Phrase-Level Simplification for Non-Native Speakers

Gustavo H. Paetzold[1] and Lucia Specia[2]

[1]Universidade Tecnológica Federal do Paraná, Brazil
[2]The University of Sheffield and Imperial College London, United Kingdom
ghpaetzold@utfpr.edu.br, l.specia@imperial.ac.uk

Abstract. Typical Lexical Simplification systems replace single words with simpler alternatives. We introduce the task of Phrase-Level Simplification, a variant of Lexical Simplification where sequences of words are replaced as a whole, allowing for the substitution of compositional expressions. We tackle this task with a novel pipeline approach by generating candidate replacements with lexicon-retrofitted POS-aware phrase embedding models, selecting them through an unsupervised comparison-based method, then ranking them with rankers trained with features that capture phrase simplicity more effectively than other popularly used feature sets. We train and evaluate this approach using BenchPS, a new dataset we created for the task that focuses on annotations on the needs of non-native English speakers. Our methods and resources result in a state-of-the-art phrase simplifier that correctly simplifies complex phrases 61% of the time.

Keywords: Phrase Simplification, Lexical Simplification, Text Simplification

1 Introduction

Text simplification strategies can take various forms: lexical simplifiers (LS) replace complex words – referred to as target words – with simpler alternatives, syntactic simplifiers (SS) apply sentence-level transformations such as sentence splitting, and data-driven simplifiers (DDS) learn lexico-syntactic simplification operations from parallel data.

While LS is usually addressed using word embeddings and machine learning models that rank candidate replacements, sentence simplification is more often tackled using translation-based DDS approaches such as sequence-to-sequence neural models, which learn transformations from complex-simple parallel corpora. To date, using DDS methods is the only general alternative to simplifying sequences longer than individual words, i.e. phrases [32, 41, 36, 38], except for some domain-specific expressions, such as medical terms, which have been addressed in early LS strategies [9, 10].

Using DDS methods to simplify phrases is, however, "risky", as they can also perform other, spurious transformations in the sentence. In addition, although

in theory DDS simplifiers can learn how to replace or remove complex phrases, complex-to-simple modifications are bounded by the often limited coverage of phrase simplifications present in the training corpus available, which tend to be much smaller than traditional bilingual parallel corpora used areas such as machine translation [15, 37]. Moreover, DDS models offer little flexibility during simplification: it is not easy to control which phrases should be simplified. In user-driven systems [8, 1], where the user chooses which portions of text to simplify, this approach would not be suitable.

LS approaches [24] can be adapted to address phrase simplification. By "phrase" we mean any sequence with more than one word. LS approaches usually simplify a complex word through a pipeline of four steps:

- **Complex Word Identification (CWI):** Consists in finding the words in the text that would challenge the target audience being addressed. There have been some efforts in creating effective supervised complex word identifiers [31, 26], but most LS approaches do not explicitly address this step, and choose instead to perform it implicitly, such as by considering the target complex word as a candidate substitution for itself [14], or by checking whether or not the replacement produced by the lexical simplifier at the end of the pipeline is actually simpler than the target complex word [13, 21].
- **Substitution Generation (SG):** Consists in finding a set of candidate substitutions for a target complex word. Different ways to generate these candidates have been devised, such as through lexicons and thesauri [7, 6], complex-to-simple parallel corpora [2, 14, 21], and word embedding models [13, 27]. Generators often produce candidate substitutions without taking into account the context of the target complex word, which means that they are rarely able to differentiate between the multiple senses of ambiguous complex words.
- **Substitution Selection (SS):** Consists in choosing among the candidate substitutions generated in the previous step best fit the context of the target complex word. Much like CWI, this step is rarely addressed explicitly, the exception being some word sense disambiguation [19] and unsupervised approaches [2, 27]. Many lexical simplifiers usually skip this step and instead train context-aware Substitution Ranking approaches.
- **Substitution Ranking (SR):** Consists in ranking candidate substitutions, either all candidates or those remaining after the SS step, according to their simplicity. Ranking candidates by their frequencies in large corpora is a very popular way of doing so [7, 27], but most recent lexical simplifiers use more elaborate supervised strategies [14, 23, 21].

Pipelined approaches are inherently flexible as they allow to target specific words. They have led to substantial performance improvements over early approaches [14, 13, 21]. However, since they only target single words, these strategies cannot be directly applied to simplify phrases, particularly in the case of phrases with compositional meaning.

In this paper we address the task of **Phrase-Level Simplification (PS)**, which has a clear motivation: many phrases cannot be simplified word for word.

Take the phrase "*real estate*" in the sentence "*John had to hire a company to manage his real estate.*", as example. Using a single-word LS to replace *real* and *estate* individually for, say *factual* and *land*, the phrase's original meaning would be lost. A phrase simplifier should be able to replace "*real estate*" with a simpler alternative, say *property*, or "*land*".

In an effort to address this problem, we propose an approach to PS that builds on the inherently flexible pipelined LS approaches. Similar to previous LS work, we do not address the challenge of identifying complex phrases, under the assumption this will be done by the user. The main **contributions** of this paper are:

- A novel phrase embeddings model for Substitution Generation that incorporates part-of-speech tags and lexicon retrofitting and outperforms well-known resources (§2);
- A cost-effective comparison-based technique for configurable Substitution Selection (§3);
- A study on phrase-level features that can improve the performance of state-of-the-art Substitution Ranking models (§4); and
- BenchPS: an annotated dataset for PS targeting the needs of non-native English speakers, which maximizes simplification coverage and efficiently handles the problem of ranking large candidate sets (§5). BenchPS contains 400 instances composed by a complex phrase in a sentence, and gold replacements ranked by simplicity. Between 1,000 and 24,000 annotations were collected in each annotation step, totaling 36,170.

These contributions as well as our experiments (§6) are described in what follows.

2 Phrase-Level Substitution Generation

The most recent LS approaches employ word embeddings for SG. [13] use a typical GloVe [30] model to create an unsupervised approach that performs comparably to other strategies that use complex-to-simple parallel corpora [14]. [27] introduce another unsupervised approach that uses embeddings trained over a corpus annotated with universal part-of-speech (POS) tags. [21] improve on the latter by incorporating lexicon retrofitting the embeddings. To generate candidates using these models, they simply extract the words with the highest cosine similarity with a given target complex word, then filter any morphological variants amongst them.

These models cannot be used for our purposes since they only contain embeddings for single words. Obtaining phrase-level embeddings has also been addressed in previous work. [35] surveys supervised phrase composition models. These models are usually tested on phrase similarity tasks, and in order to perform well at those, they are trained over large resources, such as PPDB-2.0 [29], which are used as phrase similarity databases. However, preliminary experiments with such models did not yield promising results since, as pointed out by [22],

models that perform well in similarity tasks do not necessarily perform well in LS.

In contrast, simpler unsupervised approaches are a promising fit for PS. For example, [40] and [39] simply join words that form phrases extracted from lexicons in the corpus with a special symbol (e.g. underscore) and use off-the-shelf word embedding models. We go a step further to obtain better informed phrase embeddings by using POS tag annotation and retrofitting, which have been shown to improve the performance of word embeddings in LS [21].

2.1 Retrofitted POS-Aware Phrase Embeddings

To create our phrase embeddings, we take the following approach: annotate a corpus with universal POS tags and phrases, train a typical word embeddings model over the corpus, and then retrofit the embeddings over a lexicon of synonyms, such as WordNet [12]. As we will show in §6, this improves over existing phrase embedding models.

