22 (25%)</td>
</tr>
<tr>
<td>Horn</td>
<td>Paetzold</td>
<td>Devlin</td>
<td>145 (60%)</td>
<td>30 (12%)</td>
<td>36 (55%)</td>
<td>10 (15%)</td>
<td>19 (29%)</td>
</tr>
<tr>
<td>Horn</td>
<td>Paetzold</td>
<td>Horn</td>
<td>145 (60%)</td>
<td>30 (12%)</td>
<td>32 (49%)</td>
<td>12 (18%)</td>
<td>21 (32%)</td>
</tr>
<tr>
<td>Horn</td>
<td>Paetzold</td>
<td>Glavas</td>
<td>145 (60%)</td>
<td>30 (12%)</td>
<td>32 (49%)</td>
<td>11 (13%)</td>
<td>38 (38%)</td>
</tr>
<tr>
<td>Horn</td>
<td>Paetzold</td>
<td>Paetzold-U</td>
<td>145 (60%)</td>
<td>30 (12%)</td>
<td>30 (46%)</td>
<td>5 (8%)</td>
<td>30 (46%)</td>
</tr>
<tr>
<td>Horn</td>
<td>Paetzold</td>
<td>Paetzold-S</td>
<td>145 (60%)</td>
<td>30 (12%)</td>
<td>42 (65%)</td>
<td>7 (11%)</td>
<td>16 (25%)</td>
</tr>
<tr>
<td>Glavas</td>
<td>None</td>
<td>Devlin</td>
<td>127 (53%)</td>
<td>12 (5%)</td>
<td>40 (40%)</td>
<td>16 (16%)</td>
<td>45 (45%)</td>
</tr>
<tr>
<td>Glavas</td>
<td>None</td>
<td>Horn</td>
<td>127 (53%)</td>
<td>12 (5%)</td>
<td>50 (57%)</td>
<td>13 (15%)</td>
<td>28 (32%)</td>
</tr>
<tr>
<td>Glavas</td>
<td>None</td>
<td>Glavas</td>
<td>127 (53%)</td>
<td>12 (5%)</td>
<td>46 (46%)</td>
<td>17 (17%)</td>
<td>38 (38%)</td>
</tr>
<tr>
<td>Glavas</td>
<td>None</td>
<td>Paetzold-U</td>
<td>127 (53%)</td>
<td>12 (5%)</td>
<td>49 (49%)</td>
<td>13 (13%)</td>
<td>39 (39%)</td>
</tr>
<tr>
<td>Glavas</td>
<td>None</td>
<td>Paetzold-S</td>
<td>127 (53%)</td>
<td>12 (5%)</td>
<td>56 (55%)</td>
<td>9 (9%)</td>
<td>36 (36%)</td>
</tr>
<tr>
<td>Glavas</td>
<td>Paetzold</td>
<td>Devlin</td>
<td>130 (54%)</td>
<td>14 (6%)</td>
<td>29 (30%)</td>
<td>17 (18%)</td>
<td>50 (52%)</td>
</tr>
<tr>
<td>Glavas</td>
<td>Paetzold</td>
<td>Horn</td>
<td>130 (54%)</td>
<td>14 (6%)</td>
<td>26 (27%)</td>
<td>18 (19%)</td>
<td>52 (54%)</td>
</tr>
<tr>
<td>Glavas</td>
<td>Paetzold</td>
<td>Glavas</td>
<td>130 (54%)</td>
<td>14 (6%)</td>
<td>37 (39%)</td>
<td>17 (18%)</td>
<td>42 (44%)</td>
</tr>
<tr>
<td>Glavas</td>
<td>Paetzold</td>
<td>Paetzold-U</td>
<td>130 (54%)</td>
<td>14 (6%)</td>
<td>24 (25%)</td>
<td>18 (19%)</td>
<td>54 (56%)</td>
</tr>
<tr>
<td>Glavas</td>
<td>Paetzold</td>
<td>Paetzold-S</td>
<td>130 (54%)</td>
<td>14 (6%)</td>
<td>40 (42%)</td>
<td>9 (9%)</td>
<td>47 (49%)</td>
</tr>
<tr>
<td>Paetzold</td>
<td>None</td>
<td>Devlin</td>
<td>115 (48%)</td>
<td>8 (3%)</td>
<td>59 (50%)</td>
<td>18 (15%)</td>
<td>40 (34%)</td>
</tr>
<tr>
<td>Paetzold</td>
<td>None</td>
<td>Horn</td>
<td>115 (48%)</td>
<td>8 (3%)</td>
<td>58 (50%)</td>
<td>15 (13%)</td>
<td>44 (38%)</td>
</tr>
<tr>
<td>Paetzold</td>
<td>None</td>
<td>Glavas</td>
<td>115 (48%)</td>
<td>8 (3%)</td>
<td>56 (48%)</td>
<td>25 (21%)</td>
<td>36 (31%)</td>
</tr>
<tr>
<td>Paetzold</td>
<td>None</td>
<td>Paetzold-U</td>
<td>115 (48%)</td>
<td>8 (3%)</td>
<td>48 (41%)</td>
<td>21 (18%)</td>
<td>48 (41%)</td>
</tr>
<tr>
<td>Paetzold</td>
<td>None</td>
<td>Paetzold-S</td>
<td>115 (48%)</td>
<td>8 (3%)</td>
<td>60 (51%)</td>
<td>16 (14%)</td>
<td>41 (35%)</td>
</tr>
<tr>
<td>Paetzold</td>
<td>Paetzold</td>
<td>Devlin</td>
<td>126 (52%)</td>
<td>9 (4%)</td>
<td>40 (38%)</td>
<td>21 (20%)</td>
<td>44 (42%)</td>
</tr>
<tr>
<td>Paetzold</td>
<td>Paetzold</td>
<td>Horn</td>
<td>126 (52%)</td>
<td>9 (4%)</td>
<td>42 (40%)</td>
<td>18 (17%)</td>
<td>45 (43%)</td>
</tr>
<tr>
<td>Paetzold</td>
<td>Paetzold</td>
<td>Glavas</td>
<td>126 (52%)</td>
<td>9 (4%)</td>
<td>45 (43%)</td>
<td>19 (18%)</td>
<td>41 (39%)</td>
</tr>
<tr>
<td>Paetzold</td>
<td>Paetzold</td>
<td>Paetzold-U</td>
<td>126 (52%)</td>
<td>9 (4%)</td>
<td>45 (43%)</td>
<td>21 (20%)</td>
<td>39 (37%)</td>
</tr>
<tr>
<td>Paetzold</td>
<td>Paetzold</td>
<td>Paetzold-S</td>
<td>126 (52%)</td>
<td>9 (4%)</td>
<td>39 (37%)</td>
<td>14 (13%)</td>
<td>52 (50%)</td>
</tr>
</tbody>
</table>

The results reveal that the approach with the highest G-score in the Complex Word Identification task of SemEval 2016 is the most reliable alternative among the ones evaluated. This observation supports the claim made by (Paetzold and Specia, 2016a) that the G-score is a better evaluator of an identifier’s quality in practice than the traditional F-score. The Supervised Paetzold Simplifier (Paetzold and Specia, 2016d; Paetzold and Specia, 2015) was the one to offer the most consistent results in our cumulative and non-cumulative analyses, but was outperformed by other combinations in our cumulative combinatory analysis. We have also found that including an effective Substitution Selection solution in the simplification process allows the ranker to make less mistakes, and hence avoid ungrammatical and/or incoherent replacements.

Comparing the automatic approach of PLUMBErr with the manual strategy of (Shardlow, 2014) shows many similarities between them, but suggests that a different annotation strategy for BenchLS might have increased its coverage, and consequently its reliability.

PLUMBErr’s analysis algorithms can be found in the LEXenstein framework⁴, while the BenchLS and NNSVocab datasets are available for downloaded⁵.

In future work, we aim to launch PLUMBErr 2.0, containing a more reliably annotated BenchLS, as well as a larger NNSVocab.

---

⁴http://ghpaetzold.github.io/LEXenstein/
⁵http://ghpaetzold.github.io/data/PLUMBErr.zip
References


Burns, P. R. (2013). Morphadorner v2: A java library for the morphological adornment of english language texts. Northwestern University, Evanston, IL.


How Hard Can it Be? The E-Score - A Scoring Metric to Assess the Complexity of Text

Sandeep Mathias, Pushpak Bhattacharyya
Department of Computer Science and Engineering
IIT Bombay, India
{sam.pb}@cse.iitb.ac.in

Abstract
In this paper, we present an evaluation metric, the E-Score, to calculate the complexity of text, that utilizes structural complexity of sentences and language modelling of simple and normal English to come up with a score that tells us how simple / complex the document is. We gather gold standard human data by having human participants take a comprehension test, in which they read articles from the English and Simple English Wikipedias. We use this data to evaluate our metric against a pair of popular existing metrics - the Flesch Reading Ease Score, and the Lexile Framework.

Keywords: text complexity, structural complexity, lexical complexity

1. Introduction
Today, there are many readability formulae that are used for evaluating the readability / complexity of text. Some of them, like the Flesch Reading Ease (Flesch, 1948) score (FRES) are based on surface values, like average words per sentence and average syllables per word. Others, like the Lexile Framework (Stenner, 1996) make use of the fact that rarer words are more complex than words that occur more commonly in a general corpus. The C-Score (Temnikova and Maneva, 2013) is yet another means of evaluating the difficulty of a text. However, unlike the Lexile Framework and Flesch Reading Ease, the C-Score is calculated using human readers produce the data necessary to calculate it. Yet, because the data used to calculated C-Score is gotten manually, it is one of the best metrics for getting gold standard data about the complexity of text. Because of this, we use the C-Score as the gold standard for estimating the complexity of the texts used in our experiment.

The Flesch Reading Ease is one of the earliest readability tests. In the Flesch Reading Ease score, higher valued texts are said to be simpler to read. This readability formula takes into account only the average number of syllables per word, and the average number of words in a sentence of the document.

The Lexile Framework (Stenner, 1996) makes use of the frequency of words in a training corpus, as well as the number of words in a sentence. It takes into account the mean of the log of the frequency of the word in a corpus, as well as the log of the mean sentence length.

Despite the fact that Lexile is a data-driven formula, it still suffers from criticism. Certain books, like The Library Mouse by Daniel Kirk have an abnormally high Lexile rating despite being a children’s book, as compared to a young adult book, Twilight by Stephanie Meyer.

More recently, (Schwarm and Ostendorf, 2005) demonstrated a means of classifying texts based on their complexity into appropriate grade levels. (Schwarm and Ostendorf, 2005) made use of support vector machines and language models and showed that it performed significantly better than FRES and Lexile when it came to assigning a grade-level for a document. While our work also makes use of language models, it differs from (Schwarm and Ostendorf, 2005) as it gives a raw score to the difficulty of the document, rather than the grade-level it is meant for.

2. The E-Score - Our Complexity Metric
To calculate the E-Score, we make use of two types of complexity, namely:
1. Structural complexity; and
2. Lexical complexity

2.1. Structural Complexity
Structural complexity is a measure of how complex the sentence is, based on its parse tree. There are many measures of defining structural complexity. We define structural complexity as follows for calculating the E-Score. For a given sentence S, we define the structural complexity $S_c$, as the number of factual statements extracted from Michael Heilman’s factual statement extractor (Heilman and Smith, 2010). A factual statement is a simple sentence that contains a single fact. For example, in the sentence

4. The system can be downloaded from www.cs.cmu.edu/~ark/mheilman/qg-2010-workshop
“Bernie Sanders, the Senator from Vermont, is campaigning against Hillary Clinton, the wife of former President Bill Clinton, to become the President of the United States.”

gives rise to the following factual statements:

- Bernie Sanders is the Senator from Vermont. (Appositive)
- Hillary Clinton is the wife of former President Bill Clinton. (Appositive)
- Bernie Sanders is campaigning against Hillary Clinton to become the President of the United States. (Main Clause)

The different types of simplified factual statements we extract from an input sentence are:

1. Main clause sentences
2. Factual statements from relative clauses
3. Factual statements from appositives
4. Factual statements from noun and verb participial phrases
5. Factual statements from other subordinate clauses

We use this definition of structural complexity because a sentence that is more complex would have more clauses in it that can be extracted into simpler factual statements.

2.2. Lexical Complexity

Lexical complexity is the complexity of the text based on its vocabulary. It is based on the complexity of the words and phrases used in the text. We use a unigram and bigram language model of a Simple English-English parallel corpus to calculate the lexical complexity of each n-gram. The complexity of an n-gram is comprised of 2 parts, namely the corpus complexity and the syllable count.

1. Corpus complexity For each n-gram \((g)\) of the sentence, we calculate its corpus complexity (Biran et al., 2011), \(C_c(g)\), defined as the ratio of the log likelihood of \(g\) in the English corpus to the log likelihood of \(g\) in the Simple English corpus. In other words,

   \[
   C_c(g) = \frac{LL(g|\text{normal})}{LL(g|\text{simpler})}
   \]

   Here, we assume that every n-gram in the Simple English corpus has to occur at least once in the English corpus. Section 4 contains more details about the corpus used.

2. Syllable count We consider that readers read words one syllable at a time. The syllable count, \(s(g)\), of an n-gram \((g)\) is defined as the sum of syllables of the words in that n-gram.

With these two ideas, we go ahead and calculate the lexical complexity of an n-gram \((g)\) as:

\[
Lc(g) = s(g) \times C_c(g)
\]

Hence, for a given sentence \(S\), and an n-gram size, the lexical complexity is given by

\[
Lc(S,n) = \sum_g s(g) \times C_c(g),
\]

where \(g\) is an n-gram of size \(n\).

In addition to this, we also attach a weight \(W_n\) to the lexical complexity calculated for a particular n-gram. For a given n-gram size of \(n\), the weight is \(\frac{1}{n}\). This is because of the unigrams in the n-gram are added n-times. For example, if \(n = 2\), and we have an n-gram sequence “a b c d e f g ...”, unigrams like b, c, d, e, f, etc. get added twice.

Therefore, we can say that the lexical complexity of a sentence is given by

\[
Lc(S) = \sum_n W_n \sum_g s(g) \times C_c(g).
\]

2.3. Calculating the E-Score

Both the structural complexity and the lexical complexity contribute to the overall complexity of the text. Hence, the formula used to calculate the E-Score is:

\[
E = \sum_{s\in S} \frac{S_r(s)+L_c(s)}{|S|}
\]

where \(S\) is the set of sentences in the text, and \(S_r\) and \(L_c\) are the structural and lexical complexities respectively.

3. Data

3.1. The C-Score

The C-Score (Temnikova and Maneva, 2013), unlike the earlier readability formulae is calculated using manual data. It is calculated based on participants taking a multiple choice comprehension test. It takes into account factors like number of correct answers that the participants got, the amount of time they took to read the passage, and the amount of time they took to solve the individual questions. Due to the vast differences in size of the individual articles (ranging from 84 words to 939 words), we allowed the participants to take as much time as they needed to read the articles (unlike (Temnikova and Maneva, 2013) which required participants to read them in a limited time), and normalized the C-Score based on reading time.

Like the Flesch Reading Ease Score, the C-Score is also a measure of simplicity. The higher the value, the simpler the text is. The formula for C-Score of a passage is

\[
C - \text{Score} = \frac{P_r \cdot T_r \cdot \sum_{q=1}^{N_q} Q_s(q) / t_{\text{mean}}(q)}{N_q}
\]

where \(C - \text{Score}\) is the C-Score of the passage, \(P_r\) is the percentage of correct answers, \(T_r\) is the size of the text, \(T_r\) is the mean time taken to read the text, \(N_q\) is the number of questions in the text, \(Q_s(q)\) is the size of question \(q\), and \(t_{\text{mean}}(q)\) is the mean time spent in answering question \(q\).

The question size is given by

\[
Q_s(q) = N_q(q) \times (L_q(q) + L_a(q))
\]

where \(N_q(q)\) is the number of options for question \(q\) and \(L_q\) and \(L_a\) are the lengths of the question and answers respectively.
3.2. Getting the Data

We set up a reading comprehension test in which participants had to read a set of 8 passages, alternating between Simple English\(^5\) and English\(^6\) Wikipedia articles. Since a few of the articles in the English Wikipedia were too long, only a small part was provided to the participants for reading. The topics of the passages chosen were generic in nature, such as art, culture, history, film, music, sports, science and world\(^7\). Table 1 shows the sizes of various passages.

<table>
<thead>
<tr>
<th>Passage</th>
<th>Simple</th>
<th>Normal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Art</td>
<td>520</td>
<td>939</td>
</tr>
<tr>
<td>Culture</td>
<td>235</td>
<td>705</td>
</tr>
<tr>
<td>History</td>
<td>196</td>
<td>342</td>
</tr>
<tr>
<td>Film</td>
<td>275</td>
<td>538</td>
</tr>
<tr>
<td>Music</td>
<td>373</td>
<td>284</td>
</tr>
<tr>
<td>Sports</td>
<td>174</td>
<td>381</td>
</tr>
<tr>
<td>Science</td>
<td>131</td>
<td>253</td>
</tr>
<tr>
<td>World</td>
<td>84</td>
<td>223</td>
</tr>
</tbody>
</table>

Table 1: Lengths of various passages

A total of 30 people took part in the experiment. Their educational qualifications ranged from high school graduates to PhD graduates. 19 of the participants were L2 English learners, while the rest were L1 English learners. 10 of them had won prizes in either the inter-school or intra-college level in literary activities like creative writing, quizzing, word games, scrabble, etc.

Each participant read 8 articles, alternating between Simple English and English Wikipedia articles. After reading each article, they had to answer 5 multiple choice questions (with 4 options each) on that passage. We measured the time taken to read the passages, as well as attempt each question for calculating the C-Score for various passages.

The results of the C-Score test are as shown in Table 2. In most cases, the normal shows a lower score than the simple (in Art, the ratio between simple to normal is more than 2). However, in a few cases, the C-Score of the simple article is lower than that of the normal. Film has the largest disparity, but so also does World. Film has a very high normal value and a lower simple value because of the fact that many respondents claimed to have knowledge of films, as compared to other fields (the number was nearly as much as Science). The Sports simple passage had a very long sentence at the end of it, that while structurally simple, had over 50 words. The World passage also showed the simple being harder than the normal. One of the main reasons is the fact that the size of the World “simple” passage was by far, the shortest passage.

Table 2: C-Score values of different passages

<table>
<thead>
<tr>
<th>Passage</th>
<th>Simple Score</th>
<th>Normal Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Art</td>
<td>45.81</td>
<td>22.68</td>
</tr>
<tr>
<td>Culture</td>
<td>43.09</td>
<td>49.11</td>
</tr>
<tr>
<td>History</td>
<td>59.13</td>
<td>38.13</td>
</tr>
<tr>
<td>Film</td>
<td>52.00</td>
<td>110.92</td>
</tr>
<tr>
<td>Music</td>
<td>55.18</td>
<td>37.07</td>
</tr>
<tr>
<td>Sports</td>
<td>38.02</td>
<td>68.26</td>
</tr>
<tr>
<td>Science</td>
<td>49.73</td>
<td>46.72</td>
</tr>
<tr>
<td>World</td>
<td>47.41</td>
<td>79.95</td>
</tr>
</tbody>
</table>

4. Experimental Setup

In the previous section, we described how to get the data against which we will be comparing our metric, as well as the FRES and Lexile scores. We make use of the English Wikipedia - Simple English Wikipedia (Kauchak, 2013) parallel corpus for calculating the corpus complexity of the n-grams. Since the corpus provides a sentence-aligned and a document-aligned corpus, we make use of the document-aligned corpus only for calculating the corpus complexity. The Simple English Wikipedia has around 60,000 articles, each with a corresponding English Wikipedia entry. The document-aligned corpus has all these Simple English Wikipedia articles as well as all their corresponding articles in the English Wikipedia. For each of the 16 articles (8 Simple English Wikipedia and 8 English Wikipedia articles), for which we calculated the C-Score, we calculate the E-Score, using:

1. Michael Heilman’s factual statement extractor (Heilman and Smith, 2010)
2. The unigram and bigram lexical complexities from the English Wikipedia - Simple English Wikipedia parallel corpus
3. MorphAdorner\(^8\) to count the syllables in each unigram and bigram

We also calculate the FRES and Lexile scores for each of the articles.

5. Results and Analysis

Table 3 shows the comparison of our metric, the E-score, with other metrics, such as Flesch Reading Ease Score and the Lexile Framework. The values in the table are to show how much more complex the English Wikipedia article is, with respect to the Simple English Wikipedia article.

We use the ratios, rather than the individual text values, because each of the different metrics give different ranges and directions for their scores. Flesch Readability Ease Score (Flesch, 1948) has a range between 0 and 120 (although individual sentences can have a negative value) and has simpler text getting a higher score. Lexile (Stenner, 1996) has a range between 0 and over 2000 and has more complex texts getting a higher score, unlike the C-Score. The E-Score has a range between 0 and about 2, also with more complex

\(^5\)http://simple.wikipedia.org
\(^6\)http://en.wikipedia.org
\(^7\)The Simple English article for art would be from http://simple.wikipedia.org/wiki/Art while that for the English Wikipedia article would be an extract from http://en.wikipedia.org/wiki/Art
\(^8\)http://morphadorner.northwestern.edu
texts getting a higher score. Therefore, in order to normalize the values for comparison, we take the ratio of complexity (i.e. how complex the English Wikipedia article is compared to the equivalent Simple English Wikipedia article). To see how close we are to the gold-standard ratios (ratio of the article’s Simple English Wikipedia C-Score value to that of its corresponding English Wikipedia C-Score value) that we got from our C-Score experiment, we use the following error metrics (lower is better).

1. \( S_0 = \frac{\sum_{i=1}^{n} |x_i|}{n} \cdot x_i = 0 \) if the metric is closest to the gold standard ratio and \( x_i = 1 \) otherwise. This measures percentage of the metric’s ratio not agreeing with that of the gold standard ratio.

2. \( S_1 = \frac{\sum_{i=1}^{n} (|metric_i - gold_i|)}{n} \) is the mean absolute error between the metric’s ratio and the gold standard ratio.

3. \( S_2 = \frac{\sum_{i=1}^{n} (|metric_i - gold_i|)^2}{n} \) is the mean square error between the metric’s ratio and the gold standard ratio.

\[
\begin{array}{|c|c|c|c|c|}
\hline
\text{Evaluation Metric} & \text{S0} & \text{S1} & \text{S2} \\
\hline
\text{Lexile} & 0.63 & 0.54 & 0.38 \\
\text{Flesch} & 0.88 & 0.87 & 1.05 \\
\text{E-Score} & \textbf{0.50} & \textbf{0.47} & \textbf{0.29} \\
\hline
\end{array}
\]

Table 4: Results of error analysis. Bold denotes the evaluation metric with the least error.

The Flesch Reading Ease Score assumes that the complexity of the text is dependent only on the sentence length and the number of syllables per word. It considers words like “automobile” and “procrastinate” to be of same complexity because both words have 4 syllables. With the use of data though, it can be shown that “automobile” is far more easier as compared to “procrastinate” (because “automobile” is on the Dale Chall Word List⁹, while “procrastinate” is not).

The Lexile Score makes use of a corpus, in which it assumes that the frequency of a word determines its simplicity / complexity. More frequent the word is, simpler it is.

While this is probably true in most cases, one of the issues is that it is corpus dependent. For instance, a medical corpus would have terms like disease names, drugs, etc. being as common / more common than common everyday phrases like “traffic light”. The E-Score outperforms the other two because it takes into account factors like corpus complexity, and syllable count. Corpus complexity gives a more precise measure than just frequency, of how complex an n-gram is, by measuring how much more probable it is in a parallel simplified corpus. Our metric’s measure of structural complexity also measures the fact a complex sentence is shown by having more information in it, as compared to just the number of words in it.

6. Conclusions

Using the document aligned English - Simple English Wikipedia Corpus, we are able to assign weights (i.e. the corpus complexity) to n-grams that occur in text, unlike FRES. We also look at quantities like corpus complexity (Biran et al., 2011) while assigning the complexity of a word, as well as the number of syllables, unlike Lexile, which only looks at the frequency of words in a given corpus. Our language modelling approach, in which we measure lexical complexity using n-grams, rather than just words is also an improvement over Lexile and FRES. If we were to, say, reorder the phrases of the sentence (so that we still end up with the same structural complexity), FRES and Lexile would give the same score, but our approach would give a different score, showing that the reordered sentence may be harder than the original. For example, the sentence, “Join the Dark Side, the boy will¹⁰” will give a different E-Score value, compared to the “The boy will join the Dark Side”. The FRES and Lexile scores for both sentences though will remain the same. Using structural complexity in our calculation of complexity is also better than that of the FRES and Lexile scores which take into account only the the number of words of the sentence and not its structure.

7. Bibliographical References


¹⁰Quote from Yodha in Star Wars Episode I: The Phantom Menace


Quality Estimation for Text Simplification

Sanja Štajner¹, Maja Popovic², Hanna Béchara³

¹Data and Web Science Group, University of Mannheim, Germany
²Humboldt University of Berlin, Germany
³Research Group in Computational Linguistics, University of Wolverhampton, UK
sanja@uni-mannheim.de, maja.popovic@hu-berlin.de, hanna.bechara@wlv.ac.uk

Abstract
The quality of the output generated by automatic Text Simplification (TS) systems is traditionally assessed by human annotators. In spite of the fact that the automation of that process would enable faster and more consistent evaluation, there have been almost no studies addressing this problem. We propose several decision-making procedures for automatic classification of the simplified sentences into three classes (bad, OK, good) depending on their grammaticality, meaning preservation, and simplicity. We experiment with ten different classification algorithms and 12 different feature sets on three TS datasets obtained using different text simplification strategies, achieving the results significantly above the state of the art. Additionally, we propose to use an unique measure (Total2 or Total3) for classifying the quality of the automatically simplified sentences into two (discard or keep) or three (bad, OK, good) classes.

Keywords: text simplification, automatic evaluation, quality estimation

1. Introduction
Text Simplification (TS) systems aim to transform complex sentences into their simpler variants, thus making them more understandable for various audiences: foreign language learners, children, and people with low literacy levels, cognitive or reading impairments, dyslexia, aphasia, autism, or congenital deafness. Usefulness of TS systems is usually assessed by measuring reading speed and comprehension by target users (Rello et al., 2013; Fajardo et al., 2014). However, as access to the target users might be difficult, it is common to first assess the quality of the generated sentences in terms of their grammaticality, meaning preservation and simplicity on the sentence level, by human annotators (Woodsend and Lapata, 2011; Saggion et al., 2015).

Human evaluation of the quality of automatically simplified sentences has several shortcomings. First, it is costly and time consuming. Ideally, it requires native speakers with linguistic knowledge for the evaluation of grammaticality and meaning preservation, and native speakers familiar with the specific needs of each target population for the evaluation of the simplicity of generated sentences. In practice, neither of the two conditions is usually satisfied. That introduces a new problem. The performances of different TS systems cannot be directly compared, as the human annotators differ from one study to another. So far, the annotators have been unpaid native speakers (Woodsend and Lapata, 2011), native speakers from Mechanical Turk (Siddharthan and Angrosh, 2014), and non-native speakers with high command of the required language (Glavaš and Štajner, 2013). Finally, direct comparison of performances of different TS systems requires direct access to those systems and repetition of their evaluation, in order to evaluate them under the same conditions (using the same guidelines and the same annotation scale) and by the same human annotators.

Automatic evaluation of the quality of automatically simplified sentences would provide a faster and more consistent evaluation, and allow a fairer comparison of different TS systems. However, there have been almost no studies addressing this problem.

We experiment with 10 classification algorithms and 12 feature sets – combining the state-of-the-art machine translation (MT) evaluation metrics and the machine translation quality estimation features – and investigate which of them lead to the best performances in several classification tasks. We approach the problem as a quality estimation task, comparing the original sentence with its automatically simplified version, without need for the manual simplification ‘gold standard’. The proposed classifiers and feature sets show good results on three TS datasets, obtained using different simplification strategies (MT-based TS, event-based TS, and unsupervised lexical simplification based on word-embeddings), and they outperform the state of the art (Štajner et al., 2014). Furthermore, we propose the use of an unique score for classifying the quality of the automatically simplified sentences into two (discard or keep) or three (bad, OK, good) classes, taking into account their grammaticality, meaning preservation, and simplicity, all at the same time.

2. Related Work
Automatic evaluation and quality estimation have attracted a lot of attention in machine translation. However, only one study addressed this issue in text simplification in spite of the many similarities of those tasks. Štajner et al. (2014) showed that the three most widely used MT evaluation metrics – BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2011) and TER (Snover et al., 2006), as well as TINE (Rios et al., 2011) – have good correlation with human judgements of grammaticality and meaning preservation, and suggested the use of a unique score for measuring the quality of the automatically

¹In some studies, e.g. (Woodsend and Lapata, 2011; Glavaš and Štajner, 2013), simplicity is additionally evaluated on a text level using readability metrics.
simplified sentences (taking into account only their grammaticality and meaning preservation). The main limitation was that the experiments were conducted on a very particular TS corpus consisting of sentences simplified using only syntactic simplification and considerable content reduction. Therefore, it remained unclear whether the same approach would work equally well on a more typical TS dataset consisting of lexically simplified sentences with almost no content reduction (the most common case in TS). In this paper, we experiment with a large number of features borrowed from MT evaluation and quality estimation tasks on three TS datasets obtained using very different simplification strategies. The main advantage of the features we propose is that they can easily be computed for other languages as well, as they do not require resources which are not available for languages other than English (though some of them require a POS tagger or/and lemmatiser) in contrast to the TINE metric used in the previous study (Štajner et al., 2014).

3. Methodology

The datasets, features, and experimental setup are presented in the next three subsections.

3.1. Datasets

We used three TS datasets obtained by applying different simplification strategies:

1. EventS dataset, previously used by Štajner et al. (2014), consists of 272 original sentences and their syntactically simplified versions (with significant content reduction) obtained by using the EventSimplify TS system (Glavaš and Štajner, 2013).2

2. EBritMT dataset consists of 120 original sentences from the Encyclopedia Britannica (Barzilay and El-hadad, 2003) and their automatic simplifications obtained by using several MT-based (phrase-based statistical machine translation) TS systems trained on Wikipedia TS corpus (Štajner et al., 2015).

3. LSLight dataset consists of 240 original sentences from English Wikipedia and their automatic simplifications obtained by using three different lexical simplification systems (Glavaš and Štajner, 2015).

The marks in the EventS dataset were given on a 1–3 scale, and in the other two datasets on a 1–5 scale (in both cases, the higher the mark the better). Therefore, we converted the marks of the EBritMT and LSLight datasets onto a 1–3 scale (1; 2 → 1; 3 → 2; 4, 5 → 3), in order to have the same scale in all three datasets. As the final score for grammaticality (G), meaning preservation (M), and simplicity (S) we used the arithmetic mean of the marks given by all annotators (rounded to an integer). Following the idea presented by Štajner et al. (2014) to have a unique score for evaluating the quality of the generated output, we calculated the Total3 (eq. 1) and Total2 (eq. 2) scores for each sentence x, but this time also taking into the account the simplicity score.

\[
\text{Total3}(x) = \begin{cases} 
3 & \text{if } G(x) = M(x) = S(x) = 3 \\
1 & \text{if } M(x) = 1 \text{ or } S(x) = 1 \\
2 & \text{otherwise}
\end{cases}
\]  

\[
\text{Total2}(x) = \begin{cases} 
0 & \text{if } \text{Total3}(x) = 1 \\
1 & \text{otherwise}
\end{cases}
\]

We opted for Total3 and Total2 scores which take into account the simplicity score, for two reasons: (1) we wanted to penalise the automatically simplified sentences which are not simpler than their originals and the automatically simplified sentences which are exactly the same as their originals (which is the common case in all, and especially MT-based, TS systems); and (2) because we noticed that newly proposed feature sets lead to reasonably good results on the Simplicity classification task.

3.2. Features

We experimented with a total of 39 features (Table 1), combining 22 MT evaluation features and the 17 baseline MT quality estimation (QuEst) features (Specia et al., 2009). All MT evaluation metrics were calculated using original sentences as references and automatically simplified sentences as their translation hypotheses (our approach does not require any ‘gold standard’ manually simplified sentences). The first three features (1–3) can be seen as baseline MT evaluation features, as they were used in the previous study (Štajner et al., 2014).3 We further included the separate fine-grained components of the TER metric (features 4–10) and the following additional MT evaluation metrics:

- n-gram F1 scores based on words, characters, base forms, morphemes and POS tags (features 11–15);
- class error rates (features 16–22):
  - the five class error rates on the word level produced by Hjerson error classification tool (Popović, 2011a) (features 16–20);
  - the sum of the five class error rates on the word level (feature 21);
  - the sum of the five class error rates on the block level (feature 22) where a block is defined as a group of consecutive words belonging to the same error class.

Both n-gram F metrics (Popović, 2011b) and sums of class error rates (Popović, 2012) have shown good correlations with human rankings of MT outputs in the WMT4 shared evaluation tasks.

The quality estimation (QE) feature set includes surface features such as the number of tokens, the average token length, n-gram frequencies, language model probabilities, and translations per source word. We used the open-source feature extractor (Specia et al., 2013) in order to obtain these features (Table 2).

---

2http://takelab.fer.hr/evsimplify

3We included only the features which showed the highest correlation with the human judgements.

4Worshop on Statistical Machine Translation (http://www.statmt.org/wmt15/)
<table>
<thead>
<tr>
<th>Group</th>
<th>#</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseMT</td>
<td>1</td>
<td>S-BLEU</td>
<td>Sentence-wise BLEU score (Papineni et al., 2002)</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>METEOR</td>
<td>METEOR score (Lavie and Denkowski, 2010)</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>TER</td>
<td>Translation Edit Rate (Snover et al., 2006)</td>
</tr>
<tr>
<td>TERc</td>
<td>4</td>
<td>Ins</td>
<td># of insertions (TER component)</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Sub</td>
<td># of substitutions (TER component)</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>Del</td>
<td># of deletions (TER component)</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>Shift</td>
<td># of block shifts (TER component)</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>WdSh</td>
<td># of word shifts (TER component)</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>NumErr</td>
<td># of block errors (TER component)</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>NumWd</td>
<td># of word errors (TER component)</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>WordF</td>
<td>word 4-gram F1 score</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>ChrF</td>
<td>character 6-gram F1 score</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>BaseF</td>
<td>lemma 4-gram F1 score</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>MorphF</td>
<td>morpheme 4-gram F1 score</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>PosF</td>
<td>POS 4-gram F1-score</td>
</tr>
<tr>
<td>Err</td>
<td>16</td>
<td>LexErr</td>
<td>lexical error rate</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>ExtErr</td>
<td>addition error rate</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>MissErr</td>
<td>omission error rate</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>RErr</td>
<td>reordering error rate</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>InfErr</td>
<td>inflection error rate</td>
</tr>
<tr>
<td></td>
<td>21</td>
<td>wΣerr</td>
<td>sum of word level error rates</td>
</tr>
<tr>
<td></td>
<td>22</td>
<td>bΣerr</td>
<td>sum of block level error rates</td>
</tr>
<tr>
<td>QE</td>
<td>23–39</td>
<td>QuEst</td>
<td>17 baseline Quality Estimation features</td>
</tr>
</tbody>
</table>

Table 1: Features (baseMT – baseline MT evaluation metrics used by Štajner et al. (2014); TERc – components of the TER metric; F – b-gram F metrics (Popović, 2011b); Err – sums of class error rates (Popović, 2012); QE – 17 quality estimation baseline features (Specia et al., 2009))

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Number of tokens in the source sentence</td>
</tr>
<tr>
<td>2</td>
<td>Number of tokens in the target sentence</td>
</tr>
<tr>
<td>3</td>
<td>Average source token length</td>
</tr>
<tr>
<td>4</td>
<td>LM probability of source sentence</td>
</tr>
<tr>
<td>5</td>
<td>LM probability of the target sentence</td>
</tr>
<tr>
<td>6</td>
<td>Average number of occurrences of the target word within the target sentence</td>
</tr>
<tr>
<td>7</td>
<td>Average number of translations per source word in the sentence (as given by IBM 1 table thresholded so that ( \text{prob}(t</td>
</tr>
<tr>
<td>8</td>
<td>Average number of translations per source word in the sentence (as given by IBM 1 table thresholded so that ( \text{prob}(t</td>
</tr>
<tr>
<td>9</td>
<td>Percentage of unigrams in quartile 1 of frequency (lower frequency words) in a corpus of the source language (SMT training corpus)</td>
</tr>
<tr>
<td>10</td>
<td>Percentage of unigrams in quartile 4 of frequency (higher frequency words) in a corpus of the source language</td>
</tr>
<tr>
<td>11</td>
<td>Percentage of bigrams in quartile 1 of frequency of source words in a corpus of the source language</td>
</tr>
<tr>
<td>12</td>
<td>Percentage of bigrams in quartile 4 of frequency of source words in a corpus of the source language</td>
</tr>
<tr>
<td>13</td>
<td>Percentage of trigrams in quartile 1 of frequency of source words in a corpus of the source language</td>
</tr>
<tr>
<td>14</td>
<td>Percentage of trigrams in quartile 4 of frequency of source words in a corpus of the source language</td>
</tr>
<tr>
<td>15</td>
<td>Percentage of unigrams in the source sentence seen in a corpus (SMT training corpus)</td>
</tr>
<tr>
<td>16</td>
<td>Number of punctuation marks in source sentence</td>
</tr>
<tr>
<td>17</td>
<td>Number of punctuation marks in target sentence</td>
</tr>
</tbody>
</table>

Table 2: The 17 baseline quality estimation (QE) baseline features (Specia et al., 2009)

3.3. Experiments

We performed five sets of experiments. Four of them were classification experiments into three classes (bad, OK, or good) according to the \( G, M, S \) or Total3 score. The fifth was the classification experiment into two classes (discard or keep) according to the Total2 score. Each set of experiments was performed on each of the three datasets separately, using 10 classification algorithms implemented in Weka Experimenter (Hall et al., 2009): Logistic (de Cessie and van Houwelingen, 1992), Simple Logistic (Sumner et
Table 3: Weighted F-measure (Random Forest for all except the last two rows). The results which are significantly different (p<0.5, paired t-test) from those achieved by the baseline (baseMT) are presented in bold. The highest F-measure achieved on each dataset and for each task is presented with an ‘*’. The State of the art row contains the best results obtained by Štajner et al. (2014).

Table 4: Weighted F-measure (Random Forest for all except the last row). The results which are significantly different (p<0.5, paired t-test) from those achieved by the baseline (baseMT) are presented in bold. The highest F-measure achieved on each dataset and for each task is presented with an ‘*’.

4. Results and Discussion

We performed all experiments using all 10 classification algorithms. Random Forest performed equally good, or better than, all other algorithms on all feature sets and on all classification tasks. Therefore, we present only the results achieved using this algorithm (Tables 3 and 4).

The use of Random Forest algorithm and the newly proposed feature set – the combination of n-gram F1 scores, class error rates and fine-grained components of the TER (F+Err+TERc) – led to significantly better classification performance than the state of the art (ˇStajner et al., 2014) for the Grammaticality (G) task. The same algorithm applied on the feature set combining only n-gram F1 scores and class error rates (F+Err) significantly outperformed the state of the art for the Meaning preservation (M) task as well. The combination of n-gram F1 scores and class error rates

Note that Štajner et al. (2014) did not report on the results of the Simplicity (S) task, their definition of Total2 and Total3 scores differed from ours as they did not take into account the simplicity score, and they performed experiments only on the EventS dataset.
(F+Err) appears to be the best choice on the M task not only for the EventS dataset but also for the EBriMT dataset. The LSLight dataset has heavily unequal distribution of classes on the G and M score and thus none of the feature sets can significantly outperform the majority class baseline.

For the Simplicity (S) task, the QE feature set performed the best on the EventS and EBriMT datasets, while the best results on the LSLight dataset were achieved by using the combination of all newly proposed features (F+Err+TERc+QE).

On the newly proposed classification tasks, using the unique quality score (Total3 or Total2), the QE feature set led to the best results on the EBriMT dataset on both tasks. On the EventS dataset, the combination of n-gram F1 scores and class error rates (F+Err) led to the best results for the classification into two classes (discard or keep), while the classification into three classes (bad, OK, good) was improved by combining this feature set with the fine-grained components of TER (F+Err+TERc).

5. Participation in Shared Task

We participated in the shared task on Quality Assessment for Text Simplification (QATS)\(^6\), proposing three classification systems which used the above-mentioned features.

5.1. Shared Task Description

The shared task consisted in proposing systems which are able to correctly classify automatically simplified sentences into three classes (good, ok, and bad) according to their grammaticality, meaning preservation, simplicity and overall quality.

The training set consisted of 505 sentence pairs with manually assigned (‘gold standard’) scores for all four aspects (grammaticality, meaning preservation, simplicity, and overall quality), while the test set consisted of 126 sentence pairs without ‘gold standard’ labels.

The participating systems were ranked according to their accuracy, mean absolute error, root mean squared error, and weighted F-score. Six baseline classifiers were provided: the majority-class baseline, an SVM classifier trained on four MT evaluation metrics (BLEU, METEOR, WER, and TER), as well as the four baselines based on mean value and standard deviation of each of those MT evaluation metrics (Štajner et al., 2016).

5.2. Results on the Training Dataset

Using the full set of 39 aforementioned MT evaluation metrics and QE features, for each of the four aspects (G, M, S, and Overall), we trained eight different classifiers implemented in Weka Experiment (Hall et al., 2009):

1. Logistic – Logistic Regression (le Cessie and van Houwelingen, 1992)
2. NB – Naïve Bayes (John and Langley, 1995)
3. SVM-n – Support Vector Machines with feature normalisation
4. SVM-s – Support Vector Machines with feature standardisation
5. IBk – K-nearest neighbours (Aha and Kibler, 1991)
6. JRip – a propositional rule learner (Cohen, 1995)
7. J48 – C4.5 decision tree (Quinlan, 1993)
8. RandF – Random Forest (Breiman, 2001)

All experiments were run in a 10-fold cross-validation setup with 10 repetitions, using the provided training dataset of 505 sentence pairs.\(^7\) The results are presented in Table 5. The Random Forest classification algorithm achieved the best weighted F-score for three tasks (G, M, and S), while the logistic regression obtained the best result on the fourth task (Overall). All eight classifiers outperformed the majority-class baseline. The two best systems for each aspect were submitted to the shared task.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>G</th>
<th>M</th>
<th>S</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic</td>
<td>0.716</td>
<td>0.691</td>
<td>0.608</td>
<td>0.594*</td>
</tr>
<tr>
<td>Naïve Bayes</td>
<td>0.671</td>
<td>0.633</td>
<td>0.565</td>
<td>0.534</td>
</tr>
<tr>
<td>SVM-n</td>
<td>0.656</td>
<td>0.671</td>
<td>0.597</td>
<td>0.582</td>
</tr>
<tr>
<td>SVM-s</td>
<td>0.706</td>
<td>0.677</td>
<td>0.584</td>
<td>0.549</td>
</tr>
<tr>
<td>IBk</td>
<td>0.747</td>
<td>0.644</td>
<td>0.613</td>
<td>0.588</td>
</tr>
<tr>
<td>JRip</td>
<td>0.708</td>
<td>0.639</td>
<td>0.603</td>
<td>0.515</td>
</tr>
<tr>
<td>J48</td>
<td>0.745</td>
<td>0.675</td>
<td>0.585</td>
<td>0.529</td>
</tr>
<tr>
<td>RandF</td>
<td>0.754*</td>
<td>0.708*</td>
<td>0.614*</td>
<td>0.589</td>
</tr>
<tr>
<td>Majority class</td>
<td>0.652</td>
<td>0.363</td>
<td>0.428</td>
<td>0.240</td>
</tr>
</tbody>
</table>

Table 5: Results of the classification experiments. The two best results (weighted F-measure) for each aspect (G, M, S, and Overall) are presented in bold, and the best of the two is marked with an ‘*’.