The corpus we annotated has 7 billion words taken from the SubIMDB corpus [25], UMBC webbase[1], News Crawl[2], SUBTLEX [3], Wikipedia and Simple Wikipedia [15]. We POS-tagged the corpus using the Stanford Tagger [34]. To identify phrases, we resorted to SimplePPDB: a corpus containing complex-to-simple English phrase pairs [28]. From SimplePPDB we extracted all bigrams and trigrams as our phrase set, totalling 409,064 phrases. We chose SimplePPDB because it contains a wide variety of phrase types, ranging from non-compositional phrasal constructs to compound nouns and multi-word expressions.

Many of our phrases share n-grams. The phrases "*administrative council body*" and "*council body representative*", for example, have distinct meanings, and should hence compose individual units in our corpus. The challenge is how to annotate both of them in a sentence such as "*The administrative council body representative resigned*" without turning them into a single unit. To do so, we introduce Algorithm 1, which takes a sentence S and a set of phrases P, and returns as output a set R containing various copies of S, each annotated with a different subset of phrases in P that can be found in S. Function copy(S) returns a copy of a sentence, and join(p, S) replaces the spaces separating the tokens of phrase p in sentence S with underscores ("_").

According to Algorithm 1, if the phrase "*administrative council body*" is annotated onto a copy of the sentence "*The administrative council body representative resigned*", it will result in the sentence "*The administrative_council_body representative resigned*", which no longer allows for the annotation of "*council body representative*", given that it does not exactly match with "*council_body representative*". Consequently, another copy of the sentence will be made for this annotation.

After phrase annotation, we tagged each content word that was not part of a phrase with its universal POS tag (V for verbs, N for nouns, J for adjectives and A for adverbs). Phrases were not annotated with sequences of POS

[1] http://ebiquity.umbc.edu/resource/html/id/351
[2] http://www.statmt.org/wmt11/translation-task.html

Algorithm 1: Phrase Annotation

```
input: S, P;
output: R;
R ← {S};
S_c ← copy(S);
while ‖P‖ > 0 do
    foreach p ∈ P ∩ n-grams(S_c) do
        join(p, S_c);
        P.remove(p);
    end
    R.add(S_c);
    S_c ← copy(S);
end
return R;
```

tags to reduce sparsity. The previously mentioned example would be POS annotated as "*The administrative_council_body representative-N resigned-V*". We then trained a continuous bag-of-words model (CBOW) with 1,300 dimensions using WORD2VEC [17]. We chose these settings based on findings in previous work for LS [22]. The final step was the retrofitting the model. Through retrofitting, it is possible to approximate vectors of words that share linguistic relationships such as synonymy, which are useful in simplification. We retrofitted the model over the synonym relations between all words and phrases in WordNet using the algorithm of [11]. We refer to this retrofitted POS-aware model as **Embeddings-PR**.

For comparison purposes, we built three other models:

- **Embeddings-B**: A base model, trained without POS-annotation nor retrofitting.
- **Embeddings-R**: A model enhanced with retrofitting only.
- **Embeddings-P**: A model enhanced with POS annotations only.

We explain how we apply these models for SG in the following section.

2.2 Generating Candidates with Phrase Embeddings

To find candidate substitutions for a given target complex phrase with our embeddings, we use a three-step process: vocabulary pruning, vocabulary ranking, and ranking pruning.

Vocabulary Pruning The first step of SG strategy is to discard any words and phrases from the embedding model's vocabulary that are too unlikely to yield useful candidate substitutions. We employ a simple heuristic pruning approach for that. Given a target complex phrase in a sentence, we discard from the model's vocabulary any words/phrases that:

- have numbers or spurious characters, e.g. @#$%,
- are a substring of the target complex phrase, or vice-versa,
- lead to a grammatical error with determiners, e.g. *a* vs *an* preceding candidate,
- lead to a comparative/superlative accordance error, e.g. *more* vs *most* preceding candidate.

Vocabulary Ranking Once the model's vocabulary is pruned, we then rank the remaining words and phrases according to their cosine similarity with the target complex phrase. This is a very intuitive way of determining which candidates in a CBOW model are most suitable for the simplification of a target phrase, since this type of model tends to group synonyms together.

Ranking Pruning The final step is to choose the value of α, which determines how many of the best ranking candidates will be passed onto SS. This step is crucial in ensuring the quality of the phrase simplifications produced by our PS approach. If too few candidates are pruned, then the tasks of SS and SR will be more complex to filter and rank the remaining unsuitable candidates, which can lead to frequent ungrammatical and/or meaningless replacements. If too many candidates are pruned, this could compromise the simplicity of the output produced, since the ranker would not have enough options to choose from.

[13], [27] and [21] achieve good results by using $\alpha = 10$, but provide no explanation with respect to how they arrived at this number. If there is no training/tuning data available, keeping only the 10 best candidates is a good choice, since it at least allows for more meaningful comparisons with previous work. If there is training/tuning data available, however, one can simply perform an exhaustive search with a large range of pruning settings to determine the best value. In the following section, we discuss how we optimise pruning jointly with the parameters of the SS approach.

3 Comparison-Based Substitution Selection

With few exceptions [19, 2, 27], most lexical simplifiers do not perform explicit SS.

[19] use typical word sense disambiguation methods trained over thesauri. These methods are inherently unsuitable for PS, since these thesauri rarely contain phrases. [2] introduces an unsupervised approach that does not rely on thesauri. They instead use a word co-occurrence model to calculate the similarity between the target word and the candidates, then discard any candidates that are either too similar or not similar enough to the target word. Although this approach could be adapted to our purposes, it has been shown not to suitable for embedding-based candidate generators [24]. To address this problem, [27] present a method called Unsupervised Boundary Ranking (UBR). They calculate features of the target word and generated candidates, create a binary classification dataset by assigning label 1 to the target word and 0 to all candidates, train

a linear model over the data, rank the generated candidates themselves (which were used for training) based on the boundary between 1's and 0's, and select a proportion of the highest ranking ones. [27] show that this approach yields noticeable improvements for embedding-based models.

The SS approach of [27] has two main strong points: it does not require manually annotated data for training, and it is configurable, allowing one to decide how conservative the selector should be depending on the type of generator used. We devised an unsupervised comparison-based SS approach to further improve this selector.

The intuition behind our approach is as follows: the more evidence there is that a candidate fits the context of a target complex phrase even better than the target phrase itself, the more likely it is to be a valid candidate substitution. Given a set of generated candidate substitutions for a target complex phrase in a sentence, our approach:

1. Calculates the following 40 features for the target phrase and each candidate:
   - The n-gram frequency of all n-grams composed by the target/candidate and the i preceding words and j succeeding words in the sentence for all possible combination of values of i and j between 0 and 2 (9 in total). We calculate these features using frequencies from 4 distinct corpora: SubIMDB [25], SUBTLEX [4], Wikipedia [15], and OpenSubtitles 2016 [16] ($9 * 4 = 36$ total features of this kind).
   - The language model probability of the sentence in its original form (for the target), and with the target replaced by each candidate. We train 3-gram language models for all four of the aforementioned corpora using SRILM (totalling 4 features of this kind).

   The features were chosen based on the findings of [20], where they proved effective in capturing grammaticality and meaning preservation in simplification.
2. Compares the feature values of the candidates and the target phrases, and, for each candidate phrase, calculates the proportion of features that yield a higher value for the candidate than for the target phrase (which we call "winning features").
3. Discards any candidates for which the proportion of winning features is smaller than a selected β value.

If there is no training/tuning data available, one can choose β empirically depending on the type of simplifier and/or the type of candidate generator that is used. If it is known that the generator tends to produce a large number of spurious candidates, a large β can be used, for example. If there is training/tuning data available, exhaustive search over all possible β values is not costly. The experiments in §6.1 show how β can be optimised in conjunction with the number of pruned candidates during SG.

Once the candidate substitutions are filtered by our comparison-based selector, they are passed onto our SR approach, the last step of the phrase simplification process.

4 Ranking with Phrase-Level Features

The SR approaches of [21] and [13] have been shown the most effective supervised and unsupervised LS ranking strategies, respectively.

The rank averaging approach by [13] ranks candidates according to various features, then averages their ranks in order to produce a final ranking. [21], on the other hand, employ a multi-layer perceptron that quantifies the simplicity difference between two candidate substitutions. To train their model, they calculate features for the gold candidates ranked by simplicity in a manually produced training set, pass them in pairs to the multi-layer percetron, and use as output the rank difference between the two. Given an unseen set of candidate substitutions for a complex word in a sentence, they calculate the simplicity difference between every pair of candidates, then average the differences for each candidate to create a full candidate ranking list.

However, these approaches cannot be directly used in our work, since the features used, which are in their majority n-gram frequencies, are not enough to capture phrase simplicity. We propose a set of 16 additional features:

- the phrase's number of characters,
- the phrase's number of tokens,
- the phrase's raw frequency in the corpus of 7 billion words described in §2.1, as well as SubIMDB [25], Wikipedia and Simple Wikipedia [15],
- the sentence probability after replacing the target phrase with a candidate phrase according to 3-gram language models trained on the same corpora,
- the maximum, minimum and average raw frequency of the phrase's tokens in the 7 billion word corpus,
- the maximum, minimum and average cosine similarity between the target phrase and each of the candidate's tokens according to our retrofitted POS-aware phrase embeddings model (Embeddings-PR).

In order to evaluate this pipeline for phrase simplification, we created a new dataset, as we describe in what follows.

5 BenchPS: A New Dataset

Since PS has not yet been formally addressed, there are no dedicated datasets for the training and/or evaluation of PS systems. We created a dataset that follows the same format of typical LS datasets [14, 5, 27, 24], where each instance is composed of:

- a sentence,
- a target complex word in that sentence, and
- a set of gold candidates ranked by simplicity.

An important step in creating a dataset of this kind is deciding on which target audience to focus. We chose non-native English speakers, which has been

a popular target audience in previous work, including the SemEval 2016 shared task on LS. Four challenges were involved in creating such a resource: finding complex phrases that can be potentially replaced, finding sentences containing such complex phrases, collecting simpler alternatives for the complex phrases, and ranking them according to their simplicity.

5.1 Collecting Complex Replaceable Phrases

This step involved finding a set of phrases that can pose a challenge to non-native English speakers, but also are replaced by alternatives. If the phrases selected do not have any semantically equivalent alternatives, it would be impossible to gather candidate simplifications.

In order to find replaceable complex phrases, we first collected an initial set of complex phrases from SimplePPDB. From the complex side of SimplePPDB, we gathered all two and three-word phrases that did not contain any number. After the initial filtering, we ranked the remaining phrases according to how many distinct simpler phrases they were aligned to in the SimplePPDB database with a probability $>= 0.7$, and then selected the 1,000 highest ranking phrases as an initial set of target complex phrases. We chose 0.7 based on a manual inspection of phrase pairs in SimplePPDB. This heuristic also serves as an initial clue as to how replaceable the phrases are.

Manual inspection of a sample selected phrases and their given simplifications in the database showed that the alignments alone were not enough to determine whether the phrases were indeed replaceable, given that a lot of the simplification pairs are very similar to each other. The word *establish*, for example, is aligned to "*as to determine*", "*in order to determine*", "*determine whether*" and "*determine that*", among many others. To remedy this, we conducted a user study on the replaceability of phrases. We presented annotators with a set of complex phrases, each accompanied by ten candidate replacements: the five alignments in SimplePPDB with the highest probability, and the five words/phrases with the highest cosine similarity with the complex phrase in our Embeddings-B phrase embeddings model (trained without POS annotation or retrofitting). We did so to minimise any biases in our subsequent experiments with the retrofitted POS-aware embedding models, which is one of our main contributions. Annotators where then tasked to judge each complex phrase as being either:

- **Highly replaceable:** You can think of at least three replacements for this phrase.
- **Mildly replaceable:** You can think of up to two replacements for this phrase.
- **Non-replaceable:** You can think of no replacements for this phrase.

The candidates presented serve as suggestions on how the complex phrase could be replaced, but annotators could think of different replacements. It is important to highlight the fact that this annotation methodology was used for the purpose of fulfilling our ultimate goal of finding phrases that could be replaced by others, rather than to quantify phrase replaceability in general.

We hired four volunteers who are computational linguists to annotate the phrases. Each annotator judged 250 phrases. Figure 1 illustrates the interface used for annotation through an example. This user study, as well as all others described henceforth, were conducted using Google Forms[3].

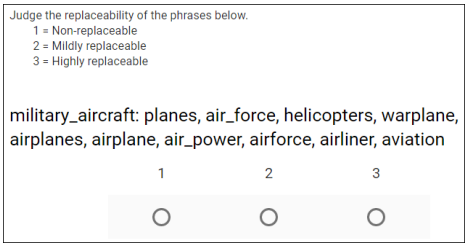

Fig. 1. Annotation interface used for our user study on phrase replaceability

Table 1 shows counts and examples of each annotations on each category. While the phrases judged either highly or mildly replaceble are quite apparently so, not all of the candidates judged non-replaceable by the annotators are, indeed, non-replaceable. For example, the phrase "*shall be able*" could be replaced with "*will be able*" in the sentence "*One of our attendants shall be able to help you with any outstanding matters*". However, this is not a problem since we are strictly looking for replaceable target phrases to compose our dataset, so high precision is more important than high recall.

Type	Frequency	Examples
Highly replaceable	285	hazardous substance, walk away, formidable challenge, very beautiful, proceed along, critical phase
Mildly replaceable	297	principal aim, more apparent, alternative form, main tool, fully understand, agricultural product
Non-replaceable	418	establish clear, shall be able, more widespread, question remain, provides opportunity, exist means

Table 1. Replaceability degrees and respective example phrases

To compose a final set of target complex phrases, we selected all 285 very replaceable phrases along with 115 randomly selected mildly replaceable phrases, totalling 400. 351 (87.7%) of them are two-word phrases, while 49 (12.3%) are three-word phrases. The next step in our dataset creation process was to find sentences which contain these complex phrases. To do so, we used the 7 billion word corpus described in §2.1, whose sentences have 32 words on average, to search for one distinct sentence for each of the 400 complex phrases selected.

[3] https://www.google.co.uk/forms/about

5.2 Collecting Phrase Simplifications

With sentences and target complex phrases at hand, we began the process of collecting gold simplifications for them. For that, we employed a two-step annotation process: open and suggested simplification judgements.