Additionally, we used the CfsSubsetEval feature selection algorithm (Hall and Smith, 1998) implemented in Weka toolkit to select only the subset of the best features for each aspect and submit the third system for the shared task (RandF classification algorithm which uses only that subset of features). The CfsSubsetEval algorithm returned the following sets of features as the best subsets:\(^8\):

- For grammaticality (20 features): all baseMT features (BLEU, METEOR, TER), all F features (WordF, ChrF, BaseF, MorphF, PosF), two Err features (bΣerr and RErr), seven baseline QE features (# 1, 2, 4, 8, 10, 12, 14), and three TERc features (Del, NumErr, and NumWd);
- For meaning preservation (23 features): all baseMT features (BLEU, METEOR, TER), all F features (WordF, ChrF, BaseF, MorphF, PosF), three Err features (wΣerr, bΣerr, and RErr), eight baseline QE features (# 1, 2, 4, 5, 7, 8, 12, 15), and three TERc features (Del, WdSh, and NumWd);

---

\(^6\)http://qats2016.github.io/shared.html

\(^7\)http://qats2016.github.io/shared.html

\(^8\)For the full description of features see Section 3.2. and Tables 1 and 2.
Table 6: Results on the shared task (performances better than those of the baseline are presented in bold and the best achieved results are marked with an "*""). The first run (Run 1) contains the results of the Logistic classification algorithm for Grammaticality and Simplicity, and the results of the IBk classification algorithm for Meaning preservation and Overall score.

<table>
<thead>
<tr>
<th>System</th>
<th>Grammaticality accuracy</th>
<th>Grammaticality weighted-F</th>
<th>Meaning accuracy</th>
<th>Meaning weighted-F</th>
<th>Simplicity accuracy</th>
<th>Simplicity weighted-F</th>
<th>Overall accuracy</th>
<th>Overall weighted-F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run 1 (IBk/Logistic)</td>
<td>70.63</td>
<td>0.716</td>
<td>69.05</td>
<td>0.681</td>
<td>50.00</td>
<td>0.511</td>
<td>47.62</td>
<td>0.475</td>
</tr>
<tr>
<td>Run 2 (RandF)</td>
<td>75.40</td>
<td>0.718</td>
<td>65.87</td>
<td>0.644</td>
<td>52.38</td>
<td>0.530</td>
<td>44.44</td>
<td>0.445</td>
</tr>
<tr>
<td>Run 3 (RandF-best)</td>
<td>75.40</td>
<td>0.700</td>
<td>61.90</td>
<td>0.597</td>
<td>57.14</td>
<td>0.564</td>
<td>48.41</td>
<td>0.486</td>
</tr>
<tr>
<td>Majority class</td>
<td>76.19</td>
<td>0.659</td>
<td>57.94</td>
<td>0.425</td>
<td>55.56</td>
<td>0.397</td>
<td>43.65</td>
<td>0.265</td>
</tr>
</tbody>
</table>

- For simplicity (15 features): two F features (ChrF and MorphF), 12 QE features (# 1–6, 8, 9, 13, 14, 16, 17), and one TERc feature (NumWd);
- For overall quality (19 features): two baseMT features (BLEU and TER), one F feature (PosF), one Err feature (RErr), 12 QE features (# 1–8, 11, 13, 16, 17), and three TERc features (Shift, NumErr, and NumWd).

5.3. Results on the Test Dataset

The achieved accuracy and weighted F-measure of our three systems submitted to the shared task dataset are presented in Table 6 together with the results obtained by the majority-class baseline. As can be noted, all our classifiers achieved a higher weighted F-score than the majority-class baseline for all four tasks. On the Meaning preservation and Overall tasks, our classifiers also obtained higher accuracy than the majority-class baseline.

Compared to all other participating systems in the shared task (Štajner et al., 2016), our systems achieved the best weighted F-score results for all four tasks, and the best accuracy on three out of four tasks (all except Overall):

- On the Grammaticality task, our three systems were the three best ranked systems, and the only systems which obtained higher weighted F-score than all six official baselines.
- On the Meaning preservation task, our Logistic classifier was the only classifier that achieved higher accuracy than the majority-class baseline, and the best ranked classifier according to the weighted F-score.
- On the Simplicity task, our Random Forest classifier trained on the subset of initial features (the best subset of features returned by the CfsSubsetEval feature selection algorithm) was the only classifier that achieved higher accuracy than the majority-class baseline.
- On the Overall quality task, our Random forest classifier trained on the best subset of initial features was ranked first according to the weighted F-score, and third according to the accuracy score. In both cases, it outperformed all six official baselines.

6. Conclusions

We presented several decision-making procedures for automatic evaluation of TS systems, proposing the use of 22 fine-grained MT evaluation metrics and 17 baseline MT quality estimation features, which could also be easily calculated for languages other than English.

Our approach reported promising results on three different TS datasets, showing the potential of our approach being used for automatic evaluation of various TS systems. The experiments also indicated that, depending on the type of the dataset that needs to be evaluated (whether it was obtained by MT-based TS, event-based TS, or unsupervised lexical simplification), different feature sets lead to the best results.

On the Shared Task on Quality Assessment for Text Simplification (QATS), our classification systems which use the newly proposed 39 features were ranked best among all participating systems.

Acknowledgements

Hanna Béchara is supported by the People Programme (Marie Curie Actions) of the European Union’s Framework Programme (FP7/2007-2013) under REA grant agreement 317471.

7. References


Workshop held in conjunction with RANLP 2013, Hiszar, Bulgaria, pages 71–78.


Shared Task on Quality Assessment for Text Simplification

Sanja Štajner¹, Maja Popović², Horacio Saggion³, Lucia Specia⁴, Mark Fishel⁵

¹University of Mannheim, Germany, ²Humboldt University of Berlin, Germany
³Universitat Pompeu Fabra, Spain, ⁴University of Sheffield, UK, ⁵University of Tartu, Estonia

Abstract

This paper presents the results of the shared task of the Workshop on Quality Assessment for Text Simplification (QATS), which consisted in automatically assigning one of the three labels (good, ok, and bad) for each of the four aspects of automatically simplified English sentences, i.e. their grammaticality, meaning preservation, simplicity, and overall quality. We asked participants to submit a maximum of three systems (raw metrics and/or classifiers) for each aspect. We received a total of 10 raw metrics and 16 classifiers for each of the four aspects. In addition to that, we computed correlations for four standard MT metrics (BLEU, METEOR, TER and WER) as baselines. The collected scores were evaluated by Pearson correlation (how well each score metric correlates with the manually assigned values) and the classifiers were evaluated in terms of their accuracy, mean average error, root squared mean error and weighted F-scores.

Keywords: automatic text simplification, quality assessment, automatic evaluation

1. Introduction

In recent years, there has been an increasing interest in automatic text simplification (ATS) and text adaptation to various target populations (e.g. non-native speakers, children, people with low literacy or cognitive disabilities). Various ATS systems have been proposed for many languages, e.g. English (Angrosh et al., 2014; Glavaš and Štajner, 2015), Portuguese (Specia, 2010), Spanish (Saggion et al., 2015; Štajner et al., 2015b), French (Brouwers et al., 2014), Italian (Barlacchi and Tonelli, 2013), and Basque (Aranzabe et al., 2012).

ATS systems have usually been evaluated for the quality of the generated output by one (or a combination) of the following:

- Readability metrics (on a text level), e.g. (Zhu et al., 2010; Woodsend and Lapata, 2011; Glavaš and Štajner, 2013; Saggion et al., 2015)

- Human assessment of grammaticality (G), meaning preservation (M), and simplicity (S) on sentence level, e.g. (Woodsend and Lapata, 2011; Glavaš and Štajner, 2013; Saggion et al., 2015)

- Machine translation (MT) evaluation metrics, such as BLEU (Papineni et al., 2002), NIST (Doddington, 2002), or TER (Snover et al., 2006) in case of MT-based ATS systems (Specia, 2010; Zhu et al., 2010; Woodsend and Lapata, 2011; Coster and Kauchak, 2011a; Štajner et al., 2015b)

Readability metrics are not sufficient on their own as they do not take into account the grammaticality and meaning of the sentence. Instead, they rely on the shallow features, such as average sentence length (in words) and average word length (in syllables or characters). Therefore, an ATS system which produces very short ungrammatical and meaningless sentences could be rated very highly by most readability metrics.

The MT evaluation metrics also have several significant drawbacks when used on their own. First, they require ‘gold standard’ simplifications which are not always available (and when they are, there is usually only one ‘gold standard’ provided for each sentence). Second, they penalise word reordering and sentence shortening, which are common operations in TS. Third, it seems that they are not a good measure of systems performance as they heavily depend on the type of manual simplification performed on the ‘gold standard’ test set (Štajner et al., 2015a).

The most reliable type of ATS evaluation is, thus, the human assessment of the generated output in three aspects: grammaticality, meaning preservation and simplicity. In this case, the evaluators – usually native speakers (for evaluating grammaticality and meaning preservation) or target users (for evaluating simplicity) – are presented with a sentence (original or simplified) and asked to assess its grammaticality and simplicity on a 1–3 or 1–5 scale (this varies from study to study, but the most common is to use a 1-5 scale), where 1 denotes the worst (i.e. completely ungrammatical or very complex) and 3 or 5 denotes the best (i.e. completely grammatical or very simple). In order to assess the meaning preservation, the annotators are presented with a pair of sentences (original and simplified) and asked to rate (on a 1–3 or 1–5 scale) how similar is the meaning of the sentences, where 1 denotes very different (or opposite) meaning and 5 denotes that both sentences have the same meaning.

As any other human evaluation, this is a costly and time-consuming task. Additionally, it is difficult to directly compare the quality of different ATS systems if they were assessed by different annotators. Therefore, automatic methods are needed in order to provide a faster and more consistent evaluation.

Štajner et al. (2014) investigated the possibility of replacing the human assessment of grammaticality and meaning preservation with a combination of cosine similarity and several MT evaluation metrics: BLEU (Papineni et
al., 2002), METEOR (Denkowski and Lavie, 2011), and TINE (Rios et al., 2011), showing promising results on one of the datasets we used in this shared task (EventS).

Inspired by that study and the close relationship between the problem of automatic evaluation of ATS systems and well-studied problems of automatic evaluation and quality estimation in machine translation (MT), this shared task focused on automatic evaluation (quality assessment) of ATS systems. The goal of the shared task was to bring together researchers working on ATS and those working on automatic evaluation and quality estimation of machine translation output, who could try to adapt their metrics to this closely related task. It provided an opportunity to make important first steps in establishing metrics for automatic evaluation of ATS systems which enable direct comparison of the quality of the generated outputs, as well as less time consuming assessment of each ATS system.

2. Datasets

The shared task dataset was a combination of the following three datasets:

- **EventS**: 272 original sentences from the EMM News-Brief\(^1\) and their syntactically simplified versions (with significant content reduction) obtained by using the EventSimplify TS system (Glavaš and Štajner, 2013).\(^2\)

- **EncBrit**: 119 original sentences from the Encyclopædia Britannica (Barzilay and Elhadad, 2003) and their automatic simplifications obtained by using ATS systems based on several phrase-based statistical machine translation systems (Štajner et al., 2015a) trained on Wikipedia TS corpus (Coster and Kauchak, 2011b).

- **LSLight**: 240 original sentences from English Wikipedia and their automatic simplifications (Glavaš and Štajner, 2015) obtained by using three different lexical simplification systems (Biran et al., 2011; Horn et al., 2014; Glavaš and Štajner, 2015).

The training and test sets used in the shared task contain sentence pairs (original, simplified) from all three sets proportionally. The statistics is shown in Table 1.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>EventS</th>
<th>EncBrit</th>
<th>LSLight</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>218</td>
<td>95</td>
<td>192</td>
<td>505</td>
</tr>
<tr>
<td>Test</td>
<td>54</td>
<td>24</td>
<td>48</td>
<td>126</td>
</tr>
</tbody>
</table>

Table 1: Datasets statistics (number of sentence pairs).

Each simplified sentence in the training dataset contained human scores (good, ok, bad) for each of the following three aspects\(^3\):

- **Grammaticality (G)**, where: bad = ungrammatical, ok = somewhat ungrammatical but the mistakes do not impede understanding, and good = completely grammatically correct;

- **Meaning preservation (M)**, where: bad = no meaning at all or completely opposite meaning from the original, ok = somewhat changed nuance of meaning or missing an unimportant part, and good = preserved original meaning;

- **Simplicity (S)**, where: bad = very difficult to understand, ok = somewhat difficult to understand, and good = easy to understand.

For each simplified sentence in the training dataset, we also provided an overall score, Overall (O), that represents a combination of the previous three scores and rewards more meaning preservation and simplicity than grammaticality. It should help decide whether the automatically simplified sentence is ready to be presented to a final user (good), needs post-editing (ok), or should better be discarded and simplified using some different technique or left in the original form (bad). The formula used for the overall score is given in Equation 1:

\[
O(x) = \begin{cases} 
  \text{good} & \text{if } G(x) = M(x) = S(x) = \text{good} \\
  \text{bad} & \text{if } M(x) = \text{bad} \text{ or } S(x) = \text{bad} \\
  \text{ok} & \text{otherwise} 
\end{cases}
\]

(1)

The distribution of classes in the training and test sets can be seen in Table 2. Several examples of the sentence pairs from the training datasets are presented in Table 3.

3. Automatic Evaluation

The shared task participants were asked to submit their systems in the form of assigned classes (for the classification task) and/or raw continuous numerical scores (for the metrics task).

3.1. Raw Score Metrics

Raw score metrics were assessed using Pearson’s correlation coefficients:

\[
r = \frac{\sum_{i=1}^{N} (a_i - \overline{a})(g_i - \overline{g})}{\sqrt{\sum_{i=1}^{N} (a_i - \overline{a})^2} \sqrt{\sum_{i=1}^{N} (g_i - \overline{g})^2}}
\]

(2)

where \(a_i\) is the automatically assigned class for the \(i\)-th sentence, \(g_i\) is the ‘gold standard’ class (i.e. manually assigned class) for this sentence, \(\overline{a}\) and \(\overline{g}\) are respective mean values, and \(N\) is the total number of sentences.

The Pearson’s correlation coefficient is used to measure the strength of a linear association between two variables, where the value \(r = 1\) means a perfect positive correlation and the value \(r = -1\) means a perfect negative correlation.
Table 2: Distribution of classes for each aspect (percentages).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>G</th>
<th>M</th>
<th>S</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>75.6</td>
<td>14.3</td>
<td>10.1</td>
<td>58.2</td>
</tr>
<tr>
<td>Test</td>
<td>76.2</td>
<td>11.1</td>
<td>12.7</td>
<td>57.9</td>
</tr>
<tr>
<td>EventS</td>
<td>62.9</td>
<td>21.3</td>
<td>15.8</td>
<td>60.3</td>
</tr>
<tr>
<td>EncBrit</td>
<td>64.7</td>
<td>18.5</td>
<td>16.8</td>
<td>63.0</td>
</tr>
<tr>
<td>LSLight</td>
<td>95.8</td>
<td>2.5</td>
<td>1.7</td>
<td>85.0</td>
</tr>
</tbody>
</table>

Table 3: Examples from the training dataset (differences between the original and simplified versions are presented in bold)

<table>
<thead>
<tr>
<th>Version</th>
<th>Sentence</th>
<th>G</th>
<th>M</th>
<th>S</th>
<th>O</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>All three were arrested in the Toome area and have been taken to the Serious Crime Suite at Antrim police station.</td>
<td>good</td>
<td>good</td>
<td>good</td>
<td>good</td>
<td>syntactic</td>
</tr>
<tr>
<td>Simple</td>
<td>All three were arrested in the Toome area. All three have been taken to the Serious Crime Suite at Antrim police station.</td>
<td>good</td>
<td>bad</td>
<td>ok</td>
<td>bad</td>
<td>content reduction</td>
</tr>
<tr>
<td>Original</td>
<td>For years the former Bosnia Serb army commander Ratko Mladic had evaded capture and was one of the world’s most wanted men, but his time on the run finally ended last year when he was arrested near Belgrade.</td>
<td>good</td>
<td>bad</td>
<td>ok</td>
<td>bad</td>
<td></td>
</tr>
<tr>
<td>Simple</td>
<td>For years the former Bosnia Serb army commander Ratko Mladic had evaded capture.</td>
<td>good</td>
<td>good</td>
<td>good</td>
<td>good</td>
<td>lexical</td>
</tr>
<tr>
<td>Original</td>
<td>Keeping articles with potential encourages editors, especially unregistered users, to be bold and improve the article to allow it to evolve over time.</td>
<td>bad</td>
<td>bad</td>
<td>ok</td>
<td>bad</td>
<td>dropping</td>
</tr>
<tr>
<td>Simple</td>
<td>Keeping articles with potential editors, especially unregistered users, to be bold and improve the article to allow it to evolve over time.</td>
<td>bad</td>
<td>bad</td>
<td>ok</td>
<td>bad</td>
<td></td>
</tr>
</tbody>
</table>

We also considered using the Spearman’s correlation coefficient $\rho$. However, $\rho$ takes into account only ranks, not absolute values of the scores, thus making no assumptions about linearity. For the shared task, it is also important to assess how far the automatic score is from the ‘gold standard’. Therefore, we decided to calculate only Pearson’s correlations. These are reported in Section 5.1. In order to compute the correlations, the manual class labels were converted into three equally distant numerical values (bad $\rightarrow$ 0, ok $\rightarrow$ 50, and good $\rightarrow$ 100).

3.2. Classifiers

Classifiers were assessed using the following four criteria:

1. **Accuracy** – the percentage of sentences for which the automatically assigned class ($a$) is the same as the ‘gold standard’ class ($g$):

   $$\text{acc} = \frac{1}{N} \sum_{n=1}^{N} \delta(a_n, g_n)$$  \hspace{1cm} (3)

2. **Mean absolute error (MAE)** – the average of the absolute values of the differences between automatic ($a$) and ‘gold standard’ ($g$) classes:

   $$\text{MAE} = \frac{1}{N} \sum_{n=1}^{N} |a_n - g_n|$$  \hspace{1cm} (4)

3. **Root mean squared error (RMSE)** – the square root of the average of the squared differences between automatic ($a$) and ‘gold standard’ ($g$) classes.

   $$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{n=1}^{N} (a_n - g_n)^2}$$  \hspace{1cm} (5)

4. **Weighted F-score** – the weighted sum of F-scores of each class where each weight is the relative frequency of the given class:
\[
\text{weighted} F = \frac{C}{\sum_{c=1}^{C} \frac{N(g_c) 2 \cdot P_c \cdot R_c}{N}} \cdot \frac{P_c + R_c}{2}
\]  

where \(N(g_c)\) is number of occurrences of the ‘gold standard’ class \(c\), \(C\) is the total number of classes, \(P_c\) is precision for the class \(c\):

\[
P_c = \frac{N(a_c, g_c)}{N(a_c)}
\]

and \(R_c\) is recall for the class \(c\):

\[
R_c = \frac{N(a_c, g_c)}{N(g_c)}
\]

where \(N(a_c, g_c)\) is the number of matches (‘gold standard’ and automatic class are the same), \(N(a_c)\) is the total number of automatic \(c\) labels, and \(N(g_c)\) is the total number of ‘gold standard’ \(c\) labels.

The main disadvantage of accuracy is that it does not take into account whether they are equal or not. MAE and RMSE are able to capture the difference. However, all three methods have problems of overly awarding the most frequent class (if any). Therefore the weighted F-score has been introduced. All four scores in the form of percentages (multiplied by 100) are reported in Section 5.2.

4. Participating Systems

We received a total of 10 metrics and 16 classifiers for each aspect from eight groups that participated in the shared task. Additionally, we provided several baselines.

4.1. Baselines

The following six classifiers were used as baselines for the classification task:

1. Majority class, which labels each sentence with the most frequent class (in the training dataset) for the given aspect.

2. Four classifiers based on the metrics frequently used for evaluation of machine translation output – BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2011), TER (Snover et al., 2006) and WER\(^4\) (Levenshtein, 1966). Each of the four metrics was used to build a classifier based on arithmetic mean in the following way:

- Classification was performed as follows:
  - Scores higher than \(b_h\) were classified as \(c(\text{max} \, \text{AM})\)
  - Scores lower than \(b_l\) were classified as \(c(\text{min} \, \text{AM})\)
  - The rest was classified as the third class.

It should be noted that \(c(\text{max} \, \text{AM}) = \text{good}\) and \(c(\text{min} \, \text{AM}) = \text{bad}\) for Grammaticality, Meaning preservation and Overall aspects, whereas for Simplicity, \(c(\text{max} \, \text{AM}) = \text{ok}\) and \(c(\text{min} \, \text{AM}) = \text{good}\).

3. SVM classifier (with no feature standardisation or normalisation) which uses BLEU, METEOR, TER, and WER as features.

For the raw score metrics task, we used BLEU, METEOR, WER, and TER scores as four baselines. For calculating the MT evaluation metrics, the original sentences were used as reference translations and the simplified sentences as MT outputs.

4.2. Raw Score Metrics

The following submissions were received for the (raw score) metrics task:

UoLGP (Rios and Sharoff, 2015)

Three different systems trained with GP (Gaussian processes) regression\(^5\) on three different sets of features were submitted:

1. The QuEst baseline features\(^6\) (UoLGP-quest)
2. The embedding features based on bilingual embeddings\(^7\) trained on ‘original’ Wikipedia and Simple Wikipedia, where each word is presented as a 100-dimensional vector and the sentence vector is the sum of the word vectors. The final vector consists of 200 features, 100 for the original sentence and 100 for the simplified sentence (UoLGP-emb).
3. The combination of the previous two feature sets (UoLGP-combo).

Simple Wikipedia sentence alignments were used to train both the QuEst baseline and the bilingual embeddings to extract features from the training and test sets, using original sentences as source sentences and simplified sentences as target sentences.

OSVCML (Nisioi and Nauze, 2016)

Three different metrics were submitted:

1. Ensemble – A metric constructed using an ensemble of classifiers trained on features ranging from bag-of-words, POS tags, sentiment information (from SentiWordNet lexicon), negation, readability measures, character n-grams and

\(^4\)Not so frequently used anymore, but it is the basic edit distance and TER has been derived from it.

\(^5\)https://sheffieldml.github.io/GPy/

\(^6\)http://www.quest.dcs.shef.ac.uk

\(^7\)https://github.com/karlmoritz/bicvm
BLEU metric between original and simplified sentences. The parameters for the ensemble were obtained using particle swarm optimisation under multiple cross-validation scenarios.

2. TreeLSTM – The metric uses GloVe word vectors8 trained on the Common Crawl corpus and dependency parse trees. The architecture is based on a 3-layer recurrent neural networks, adapted and fine-tuned for the current task, inspired by the work of Tai et al. (2015).

3. Ensemble_treeLSTM – A combination of the previously described metrics.

SimpleNets (Paetzold, 2016)

Two types of resource-light neural networks:

1. Multi-layer perceptron (MLP), which uses a set of 14 features based on sentence lengths, vocabulary size and language model probabilities (SimpleNets-MLP).

2. Recurrent neural network (RNN) using n-gram model and word embeddings. One version (SimpleNets-RNN2) employs bigram models, while the other version (SimpleNets-RNN3) employs trigram models.

IIT (Mathias and Bhattacharyya, 2016)

METEOR (Denkowski and Lavie, 2014) was submitted as a raw metric score for the meaning preservation aspect.

4.3. Classifiers

The following submissions were received for the classification task:

CLaC (Davoodi and Kosseim, 2016)

Random Forest classifiers trained independently for each of the three aspects (G, M, S) using different sets of features for each aspect:

- Perplexities of language model built on Google Ngram corpus for grammaticality (G);
- Word embeddings and cosine measures for meaning preservation (M);
- A wider range of features including TF-IDF, sentence length, frequency of cue phrases, etc. for simplicity (S).

The classification system for the overall score (O) was rule-based, taking into account predicted classes for meaning preservation and simplicity.

Deep(Indi)Bow (Fishel, 2016)9

DeepIndiBow uses four individual multi-layer perceptron (MLP) networks for each of the evaluation aspects (G, M, S, and O) whereas DeepBow uses a single output layer corresponding to the concatenation of all four outputs.

Both variants use the same input, i.e. bag-of-word vectors extracted from original and simplified sentences where each element corresponds to a certain vocabulary entry and is equal to 1 if the entry is present in the sentence or 0 otherwise.

IIT (Mathias and Bhattacharyya, 2016)

One bagging classifier (Breiman, 1996) trained in Weka for each of the three aspects (G, M, and S) using different sets of features for each aspect:

- Language model with several additional features for grammaticality (G);
- METEOR (Denkowski and Lavie, 2014) for meaning preservation (M);
- A combination of features measuring lexical and structural complexity for simplicity (S).

Two systems for predicting the overall score were submitted based on:

1. Output classes of the classifiers for G, M, and S only (IIT-Default);
2. A combination of output classes of the three classifiers (for G, M, and S) and all previously used features (IIT-Metrics).

SimpleNets (Paetzold, 2016)

The same systems (two types of resource-light neural networks) as those described in Section 4.2. They participated as both, raw score metrics and classifiers.

SMH (Štajner et al., 2016)

Three systems were proposed for each aspect. Random Forest (Breiman, 2001) and IBk/K-nearest neighbours (Aha and Kibler, 1991) classifiers for grammaticality (G) and simplicity (S), and Random Forest and Logistic (le Cessie and van Houwelingen, 1992) for meaning preservation (M) and overall score (O). All classifiers were trained on a set of 22 MT evaluation metrics and 17 quality estimation (QuEst) baseline features (Specia et al., 2013). The third classifier for each aspect was the Random Forest (which gave best results on the training dataset in a 10-fold cross-validation setup) applied only on a subset of initial features returned by the CfsSubsetEval feature selection algorithm (Hall and Smith, 1998).

MS (Popović and Štajner, 2016)

Three systems were proposed for each aspect. The first two, Random Forest (Breiman, 2001) and IBk/K-nearest neighbours (Aha and Kibler, 1991) use the set of 13 MT evaluation metrics as features. The 13 features were selected among initial 26 MT evaluation metrics, as those with the highest Pearson correlation with respect to the overall quality aspect. The two classification algorithms were chosen among ten different, initially explored algorithms, as best performing on the training dataset in a 10-fold cross-validation setup. The third classifier (MS-RandForest-b for G and

8http://nlp.stanford.edu/projects/glove/
9https://github.com/fishel/deepbow
Table 4: Sorted Pearson’s correlation coefficients for raw score metrics (the baseline systems are underlined).

<table>
<thead>
<tr>
<th>Grammaticality</th>
<th>Meaning</th>
<th>Simplicity</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.482 OSVCML1</td>
<td>0.588 IIT-Meteor</td>
<td>0.382 OSVCML1</td>
<td>0.343 OSVCML</td>
</tr>
<tr>
<td>0.384 METEOR</td>
<td>0.585 OSVCML</td>
<td>0.376 OSVCML2</td>
<td>0.334 OSVCML</td>
</tr>
<tr>
<td>0.344 BLEU</td>
<td>0.573 OSVCML2</td>
<td>0.339 OSVCML</td>
<td>0.232 SimpleNets-RNN2</td>
</tr>
<tr>
<td>0.340 OSVCML</td>
<td>0.533 BLEU</td>
<td>0.320 SimpleNets-MLP</td>
<td>0.230 OSVCML1</td>
</tr>
<tr>
<td>0.323 TER</td>
<td>0.527 METEOR</td>
<td>0.307 SimpleNets-RNN3</td>
<td>0.205 UoLG-emb</td>
</tr>
<tr>
<td>0.308 SimpleNets-MLP</td>
<td>0.513 TER</td>
<td>0.240 SimpleNets-RNN2</td>
<td>0.198 SimpleNets-MLP</td>
</tr>
<tr>
<td>0.308 WER</td>
<td>0.495 WER</td>
<td>0.123 UoLGP-combo</td>
<td>0.196 METEOR</td>
</tr>
<tr>
<td>0.256 UoLGP-emb</td>
<td>0.482 OSVCML1</td>
<td>0.120 UoLGP-emb</td>
<td>0.189 UoLGP-combo</td>
</tr>
<tr>
<td>0.256 UoLGP-combo</td>
<td>0.465 SimpleNets-MLP</td>
<td>0.086 UoLGP-quest</td>
<td>0.144 UoLGP.quest</td>
</tr>
<tr>
<td>0.208 UoLGP-quest</td>
<td>0.285 UoLGP-quest</td>
<td>0.052 IIT-S</td>
<td>0.130 TER</td>
</tr>
<tr>
<td>0.064 SimpleNets-RNN3</td>
<td>0.262 SimpleNets-RNN3</td>
<td>-0.169 METEOR</td>
<td>0.112 SimpleNets-RNN3</td>
</tr>
<tr>
<td>0.056 SimpleNets-RNN2</td>
<td>0.262 SimpleNets-RNN2</td>
<td>-0.242 TER</td>
<td>0.111 WER</td>
</tr>
<tr>
<td></td>
<td>0.250 UoLGP-combo</td>
<td>-0.260 WER</td>
<td>0.107 BLEU</td>
</tr>
<tr>
<td></td>
<td>0.188 UoLGP-emb</td>
<td>-0.267 BLEU</td>
<td></td>
</tr>
</tbody>
</table>

For the simplicity aspect, all participating systems performed better than all baselines, with OSVCML systems being the best, followed by the three SimpleNets systems. For the overall aspect, several systems outperformed all four baselines. Among them, the OSVCML2 and OSVCML systems were the best, followed by the SimpleNets-RNN2, OSVCML1, UoLGP-emb, and SimpleNets-MLP.

If taking into account results on all four aspects, the OSVCML approach achieved best results on this task.

5.2. Classification Task

On the classification task, for grammaticality, the baseline majority-class (together with the MT-baseline, DeepIndiBow and DeepBow) achieved the highest accuracy (76.19%). Given the highly unbalanced distribution of classes for this aspect in both training and test datasets (Table 2), it is not surprising that this was a baseline difficult to outperform in terms of accuracy. However, the weighted F-score was the highest for the three SMH systems, which were the only systems able to outperform the BLEU baseline in terms of the weighted F-score (0.718, 0.716, and 0.700, respectively).

On the meaning preservation task, the best accuracy (69.05%) was achieved by the SMH-Logistic system. This was the only system able to outperform the accuracy of the MT-baseline. In terms of the weighted F-score, only two systems (SMH-Logistic, MS-RandForest, and SMH-RandForest) achieved better scores than the TER-baseline. On the simplicity task, the best accuracy (57.14%) was achieved by the SMH-RandForest-b system, which was the only system to achieve better accuracy than the majority-class baseline (55.56%). In terms of weighted F-score, however, five systems outperformed the highest baseline (0.483) achieved by the MT-baseline system. The SMH-RandForest-best obtained the highest weighted F-score (0.564), followed by the other two SMH systems, and SimpleNets-RNN3 and SimpleNets-MLP systems.

On the overall prediction task, as many as six systems managed to achieve higher accuracy than the majority-class

---

MS-IBk-b (for M, S, and O) was built using the better performing of the previous two, but this time only on a subset of those 13 features chosen by the CfsSubsetEval feature selection algorithm (Hall and Smith, 1998).

UoW (Bechara et al., 2015)

An SVM regression system trained using LibSVM\(^{10}\) with RBF kernel. Different sets of features were used for each aspect:

- **Grammaticality**: 17 QuEst baseline features for MT quality estimation (Specia et al., 2013)
- **Meaning preservation**: MiniExpert’s Semantic Textual Similarity features (Bechara et al., 2015)
- **Simplicity**: the features proposed by Yaneva and Evans (2015) as a means to estimate text difficulty for readers with autism spectrum disorder.

5. Results

In the following two subsections, we present the results of all raw score metrics (Section 5.1.) and classifiers (Section 5.2.) which participated in the shared task.

5.1. Raw Score Metrics Task

Pearson’s correlation coefficients for the standard MT metrics and the submitted raw score metrics are shown in Table 4 for all four TS evaluation aspects.

For the grammaticality aspect, only one system (OSVCML) was able to outperform METEOR and BLEU baselines, reaching a 0.482 correlation with the human scores (the baseline METEOR, as the second best, had a 0.384 correlation).

For the meaning preservation, three systems (IIT-Meteor, OSVCML and OSCCML2) outperformed all four baselines (BLEU, METEOR, TER, and WER).

For the simplicity aspect, all participating systems performed better than all baselines, with OSVCML systems being the best, followed by the three SimpleNets systems. For the overall aspect, several systems outperformed all four baselines. Among them, the OSVCML2 and OSVCML systems were the best, followed by the SimpleNets-RNN2, OSVCML1, UoLGP-emb, and SimpleNets-MLP.

If taking into account results on all four aspects, the OSVCML approach achieved best results on this task.

---

\(^{10}\)https://www.csie.ntu.edu.tw/~cjlin/libsvm/
Table 5: Accuracy, mean average error (MAE), root mean squared error (RMSE) and weighted F score of the classifiers on the task of predicting the grammaticality (a) and meaning preservation (b). The baseline systems are underlined.
<table>
<thead>
<tr>
<th>Accuracy</th>
<th>Mean average error</th>
<th>Root mean squared error</th>
<th>Weighted F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>57.14 SMH-RandForest-b</td>
<td>25.00 SimpleNets-RNN3</td>
<td>31.22 Majority-class</td>
<td>56.42 SMH-RandForest-b</td>
</tr>
<tr>
<td>55.56 Majority-class</td>
<td>25.40 SMH-RandForest-b</td>
<td>33.54 DeepIndiBow</td>
<td>53.02 SMH-RandForest</td>
</tr>
<tr>
<td>53.17 SimpleNets-MLP</td>
<td>26.19 MT-baseline</td>
<td>34.60 MS-IBk-best</td>
<td>51.12 SMH-IBk</td>
</tr>
<tr>
<td>52.38 SMH-RandForest</td>
<td>26.98 SimpleNets-MLP</td>
<td>35.57 SimpleNets-RNN3</td>
<td>49.96 SimpleNets-RNN3</td>
</tr>
<tr>
<td>52.38 DeepIndiBow</td>
<td>26.98 SimpleNets-RNN2</td>
<td>36.03 SimpleNets-MLP</td>
<td>49.81 SimpleNets-MLP</td>
</tr>
<tr>
<td>52.38 SimpleNets-RNN3</td>
<td>27.78 SMH-RandForest</td>
<td>36.64 MS-RandForest</td>
<td>48.31 MT-baseline</td>
</tr>
<tr>
<td>50.79 MT-baseline</td>
<td>28.17 UoW</td>
<td>36.68 MT-baseline</td>
<td>47.84 MS-IBk-b</td>
</tr>
<tr>
<td>50.00 SMH-IBk</td>
<td>28.17 SMH-IBk</td>
<td>37.01 SimpleNets-RNN2</td>
<td>47.82 MS-RandForest</td>
</tr>
<tr>
<td>49.21 MS-RandForest</td>
<td>29.37 Majority-class</td>
<td>38.08 SMH-RandForest</td>
<td>43.46 IIT-S</td>
</tr>
<tr>
<td>48.41 MS-IBk-b</td>
<td>31.35 Majority-class</td>
<td>38.85 IIT-S</td>
<td>42.57 DeepIndiBow</td>
</tr>
<tr>
<td>47.62 IIT-S</td>
<td>32.94 BLEU</td>
<td>39.51 SMH-IBk</td>
<td>40.92 UoW</td>
</tr>
<tr>
<td>44.44 UoW</td>
<td>34.13 IIT-S</td>
<td>43.87 DeepBow</td>
<td>39.68 Majority-class</td>
</tr>
<tr>
<td>42.06 DeepBow</td>
<td>34.13 TER</td>
<td>44.19 UoW</td>
<td>38.10 MS-IBk</td>
</tr>
<tr>
<td>38.10 WER</td>
<td>34.92 METEOR</td>
<td>44.48 CLaC-RF-0.6</td>
<td>35.58 DeepBow</td>
</tr>
<tr>
<td>38.10 TER</td>
<td>34.92 WER</td>
<td>44.48 CLaC-RF-0.7</td>
<td>34.88 CLaC-RF-0.5</td>
</tr>
<tr>
<td>38.10 BLEU</td>
<td>35.71 MS-IBk-best</td>
<td>45.32 CLaC-RF-0.5</td>
<td>34.66 CLaC-RF-0.6</td>
</tr>
<tr>
<td>38.10 MS-IBk</td>
<td>36.11 DeepBow</td>
<td>45.77 BLEU</td>
<td>34.48 WER</td>
</tr>
<tr>
<td>35.71 METEOR</td>
<td>38.89 MS-IBk</td>
<td>46.10 WER</td>
<td>34.30 CLaC-RF-0.7</td>
</tr>
<tr>
<td>35.71 CLaC-RF-0.7</td>
<td>40.48 CLaC-RF-0.7</td>
<td>46.12 MS-IBk</td>
<td>33.52 TER</td>
</tr>
<tr>
<td>35.71 CLaC-RF-0.6</td>
<td>41.67 CLaC-RF-0.6</td>
<td>46.23 TER</td>
<td>33.34 METEOR</td>
</tr>
<tr>
<td>34.92 CLaC-RF-0.5</td>
<td>43.25 CLaC-RF-0.5</td>
<td>47.47 METEOR</td>
<td>33.00 BLEU</td>
</tr>
</tbody>
</table>

Table 6: Sorted accuracy, mean average error (MAE), root mean squared error (RMSE) and weighted F score of the classifiers on the task of predicting the simplicity (a) and overall quality (b).
baseline (43.65%). The best performing system was the SimpleNets-RNN2, followed by the UoW, three SMH systems and another SimpleNets system. Five of those six systems were also able to outperform all baselines (METEOR being the best among them) in terms of the weighted F-score. The order of the systems was somewhat changed in comparison to their order in terms of accuracy. The highest weighted F-score (0.486) was achieved by the SMH-RandForest-b in this case, followed by the UoW, SMH-Logistic, SimpleNets-RNN2, and SMH-RandForest.

If we take into account the results achieved on all four aspects, it seems that the classifiers (Random Forest and Logistic) trained on the combination of MT evaluation metrics and MT quality estimation baseline features (the SMH systems) have the best potential to predict human scores.

### 6. Summary and outlook

This first shared task on quality assessment for text simplification had a goal of exploring how much overlap there is between this task and the closely related tasks of MT evaluation and MT quality estimation. It brought together people from all three communities.

The results indicated that MT evaluation metrics have a good potential to be used for replacing human evaluation of grammaticality and meaning preservation, but they do not seem suitable for assessing simplicity and overall quality (which strongly depends on simplicity). Furthermore, the addition of MT quality estimation features appear to significantly improve the performance of the classifiers on the Simplicity and Overall prediction tasks.

Now that we have established some baselines on this task, the next step would be to tailor TS-specific features which could boost the performance of the classifiers, especially on the Simplicity and Overall prediction tasks.

Finally, this shared task showed there is a good potential for collaboration among research communities working on MT evaluation, MT quality estimation, and text simplification.

We are currently preparing a significantly larger and richer dataset for English and a smaller dataset for Spanish, hoping to gradually extend the shared task to languages other than English.

### 7. References


Glavaš, G. and Štajner, S. (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 (Volume 2: Short Papers), pages 63–68. ACL.


Automatic text simplification (ATS) has gained a considerable attention in the last twenty years. Many ATS systems have been proposed for various languages, e.g. English (Angrosh et al., 2014; Glavaš and Štajner, 2015), Portuguese (Specia, 2010), Spanish (Saggion et al., 2015; Štajner et al., 2015), French (Brouwers et al., 2014), Italian (Barlacchi and Tonelli, 2013), and Basque (Aranzabe et al., 2012). The aim of ATS systems is to transform syntactically and lexically complex sentences into their simpler variants more accessible to wider audiences (non-native speakers, children, people with learning disabilities, etc.). The standard way of evaluating ATS systems is by human assessment of the quality of the generated sentences in terms of their grammaticality, meaning preservation, and simplicity (Woodsend and Lapata, 2011; Glavaš and Štajner, 2013; Saggion et al., 2015). The annotators are presented with pairs of original and automatically simplified sentences (for meaning preservation), and with automatically simplified sentences (for grammaticality and simplicity), and asked to evaluate them on a 1–3 or 1–5 scale (where lower score always denotes worse output).

In addition to this, some ATS systems are also evaluated by readability metrics (on a text level), e.g. (Zhu et al., 2010; Woodsend and Lapata, 2011; Glavaš and Štajner, 2013; Saggion et al., 2015), or by machine translation (MT) evaluation metrics, such as BLEU (Papineni et al., 2002), NIST (Doddington, 2002), or TER (Snover et al., 2006) in case of the MT-based ATS systems (Specia, 2010; Zhu et al., 2010; Woodsend and Lapata, 2011; Coster and Kaufchak, 2011; Štajner et al., 2015).

As any other human evaluation, the assessment of grammaticality (G), meaning preservation (M), and simplicity (S) is a costly and time-consuming task. Therefore, automatic methods are needed in order to provide a faster and more consistent evaluation. In spite of that, this task has not attracted much attention so far. To the best of our knowledge, there has been only one work (Štajner et al., 2014) tackling this problem by assessing the potential of several MT evaluation metrics: BLEU (Papineni et al., 2002), TER (Snover et al., 2006), METEOR (Denkowski and Lavie, 2011), and TINE (Rios et al., 2011). They showed that all of them correlate well with the human scores for meaning preservation, and some of them (BLEU and TER) also have a good correlation with the human scores for grammaticality (they did not investigate the correlation of those metrics with the human scores for simplicity).

Štajner et al. (2014) further built several classifiers for automatic assessment of grammaticality, meaning preservation, and their combination. One of the main limitations of their work was the dataset used for training and testing. The dataset1 was produced by the ATS system which performs syntactic simplification and content reduction (Glavaš and Štajner, 2013) and is, therefore, not the best representative for most of the ATS systems (that usually perform syntactic and/or lexical simplification and no content reduction).

In this work, we built upon previous work (Štajner et al., 2014) on several levels:

1. We significantly extended the list of MT metrics, including the MT evaluation metrics based on one of the two usual approaches: n-gram matching or edit distance.
2. We calculated Pearson’s correlation of all MT metrics with human scores for grammaticality, meaning preservation and simplicity (not only for grammaticality and meaning preservation).
3. We preselected a subset of most promising MT metrics based on their Pearson’s correlations with the human scores.
4. We tested the usefulness of MT metrics on a larger and more heterogeneous dataset (which also contains lexical simplifications and simplifications without content reduction) provided for the shared task.