Open Simplification Judgements In this step we presented annotators with the complex phrase in its respective sentence and asked them to suggest as many simpler equivalent words or phrases as they could think of. 307 fluent English speakers annotators, all of which are volunteer students or academic staff of various universities, participated in the process. Each annotator received 10 complex phrases to simplify, and each complex phrase was annotated by at least 5 annotators. Figure 2 illustrates the annotation interface used for this step.

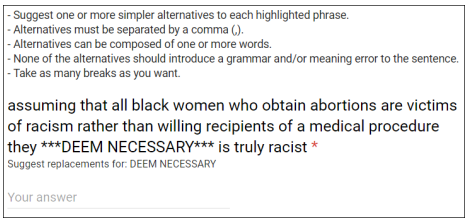

Fig. 2. Annotation interface used for the open simplification step

In total, 3,070 annotations were produced ($307 * 10 = 3,070$) and 3,367 gold simplifications suggested. The average number of gold simplifications suggested per complex phrase is 8.4 (± 3.1), ranging from 1 to 23. Table 2 shows some examples of annotations produced.

While inspecting the simplifications suggested by annotators along with those automatically produced for our replaceability user study, we noticed that there were many simpler, grammatical and meaningful alternatives among the candidates that were not suggested by any annotators. To address this problem and hence complement the simplifications in our dataset, we conducted a second annotation step.

Suggested Simplification Judgements In this step we presented annotators with the complex phrase in its context along with five automatically produced candidate replacements, and asked them to select those, if any, that could simplify the complex phrase without compromising the meaning or grammaticality of the sentence. For each complex phrase, we randomly selected up to five out of the ten candidate replacements used in our replaceability user study described in §5.1 which had not yet been suggested by any annotators in the previous annotation step.

Sentence with target phrase	Suggestions
This is not an obscure philosophical argument but a practical issue of **{considerable importance}**.	significant importance, serious importance, considerable significance, great relevance
If elected, he would also be eligible for immunity from **{criminal prosecution}**.	criminal judgment, facing trial, legal action, being punished
He said a special **{educational curriculum}** is being prepared for the youngsters.	learning timetable, educational timetable, educational programme
The IRS shall file a **{fairly significant}** claim against R. Allen Stanford he said.	big, crucial, quite significant, important, significant
Mr Hutchinson suggested that issues, such as dealing with the past, could be dealt with **{more speedily}** in devolved setting	more effectively, quicker, more rapidly, faster

Table 2. Example annotations from the open simplification step

400 fluent English speakers with the same profile as in the previous step (§5.2) participated in the annotation process. Each annotator received 12 complex phrases with five candidates each to judge ($12 * 5 = 60$ candidates in total), and each candidate was judged by at least 8 annotators. Figure 3 shows the annotation interface used for this step.

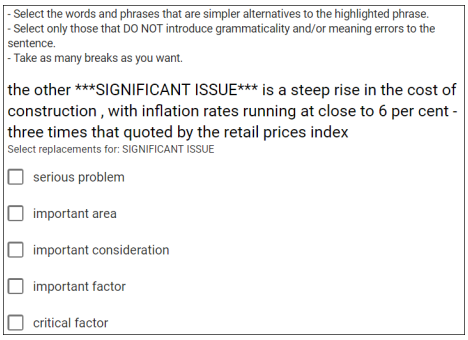

Fig. 3. Annotation interface used for the given simplification step

In total, 24,000 judgements were produced ($400 * 60 = 24,000$). To complement the pool of simplifications of a given complex phrase in our dataset, we selected only the candidates which had been judged positively by at least 4 annotators. We found through manual inspection that this number of judgements best separated good from spurious candidates. A total of 697 new simplifications were produced through this step. After adding these new simplifications to our dataset, the average number of gold simplifications per complex phrase moved from 8.4 ($\pm$3.1) to 9.7 ($\pm$3.4), ranging from 1 to 24. Amongst our gold candidates, 1,161 (29.8%) are single words, 2,213 (56.8%) phrases with two words, 392

(10%) phrases with three words, and 94 (2.4%), 25 (0.6%) and 14 (0.4%) are phrases with four, five and six words, respectively. Table 3 shows some examples of annotations produced in this step.

Sentence with target phrase	Selected suggestions
This is not an obscure philosophical argument but a practical issue of **{considerable importance}**.	great interest, great significance, particular interest, significance
If elected, he would also be eligible for immunity from **{criminal prosecution}**.	disciplinary action, prosecutions
He said a special **{educational curriculum}** is being prepared for the youngsters.	course of study, education program, syllabus
The IRS shall file a **{fairly significant}** claim against R. Allen Stanford he said.	sizable, substantial, very big
Mr Hutchinson suggested that issues, such as dealing with the past, could be dealt with **{more speedily}** in devolved setting	right away, very quickly, without delay

Table 3. Example annotations from the given simplification step

5.3 Simplification Ranking

The last step in our dataset creation process was to rank our gold simplifications. This is arguably the most challenging step in this process. It is very difficult for a human to confidently rank a large number of gold simplifications such as what we have in our dataset. While ranking 3 or 4 candidates is not too challenging, providing a full rank for 24 candidates (our maximum number of gold candidates for a given complex phrase) is most likely not feasible, and hence would yield unreliable data.

To address this problem, instead of asking annotators to produce full rankings, we could generate all possible candidate pairs from the set, present each pair to the annotators, and then ask them to select which of the candidates is simpler. With all pairs annotated, we could then use inference algorithms to produce full rankings. This annotation approach would reduce the amount of information presented to the annotator, but it introduces a new problem: The number of instances that have to be annotated increases in combinatorial fashion with respect to the number of candidates available. For a phrase with 24 candidate substitutes, for example, 276 pairs would have to be annotated. Because of this limitation, we conceived a new ranking-by-comparison annotation approach. We reduce the number of comparisons in two ways: by presenting the annotator with more than two candidates at a time, and by sampling the space of comparisons based on the minimum number of comparisons in which each candidate must be featured.

Suppose we have a set of five candidates $\langle a, b, c, d, e \rangle$ and are willing to compare three candidates at a time such that all of them appear in at least three

comparisons. To do so, we could present annotators with six comparison tasks featuring the following sets of candidates:

$$\langle a, b, c\rangle\ \langle c, d, e\rangle\ \langle a, c, d\rangle$$
$$\langle a, b, e\rangle\ \langle b, d, e\rangle\ \langle a, b, d\rangle$$

By doing so, we would reduce the number of annotations required from 10, which is the number of distinct pairs that can be extracted from $\langle a, b, c, d, e\rangle$, to only six, which is 40% less. Notice also that this approach is flexible, given that one can adjust the number of candidates presented at a time and the minimum number of comparisons per candidate based on the number of candidates to be ranked, degree of informativeness desired and annotation budget.

For this annotation step, we presented annotators with three candidates at a time and required each candidate to appear in at least five comparisons. Each instance presented to annotators was composed of a sentence with a gap in place of the complex phrase and three candidate simplifications. Annotators were asked to select which candidate, when placed in the gap, would make the sentence easiest to understand, and which candidate would make the sentence most difficult to understand. By asking for both the easiest and most difficult of the candidates, we get a full local ranking between the three candidates presented, which allows us to more accurately quantify the simplicity differences between them. In order to produce a set of comparisons to present to the annotators, we randomly selected comparisons from the set of possible 3-candidate combinations that could be extracted from the candidate set until every candidate was featured in at least five comparisons. The annotation interface we used is illustrated in Figure 4.