---

1 takelab.fer.hr/data/evsimplify/
Mladic reportedly gave a thumbs-up and clapped to supporters in the court's public gallery as the trial got under way. ok ok ok ok syntactic + content reduction

Original Philippine President Benigno Aquino said he was looking to end the standoff through diplomatic means. bad bad good bad content reduction

Simple Philippine President Benigno Aquino said. Philippine President Benigno Aquino said.

Her mother wanted her to leave school and marry, but she rebelled. good good good good lexical

Original The novel received favorable reviews in several major newspapers. bad ok ok ok dropping

Simple The received favorable reviews in several major newspapers. The received favorable reviews in several major newspapers.

Original The place where Waste was executed is now the site of a Roman Catholic church. good good good good lexical + insertion

Simple The place where Joan Waste was killed is now the site of a church. The place where Joan Waste was killed is now the site of a church.

Table 1: Examples from the training dataset (differences between the original and simplified versions are presented in bold)

We calculated a total of 26 metrics on the provided training dataset, using the original English sentences as references, and their simplifications as hypotheses. In the next step, we selected the most promising metrics based on Pearson’s correlation coefficients between them and the assigned human scores for the overall quality of the sentences. Finally, we used that subset of MT metrics to train eight classifiers, out of which we submitted the two best ones to the shared task. Additionally, we experimented with one attribute selection algorithm and submitted the best of the two classifiers trained on that subset of features.

2. Shared Task Description

The shared task participants were provided with a training dataset of 505 sentence pairs and a test set (without ‘gold standard’ scores) of 126 sentence pairs from news articles and Wikipedia articles. The automatically simplified sentences were obtained by various automatic text simplification systems and thus cover different simplification phenomena (only lexical simplification, only syntactic simplification, mixture of lexical and syntactic simplification, content reduction, etc.).

The training dataset contained a quality label (good, ok, or bad) for each sentence according to four aspects:

- Grammaticality (G)
- Meaning preservation (M)
- Simplicity (S)
- Overall (O)

where the overall score presents a combination of the previous three scores which rewards more meaning preservation and simplicity than grammaticality.2

Several examples from the training dataset are presented in Table 1. It can be noted that some of the changes to the original sentence affect only one word (lexical changes, dropping, or insertion), while others reflect on larger portions of the sentence (syntactic changes and content reduction).

The MT evaluation metrics calculate similarity between the obtained translation and a reference translation. The higher similarity indicates better translation quality. For TS evaluation, which is a similar – but nevertheless distinct – task, we expected that the MT evaluation metrics could be good indicators of grammaticality and meaning preservation. We also expected that the simplicity aspect will be difficult to capture using the overall MT evaluation scores. However, we thought that certain components, such as insertions or deletions, might be able to capture relations between text reduction and quality.

We did not investigate complex MT evaluation metrics which require language dependent knowledge such as semantic roles, etc. Most of the metrics we used are completely language independent, while some of them require part-of-speech taggers and/or lemmatisers for the given language.

3. MT Evaluation Metrics

We focused on 26 automatic MT metrics:

- N-gram based metrics (9)

2http://qats2016.github.io/shared.html
BLEU and METEOR are widely used for MT evaluation, and the other metrics have shown very good correlations with human judgments in recent years, especially the character n-gram F3 score (chrF3) for morphologically rich languages.

- Edit-distance based metrics (4)
- WER – Levenshtein (edit) distance (Levenshtein, 1966);
- TER – modified edit distance taking into account shifts of word sequences (Snover et al., 2006);
- Serr – sum of word-level error rates provided by Hjerson, automatic tool for MT error analysis (Popović, 2011a);
- bSerr – Hjerson’s sum of block-level error rates.

WER is the basic edit-distance metric widely used in speech recognition and at the beginnings of machine translation development. It was later substituted by TER which shows better correlations with human judgments as it does not penalise small differences as much as WER. Recently, Hjerson’s error rates have also shown good correlations with human judgments.

- Components of edit-distance based metrics (13)
- WER and TER substitutions, deletions and insertions (wer-sub, ter-sub, wer-del, ter-del, wer-ins, ter-ins);
- TER shifts and word shifts (ter-sh, ter-wsh);
- Hjerson’s error classes:
  - Inflectional errors (infl)
  - Reordering errors (reord)
  - Missing words (miss)
  - Extra words (ext)
  - Lexical errors (lex).

It should be noted that the n-gram based metrics represent scores, i.e. the higher the value, the more similar the segments, whereas edit-distance based metrics represent error rates, i.e. the lower the value, the more similarity between the segments.

### 4. Correlations with Human Scores

The first step in exploring those 26 MT metrics consisted in calculating Pearson’s correlation coefficients between each metric and human scores for four aspects (grammaticality, meaning preservation, simplicity and overall score), which are presented in Table 2. The metrics are sorted from best to worst according its correlation with the human scores for the overall quality of sentences.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Overall</th>
<th>G</th>
<th>M</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>posF</td>
<td>0.288</td>
<td>0.386</td>
<td>0.559</td>
<td>-0.043</td>
</tr>
<tr>
<td>wer-del</td>
<td>-0.265</td>
<td>-0.297</td>
<td>-0.542</td>
<td>0.097</td>
</tr>
<tr>
<td>ter-del</td>
<td>-0.257</td>
<td>-0.264</td>
<td>-0.496</td>
<td>0.059</td>
</tr>
<tr>
<td>miss</td>
<td>-0.252</td>
<td>-0.267</td>
<td>-0.529</td>
<td>0.117</td>
</tr>
<tr>
<td>BLEU</td>
<td>0.251</td>
<td>0.326</td>
<td>0.582</td>
<td>-0.122</td>
</tr>
<tr>
<td>wordF3</td>
<td>0.250</td>
<td>0.333</td>
<td>0.588</td>
<td>-0.124</td>
</tr>
<tr>
<td>baseF</td>
<td>0.248</td>
<td>0.340</td>
<td>0.578</td>
<td>-0.111</td>
</tr>
<tr>
<td>morphF</td>
<td>0.239</td>
<td>0.332</td>
<td>0.559</td>
<td>-0.104</td>
</tr>
<tr>
<td>WER</td>
<td>-0.237</td>
<td>-0.302</td>
<td>-0.543</td>
<td>0.104</td>
</tr>
<tr>
<td>chrF3</td>
<td>0.231</td>
<td>0.303</td>
<td>0.575</td>
<td>-0.147</td>
</tr>
<tr>
<td>TER</td>
<td>-0.229</td>
<td>-0.296</td>
<td>-0.539</td>
<td>0.119</td>
</tr>
<tr>
<td>METEOR</td>
<td>0.228</td>
<td>0.262</td>
<td>0.527</td>
<td>-0.140</td>
</tr>
<tr>
<td>Serr</td>
<td>-0.223</td>
<td>-0.276</td>
<td>-0.536</td>
<td>0.127</td>
</tr>
<tr>
<td>wordF</td>
<td>0.219</td>
<td>0.316</td>
<td>0.552</td>
<td>-0.129</td>
</tr>
<tr>
<td>chrF</td>
<td>0.216</td>
<td>0.299</td>
<td>0.549</td>
<td>-0.140</td>
</tr>
<tr>
<td>ter-sh</td>
<td>-0.155</td>
<td>-0.261</td>
<td>-0.186</td>
<td>-0.175</td>
</tr>
<tr>
<td>ter-wsh</td>
<td>-0.151</td>
<td>-0.150</td>
<td>-0.157</td>
<td>0.052</td>
</tr>
<tr>
<td>reord</td>
<td>-0.142</td>
<td>-0.172</td>
<td>-0.156</td>
<td>-0.124</td>
</tr>
<tr>
<td>bSerr</td>
<td>-0.097</td>
<td>-0.222</td>
<td>-0.345</td>
<td>0.097</td>
</tr>
<tr>
<td>infl</td>
<td>-0.033</td>
<td>-0.024</td>
<td>-0.136</td>
<td>0.104</td>
</tr>
<tr>
<td>ter-ins</td>
<td>-0.025</td>
<td>-0.111</td>
<td>-0.069</td>
<td>-0.034</td>
</tr>
<tr>
<td>ter-sub</td>
<td>-0.021</td>
<td>-0.052</td>
<td>-0.130</td>
<td>-0.129</td>
</tr>
<tr>
<td>wer-sub</td>
<td>-0.001</td>
<td>-0.029</td>
<td>-0.149</td>
<td>0.085</td>
</tr>
<tr>
<td>wer-ins</td>
<td>0.016</td>
<td>-0.029</td>
<td>0.020</td>
<td>-0.058</td>
</tr>
<tr>
<td>lex</td>
<td>0.065</td>
<td>-0.012</td>
<td>-0.119</td>
<td>0.140</td>
</tr>
<tr>
<td>ext</td>
<td>0.074</td>
<td>-0.025</td>
<td>0.014</td>
<td>0.013</td>
</tr>
</tbody>
</table>

Table 2: Pearson’s correlation coefficients between automatic metrics and human scores.

First of all, it can be confirmed that the TS evaluation task, although similar to the MT evaluation task, requires different approaches to evaluation of certain aspects. The MT evaluation metrics show the best correlations for the meaning preservation where a number of metrics have correlations over 0.5. Grammaticality seems to be more difficult aspect (the MT metrics achieve maximum correlation of about 0.390). Simplicity is, as intuitively expected, the most difficult aspect. While in MT evaluation the similarity between the reference and the hypothesis should be rewarded, this is not the case for the simplicity. The correlations with the overall score are not greater than 0.290; this score is also difficult for MT metrics since it takes all aspects (including simplicity) into account. Contrary to the expected, simplicity is not well captured by deletions/omissions. Those metrics, instead, have rather high correlation with meaning preservation scores, probably due to the fact that text reduction can influence the meaning.
For building the classifiers, we selected only those metrics which had the correlation greater than 0.200 with the overall score, i.e. the first 13 rows in Table 2.1

5. Classification Experiments

After selection of the 13 best correlating features, for each of the four aspects (G, M, S, and Overall), we trained eight different classifiers implemented in Weka Experimenter (Hall et al., 2009):

1. Log – Logistic Regression (le Cessie and van Houwelingen, 1992)
2. NB – Naïve Bayes (John and Langley, 1995)
3. SVM-n – Support Vector Machines with feature normalisation
4. SVM-s – Support Vector Machines with feature standardisation
5. IBk – K-nearest neighbours (Aha and Kibler, 1991)
6. JRip – a propositional rule learner (Cohen, 1995)
7. J48 – C4.5 decision tree (Quinlan, 1993)
8. RandF – Random Forest (Breiman, 2001)

All experiments were conducted in a 10-fold cross-validation setup with 10 repetitions, using the provided training dataset of 505 sentence pairs.4

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Aspect</th>
<th>G</th>
<th>S</th>
<th>M</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log</td>
<td></td>
<td>0.710</td>
<td>0.452</td>
<td>0.641</td>
<td>0.493</td>
</tr>
<tr>
<td>NB</td>
<td></td>
<td>0.653</td>
<td>0.416</td>
<td>0.628</td>
<td>0.373</td>
</tr>
<tr>
<td>SVM-n</td>
<td></td>
<td>0.652</td>
<td>0.363</td>
<td>0.633</td>
<td>0.424</td>
</tr>
<tr>
<td>SVM-s</td>
<td></td>
<td>0.655</td>
<td>0.363</td>
<td>0.653</td>
<td>0.494</td>
</tr>
<tr>
<td>IBk</td>
<td></td>
<td>0.737</td>
<td>0.532</td>
<td>0.655</td>
<td>0.530</td>
</tr>
<tr>
<td>JRip</td>
<td></td>
<td>0.671</td>
<td>0.458</td>
<td>0.620</td>
<td>0.461</td>
</tr>
<tr>
<td>J48</td>
<td></td>
<td>0.718</td>
<td>0.507</td>
<td>0.609</td>
<td>0.470</td>
</tr>
<tr>
<td>RandF</td>
<td></td>
<td>0.747</td>
<td>0.519</td>
<td>0.653</td>
<td>0.499</td>
</tr>
<tr>
<td>Majority class</td>
<td></td>
<td>0.652</td>
<td>0.363</td>
<td>0.428</td>
<td>0.240</td>
</tr>
</tbody>
</table>

Table 3: Results of the classification experiments (weighted F-score). The two best results for each aspect (G, S, M, and Overall) are presented in bold.

5.2. Feature Selection

We further selected a subset of best features using the CfsSubsetEval attribute selection algorithm (Hall and Smith, 1998) implemented in Weka by applying it to the whole training dataset (in a 10-fold cross-validation setup). Next, for each aspect, we trained a classifier (the most successful one for that aspect according to the results in Table 3) only on that subset of features. The CfsSubsetEval attribute selection algorithm uses a correlation-based approach to the feature selection problem, following the idea that “good feature sets contain features that are highly correlated with the class, yet uncorrelated with each other” (Hall, 1999). On small datasets, the CfsSubsetEval gives results similar to, or better than, those obtained by using a wrapper (Hall, 1999).

The CfsSubsetEval attribute selection algorithm returned the following subsets of best features:

- For Grammaticality (G): \{BLEU, METEOR, TER, WER, chrF3, morphF, posF, ter-d, wer-d\} – a total of 9 features
- For Simplicity (S): \{BLEU, chrF3, morphF\} – a total of 3 features
- For Meaning preservation (M): All except Serr – a total of 12 features
- For Overall: \{BLEU, WER, chrF3, posF, miss, wer-d\} – a total of 6 features

5.3. Results on the Test Dataset

We submitted three runs for each aspect to the shared task. The first two runs were the two classifiers which led to the best results of the 10-fold cross-validation experiments on the training dataset, the IBk and RandF classifiers. In the third run, we built a classifier – either IBk or RandF, depending on which of them was more successful in the cross-validation experiments on the training dataset (the IBk for Overall, M, and S, and RandF for G) – using only the subset of features returned by the CfsSubsetEval attribute selection algorithm (Section 5.2.).

The results of all three runs are presented in Table 4, together with the majority-class baseline, which turned out to be a very strong baseline for this task (Štajner et al., 2016). On the task of predicting the meaning preservation score (M), all three of our systems outperformed the majority-class baseline in terms of accuracy and the weighted average F-score. On the task of predicting the overall score, all our systems achieved higher weighted F-scores than the majority-class baseline. On the other two tasks, predicting grammaticality and simplicity, only some of our systems (two for grammaticality, and one for simplicity) succeeded in outperforming the majority-class baseline in terms of the weighted F-score, and none of the systems achieved a higher accuracy score than the majority-class baseline.

Among all our systems, the Random Forest classification algorithm with all 13 preselected features (RandF) achieved the best results on all four tasks.

---

3 Although wordF and chrF scores also fulfill this criterion, they are not used in the best set because their F3 versions showed better performance.
4 http://qats2016.github.io/shared.html
### Table 4: Results on the shared task (performances better than those of the baseline are presented in bold)

<table>
<thead>
<tr>
<th>System</th>
<th>Grammaticality accuracy</th>
<th>Grammaticality weighted-F</th>
<th>Meaning accuracy</th>
<th>Meaning weighted-F</th>
<th>Simplicity accuracy</th>
<th>Simplicity weighted-F</th>
<th>Overall accuracy</th>
<th>Overall weighted-F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run 1 (IBk)</td>
<td>60.32</td>
<td>0.620</td>
<td>61.90</td>
<td>0.615</td>
<td>38.10</td>
<td>0.381</td>
<td>38.10</td>
<td>0.383</td>
</tr>
<tr>
<td>Run 2 (RandF)</td>
<td>72.22</td>
<td>0.675</td>
<td>66.67</td>
<td>0.656</td>
<td>49.21</td>
<td>0.478</td>
<td>39.68</td>
<td>0.398</td>
</tr>
<tr>
<td>Run 3 (best)</td>
<td>69.84</td>
<td>0.667</td>
<td>62.70</td>
<td>0.617</td>
<td>37.30</td>
<td>0.377</td>
<td>38.10</td>
<td>0.382</td>
</tr>
<tr>
<td>Majority class</td>
<td>76.19</td>
<td>0.659</td>
<td>57.94</td>
<td>0.425</td>
<td>55.56</td>
<td>0.397</td>
<td>43.65</td>
<td>0.265</td>
</tr>
</tbody>
</table>

### Table 5: Confusion matrices (RandF using all 13 features)

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Predicted</th>
<th>Actual class</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>good</td>
<td>ok</td>
</tr>
<tr>
<td>Grammaticality</td>
<td></td>
<td></td>
</tr>
<tr>
<td>good</td>
<td>87</td>
<td>9</td>
</tr>
<tr>
<td>ok</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>bad</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Meaning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>good</td>
<td>61</td>
<td>11</td>
</tr>
<tr>
<td>ok</td>
<td>9</td>
<td>17</td>
</tr>
<tr>
<td>bad</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Simplicity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>good</td>
<td>47</td>
<td>22</td>
</tr>
<tr>
<td>ok</td>
<td>18</td>
<td>12</td>
</tr>
<tr>
<td>bad</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Overall</td>
<td></td>
<td></td>
</tr>
<tr>
<td>good</td>
<td>15</td>
<td>13</td>
</tr>
<tr>
<td>ok</td>
<td>13</td>
<td>23</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>19</td>
</tr>
</tbody>
</table>

In most misclassification cases (all except those for the ok sentences for the Overall aspect), our systems have a tendency of assigning a higher class than the actual class. This is particularly accentuated in classifications according to sentence grammaticality and simplicity, where more than a half of sentences that should be classified as bad were classified as good.

### 5.4. Error Analysis
In order to better understand which sentences pose most difficulties in these classification tasks, the confusion matrices for our best systems in each of the four aspects (according to Table 4) are presented in Table 5.

### 6. Conclusions
Automatic evaluation of the quality of sentences produced by automatic text simplification (ATS) systems is an important task which could significantly speed up evaluation process and offer a fairer comparison among different ATS systems. Nevertheless, it has hardly been addressed so far. In this paper, we reported on results of our classification systems, which were submitted to the QATS shared task on quality assessment for text simplification. The proposed feature sets were based on the use of standard – as well as some more recent – machine translation (MT) evaluation metrics.

More importantly, we explored the correlation of 26 MT evaluation metrics with human scores for grammaticality, meaning preservation, simplicity, and overall quality of automatically simplified sentences. The results revealed some important differences between evaluation tasks in MT and TS, which may seem very similar at first sight. They indicated that it is necessary to propose different, TS-specific, features in order to better assess the simplicity of automatically simplified sentences.

### 7. References


Glavaš, G. and Štajner, S. (2013). Event-Centered Simplifi-


Using Machine Translation Evaluation Techniques to Evaluate Text Simplification Systems

Sandeep Mathias, Pushpak Bhattacharyya
Department of Computer Science and Engineering
IIT Bombay, India
{sam.pb}@cse.iitb.ac.in

Abstract
In this paper, we discuss our approaches to find out ways to evaluate automated text simplification systems, based on the grammaticality and simplicity of their output, as well as the meaning preserved from the input, and the overall quality of simplification of the system. In this paper, we discuss existing techniques currently used in the area of machine translation, as well as a novel technique for text complexity analysis, to assess the quality of the text simplification system.

Keywords: METEOR, Language modelling, Lexical complexity

1. Introduction
While there has been a lot of research in automated text simplification, there has been relatively little work done in evaluating automated text simplification systems. One of the main bottlenecks in coming up with solutions to this problem lies in what should be measured. Should we provide more emphasis on the grammar of the output, or the retention of the meaning of the output, or the difficulty of the output, or a combination of all three? Our paper aims to describe different techniques to solve these problems. Some of them are already in use in the field of machine translation, while others are relatively new techniques specifically for text simplification.

2. Problem Statement
The LREC (Language Resources and Evaluation Conference) 2016 had a workshop called the Quality Assessment for Text Simplification (QATS) workshop that had a shared task which aims to find out ways to evaluate various aspects of assessing the output from text simplification systems. The main aim of the shared task is to find out ways to evaluate different text simplification systems based on their output. For this task, we consider the following questions:

1. How grammatically correct is the output of the system?
2. How simple is the output of the system?
3. How much of the meaning of the input sentence is preserved in the output of the system?
4. How good is the overall quality of the system?

The training data has 505 sentence pairs that are manually scored for various aspects, such as grammaticality, meaning preservation, simplicity, and overall quality. The sentence pairs (or output sentences in the case of grammaticality and simplicity) are classified as either “good”, “ok”, or “bad”. The following sections will explain our approaches to solve each of these questions.

3. Grammaticality
Grammaticality is a means of finding out how grammatically correct a sentence is. Grammaticality is scored based on the quality of the simplified sentence only. To evaluate grammaticality, we make use of language modelling - one of the tasks in machine translation. In machine translation, the language model is used to check how likely a string in the target language is. Hence, we use language modelling to score the grammaticality of the system.

To build the language model, we make use of the English Simple Wikipedia from the English Wikipedia - Simple English Wikipedia corpus. The corpus consists of a sentence aligned and a document aligned parallel corpus between articles in English Wikipedia and Simple English Wikipedia. The sentence aligned corpus has sentences in the Simple English Wikipedia aligned with their corresponding sentence in the English Wikipedia. On the other hand, the document aligned corpus has the article in the Simple English Wikipedia aligned with the corresponding article in the English Wikipedia. Since every article in the Simple English Wikipedia has a corresponding article in the English Wikipedia, using the document-aligned Simple English Wikipedia is equivalent to using the entire Simple English Wikipedia (at least up to the point of the corpus’ creation).

For our task, we take all the Simple English Wikipedia articles from the document aligned corpus to calculate the language modelling score. We make use of the SRILM toolkit used for machine translation to calculate the language modelling score. We make use of the SRILM toolkit used for machine translation to train the language model. Using SRILM, we also find out the out of vocabulary words (OOVs), the perplexity (ppl), and the average perplexity per word (ppl1) in each sentence. The following features are used to help classify the how grammatical the output sentence is:

1. Number of words in the sentence
2. Number of OOVs

---

1http://en.wikipedia.org
2http://simple.wikipedia.org
3. Log of the probability score (Language model score for the sentence)
4. Perplexity of the sentence
5. Average perplexity per word of the sentence

Details of the experiment and results are covered in Section 6.