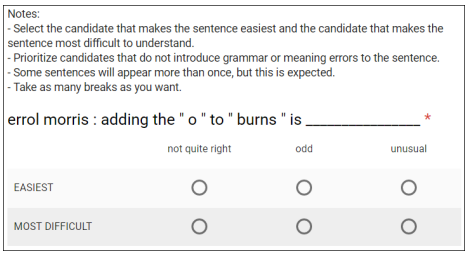

Fig. 4. Annotation interface used for the candidate ranking step

300 non-native English speakers took part in this annotation step (same profile as in §5.2). Each annotator received 27 comparisons with three candidates each to judge. Due to the limited annotation budget available, each comparison was judged by only one annotator. In total, 8,100 judgements were produced ($300 * 27 = 8,100$). To generate a final ranking of candidates, we first assigned

score 3 to the candidates judged easiest, 1 to the candidates judged most complex, and 2 to the remaining candidates. Then we averaged the scores of each candidate across all comparisons in which they appeared, and used the average to rank them in decreasing order (ties were allowed). Table 4 shows some examples of annotations produced in this step.

Sentence with target phrase	Ranked candidates
This is not an obscure philosophical argument but a practical issue of {_____}.	1 - significant importance 2 - great relevance 3 - concern
If elected, he would also be eligible for immunity from {_____}.	1 - being punished 2 - criminal judgment 3 - criminal prosecution
He said a special {_____} is being prepared for the youngsters.	1 - teaching method 2 - educational curriculum 3 - learning timetable
The IRS shall file a {_____} claim against R. Allen Stanford.	1 - significant 2 - very big 3 - quite significant
Mr Hutchinson suggested that issues, such as dealing with the past, could be dealt with {_____} in devolved setting.	1 - faster 2 - more effectively 3 - more speedily

Table 4. Example annotations from the candidate ranking step. Candidates are ranked from simplest (1) to most complex (3).

The full process resulted in a new dataset – **BenchPS**. It contains 400 instances composed of a sentence, a target complex phrase, and an average of 9.7 gold replacements ranked by simplicity. Table 5 shows examples of instances in BenchPS.

6 Experiments

In this section we present experiments conducted with the approach and resources presented earlier.

6.1 Candidate Phrase Generation

In our first experiment we conducted an exhaustive search for the best values of the hyperparameters of our SG and SS approaches jointly:

- α**:** The number of candidates to generate with our retrofitted POS-aware embeddings (Embeddings-PR) for SG.
- β**:** The proportion of winning comparisons required by our comparison-based SS approach to keep a candidate substitution.

Sentence with target phrase	Gold replacements
A **{medical professional}** in the government recommended testing on entry, but was told it was too expensive to test all	1:physician, 2:doctor, 2:health care provider, 3:medical advisor, 4:medical doctor
The solution for these symptoms is **{very straightforward}** – just go back on the antidepressant and they go away	1:simple, 2:relatively simple, 3:easy, 4:very easy, 4:fairly simple, 5:not at all difficult, 6:pretty simple, 7:fortright
In response, Coelho has digitalized all 16 of his titles in Farsi and posted them today on the internet for anyone to download, **{free of charge}**.	1:free-of-charge, 2:for free, 3:freely, 3:free, 4:at no cost, 5:without payment, 5:for no pay, 6:gratis
And the painful irony is that the source of their money woes is exactly what makes them **{most happy}**: cars	1:happiest, 2:pleased, 3:glad
My father died when I was 13 and my mother later remarried, to a man much more like her in temperament and who was the **{most wonderful}** stepfather imaginable	1:best, 2:nicest, 3:greatest, 4:finest

Table 5. Example instances from the completed BenchPS. Candidate substitutions are ranked from simplest to most complex.

For α, we checked all integer values between 1 and 50, and for β, all values between 0.0 and 1.0 in intervals of 0.01 (100 values), totalling 5,000 value pairs. We also performed a joint exhaustive search between α and a third parameter γ, which is the proportion of candidates discarded by an unsupervised boundary ranking selector (UBR). We do this so we can compare our performance against the current state-of-the-art approach. For γ, we also search for all values between 0.0 and 1.0 in intervals of 0.01. Both SS approaches use the same set of 40 features described in §3.

We split BenchPS into two equally sized portions (200 instances): One for training, and one for testing. The unsupervised boundary ranking selector is trained using the same procedure described by [27] over the training set. For each possible combination of $\alpha \times \beta$ and $\alpha \times \gamma$, we calculate the following three metrics over the training set:

- **Precision:** Proportion of selected candidates that are in the gold simplifications.
- **Recall:** Proportion of gold simplifications that are in the set of selected candidates.
- **F-measure:** Harmonic mean between Precision and Recall.

The values obtained are illustrated in Figure 6 in form of heatmaps. Brighter colors indicate better metric scores. While the behaviours of our comparison-based approach and UBR are rather similar for Recall, the same cannot be said for Precision and F-measure. The top left corner of the Precision map of our

approach is much brighter-colored than that of UBR, which suggests that our approach is more suitable for the creation of more conservative phrase simplifiers. And as it can be noticed, the F-measure heatmap of our approach is overall much brighter colored than that of UBR, specially in the α range between 5 and 20.

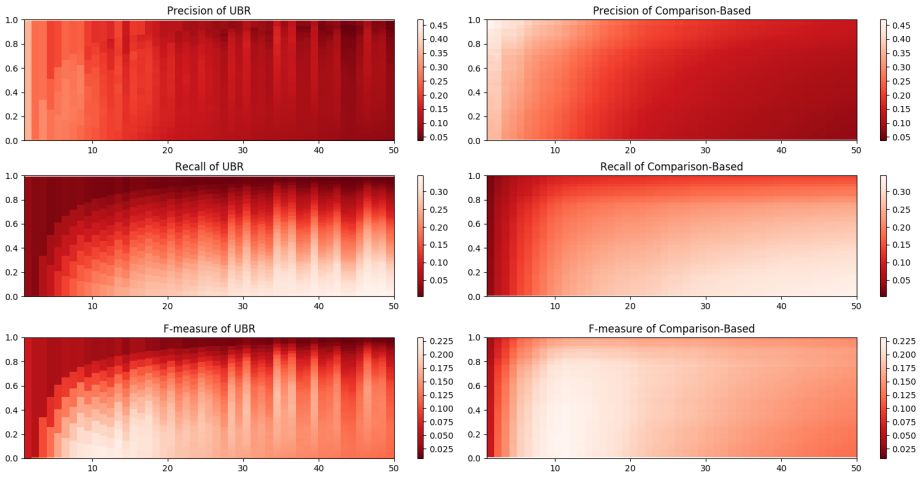

Fig. 5. Results of exhaustive hyperparameter search for SS approaches with respect to our SG approach. The horizontal axes represent α values. Vertical axes on the left and right column represent γ and β values, respectively.

Interestingly, the α value that yielded the highest F-measure scores for both approaches was 11. The β and γ values, on the other hand, were 0.32 and 0.0, respectively. In other words, our SG approach benefits from being paired with our comparison-based SS strategy, but performs better on its own than when paired with an unsupervised boundary ranking selector.

With these hyperparameter values selected, we compared our approach to others. For SG, we include nine strategies in the comparison:

- **WordNet:** Extracts synonyms, hypernyms, and hyponyms of phrases as candidates from WordNet.
- **SimplePPDB:** Extracts aligned paraphrases as candidates from SimplePPDB. We include four variants of SimplePPDB generators: one that extracts all aligned paraphrases as candidates (SimplePPDB), and three others that consider only paraphrases aligned with a probability > 0.5 (SimplePPDB-0.5), > 0.7 (SimplePPDB-0.7), and > 0.9 (SimplePPDB-0.9).
- **Embeddings:** Extracts candidates using our SG approach. We create one system for each of the embedding models described in §2.1: the base model (Embeddings-B), the base model retrofitted over WordNet (Embeddings-R), the POS-aware model (Embeddings-P) and the POS-aware retrofitted model (Embeddings-PR).

We pair each of these nine generators with three SS methods:

- no selection,
- unsupervised boundary ranking (UBR), and
- our comparison-based approach (CB).

To find the best values of β and γ, we perform exhaustive search over the training set with respect to the F-measure. To evaluate all 27 combinations of SG and SS approaches we use the same previously introduced Precision (P), Recall (R), and F-measure (F) metrics over the test set.

The results in Table 6 reveal that, regardless of the selector used, our POS-tagged retrofitted phrase-annotated embeddings outperformed all other generators. Pairing them with our comparison-based selector yielded the highest scores overall. The WordNet generator could not produce a single valid candidate substitution due to its low coverage for phrases. Finally, due to the large number of spurious candidates produced, none of the selectors managed to consistently improve the Precision scores of SimplePPDB generators without detrimental compromises in Recall.

	No Selection			UBR			CB		
	P	R	F	P	R	F	P	R	F
WordNet	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
SimplePPDB	0.055	0.173	0.084	0.087	0.135	0.106	0.061	0.163	0.089
SimplePPDB-0.5	0.083	0.149	0.107	0.095	0.123	0.107	0.089	0.140	0.109
SimplePPDB-0.7	0.120	0.098	0.108	0.119	0.069	0.088	0.120	0.098	0.108
SimplePPDB-0.9	0.175	0.018	0.032	0.177	0.013	0.024	0.178	0.018	0.032
Embeddings-B	0.140	0.155	0.147	0.163	0.132	0.146	0.163	0.143	0.153
Embeddings-R	0.154	0.171	0.162	0.181	0.146	0.162	0.177	0.157	0.166
Embeddings-P	0.190	0.210	0.200	0.177	0.143	0.158	0.205	0.203	0.204
Embeddings-PR	**0.216**	**0.240**	**0.227**	**0.197**	**0.158**	**0.175**	**0.232**	**0.231**	**0.232**

Table 6. Candidate generation and selection results: Precision (P), Recall (R) and F-measure (F)

6.2 Phrase Simplicity Ranking

In our second experiment, we assessed the performance of 20 rankers in the task of capturing phrase simplicity:

- One for each of the 16 features described in §4.
- **Glavas:** The rank averaging approach of [13], as described in §4, while using only their original features.
- **Glavas+P:** The same ranker, but with the original feature set complemented with our 16 phrase-level features described in §4.

- **Paetzold:** The supervised neural ranker of [21], as described in §4, trained over the training portion of BenchPS using only with the features introduced by [21].
- **Paetzold+P:** The same ranker, but with the original feature set complemented with our 16 phrase-level features described in §4.

We evaluated the rankers on the test set using three evaluation metrics: Spearman (s) and Pearson (r) correlation between gold and produced rankings, as well as TRank [33], which measures the proportion of times in which the simplest gold candidate was ranked first.

The results in Table 7 confirm that the proposed features help rankers capture phrase simplicity: adding them to the Glavas and Paetzold rankers resulted in substantial performance gains. The supervised neural ranker complemented with our features (Paetzold+P) obtained the highest correlation scores, but was outperformed in TRank by the neural ranker without our features (Paetzold). We believe this is due to the fact that the Paetzold ranker is better at placing simpler candidates at the top rank, but worse than the Paetzold+P ranker in ordering all candidates properly. Additionally, the length and token count of a phrase do not seem to play a significant role in its simplicity. And contrary to what was observed in previous benchmarks for single-word LS [24], language model probabilities tend to capture simplicity at phrase-level more effectively than raw frequencies. This suggests that the context has a stronger influence in determining the simplicity of phrases than the simplicity of single words.

6.3 Full Pipeline Evaluation

In this experiment, we evaluated various combinations of candidate generators and rankers in the creation of full phrase simplifiers. We include two top performing generators from the ones tested in the previous experiments: SimplePPDB-0.5 and Embeddings-PR. We pair both generators with the three selectors described in the previous experiment, and hence consider six different ways of producing candidates.

We paired each of the aforementioned generator/selector pairs with the 20 rankers described in §6.2. The metric used here is **Accuracy**: the proportion of instances in which the simplifier replaced the complex phrase with one of the gold simplifications from the dataset. In this context, Accuracy encompasses all aspects of simplification quality by assuming that the gold candidates in our dataset ensure grammaticality, meaning preservation and simplicity.

The results are illustrated in Table 8. While the addition of our phrase-level features did not have a significant impact on the performance of the rankers for the SimplePPDB-0.5 generator, it led to noticeably higher Accuracy scores for our Embeddings-PR model. The differences observed are statistically significant (10-fold bootstrapping significance tests with $p<0.01$).

Interestingly, although Table 8 confirms the finding of §6.1 that our comparison-based approach is indeed more reliable than unsupervised boundary ranking,

	TRank	r	s
Length	0.070	0.003	0.071
Token Count	0.090	0.066	0.123
SimpleWiki-PF	0.230	0.288	0.305
SubIMDB-PF	0.235	0.257	0.270
Wikipedia-PF	0.220	0.254	0.280
7billion-PF	0.225	0.252	0.275
SimpleWiki-LM	0.250	0.319	0.342
SubIMDB-LM	0.250	0.305	0.326
Wikipedia-LM	0.245	0.325	0.366
7billion-LM	0.275	0.338	0.377
Min. Frequency	0.195	0.293	0.319
Max. Frequency	0.140	0.179	0.213
Avg. Frequency	0.185	0.255	0.279
Min. Similarity	0.155	0.162	0.198
Max. Similarity	0.130	0.175	0.219
Avg. Similarity	0.155	0.174	0.221
Glavas	0.255	0.281	0.291
Glavas+P	0.275	0.348	0.372
Paetzold	**0.365**	0.375	0.345
Paetzold+P	0.355	**0.420**	**0.392**

Table 7. Simplicity correlation results

neither of these SS approaches proved more effective than not performing selection at all. Inspecting the results we found that, although both selectors do manage to improve the precision of the SG approaches evaluated, they discard too many useful candidates, sometimes leaving none behind.

Overall, the most accurate simplifier combines our phrase and POS-aware retrofitted embeddings (Embeddings-PR), no selection (No Sel.), and the supervised neural ranker with our phrase-level features (Paetzold+P).