4. Meaning Preservation
The second aspect that we score is the amount of meaning retained in the output, given a particular input sentence. Consider the following sentence:

Warsaw lies on the Vistula River, about 240 miles southeast of the Baltic coast city of Gdansk.\(^3\)

An example of a good meaning preservation would be Warsaw is on the Vistula River, about 240 miles southeast of Gdansk. Gdansk is a Baltic coast city. However, a sentence with bad meaning preservation would be something like Vistula is on the Warsaw River, about 240 miles southeast of the Baltic coast city of Gdansk. To accomplish this task, we make use of the METEOR (Denkowski and Lavie, 2014) metric, already being used in machine translation. Unlike BLEU (Papineni et al., 2002), the advantages of using METEOR are as follows:

1. BLEU matches words only if they are completely matched (i.e. their surface forms are the same). For instance, BLEU would score a match of live → lives as 0. However, since both live and lives have the same stem, METEOR will award some value to the match.
2. BLEU does not match synonyms. A word like jail → prison will be scored as 0. However, in METEOR, it would be awarded a score, since prison is a synonym of jail.
3. BLEU does not score paraphrases. At times, the output sentence may have a paraphrase of the reference phrase - for example, kicked the bucket → died. BLEU will score it as 0, but METEOR will award it a score.

METEOR works by identifying all possible matches between the input and simplified sentences. We consider 4 different types of matches, namely

1. **Exact** If the surface words are the same.
2. **Stem** If the stems of the words unmatched by the previous matcher are the same. This is done using the Snowball Stemmer for English (Porter, 2001).
3. **Synonym** If the stems of the words also remain unmatched, the remaining words are matched if they belong to the same synset in WordNet (Fellbaum, 1998).
4. **Paraphrase** Among the remaining unmatched words, match them if they occur as paraphrases in a paraphrase table (Denkowski and Lavie, 2014).

For each type of match, a weight is used to calculate the score. We use the weights described in (Denkowski and Lavie, 2014) for exact, stem, synonym, and paraphrase matching. Table 1 gives the weights of different types of matches used. We make use of the METEOR scores for estimating the meaning preservation of the input sentence in the simplified output sentence. Here, for each sentence pair, we calculate the METEOR score for the simplified sentence with respect to the original input sentence. We use the METEOR score as the only feature to measure the meaning of the input sentence preserved in the output sentence.

<table>
<thead>
<tr>
<th>Type of match</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact</td>
<td>1.00</td>
</tr>
<tr>
<td>Stem</td>
<td>0.60</td>
</tr>
<tr>
<td>Synonym</td>
<td>0.80</td>
</tr>
<tr>
<td>Paraphrase</td>
<td>0.60</td>
</tr>
</tbody>
</table>

Table 1: Weights of different matchers in METEOR

5. Simplicity
Along with measuring the grammaticality of the simplified output sentence, as well as the meaning of the input sentence preserved in the output sentence, we also look at how simple the output sentence is. This is important in judging the quality of the text simplification. There are many measures to judge the quality of the simplified output. One of the earliest methods was the Flesch Reading Ease Score (FRES) (Flesch, 1948), proposed by Rudolph Flesch in 1948. This approach took into account mainly the average number of words per sentence, and the average number of syllables per word. It used a simple formula to calculate the reading ease of a piece of text, without the use of any extra data. More recently, the presence of data, in the form of the English Wikipedia - Simple English Wikipedia (Kauchak, 2013) gave rise to data driven approaches to simplify texts. For calculating the complexity of the texts, we take into account mainly two types of complexity, namely structural complexity and lexical complexity. Structural complexity is a measure of how complex the sentence is, based on its parse tree. For structural complexity, we calculate the number of

1. Main clause sentences
2. Sentences from relative clauses
3. Sentences from appositives
4. Sentences from noun and verb participial phrases
5. Sentences from other subordinate clauses

that we can extract from a single input sentence using Michael Heilman’s factual statement extractor\(^4\) (Heilman and Smith, 2010).

\(^3\)This was one of the sentences in the test set of the shared task.

\(^4\)The system can be downloaded from www.cs.cmu.edu/~ark/mheilman/qg-2010-workshop
Lexical complexity is the complexity of the text based on its vocabulary. It is based on the complexity of the words and phrases used in the text.

We use a unigram and bigram language model of the English Wikipedia - Simple English Wikipedia (Kauchak, 2013) parallel corpus to calculate the lexical complexity of each n-gram. The complexity of an n-gram is comprised of 2 parts, namely the corpus complexity and the syllable count.

1. **Corpus complexity** For each n-gram \((g)\) of the sentence, we calculate its corpus complexity (Biran et al., 2011), \(C_c(g)\), defined as the ratio of the log likelihood of \(g\) in the English corpus to the log likelihood of \(g\) in the Simple English corpus. In other words,

\[
C_c(g) = \frac{LL(g|\text{normal})}{LL(g|\text{simple})}
\]

Here, we assume that every n-gram in the Simple English corpus has to occur at least once in the English corpus.

2. **Syllable count** We consider that readers read words one syllable at a time. The syllable count, \(s(g)\), of an n-gram \((g)\) is defined as the sum of syllables of the words in that n-gram.

With these two ideas, we go ahead and calculate the lexical complexity of an n-gram \((g)\) as:

\[
Lc(g) = s(g) \times C_c(g)
\]

Hence, for a given sentence \(S\), and an n-gram size, the lexical complexity is given by

\[
Lc(S, n) = \sum_g s(g) \times C_c(g),
\]

where \(g\) is an n-gram of size \(n\).

In addition to this, we also attach a weight \(W_n\) to the lexical complexity calculated for a particular n-gram. For a given n-gram size of \(n\), the weight is \(\frac{1}{n}\). This is because the unigrams in the n-gram are added n-times. For example, if \(n\) is 2, and we have an n-gram sequence “a b c d e f g ...”, unigrams like b, c, d, e, f, etc. get added twice.

Therefore, we can say that the lexical complexity of a sentence is given by

\[
Lc(S) = \sum_n W_n \sum_g s(g) \times C_c(g),
\]

The final complexity of the text is calculated as the sum of the lexical and structural complexity. We use this value as the feature in calculating the complexity of the simplified sentence.

### 6. Shared Task Results

For each of the tasks - grammaticality, meaning preservation, simplicity - we treat them as a classification problem and classify the outputs of the simplification systems as either “good”, “ok”, or “bad”. We use Bagging with the REP-Tree classifier in Weka running 10 iterations to train our model. The training data was a set of 505 sentence pairs, and the test data, an additional 126 sentence pairs. Accuracy (Acc.) is the percentage of sentences correctly classified. To calculate the mean absolute error (MAE), and the root mean square error (RMSE), 100, 50 and 0 were given to the classes “good”, “ok”, and “bad” respectively. The baseline we use is the majority class of our training data. The following are the results of various aspects of our task.

#### 6.1. Grammaticality

The following table gives the results for our experiments with the grammaticality of the output sentences.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Acc. (%)</th>
<th>MAE</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training set - Baseline</td>
<td>75.64</td>
<td>17.23</td>
<td>36.96</td>
</tr>
<tr>
<td>Training set</td>
<td>76.04</td>
<td>16.63</td>
<td>36.01</td>
</tr>
<tr>
<td>Test set - Baseline</td>
<td>76.19</td>
<td>18.25</td>
<td>21.63</td>
</tr>
<tr>
<td>Test set</td>
<td>72.22</td>
<td>21.43</td>
<td>25.78</td>
</tr>
</tbody>
</table>

Table 2: Results of grammaticality classification

From the results, we can clearly see that existing language modelling toolkits (like SRILM) can provide reasonably accurate results for grammaticality.

#### 6.2. Meaning Preservation

The following table gives the results for our experiments in estimating the meaning preservation of the input in the output sentences.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Acc. (%)</th>
<th>MAE</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training set - Baseline</td>
<td>58.21</td>
<td>28.61</td>
<td>46.94</td>
</tr>
<tr>
<td>Training set</td>
<td>66.34</td>
<td>19.50</td>
<td>35.25</td>
</tr>
<tr>
<td>Test set - Baseline</td>
<td>57.94</td>
<td>28.97</td>
<td>35.30</td>
</tr>
<tr>
<td>Test set</td>
<td>63.49</td>
<td>20.63</td>
<td>26.75</td>
</tr>
</tbody>
</table>

Table 3: Results of Meaning Preservation using METEOR

From the above results, we see that the use of METEOR for classifying the output gives satisfactory results.

#### 6.3. Simplicity

The following table gives the results for our experiments in estimating the simplicity of the output of the text simplification system.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Acc. (%)</th>
<th>MAE</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training set - Baseline</td>
<td>52.67</td>
<td>32.18</td>
<td>49.60</td>
</tr>
<tr>
<td>Training set</td>
<td>48.31</td>
<td>32.87</td>
<td>48.59</td>
</tr>
<tr>
<td>Test set - Baseline</td>
<td>55.56</td>
<td>29.37</td>
<td>31.22</td>
</tr>
<tr>
<td>Test set</td>
<td>47.62</td>
<td>34.13</td>
<td>38.85</td>
</tr>
</tbody>
</table>

Table 4: Results of Simplicity of output

From the above results, we realize that calculating the simplicity of the output of the system requires a lot of research to solve.
6.4. Overall quality

In addition to calculate the above metrics, we also classify the overall quality of the system. Here, we consider two experiments. The first uses only the classes output that we get from the different aspects of the simplification system, namely the class values of grammaticality, simplicity and meaning preservation as features. The second uses the classes output as well as the other values (like OOVs, lexical complexity, etc.) that we used to classify the individual aspects of simplification as features. The following table gives the results of classifying the overall quality of the text simplification system:

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Acc. (%)</th>
<th>MAE</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training set - Baseline</td>
<td>43.76</td>
<td>33.17</td>
<td>46.51</td>
</tr>
<tr>
<td>Training Set - Classes</td>
<td>45.74</td>
<td>31.39</td>
<td>44.67</td>
</tr>
<tr>
<td>Training Set - Values</td>
<td>56.23</td>
<td>23.56</td>
<td>36.70</td>
</tr>
<tr>
<td>Test Set - Baseline</td>
<td>43.65</td>
<td>28.17</td>
<td>40.52</td>
</tr>
<tr>
<td>Test Set - Classes</td>
<td>33.33</td>
<td>42.46</td>
<td>47.83</td>
</tr>
<tr>
<td>Test Set - Values</td>
<td>39.68</td>
<td>34.92</td>
<td>42.97</td>
</tr>
</tbody>
</table>

Table 5: Overall Quality Classification Results

Training Set - Classes and Test Set - Classes make use of only the classes output from our tasks, like simplicity, meaning preservation and grammaticality. Training Set - Values and Test Set - Values use all the other values calculated, like METEOR Score, lexical complexity, etc. in addition to the class values to help classify the overall quality of the output. One of the factors affecting the low quality of the overall quality classification is the fact that our simplification results are comparatively low as compared to those of meaning preservation and grammaticality.

7. Conclusions

Among the three given tasks, we have seen that, for the evaluation of text simplification systems, metrics such as METEOR and techniques like language modelling can achieve good results as compared to more complex tasks, like evaluating the simplicity of the text, which is also why the accuracy in the overall quality classification of the various text simplification systems is quite low.

8. Bibliographical References


SimpleNets: Evaluating Simplifiers with Resource-Light Neural Networks

Gustavo H. Paetzold and Lucia Specia
Department of Computer Science, University of Sheffield
Western Bank, Sheffield, UK
ghpaetzold1@sheffield.ac.uk, l.specia@sheffield.ac.uk

Abstract
We present our contribution to the shared task on Quality Assessment for Text Simplification at QATS 2016: the SimpleNets systems. We introduce a resource-light Multi-Layer Perceptron classifier, as well as a deep Recurrent Neural Network that predicts the quality of a simplification by assessing the quality of its n-grams. Our Recurrent Neural Networks have achieved the state-of-the-art solution for the task, outperforming all other system in terms of overall simplification Accuracy.

Keywords: Text Simplification, Neural Networks, Evaluation

1. Introduction
In Text Simplification, the goal is to modify a given portion of text in order to make it simpler to read. In previous research, the needs of various target audiences have been addressed, such as the aphasic (Carroll et al., 1998), dyslexic (Rello et al., 2013b; Rello et al., 2013a; Rello et al., 2013c), poor literacy readers (Watanabe et al., 2009; Aluisio and Gasperin, 2010) and non-native speakers (Paetzold, 2015; Paetzold and Specia, 2016). Simplifiers can focus on different linguistic aspects. For example, lexical simplifiers replace single words and expressions, while syntactic simplifiers apply complex transformations to the syntactic structure of sentences.

As with any other task in Natural Language Processing, Text Simplification faces many challenges. The biggest among them is producing a simplifier capable of preserving the grammaticality and meaning of a text while still making it simpler to read. Researchers have shown that, regardless of the approach used, preserving the integrity of a text is a tall order (Siddharthan, 2006; Woodsend and Lapata, 2011; Kauchak and Barzilay, 2006; Paetzold, 2013; Paetzold and Specia, 2013). Another challenge is finding ways of determining how well a simplifier is capable of doing so without relying on costly human judgments.

For automatic quality assessment, early simplification approaches such as the ones of (Specia, 2010) and (Zhu et al., 2010), have used the BLEU (Papineni et al., 2002) and NIST (Martin and Przybocki, 2003) metrics, which are, to this day, very popular in Machine Translation evaluation. However, the experiments in (Štajner et al., 2014), in which the reliability of various Machine Translation evaluation metrics is compared, reveal that they do not correlate well with human judgments: the highest performing metric obtains no more than 0.234 Pearson correlation for grammaticality, and 0.442 for meaning preservation.

In this paper we describe our contribution to the QATS 2016 shared task on Text Simplification Quality Assessment. We introduce two new, supervised approaches to the task, which use different types of features and Neural Network architectures to tackle the evaluation problem.

2. Task, Datasets and Evaluation
In this first Quality Assessment for Text Simplification shared task, participants were required to create new ways of predicting the quality of simplified English sentences. The training and test sets provided contains 505 and 126 instances, respectively. Each instance is composed of a sentence in its original form, a simplified version, and four human-produced quality labels. The sentences were taken from news articles and Wikipedia, then simplified with by various systems. The quality labels had one of three values: “Good”, “Ok” and “Bad”. Participants were asked to assign these labels to four assessment aspects: Grammaticality, Meaning Preservation, Simplicity and Overall Quality.

Participants could submit up to three solutions for each aspect. Their solutions could either treat the task as a discrete classification problem, and hence provide a Good/Ok/Bad label as output for each instance, or as a continuous scoring problem, and hence produce a numerical quality coefficient. Classifiers were evaluated with Accuracy, MAE, RMSE and weighted F-measure, and scorers with Pearson correlation.

In order to avoid confusion, we hereon refer to “simplifications” as a pair composed by the original and the simplified sentence.

3. Resource-Light Neural Networks
Our intention in participating in the Quality Assessment task was to introduce a resource-light evaluation framework that exploits the proven effectiveness of deep Neural Network architectures. Since Neural Networks are very flexible, we were able to devise two distinct systems: SimpleNets-MLP and SimpleNets-RNN.

3.1. SimpleNets-MLP: A Deep Classifier
The SimpleNets-MLP is the simpler of the two systems. It consists on a Multi-Layer Perceptron composed of various dense hidden layers and a sigmoid activation layer with three nodes (one for each possible label). The architecture of SimpleNets-MLP is illustrated in Figure 1. SimpleNets-MLP receives as input a set of features that describe a given simplification, and outputs its quality label.
for a given aspect. For training, it uses a Categorical Cross-Entropy optimization function along with a Stochastic Gradient Descent learning algorithm.

We use the same 14 resource-light features for all quality aspects:

- Original and simplified sentences’ number of characters.
- Original and simplified sentences’ number of tokens.
- Original and simplified sentences’ number of types.
- Original and simplified sentences’ 5-gram language model probabilities from Wikipedia (Kauchak, 2013).
- Original and simplified sentences’ 5-gram language model probabilities from Simple Wikipedia (Kauchak, 2013).
- Original and simplified sentences’ 5-gram language model probabilities from SUBTLEX (Brysbaert and New, 2009).
- Original and simplified sentences’ 5-gram language model probabilities over SubIMDB (Paetzold and Specia, 2016).

Language models were trained with the help of SRILM (Stolcke and others, 2002).

### 3.2. SimpleNets-RNN: An N-gram Model

SimpleNets-RNN shares the same overall architecture of SimpleNets-MLP, but it is not a Multi-Layer Perceptron. Instead, it is a deep sequence-to-label network of recurrent Long Short-Term Memory (LSTM) hidden layers, tailed also by a three-noded dense sigmoid activation layer. Unlike SimpleNets-MLP, SimpleNets-RNN does not exploit engineered features, but an n-gram model and word embeddings.

The n-gram model works under the assumption that the quality of a sentence is the product of the quality of its n-grams. In other words, if the sentence is composed of grammatical, meaningful and simple n-grams, then it will be so too. In order to train a recurrent net that predicts the quality of a sentence based on the quality of its n-grams, we first decompose the instances in the dataset. For each n-gram of size $1 \leq s \leq N$ in the union of n-grams from the original and the simplified sentence, we create a new decomposed training instance. Each decomposed instance is represented by a label for a given quality aspect and a $N \times M$ matrix, in which $N$ is the maximum n-gram size of the model and $M$ the size of the word vectors being used. The label of each decomposed instance is the label of the simplification in the original training set. Each column of the matrix represents a time-step, and each line a word vector. In order to normalize the matrix sizes for n-grams with size $s < N$, we use padding. It uses the same optimization function and learning algorithm as SimpleNets-MLP.

During test time, we do the same: decompose each instance of the dataset into n-gram instances and predict their quality using the trained network. But with this setup, we obtain a series of quality predictions of individual n-grams, not a single quality label for the simplification as a whole, which is what the task requires. To transform n-gram quality labels into a single sentence quality label, we first transform Bad/Ok/Good labels into numerical scores using the rules below:

- Bad = 0.0
- Ok = 1.0
- Good = 2.0

We then average the label scores of all n-grams in a simplification and obtain a real-valued quality score. To infer a final quality label, we use a set of three equally sized label translation intervals:

- $[-\infty, 0.665] \rightarrow$ Bad
- $[0.665, 1.334] \rightarrow$ Ok
- $[1.334, \infty] \rightarrow$ Good

![Figure 1: Architecture of SimpleNets-MLP](image)

![Figure 2: SimpleNets workflow](image)


Figure 2 illustrates the SimpleNets workflow.

4. Experimental Setup

In order to assess the efficacy of our systems, we use training and test sets provided by the task organizers, which contain 505 and 126 instances, respectively. For evaluation, we resort to the two metrics described in the task’s description page: Accuracy and Pearson correlation.

We have trained an instance of SimpleNets-MLP and two instances of SimpleNets-RNN for each quality aspect included in the shared task, along with various baselines. The two versions of SimpleNets-RNN are SimpleNets-RNN2 and SimpleNets-RNN3, which employ bigram and trigram models, respectively. The architecture of each system was determined individually through 5-fold cross-validation for each quality aspect over the training set. The parameters of the architecture considered and the values tested for each one of them are:

1. Number of hidden layers: 1 to 5 in steps of 1.
2. Hidden layer size: 100 to 500 in steps of 100.
3. Word vector size: 100 to 1,500 in steps of 200.

The word vector models used were trained with word2vec (Mikolov et al., 2013). We use the bag-of-words (CBOW) model, and train the models over a corpus of around 7 billion words comprised by SubIMDB (Paetzold and Specia, 2016), UMBC webbase, SUBTLEX (Brysbaert and New, 2009), Wikipedia and Simple Wikipedia (Kauchak, 2013). The baselines included in our evaluation are:

- **All Bad**: Assigns “Bad” to all simplifications.
- **All Ok**: Assigns “Ok” to all simplifications.
- **All Good**: Assigns “Good” to all simplifications.
- **Meteor**: Computes Meteor scores between the original and simplified sentences (Denkowski and Lavie, 2011).
- **BLEU**: Computes BLEU scores between the original and simplified sentences (Papineni et al., 2002).
- **TER**: Computes TER scores between the original and simplified sentences (Snover et al., 2006).
- **WER**: Computes WER scores between the original and simplified sentences (Klavik and Peters, 2002).
- **MT**: Combines the Meteor, TER, and WER scores using Support Vector Machines trained on sentence labels.

The Meteor, BLEU, TER, WER and MT baselines were provided by the task organizers. We also compare the performance of our Neural Networks to that of other five Machine Learning classification techniques:

- Support Vector Machines.
- Ada Boosting.
- Decision Trees.
- Random Forests.
- Gradient Boosting.

As input, we use the same features described in Section 3.1. To train models using these algorithms, we use their implementation within scikit-learn (Pedregosa et al., 2011).

For correlation assessment, all classifiers, including SimpleNets-MLP, output the labels’ numerical equivalent following the method described in Section 3.2.. For the SimpleNets-RNN systems, however, correlation scores are calculated based on the raw average n-gram quality score, and Accuracy based on the label translation intervals as described in Section 3.2.

5. Results

Table 1 shows the Accuracy (A) and Pearson correlation (r) scores obtained. Complementary MAE, RMSE and F-measure scores can be found in the task’s webpage.

The Accuracy scores for most baselines suggest that their overall performance is not much superior, and often even worse, than predicting the same label for all instances. Pearson correlation scores reveal that Machine Learning approaches are, in fact, much better alternatives. Machine Translation evaluation metrics have proved a much less reliable approach to Text Simplification evaluation.

Our proposed metrics obtained very competitive scores according to all four simplification aspects. According to Accuracy, they ranked 3rd in Meaning and Simplicity, 2nd in Grammaticality, and 1st in Overall Quality. While SimpleNets-RNN classifiers have shown to be more proficient in predicting Grammaticality and Overall Quality, SimpleNets-MLP performed best in predicting Meaning and Simplicity. We believe that the learning setup used by SimpleNets-RNN is not very suitable for Meaning and Simplicity, and consequently led to the rather low scores obtained for these aspects. For Meaning, we believe that it would be more effective to train Neural Networks that attempt to learn not the quality of n-grams with respect to Meaning, but rather how compromising the deletions and additions in the simplified sentence are with respect to the original sentence. For Simplicity, an RNN that receives the entire sentence as input instead of separate n-grams could, in principle, better capture sentence length as well as long distance dependencies, which could also improve the performance of our approach.

---

1http://ebiquity.umbc.edu/resource/html/id/351
2http://www.statmt.org/wmt11/translation-task.html
<table>
<thead>
<tr>
<th>System</th>
<th>Grammaticality</th>
<th>Meaning</th>
<th>Simplicity</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>r</td>
<td>A</td>
<td>r</td>
</tr>
<tr>
<td>All Ok</td>
<td>11.11</td>
<td>0.050</td>
<td>26.19</td>
<td>-0.074</td>
</tr>
<tr>
<td>All Bad</td>
<td>12.70</td>
<td>0.050</td>
<td>15.87</td>
<td>-0.074</td>
</tr>
<tr>
<td>All Good</td>
<td>76.19</td>
<td>0.050</td>
<td>57.94</td>
<td>-0.074</td>
</tr>
<tr>
<td>Meteor</td>
<td>63.49</td>
<td>0.384</td>
<td>61.90</td>
<td>0.527</td>
</tr>
<tr>
<td>BLEU</td>
<td>69.84</td>
<td>0.344</td>
<td>65.08</td>
<td>0.533</td>
</tr>
<tr>
<td>TER</td>
<td>68.25</td>
<td>0.323</td>
<td>66.67</td>
<td>0.513</td>
</tr>
<tr>
<td>WER</td>
<td>65.87</td>
<td>0.308</td>
<td>62.70</td>
<td>0.495</td>
</tr>
<tr>
<td>MT</td>
<td>76.19</td>
<td>-</td>
<td>66.67</td>
<td>-</td>
</tr>
<tr>
<td>SVM</td>
<td>76.19</td>
<td>0.047</td>
<td>57.94</td>
<td>-0.050</td>
</tr>
<tr>
<td>Ada Boosting</td>
<td>73.81</td>
<td>0.148</td>
<td>58.73</td>
<td>0.205</td>
</tr>
<tr>
<td>Decision Trees</td>
<td>65.87</td>
<td>0.073</td>
<td>53.97</td>
<td>0.191</td>
</tr>
<tr>
<td>Random Forests</td>
<td>69.05</td>
<td>0.020</td>
<td>59.52</td>
<td>0.348</td>
</tr>
<tr>
<td>Gradient Boosting</td>
<td>69.05</td>
<td>0.027</td>
<td>65.87</td>
<td>0.391</td>
</tr>
<tr>
<td>Davoodi-0.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Davoodi-0.6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Davoodi-0.7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Davoodi-RF</td>
<td>-</td>
<td>-</td>
<td>49.21</td>
<td>-</td>
</tr>
<tr>
<td>Davoodi-RF-0.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Davoodi-RF-0.6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Davoodi-RF-0.7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Davoodi-RF-Perp</td>
<td>58.73</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DeepBow</td>
<td>76.19</td>
<td>-</td>
<td>53.17</td>
<td>-</td>
</tr>
<tr>
<td>DeepIndiBow</td>
<td>76.19</td>
<td>-</td>
<td>47.62</td>
<td>-</td>
</tr>
<tr>
<td>MS-IBk</td>
<td>63.32</td>
<td>-</td>
<td>61.90</td>
<td>-</td>
</tr>
<tr>
<td>MS-IBk-best</td>
<td>-</td>
<td>-</td>
<td>62.70</td>
<td>-</td>
</tr>
<tr>
<td>MS-RandForest</td>
<td>72.22</td>
<td>-</td>
<td>66.67</td>
<td>-</td>
</tr>
<tr>
<td>MS-RandForest-best</td>
<td>69.84</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Mathias-Default</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Mathias-LM</td>
<td>72.22</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Mathias-Meteor</td>
<td>-</td>
<td>-</td>
<td>63.49</td>
<td>-</td>
</tr>
<tr>
<td>Mathias-Metrics</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Mathias-S</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OSVCML</td>
<td>-</td>
<td>0.340</td>
<td>-</td>
<td><strong>0.585</strong></td>
</tr>
<tr>
<td>OSVCML1</td>
<td>-</td>
<td>-</td>
<td>0.482</td>
<td>-</td>
</tr>
<tr>
<td>OSVCML2</td>
<td>-</td>
<td>-</td>
<td>0.573</td>
<td>-</td>
</tr>
<tr>
<td>SMH-IBk</td>
<td>70.63</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SMH-Logistic</td>
<td>-</td>
<td>-</td>
<td><strong>69.05</strong></td>
<td>-</td>
</tr>
<tr>
<td>SMH-RandForest</td>
<td>75.40</td>
<td>-</td>
<td>65.87</td>
<td>-</td>
</tr>
<tr>
<td>SMH-RandForest-best</td>
<td>75.40</td>
<td>-</td>
<td>61.90</td>
<td>-</td>
</tr>
<tr>
<td>UoLG-combo</td>
<td>-</td>
<td>0.256</td>
<td>-</td>
<td>0.250</td>
</tr>
<tr>
<td>UoLG-emb</td>
<td>-</td>
<td>0.256</td>
<td>-</td>
<td>0.188</td>
</tr>
<tr>
<td>UoLG-quest</td>
<td>-</td>
<td>0.208</td>
<td>-</td>
<td>0.285</td>
</tr>
<tr>
<td>UoW</td>
<td>41.27</td>
<td>-</td>
<td>63.49</td>
<td>-</td>
</tr>
<tr>
<td>SimpleNets-MLP</td>
<td>74.60</td>
<td>0.309</td>
<td>65.87</td>
<td>0.461</td>
</tr>
<tr>
<td>SimpleNets-RNN2</td>
<td>75.40</td>
<td>0.058</td>
<td>57.94</td>
<td>0.262</td>
</tr>
<tr>
<td>SimpleNets-RNN3</td>
<td>74.60</td>
<td>0.068</td>
<td>51.59</td>
<td>0.258</td>
</tr>
</tbody>
</table>

Table 1: Accuracy and Pearson correlation scores

6. Conclusions

We have introduced the SimpleNets-MLP and SimpleNets-RNN systems: two different solutions for Text Simplification Quality Assessment that combine resource-light features with minimalistic Neural Network architectures. The results show that, while our Multi-Layer Perceptrons trained over simple features (SimpleNets-MLP) are more reliable at predicting Meaning and Simplicity, our Recurrent Neural Networks that predict and combine the quality of individual n-grams (SimpleNets-RNN) achieve the state-of-the-art performance in predicting the Overall Quality of simplifications.

In future work, we aim to explore the use of more sophisticated n-gram models and Recurrent Neural Network architectures in other domains of Quality Assessment, such as Machine Translation and Summarization.
References


An Ensemble Method for Quality Assessment of Text Simplification

Sergiu Nisioi\textsuperscript{\dagger}, Fabrice Nauze\textsuperscript{*}
University of Bucharest\textsuperscript{\dagger},
Oracle Service Cloud\textsuperscript{*}
sergiu.nisioi@oracle.com, fabrice.nauze@oracle.com

Abstract
In this paper we describe the Oracle Service Cloud Machine Learning (OSVCML) systems used for the Quality Assessment of Text Simplification, 2016 (QATS) shared task. We construct an ensemble method using particle swarm optimization and different scoring methods (SVM, string kernels, logistic regression, boosting trees, BLEU). The purpose is to capture relevant combinations of classifier and features for each different aspects of text simplification: simplicity, grammaticality, meaning preservation, and overall scores. In addition, we compare our approach with a deep neural network architecture and show that the generated models are stronger when combined together.

Keywords: Text Simplification Evaluation, Particle Swarm Optimization, Ensemble Methods, LSTM

1. Introduction
The need for simplified texts has largely been acknowledged by international instances like the United Nations or the European Union, with text simplification guidelines being issued to clarify the process for writers. The obvious next step has been to build computational systems to automate the text simplification process. We are now facing the last stage in this evolution where we need to evaluate our systems for automatic text simplification (ATS).

In recent years, text simplification has been viewed as a statistical machine translation process which involves translating from a complex to a simplified version of a sentence (Specia, 2010; Coster and Kauchak, 2011; Wubben et al., 2012; Stajner et al., 2015). Therefore, the structure of ATS datasets (Kauchak, 2013) is often based on sentence-aligned parallel corpora and the evaluation methods resemble the BLEU score (Papineni et al., 2002) and other machine translation metrics (Clarke and Lapata, 2006; Kauchak, 2013). However, there is no standard procedure for the automatic evaluation of text simplification, nor an expected standard baseline, rather the best evaluation is carried using supervised human annotators, which can potentially produce high quality datasets for text simplification evaluation.

Therefore, in our study we employ features widely used in text classification and train several classifiers with the judgments provided by human evaluators. Our approach consists of multiple scoring methods: support vector machines, logistic regression, gradient boosting trees, BLEU metric, string kernels and deep neural encoder-decoder architectures, aggregated using an ensemble method optimized with swarms of particles.

2. Corpus
The dataset provided for the QATS shared task\textsuperscript{1} consists of phrases from news domain and Wikipedia, and the corresponding simplified version (generated using various automatic text simplification methods). The data was annotated by judges along four dimensions: grammaticality, meaning preservation, simplicity, and overall score, with labels (good, ok, bad) representing how well the systems performed. The provided training set consists of 505 phrase pairs and judgments, and the test set contains of 126 phrase pairs.

The QATS dataset provided in the shared task is a useful and rare resource, however some critical observations are worth being made. The dataset used is not balanced in terms of different numbers of classes for each metric, a fact which makes the models harder to train properly and the systems harder to evaluate. In addition, we don’t have multiple opinions on the label of a phrase form different manual annotators, a fact that can determine a certain bias in the way the gold standard is constructed. We are aware, however, that this is probably the best dataset we currently have to evaluate text simplification.

3. Features
We combine two approaches on this task: (1) involving text classification with features extracted either from the simplified (S) or from the original (O) phrases and (2) using an architecture based on deep neural networks, which takes as input both original and simplified phrases in an attempt to learn the similarities and differences between them.

We use a wide set of features to cover different aspects that vary in simplified texts, features ranging from simple tokens, part of speech tags, presence of negation, readability measures and sentiment information. A summary of all the features we use is available in Table 1.

We consider content independent features, those that do not directly reflect topic, genre or content information from the text. For example, part of speech tags, syntactic chunks, dependency parse trees, punctuation marks, verb negation, and readability measures should reflect grammaticality irrespective of the topics addressed in the text. On the other hand, content dependent features are meant to capture semantic shifts from original to simplified sentences, meaning preservation, common content words, and possibly specific marks that are present only in certain classes of simplified examples. Among the content dependent features,

\textsuperscript{1}http://qats2016.github.io/shared.html
we count polarity information, tokens, character n-grams, and global word vectors (word embeddings (Pennington et al., 2014) extracted from Common Crawl corpus\(^2\)). We are aware that content independent features, may reflect different patterns of style, genre, and possibly semantic information present in grammatical structures.

We use two feature extractors: NLTK (Bird et al., 2009) and a proprietary software created by the Oracle Language Technology (OLT) group. OLT delivers linguistic technologies and data components for use in Oracle search and text analysis applications requiring support for numerous languages. It currently supports 86 languages, 16 technologies, and 5 APIs in integrations with over 8 Oracle products and applications (Oracle, 2014). However, we do not compare the two toolkits here, rather we use them together for more accurate sentence splitting, syntactic chunking and POS tagging. We count the number of sentences and the length (in words) of the longest sentence. Similarly, we count the number of NN and NP syntactic chunks and the length of the longest group. Tokens and parts of speech are weighted according to frequency, tf-idf and the total number of tokens / POS in the simplified and original phrase. Last but not least, character n-grams are used as features in the string kernel representation.

Overall we create three classes of features:

1. \(Ft_1 = S_{t-f-idf} \times \) tf-idf of parts of speech and text \((S_{t-f-idf})\) from the simplified examples
2. \(Ft_2 = |S_{t-f-idf} - O_{t-f-idf}| + S_{t-f-idf} \times O_{t-f-idf} \times \) tf-idf extracted from both simplified \((S_{t-f-idf})\) and original \((O_{t-f-idf})\) parts of speech and text
3. \(Ft_3\) - simple numerical features obtained from token, parts of speech, sentence, chunks and other counts, readability measure, and polarity values

These classes are used in combination with multiple scoring methods to create an ensemble based on which we will decide our final predictions.

Table 1: A complete list of content dependent and independent features used. Quantification reflects how are the features represented numerically. The last column indicates the class \((Ft_1, Ft_2, Ft_3)\) or machine learning method that is being compiled with the respective features.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Quantification</th>
<th>Feature class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tokens</td>
<td>count, total no., tf-idf</td>
<td>(Ft_1, Ft_2, Ft_3) LSTM architecture</td>
</tr>
<tr>
<td>GloVe global word vectors</td>
<td>from Common Crawl corpus n = 2, 3, 4</td>
<td>string kernel (Ft_3)</td>
</tr>
<tr>
<td>Character n-grams</td>
<td>positive and negative polarity</td>
<td>(Ft_3)</td>
</tr>
<tr>
<td>Polarity information</td>
<td></td>
<td>(Ft_3)</td>
</tr>
<tr>
<td>Parts of speech</td>
<td>count, total no., tf-idf</td>
<td>(Ft_1, Ft_2, Ft_3)</td>
</tr>
<tr>
<td>Punctuation marks</td>
<td>count</td>
<td>(Ft_3)</td>
</tr>
<tr>
<td>Sentences</td>
<td>no. of sentences, length of the longest sentence</td>
<td>(Ft_3)</td>
</tr>
<tr>
<td>NP chunks</td>
<td>length of the longest NP group</td>
<td>(Ft_3)</td>
</tr>
<tr>
<td>NN chunks</td>
<td>no. of chunks, no. of terms in each chunk</td>
<td>(Ft_3)</td>
</tr>
<tr>
<td>Flesch-Kincaid readability</td>
<td>score, minimum age, US grade</td>
<td>(Ft_3)</td>
</tr>
<tr>
<td>Negated tokens</td>
<td>no. of negated tokens</td>
<td>(Ft_3)</td>
</tr>
<tr>
<td>Dependency parse trees</td>
<td></td>
<td>(Ft_3)</td>
</tr>
</tbody>
</table>

In addition, we use the Stanford parser (Chen and Manning, 2014) to construct deep neural networks using the dependency parse relations and GloVe word embeddings. We expect this approach to cover both syntactic and semantic differences between the simplified and original sentences.

4. Our Approach

4.1. Scoring Methods

We use different scoring methods comprising classifiers, regression models, and similarity measures, to predict a value for each example. The values are aggregated and weighted using a particle swarm optimization-based ensemble approach, which we detail in Section 4.2. Table 2 contains the pairs of scoring methods and features.

**BLEU** The modified n-gram precision metrics are the building blocks of BLEU score (Papineni et al., 2002), computed as the number of common token n-grams between two sentences divided by the total number of occurrences of those sentences. In our case, we have computed the modified unigram and bigram precision - shorter n-grams indicating the preservation of meaning while longer n-grams indicate the preservation of fluency (Papineni et al., 2002). In addition to the standard score, we computed the modified n-gram precision between parts of speech, hoping to cover grammatical changes between POS sequences in original and simplified sentences.

**SVM** We decided to use a linear kernel to train support vector machines for text classification (Chang and Lin, 2011). These models proved effective in numerous text classification studies (Koppel et al., 2009; Zampieri et al., 2015), especially when the number of features is relatively large. Therefore, we trained different classifiers using the three classes of features previously defined: \(Ft_1, Ft_2\) and \(Ft_3\), to predict the final label (good, bad, ok).

**SVM + String Kernel** A kernel function can be used in combination with support vector machines either to embed the data in a higher dimensional space in order to achieve linear separability, or to replace the dot product between
vectors with values that are more appropriate for the data used. Previous studies on text classification, revealed that character n-gram-based string kernels can be effective tools for authorship attribution, native language identification or plagiarism detection (Grozea and Popescu, 2010; Ionescu et al., 2014).

On our datasets, however, we are faced with relatively short-length examples of either simplified or original phrases. Therefore, we propose a kernel suitable for short texts, computed by summing the number of common character n-grams between two examples, where \( n = 2, 3, 4 \). The kernel function, in our case, will measure common aspects in terms of content words, punctuation marks, and affixes, based on character n-grams similarities.

Formally, given an alphabet \( A \), we define the mapping function \( \Phi_n : \mathcal{D} \rightarrow \{0,1\}^{Q_n} \) for an example \( e \in \mathcal{C} \) in the dataset to be the vector of all the binary values of existence of the n-gram \( g \) in the example:

\[
\Phi_n(e) = [\phi_g(e)]_{g \in A^n}
\]

The function \( \phi_g(e) = 1 \) if the n-gram \( g \) is in the example \( e \) and equal to zero otherwise. Computationally, \( Q_n \) depends on all the possible character n-grams between two examples at certain instance. The corresponding Gram matrix \( K \) of size \( |\mathcal{C}| \times |\mathcal{C}| \) has the following elements:

\[
K_{ij} = \sum_{n=2}^{n \leq 4} \langle \Phi_n(e_i)\Phi_n(e_j) \rangle
\]

The gram matrix is then normalized to \([0,1]\) interval:

\[
K_{ij} = \frac{K_{ij}}{\sqrt{K_{ii}K_{jj}}}
\]

### Logistic Regression

In addition, we wanted to add regression values to our ensemble in order to have fine-grained predictions of the classes. For this purpose, we train a logistic regression model (Fan et al., 2008) in one-vs-rest approach, by fitting a binary classifier with \( \ell_2 \) penalty, for each label. In this context, we use the three classes of features (\( F_{t_1}, F_{t_2}, F_{t_3} \)) this situation.

### Gradient Boosting Trees

Last but not least, for each feature class, we train a gradient boosting ensemble classifier (Friedman, 2002)\(^3\) with softmax multi-class objective, 120 trees and a max depth of 12.

---

### Table 2: Different classifiers used in the ensemble

<table>
<thead>
<tr>
<th>Classifier or Metric</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modified n-gram precision</td>
<td>Original vs. Simplified parts of speech</td>
</tr>
<tr>
<td>Modified n-gram precision</td>
<td>Original vs. Simplified tokens</td>
</tr>
<tr>
<td>String Kernel, ( n=2, 3, 4 )</td>
<td>Simplified text and POS</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>( F_{t_1}, F_{t_2}, F_{t_3} )</td>
</tr>
<tr>
<td>Linear SVM</td>
<td>( F_{t_1}, F_{t_2}, F_{t_3} )</td>
</tr>
<tr>
<td>Gradient Boosting Trees</td>
<td>( F_{t_1}, F_{t_2}, F_{t_3} )</td>
</tr>
<tr>
<td>Long Short-Term Memory</td>
<td>GloVe word vectors and dependency parse trees</td>
</tr>
</tbody>
</table>

---

![Figure 1: Ensemble architecture with PSO-learned weights](https://github.com/dmlc/xgboost)

Thus, we obtain nine different scoring methods by training support vector machines, logistic regression and gradient boosting trees on each class of features defined in Section 3: \( F_{t_1}, F_{t_2}, F_{t_3} \). In addition, we added the SVM String Kernel, and the two modified n-gram precision computations (tokens and parts of speech), in total having as much as twelve different prediction values for each example. To aggregate these values, we train a particle swarm optimization-based ensemble method to learn the different weights corresponding to each predicted value as described in the following section.

### 4.2. Ensemble Method

Through our approach, we construct \( N \) scoring methods comprising classifiers, regression models and similarity metrics, which we will denote by \( C_1, C_2, \ldots, C_N \). The final prediction value is determined by a linear combination of the predictions generated by each individual scoring method:

\[
prediction = w_1C_1 + w_2C_2 + \cdots + w_NC_N
\]

The results of each individual method are weighted and summed-up together to ensemble the final prediction value. In other words, for each example in the test set, a scoring method \( C_i \) will predict either a class, a similarity score or a regression value. Then, each prediction is weighted by the corresponding \( w_i \) and summed up with other predictions in a linear combination. The architecture is represented in Figure 1. The problem now remains to properly find the optimal weights for each scoring method, given an objective function, and, at the same time, to avoid weights that over-fit individual training / testing datasets.

Pearson correlation \( \rho \in [-1,1] \) measures the degree of dependence between two random variables \( X = \{x_i\} \) and...
\[ Y = \{ y_i \} \text{ of size } n, \text{ and has the following formula:} \]
\[ p = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}} \tag{2} \]
where \( \bar{x} \text{ and } \bar{y} \text{ are the sample mean of } X \text{ and } Y, \text{ respectively.} \]
We use the negative log of squared Pearson correlation as objective function. Therefore, we maximize the correlation between the predicted and the gold standard examples by minimizing the objective function. Our objective function becomes:
\[ \text{Obj} = -\log(p^2) \tag{3} \]
To search for the optimal weights, we use particle swarm optimization (PSO) (Kennedy and Eberhart, 1995) - effective for fast optimizations when gradients are not available or when they are not trivial to compute. This strategy uses a swarm of particles (a set of different solutions) for a given problem which are individually adjusted based on nearest neighbor and a velocity variable (0.55 in our case). We initialize two hundred particles to search between [0, 1] values for each weight, and let the algorithm run for a maximum of 1000 iterations. Particle swarm optimization is able to find weights to minimize an arbitrary function, however, the weights are not necessarily optimal, since the approach can converge to a local minimum.
We treat this problem by computing a set of candidate weights from multiple cross validation (CV) epochs in which the training and development data are disjointly split. This way, we avoid over-fitting the weights on individual train-development pairs. The mean values of all the weight-candidates, computed over multiple CV epochs, are selected as final weights for the ensemble method. For reproducibility, parts of the source code and data will be made available in a public repository\(^5\).