To further illustrate the effectiveness of our phrase-level features for SR, we compared the performance of the Glavas/Paetzold and Glavas+P/Paetzold+P rankers in thousands of different settings. To do so, we paired each of these four rankers with all the 5,000 settings described in §6.1 of Embeddings-PR and our comparison-based selector. For the α value (number of candidates generated by Embeddings-PR), we considered all integer values between 1 and 50, and for β (number of winning comparisons necessary for a candidate to be kept), all values between 0.0 and 1.0 in intervals of 0.01. In total, the performance of 20,000 phrase simplifiers were tested (4 rankers * 5,000 generator/selector pairs).

Figure 6 shows heatmaps for the Accuracy scores obtained in all these settings. The brighter (whiter) the color of a given spot, the higher the Accuracy score obtained for that setting. The heatmaps reveal that, although the Accuracy of the rankers is very similar for α values no larger than 12, the rankers trained with our phrase-level features (Glavas+P/Paetzold+P) achieve much higher Accuracy scores when more candidates are generated. This suggests that

	SimplePPDB-0.5			Embeddings-PR		
	No Sel.	UBR	CB	No Sel.	UBR	CB
Length	0.000	0.115	0.020	0.100	0.100	0.165
Token Count	0.020	0.135	0.035	0.115	0.115	0.155
SimpleWiki-PF	0.130	0.290	0.130	0.350	0.290	0.355
SubIMDB-PF	0.105	0.280	0.105	0.325	0.270	0.325
Wikipedia-PF	0.135	0.285	0.140	0.325	0.275	0.335
7billion-PF	0.115	0.295	0.120	0.320	0.260	0.325
SimpleWiki-LM	0.130	0.315	0.130	0.350	0.310	0.350
SubIMDB-LM	0.145	0.320	0.150	0.365	0.290	0.365
Wikipedia-LM	0.120	0.300	0.125	0.375	0.325	0.375
7billion-LM	**0.160**	0.325	**0.165**	0.385	0.325	0.375
Min. Frequency	0.130	0.205	0.130	0.270	0.215	0.270
Max. Frequency	0.130	0.125	0.130	0.040	0.065	0.080
Avg. Frequency	0.130	0.155	0.130	0.110	0.100	0.130
Min. Similarity	0.05	0.290	0.060	0.405	0.320	**0.430**
Max. Similarity	0.05	0.195	0.060	0.270	0.210	0.320
Avg. Similarity	0.05	0.310	0.060	0.390	0.310	0.410
Glavas	0.120	0.300	0.125	0.380	0.295	0.380
Glavas+P	0.110	**0.355**	0.115	0.390	0.325	0.395
Paetzold	0.135	0.305	0.140	0.365	0.305	0.345
Paetzold+P	0.135	0.350	0.140	**0.460**	**0.370**	0.425

Table 8. Accuracy of full pipeline phrase simplification approaches

adding these features to the ranker is crucial when creating phrase simplifiers that aim to maximise the recall of candidates generated, and hence increase the simplicity of the output.

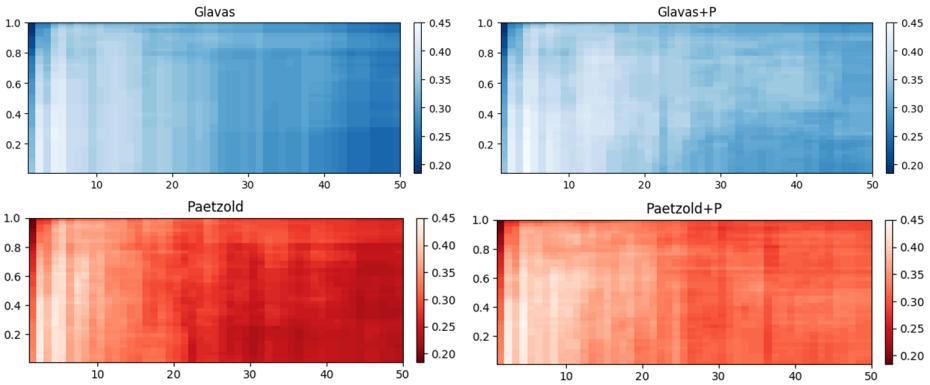

Fig. 6. Accuracy scores for of exhaustive performance comparison. The horizontal axes represent α values. Vertical axes on the left and right column represent γ values.

6.4 Error Analysis

To complement our quantitative analyses, we conducted a manual inspection of the mistakes made by two of the simplifiers featured in the experiments of §6.3:

- **Embeddings-PS:** the best performing simplifier that uses our Embeddings-PR generator, which employs no SS approach and the Paetzold+P ranker.
- **SimplePPDB-PS:** the best performing simplifier that uses the SimplePPDB-0.5 generator, which employs unsupervised boundary ranking and the Glavas+P ranker.

The authors of this paper, who are fluent non-native English speakers, conducted the analyses. They were presented with the sentence, target complex phrase and simplification produced for each instance of the BenchPS dataset in which these simplifiers failed to produce a simplification that was amongst the gold simplifications available. The annotator was tasked with deciding which of the simplifications produced were grammatical, meaning preserving, and simpler than the target phrase. This type of analysis allows us to truly quantify the proficiency of these simplifiers, since one can never be sure that every conceivable simpler alternative to the target complex phrases of a dataset were captured during its creation. An individual analysis was conducted for each of the aforementioned simplifiers.

We found that, out of the 111 simplification "errors" made by the Embeddings-PS simplifier, 33 (29.72% of 111) were grammatical, meaning preserving, and simpler than the target. And for the SimplePPDB-PS simplifier, the proportion was 47 out of 129 (36.43% of 129). Adding these numbers to the scores in §6.3 results in that Embeddings-PS has an actual Accuracy of 61% (89+33 out of 200), and SimplePPDB-PS 59% (71+47 out of 200).

The feedback provided by the annotator indicated that Embeddings-PS is more effective than SimplePPDB-PS at finding simplifications for more challenging complex phrases in the dataset. Some examples are "lodge a complaint", which was replaced by "notify" in "... the club says it will *lodge a complaint* against Dyfed-Powys police...", and "purchase and sale", which was replaced by "transaction" in "... as well as the *purchase and sale* of existing residential real estate properties...". We hypothesise that this is caused predominantly by the fact that our Embeddings-PR model is trained over a vocabulary that is much larger than that featured in SimplePPDB. However, we also found that the Embeddings-PS simplifier replaces complex phrases with antonyms much more frequently than SimplePPDB-PS. Some examples are the phrases "provides details" and "quite rapidly", which were replaced by "summarizes" and "gradually", respectively. This is caused by the fact that this type of embeddings model tends to group antonyms close together [18].

7 Final Remarks

We introduced resources and approaches that can be used to address the task of Phrase-Level Simplification. We presented a way of training and employ-

ing POS-aware retrofitted phrase embedding models for SG, introduced a new unsupervised comparison-based approach for SS, and proposed a set of phrase-level features that can complement consolidated supervised and unsupervised SR strategies.

To train and evaluate phrase simplifiers, we created BenchPS: a new dataset produced using a flexible annotation methodology that aims to maximize the recall of gold simplifications available and minimize the costs of producing simplicity rankings. Through experimentation, we found that our enhanced phrase embeddings provide a more reliable source of simplifications for complex phrases than stand-alone SimplePPDB. We also found that our comparison-based SS approach can be more effective in discarding inappropriate candidates than the former state-of-the-art strategy. Our experiments with candidate ranking show that adding our phrase-level features to state-of-the-art rankers can increase their performance significantly, specially when a large amount of candidate substitutes are produced during SG and SS. We also found that context plays a more important role in the simplification of phrases than of single words. Through a manual inspection of the mistakes made by our most reliable phrase simplifier, we found that it correctly simplifies complex phrases 61% of the time.

In future work, we intend to address the challenges of automatically identifying complex phrases, as well as simplifying out of vocabulary phrases. The BenchPS dataset[4] is already available online. The code for the phrase simplification approaches introduced will be made available once this paper is published.

References

1. M. Azab, C. Hokamp, and R. Mihalcea. Using word semantics to assist english as a second language learners. In *Proceedings of the 2015 NAACL*, 2015.
2. O. Biran, S. Brody, and N. Elhadad. Putting it simply: a context-aware approach to lexical simplification. In *Proceedings of the 49th ACL*, pages 496–501, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.
3. M. Brysbaert and B. New. Moving beyond kučera and francis: A critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for american english. *Behavior research methods*, 41:977–990, 2009.
4. M. Brysbaert and B. New. Moving beyond kucera and francis: a critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for american english. *Behavior research methods*, 41:977–90, 2009.
5. J. De Belder and M.-F. Moens. A dataset for the evaluation of lexical simplification. In *Computational Linguistics and Intelligent Text Processing*, pages 426–437. 2012.
6. S. Devlin. *Simplifying Natural Language for Aphasic Readers*. PhD thesis, University of Sunderland, 1999.
7. S. Devlin and J. Tait. The use of a psycholinguistic database in the simplification of text for aphasic readers. *Linguistic Databases*, pages 161–173, 1998.
8. S. Devlin and G. Unthank. Helping aphasic people process online information. In *Proceedings of the 8th SIGACCESS*, pages 225–226, 2006.

[4] `http://ghpaetzold.github.io/data/BenchPS.zip`

9. N. Elhadad. Comprehending technical texts: Predicting and defining unfamiliarterms. In *Proceedings of the 2006 AMIA*, 2006.
10. N. Elhadad and K. Sutaria. Mining a lexicon of technical terms and lay equivalents. In *Proceedings of the 2007 BioNLP*, pages 49–56, 2007.
11. M. Faruqui, J. Dodge, S. K. Jauhar, C. Dyer, E. Hovy, and N. Smith. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 NAACL*, pages 1606–1615, Denver, Colorado, May–June 2015. Association for Computational Linguistics.
12. C. Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.
13. G. Glavaš and S. Štajner. Simplifying lexical simplification: Do we need simplified corpora? In *Proceedings of the 53rd ACL*, pages 63–68, Beijing, China, July 2015. Association for Computational Linguistics.
14. C. Horn, C. Manduca, and D. Kauchak. Learning a lexical simplifier using wikipedia. In *Proceedings of the 52nd ACL*, pages 458–463, Baltimore, Maryland, June 2014. Association for Computational Linguistics.
15. D. Kauchak. Improving text simplification language modeling using unsimplified text data. In *Proceedings of the 51st ACL*, pages 1537–1546, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.
16. P. Lison and J. Tiedemann. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In *Proceedings of the 10th LREC*, 2016.
17. T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
18. T. Mikolov, W.-t. Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of 2013 NAACL*, pages 746–751, 2013.
19. B. P. Nunes, R. Kawase, P. Siehndel, M. a. Casanova, and S. Dietze. As simple as it gets - a sentence simplifier for different learning levels and contexts. In *Proceedings of the 13th ICALT*, pages 128–132, 2013.
20. G. Paetzold and L. Specia. Understanding the lexical simplification needs of non-native speakers of english. In *Proceedings of the 26th COLING*, pages 717–727, Osaka, Japan, 2016. The COLING 2016 Organizing Committee.
21. G. Paetzold and L. Specia. Lexical simplification with neural ranking. In *Proceedings of the 15th EACL*, pages 34–40. Association for Computational Linguistics, 2017.
22. G. H. Paetzold. *Lexical Simplification for Non-Native English Speakers*. PhD thesis, University of Sheffield, 2016.
23. G. H. Paetzold and L. Specia. Lexenstein: A framework for lexical simplification. In *Proceedings of ACL-IJCNLP 2015 System Demonstrations*, pages 85–90, Beijing, China, July 2015. Association for Computational Linguistics and The Asian Federation of Natural Language Processing.
24. G. H. Paetzold and L. Specia. Benchmarking lexical simplification systems. In *Proceedings of the 10th LREC*, Portoroz, Slovenia, 2016. European Language Resources Association (ELRA).
25. G. H. Paetzold and L. Specia. Collecting and exploring everyday language for predicting psycholinguistic properties of words. In *Proceedings of the 26th COLING*, pages 1669–1679, Osaka, Japan, December 2016.
26. G. H. Paetzold and L. Specia. Semeval 2016 task 11: Complex word identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 560–569, San Diego, California, June 2016. Association for Computational Linguistics.
27. G. H. Paetzold and L. Specia. Unsupervised lexical simplification for non-native speakers. In *Proceedings of the 13th AAAI*, pages 3761–3767. AAAI Press, 2016.

28. E. Pavlick and C. Callison-Burch. Simple ppdb: A paraphrase database for simplification. In *Proceedings of the 54th ACL*, pages 143–148, 2016.
29. E. Pavlick, P. Rastogi, J. Ganitkevitch, B. Van Durme, and C. Callison-Burch. Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd ACL*, pages 425–430. Association for Computational Linguistics, 2015.
30. J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. *Proceedings of the 2014 EMNLP*, pages 1532–1543, 2014.
31. M. Shardlow. A comparison of techniques to automatically identify complex words. In *Proceedings of the 51st ACL Student Research Workshop*, pages 103–109, 2013.
32. L. Specia. Translating from complex to simplified sentences. In *Computational Processing of the Portuguese Language*, pages 30–39. 2010.
33. L. Specia, S. K. Jauhar, and R. Mihalcea. Semeval-2012 task 1: English lexical simplification. In *Proceedings of the 1st SemEval*, pages 347–355, Montréal, Canada, 2012. Association for Computational Linguistics.
34. K. Toutanvoa and C. Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the 2000 SIGDAT*, pages 63–70, Hong Kong, China, October 2000. Association for Computational Linguistics.
35. S. Wang and C. Zong. Comparison study on critical components in composition model for phrase representation. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 16(3):16, 2017.
36. S. Wubben, A. van den Bosch, and E. Krahmer. Sentence simplification by monolingual machine translation. In *Proceedings of the 50th ACL*, pages 1015–1024, 2012.
37. W. Xu, C. Callison-Burch, and C. Napoles. Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics*, 3:283–297, 2015.
38. W. Xu, C. Napoles, E. Pavlick, Q. Chen, and C. Callison-Burch. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415, 2016.
39. W. Yin and H. Schütze. An exploration of embeddings for generalized phrases. In *Proceedings of the ACL 2014 Student Research Workshop*, pages 41–47, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics.
40. Y. Zhao, Z. Liu, and M. Sun. Phrase type sensitive tensor indexing model for semantic composition. In *Proceedings of the 2015 AAAI*, pages 2195–2202, 2015.
41. Z. Zhu, D. Bernhard, and I. Gurevych. A monolingual tree-based translation model for sentence simplification. *Computational Linguistics*, (August):1353–1361, 2010.