### 4.3. PSO for Class Prediction

A particle swarm optimization method can be used to predict the best threshold values to convert floating points to classes. Having a three-class classification problem, we can split the resulting predictions based on two thresholds (\( \text{min}_t \) and \( \text{max}_t \)); therefore, the values below \( \text{min}_t \) are considered bad, the values above \( \text{max}_t \) are considered good and the values in between are labeled ok. For our data, classes can be ranked by the following scale:

- bad < \( \text{min}_t \) ≤ ok ≤ \( \text{max}_t \) < good

We use each cross-validation epoch to search for the \( \text{min}_t \) and \( \text{max}_t \) values in the min-max interval of predictions, to maximize the classification accuracy (as objective function in the PSO algorithm). Over different CV epochs, these values can vary significantly, especially when the predicted classes are not evenly balanced. Therefore, we use the mean \( \text{min}_t \) and \( \text{max}_t \) values over all cross-validation epochs, to predict the final results.

\(^4\)By disjoint, we mean that the development examples do not contain original, unsimplified sentences from the training set.

\(^5\)https://github.com/seniiou/qats-pso-ensemble.git

### 4.4. Deep Learning Approach

Recurrent neural networks have recently been used in numerous NLP studies, offering solid results for different tasks and applications, from language modelling, machine translation, evaluation of machine translation, and semantic measures of similarity (Cho et al., 2014; Karpathy et al., 2015; Bengio et al., 2003). Our approach is based on the work of Tai et al. (2015) for semantic relatedness and Gupta et al. (2015) for machine translation evaluation. Thus, we build a recurrent neural network architecture based on dependency parse trees using both the simplified and the original sentences.

The dependency parse trees are constructed from example pairs, which are then used to build the LSTM-based recurrent neural network (TreeLSTM). We use a three layered TreeLSTM, with 30 hidden units, 0.05 learning rate and a batch size of 55 phrases. The inputs for the neural networks are word embeddings corresponding to tokens from each sentence. The metric between original and simplified sentences is defined as a LSTM unit that uses the hidden states obtained from the original and simplified TreeLSTM representations:

\[ h_{\text{mul}} = h_{\text{orig}} \cdot h_{\text{simpl}} \]
\[ h_{\text{add}} = h_{\text{orig}} + h_{\text{simpl}} \]
\[ h_s = \sigma(W^{(\text{mul})}h_{\text{mul}} + W^{(\text{add})}h_{\text{add}} + b^{(h)}) \]
\[ p_0 = \text{softmax}(W^{(p)}h_s + b^{(p)}) \]
\[ \hat{y} = r^T \hat{p}_0 \]

where \( \hat{p}_0 \) is the estimated probability distribution, and \( r^T = [1, 2, 3] \) are the annotated labels encoded as integers. The sparse target distribution, as defined in Tai et al. (2015) is:

\[ p_i = \begin{cases} \ y - \lfloor y \rfloor, & \text{if } i = \lfloor y \rfloor + 1 \\ \lfloor y \rfloor - y + 1, & \text{if } i = \lfloor y \rfloor \\ 0, & \text{otherwise} \end{cases} \]

The cost function is defined over probability distributions \( p \) and \( \hat{p}_0 \) using regularized Kullback-Leibler (KL) divergence.

This approach predicts a floating value representing the similarity relation between original and simplified sentences. In order to predict the labels, we use at each node \( j \) a softmax classifier to predict the label \( \hat{y} \) given the inputs \( \{x\} \) observed at nodes in the subtree rooted at \( j \):

\[ p_0(y|\{x\}_j) = \text{softmax}(W^{(s)}h_s + b^{(s)}) \]
\[ \hat{y}_j = \text{argmax}(p_0(y|\{x\}_j)) \]

Training the neural networks over several epochs can be time consuming, therefore, we do not use the TreeLSTM model directly to train our ensemble method. We decided to average the output generated by the deep neural networks with the predictions of the PSO ensemble, equally weighting both models.
5. Results and Discussion

We report only our final results submitted for the QATS 2016 shared task. The predictions are made on the test corpus provided by the organizers.

In addition, we convert our floating values into good, bad, ok by splitting them according to min\(_d\) and max\(_d\) computed as mean values over every cross-validation step. The TreeLSTM system is retrained to predict both scores and individual classes.

In what follows, we describe the classifiers submitted for the shared task.

Simplicity: When predicting simplicity, we use three ensembles:

1. PSO Simplifi\(_1\) - which does not contain features extracted with OLT - NP groups, OLT parts of speech, punctuation, OLT sentence splits

2. PSO Simplifi\(_2\) - which contains both features extracted with OLT and features extracted with NLTK

3. PSO Simplifi\(_3\) - uses the same features as PSO Simplifi\(_2\), but it is trained as three pairwise classifiers between: ok vs. bad, ok vs. good, and bad vs. good. The final label is decided based on the maximum number of votes from each classifier.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Correlation</th>
<th>Accuracy</th>
<th>(F_1) measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. PSO Simplifi(_1)</td>
<td>0.342</td>
<td>0.555</td>
<td>0.537</td>
</tr>
<tr>
<td>2. PSO Simplifi(_2)</td>
<td>0.382</td>
<td>0.579</td>
<td>0.527</td>
</tr>
<tr>
<td>3. PSO Simplifi(_3)</td>
<td>0.376</td>
<td>0.579</td>
<td>0.522</td>
</tr>
</tbody>
</table>

Table 3: Main results for simplicity labels

In this particular case, we do not use the TreeLSTM approach since the human judgments were made only by looking at the simplified phrase, irrespective of the original. The results for each ensemble are depicted in Table 3. The pairwise approach together with the features extracted using OLT improved the overall evaluation.

Meaning: To predict meaning preservation, we construct (1) a PSO ensemble with OLT features, (2) a TreeLSTM neural network, and (3) an average between the two. The PSO ensemble method performs better than any other model, and it’s worth noticing that the ensemble combined with recurrent neural networks (model nr. 3) lowers the accuracy values. Empirical experiments showed that it is more efficient to directly predict the overall results, than to use each individual label (meaning, grammaticality or simplicity) as a scoring method in the ensemble.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Correlation</th>
<th>Accuracy</th>
<th>(F_1) measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. PSO Gramat(_1)</td>
<td>0.323</td>
<td>0.7</td>
<td>0.657</td>
</tr>
<tr>
<td>2. PSO Gramat(_2)</td>
<td>0.483</td>
<td>0.753</td>
<td>0.678</td>
</tr>
</tbody>
</table>

Table 5: Main results for grammaticality labels

Overall: The final predictions for overall labels are built using (1) a PSO ensemble model, (2) the TreeLSTM approach described in Section 4.4. and (3) a combination between the two. The PSO ensemble proved better than the recurrent neural networks approach (see Table 6), and the combination between the two obtained the best \(F_1\) score and accuracy values. Empirical experiments showed that it is biased by the large numbers of good grammatical sentences in the test data. We submitted two PSO ensembles, one trained on the entire dataset provided by the organizers and another one trained on a random subset of the training data (a local subset) which was used in cross-validation scenarios. To our surprise, the local dataset proved to better fit the test data provided by the organizers (see Table 5) obtaining better results compared to the system trained on the entire training data (0.482 vs. 0.38). We are inclined to believe, this accidental phenomenon reveals that similarities between training and testing datasets can greatly influence the results. Given this situation, we consider additional studies to be mandatory for a better understanding of the these results.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Correlation</th>
<th>Accuracy</th>
<th>(F_1) measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. PSO ensemble</td>
<td>0.332</td>
<td>0.49</td>
<td>0.43</td>
</tr>
<tr>
<td>2. TreeLSTM</td>
<td>0.230</td>
<td>0.43</td>
<td>0.41</td>
</tr>
<tr>
<td>3. PSO ensemble and TreeLSTM</td>
<td>0.343</td>
<td>0.5</td>
<td>0.46</td>
</tr>
</tbody>
</table>

Table 6: Main results for overall labels

A brief analysis of the classifiers involved in the ensemble revealed that the simple features, the majority in \(F_{13}\) class, do not contribute significantly to the overall results. We managed to observe improvements by using negation, sentiment and readability metrics, we believe these features can capture small changes in register or meaning from original to simplification. In addition, the n-gram precision metric performed surprisingly well for meaning prediction, being the simplest and most effective single metric for meaning evaluation. Our adapted string kernel measures fine-grained similarities covering content, punctuation marks, affixes etc., but, at the same time, it obfuscate behind the Gramm matrix the actual features that contribute to classification results, making hard to interpret the features that are useful for this task. Finally, the deep neural network architecture (TreeLSTM) obtained comparable accuracy and correlation scores by itself, apart from this, it improved the prediction of the overall judgments in combination with the particle swarm optimization ensemble method.
6. Conclusions and Future Work

We describe a particle swarm optimized ensemble method trained in order to aggregate multiple strong classifiers. The overall correlation values and the accuracy values are not exceedingly high for the quality assessment of text simplification, making the evaluation of such algorithms a task not yet easily solvable by machines.

Our proposed ensemble can be inspected to observe the influence of each scoring method for different tasks, nevertheless, the results are hard to interpret in terms of individual features. The scoring methods used in the ensemble cover a wide variety of textual features and more work is needed to better understand the usefulness of each feature set. As future work, we plan to carry a more exhaustive analysis of the automatic text simplification models together with the relevant features that improve evaluation.

7. Bibliographical References


CLaC @ QATS: Quality Assessment for Text Simplification

Elnaz Davoodi and Leila Kosseim
Concordia University
Montreal, Quebec, Canada
e_davoo@encs.concordia.ca, kosseim@encs.concordia.ca

Abstract
This paper describes our approach to the 2016 QATS quality assessment shared task. We trained three independent Random Forest classifiers in order to assess the quality of the simplified texts in terms of grammaticality, meaning preservation and simplicity. We used the language model of Google-Ngram as feature to predict the grammaticality. Meaning preservation is predicted using two complementary approaches based on word embedding and WordNet synonyms. A wider range of features including TF-IDF, sentence length and frequency of cue phrases are used to evaluate the simplicity aspect. Overall, the accuracy of the system ranges from 33.33% for the overall aspect to 58.73% for grammaticality.

Keywords: Simplification, Word Embedding, Language Model

1. Introduction
Automatic text simplification is the process of reducing the complexity of a text to make it more accessible to a broader range of readers with different readability levels. While preserving its meaning as much as possible, a text’s lexical, syntactic and discourse level features should be made more simple. However, evaluating the simplicity level of a text is still a challenging task for both humans and automatic systems.

Current approaches to automate text simplification vary depending on the type of simplification. Lexical simplification was the first effort in this area. In particular, Devlin and Tait (1998) introduced an approach of replacing words with their most common synonym based on frequency (Kučera et al., 1967). More recently, publicly available resources such as Simple English Wikipedia [1] and the Google 1T corpus [2] have been used to automate lexical simplification based on similar approaches such as common synonym replacement and context vectors (e.g. (Biran et al., 2011; Bott et al., 2012; Rello et al., 2013; Kauchak, 2013)).

Another approach to automatic text simplification involves syntactic simplification. Current work in this area aims to identify and simplify complex syntactic constructions such as passive phrases, embedded clauses, long sentences, etc. Initial work on syntactic simplification focused on the use of transformation rules in order to generate simpler sentences (e.g. Chandrasekar and Srinivas (1997)). Later, work have paid more attention on sentence splitting (e.g. Carroll et al. (1998), rearranging clauses (e.g. Siddharthan (2006)) and dropping clauses (e.g. Barlacchi and Tonelli, 2013; Stajner et al., 2013)). To our knowledge, Siddharthan (2003) is the only effort that specifically addressed the preservation of a text’s discourse structure by resolving anaphora and ordering sentence.

In the remainder of this paper, we describe the methodology we used to measure the 4 simplification criteria of the QATS workshop: GRAMMATICALITY, MEANING PRESERVATION, SIMPLICITY and OVERALL. In Sections 2 and 3, the details of our submitted system are described, while Section 4 summarises our results.

2. System Overview
As can be seen in Figure 1, our system consisted of three independent supervised models in order to predict each of the three main aspects: GRAMMATICALITY, MEANING PRESERVATION and SIMPLICITY. We used 10 fold cross-validation in order to choose the best supervised models. The 4th aspect (i.e. OVERALL) was predicted using the predictions of MEANING PRESERVATION and SIMPLICITY.

2.1. Grammaticality Prediction
In order to predict the quality of the simplified sentences from the point of view of grammaticality, we have used the log likelihood score of the sentences using the Google Ngram corpus [3]. To do this, the BerkeleyLM language modeling toolkit [4] was used (Pauls and Klein, 2011) to built a language model from the Google Ngram corpus, then the perplexity of all simple sentences in the training set were calculated. These log likelihood scores were used as features to feed a Random Forest classifier.

2.2. Meaning Preservation Prediction
The purpose of meaning preservation is to evaluate how close the meaning of the original sentence is with respect to its simple counterpart. To do this, we used two complementary approaches based on word embedding and the cosine measure.

2.2.1. Word Embedding
We used the Word2Vec package (Mikolov et al., 2013a) [5] to learn the representation of words on the Wikipedia dump [6]. We then trained a skip-gram model using the deeplearning4j library. As a result, each

1. https://books.google.com/ngrams
simplification/
3. https://books.google.com/ngrams
simplification/
word in the original sentence and its simple counterpart are represented as a vector. As calculating the similarity of two sentences using word embedding is still a challenging task, our approach to this problem was to use average similarity. To do so, we calculated the similarity of each word (each vector) in the original sentence to all the words in its simpler counterpart. Then using the average length of the original and simple pairs, we calculated the average similarity between a pair of sentences. This similarity was the first feature we fed to a Random Forest classifier.

2.2.2. Cosine Similarity of WordNet Synonyms
The second feature we used for meaning preservation was based on WordNet synonyms. Each sentence was represented as a vector of its constituent words. Then, using WordNet all synonyms of each word were added to the corresponding vector of the sentence. However, as each word can have various part of speech (POS) tags, before expanding the vector, we first identified the POS of all the words in the sentence using the Stanford POS tagger (Manning et al., 2014). Afterwards, we filtered the synonyms according to the POS tags and added only those with the same POS tag of the word. As a result, each sentence was represented as a vector of words and their synonyms. Using the cosine similarity to calculate the similarity between corresponding vectors of pairs of sentences, we measured how close the meaning of two sentences were.

![Figure 1: System Overview](https://wordnet.princeton.edu/)

2.3. Simplicity Prediction
The purpose of simplicity prediction is to evaluate how simpler the simple sentences are compared to their original counterpart. As simplicity can be measured at various levels (i.e. lexical, syntactic and discourse), we considered the following sets of features in order to capture the changes at each level.

2.3.1. Vector Space Model Similarity
The first feature we considered in order to evaluate the simplicity of the simple sentences compared to their original counterpart, was the cosine similarity between the Term Frequency-Inverse Document Frequency (TF-IDF) vectors of each pair. A cosine similarity of 1 indicates that no change has been made in the simplification process. However, before transforming sentences into their corresponding TF-IDF vector, we preprocessed them. First, stop words were removed, then all words were stemmed using the Porter Stemmer (Porter, 1980). As a result, each sentence was represented as a vector of the size of all the stems in all sentences. It is worth noting that in order to compute the inverse document frequency for each stem, we considered each sentence as a document. The cosine similarity between original and simple sentences of the \( i^{th} \) pair is calculated using Formula [1], where \( O_i \) and \( S_i \) represent the vectors of the original sentence and its simple counterpart correspondingly.

\[
\text{Cosine}_{\text{Sim}}(i) = \frac{O_i \cdot S_i}{||O_i|| \times ||S_i||}
\]
2.3.2. Sentence Length

Traditional approaches to readability level assessment have identified text length as an important feature to measure complexity (e.g., Kincaid et al. (1975)). Following this, we investigated the influence of sentence length in terms of the number of open class words only. By ignoring closed class words, we eliminated the effect of words which do not contribute much to the meaning of the sentence. Thus, we considered the difference between the length of pairs of sentences as our second feature for simplicity prediction.

2.3.3. Average Word Length

According to Kincaid et al. (1975), not only can the number of words in the sentence be an indicator of simplicity level, but also its length in terms of the number of characters. To account for this, we also considered the difference between the average number of characters between pairs of sentences. Using this feature along with the number of words of each sentence (see Section 2.3.2), we investigated not only the influence of the length of sentence, but also the length of each word in the sentence.

2.3.4. Frequency in the English Wikipedia Corpus

The frequency of each word in the regular English Wikipedia can be an indicator of the simplicity level of the word. We expected that words in the original sentences would be more frequent in the regular English Wikipedia than words of the simple sentences. Thus, we calculated the difference between the average frequency of all words of the original sentence and their simple counterpart. To do this, we preprocessed both pairs of sentences and the regular English Wikipedia corpus, in order to remove stop words and then stem the remaining words.

2.3.5. Frequency in the Simple English Wikipedia Corpus

The Simple English Wikipedia corpus is an aligned corpus of 60K regular, simple pairs of Wikipedia articles. We used this corpus in order to calculate the average frequency of words of each pair of sentences. We expected the words of simpler sentences to be more frequent in the Simple Wikipedia articles compared to the original sentences. To do this, we performed the same preprocessing as described in Section 2.3.4 and used the average frequency of the sentence’s stems as features.

2.3.6. Frequency of Cue Phrases

The last feature we considered to predict the simplicity aspect was the difference in the usage of cue phrases. Cue phrases are special terms such as however, because, since, etc. which connect text segments and mark their discourse purpose. Several inventories of cue phrases have been proposed (e.g., Knott, 1996; Prasad et al., 2007). For our work, we used the list of 100 cue phrases introduced by Prasad et al. (2007) and calculated the difference between the frequency of cue phrases across pairs of sentences. It is worth noting that cue phrases may be used to explicitly signal discourse relations between text segments or may be used in a non-discourse context. However, here we considered both discourse and non-discourse usage of cue phrases.

2.4. Overall Prediction

The last aspect to be predicted evaluated the combination of all other aspects. According to our analysis of the training data set, this aspect depended mostly on the SIMPLICITY and the MEANING PRESERVATION aspects. Our prediction of this aspect was based only on a simple set of rules using the predictions of these two aspects. The following shows the rules we used to predict the value of this aspect.

- If both simplicity and meaning preservation are classified as GOOD, then overall = GOOD,
- If at least one of simplicity or meaning preservation is classified as BAD, then overall = BAD,
- otherwise, overall = OK.

3. Data and Results

The training set contains 505 pairs of original and simple sentences. The original sentences were taken from the news domain and from Wikipedia and the simple counterparts were automatically simplified using various text simplification systems. Thus, the simple counterparts may contain various types of simplifications such as lexical, syntactic or mixture of both. Table 1 shows the distribution of the data for each of the four aspects. As can be seen, none of the aspects have a normal distribution over the class-labels.

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Value(%)</th>
<th>Good</th>
<th>Ok</th>
<th>Bad</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grammaticality</td>
<td></td>
<td>75.65</td>
<td>14.26</td>
<td>10.09</td>
</tr>
<tr>
<td>Meaning preservation</td>
<td></td>
<td>58.22</td>
<td>26.33</td>
<td>15.45</td>
</tr>
<tr>
<td>Simplicity</td>
<td></td>
<td>52.68</td>
<td>30.29</td>
<td>17.03</td>
</tr>
<tr>
<td>Overall</td>
<td></td>
<td>26.33</td>
<td>46.14</td>
<td>27.53</td>
</tr>
</tbody>
</table>

Table 1: Distribution of data

For our participation, we submitted one run for GRAMMATICALITY and MEANING PRESERVATION and three runs for the SIMPLICITY and OVERALL aspects. The three runs had different classification threshold to assign class labels. Our official results are listed in Table 2. MAE and RMSE stand for Mean Average Error and Root Mean Square Error correspondingly.

4. Bibliographical References


<table>
<thead>
<tr>
<th>System Name</th>
<th>Accuracy</th>
<th>MAE</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grammaticality-Davoodi-RF-perplexity</td>
<td>58.73%</td>
<td>27.38</td>
<td>34.66</td>
</tr>
<tr>
<td>Meaning preservation-Davoodi-RF</td>
<td>49.21%</td>
<td>30.36</td>
<td>36.31</td>
</tr>
<tr>
<td>Simplicity-Davoodi-RF-0.5</td>
<td>34.92%</td>
<td>43.25</td>
<td>45.52</td>
</tr>
<tr>
<td>Simplicity-Davoodi-RF-0.6</td>
<td>35.71%</td>
<td>41.67</td>
<td>44.48</td>
</tr>
<tr>
<td>Simplicity-Davoodi-RF-0.7</td>
<td>35.71%</td>
<td>40.48</td>
<td>44.48</td>
</tr>
<tr>
<td>Overall-Davoodi-0.5</td>
<td>54.13%</td>
<td>41.27</td>
<td>44.21</td>
</tr>
<tr>
<td>Overall-Davoodi-0.6</td>
<td>32.54%</td>
<td>42.06</td>
<td>45.67</td>
</tr>
<tr>
<td>Overall-Davoodi-0.7</td>
<td>33.33%</td>
<td>41.27</td>
<td>45.16</td>
</tr>
</tbody>
</table>

Table 2: Official Results of our system at QATS.


