Neural Computing and Applications (2023) 35:6265–6280
https://doi.org/10.1007/s00521-022-07905-y

ORIGINAL ARTICLE

SimpLex: a lexical text simplification architecture

Ciprian-Octavian Truică[1,2] · Andrei-Ionuţ Stan[2] · Elena-Simona Apostol[1,2]

Received: 21 January 2022 / Accepted: 28 September 2022 / Published online: 18 November 2022

Abstract
Text simplification (TS) is the process of generating easy-to-understand sentences from a given sentence or piece of text. The aim of TS is to reduce both the lexical (which refers to vocabulary complexity and meaning) and syntactic (which refers to the sentence structure) complexity of a given text or sentence without the loss of meaning or nuance. In this paper, we present SimpLex, a novel simplification architecture for generating simplified English sentences. To generate a simplified sentence, the proposed architecture uses either word embeddings (i.e., Word2Vec) and perplexity, or sentence transformers (i.e., BERT, RoBERTa, and GPT2) and cosine similarity. The solution is incorporated into a user-friendly and simple-to-use software. We evaluate our system using two metrics, i.e., SARI and Perplexity Decrease. Experimentally, we observe that the transformer models outperform the other models in terms of the SARI score. However, in terms of perplexity, the word embedding-based models achieve the biggest decrease. Thus, the main contributions of this paper are: (1) We propose a new word embedding and transformer-based algorithm for text simplification; (2) we design SimpLex—a modular novel text simplification system—that can provide a baseline for further research; and (3) we perform an in-depth analysis of our solution and compare our results with two state-of-the-art models, i.e., LightLS as reported by Glavaš and Štajner (in: Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing, 2015) and NTS-w2v as reported by Nisioi et al. (in: Proceedings of the 55th annual meeting of the association for computational linguistics, 2017). We also make the code publicly available online.

Keywords Text simplification · Complexity prediction · Transformers · Word embeddings · Perplexity

1 Introduction

Text simplification is a complex process that requires both a good knowledge of the language and a deep understanding of what constitutes a simple text. In spite of this, recent advances in deep learning and especially natural language processing have enabled computer scientists to create systems that generate simplified text considered, even by human standards, to be of high quality [22]. The direction that has been explored in this field in the last years has been the usage of machine translation and recurrent neural networks (RNNs) to automatically generate simplified sentences, to the detriment of manually crafted syntactic simplification rules and lexical substitutions [54]. Furthermore, with the new developments in the field of machine and deep learning, new approaches for developing text simplification systems based on data-driven solution [3, 53, 59] and few-shot learning [26, 65] have emerged.

Text simplification has tremendous potential for helping specific groups of people in real-life situations [21]. More specifically, text simplification can benefit people suffering from certain conditions, such as dyslexia, autism, and aphasia [2]. These individuals have trouble reading and understanding long sentences or sentences that contain

✉ Elena-Simona Apostol
elena.apostol@upb.ro

Ciprian-Octavian Truică
ciprian.truica@upb.ro

Andrei-Ionuţ Stan
andrei_ionut.stan@stud.acs.upb.ro

[1] Department of Information Technology, Uppsala University, Uppsala, Sweden

[2] Computer Science and Engineering Department, Faculty of Automatic Control and Computers, University Politehnica of Bucharest, Bucharest, Romania

complex words. Thus, automatic text simplification can open up more works to them. Text simplification also enables non-native speakers of a given language to understand pieces of text more easily without loss of meaning [1, 54]. Current directions try to use (neural) machine translation for this task [16], with a focus on encoder–decoder RNN-based architectures [4, 44]. These models can be further improved by using the attention mechanisms [34] and different types of word embeddings, such as Word2Vec [38, 44] or GloVe [48, 58]. Although these approaches use machine translation methods for automatic language generation, they need a high level of explicit controllability, e.g., control the vocabulary size and words [44], use special tokens to represent specific grammatical attributes [36], employ domain specific dictionaries [41], use different word or sentence embeddings to find semantically and syntactical similar candidates for word [58] and sentences [27, 43, 55].

As text simplification is crucial in helping specific groups of people with reading disabilities, and improving reading comprehension for non-native speakers as well as medical practitioners, teachers, and researchers, with this work, we try to answer the following research questions:

Q_1 Can we design a modular architecture for text simplification that uses both word-based and transformer-based embeddings?
Q_2 Can our architecture be used as a baseline for further research in the field?

To answer the two research questions, the main objectives of this paper are:

O_1 Propose a new easy to use modular text simplification architecture that can be used as a baseline solution for research in the field;
O_2 Show the efficiency of the proposed architecture by comparing it with state-of-the-art text simplification systems.

In this paper, to address objective O_1, we propose SimpLex a simplification architecture that focuses on the lexical simplification of texts. SimpLex generates a simplified sentence that preserves the initial meaning of the original sentence. We propose two approaches for text simplification: (1) word embedding-based, and (2) transformer-based.

The word embedding-based approach uses the cosine similarity to select synonyms and then computes the perplexity score to determine the best-simplified sentence. Word embeddings assign a static, non-changeable embedding to each word in the vocabulary. This approach can be problematic for a simplification system, as it deprives it of knowledge about the context in which the words are used.

The transformer-based approach selects synonyms based on transformer embeddings and then uses the cosine similarity to select the best-simplified sentence. This approach generates embeddings on both the word and sentence level by taking a whole sentence and producing the embeddings. Thus, a single word can have different embeddings depending on the sentence and the context in which it is used. In this case, transformers prove to be more suited to choosing the best-fitting word than word embeddings.

We also provide a fully containerized server using docker and an easy-to-use UI that can serve as a tool for users to explore SimpLex. This is also an important contribution, as there are no commercial systems that provide automatic text simplification, mostly due to the fact that this is a very challenging task [57]. Furthermore, we make SimpLex's source code publicly available online on GitHub at https://github.com/elena-apostol/SimpLex.

To address objective O_2, we compare the results obtained by SimpLex with two state-of-the-art text simplification systems, i.e., LightLS [19] and NTS-w2v [44], on a well-established dataset in the community: WikiNet [24].

The main contributions of this work are:

(1) The task of text simplification is a real-world problem that needs to be addressed. We try to propose a solution to this problem through SimpLex.
(2) We propose a novel architecture for text simplification that uses state-of-the-art NLP and NLU models for word complexity prediction and synonym selection and ranking.
(3) We use both perplexity and cosine similarity to select the best synonym of a complex word and replace it with a simpler version without altering the meaning of the text.
(4) SimpLex uses a modular architecture, and thus, we can plug in other methods for synonym ranking (i.e., other word embeddings and other transformer models besides BERT, RoBERTa, and GPT2), even in other languages.
(5) We provide a fully functional implementation of SimpLex. As SimpLex can be deployed as a docker, this will help potential users of the software to easily use our solution.
(6) SimpLex can be used as a baseline for further research in the field of text simplification.
(7) Although evaluating human language complexity is tricky, we use a set of well-defined automatic methods to grant valuable insights into our results obtained with SimpLex.

The paper is structured as follows. In Sect. 2, we discuss the current state-of-the-art in the field. In Sect. 3, we present SimpLex and give a detailed view of each individual component. In Sect. 4, we analyze the results achieved by our novel architecture and discuss some examples of sentence simplifications. Finally, in Sect. 5, we present the conclusions and future directions for further research and improvement.

2 Related work

Nassar et al. [42] discuss the two current approaches to the text simplification problem, namely the neural approach and the non-neural one. They argue that, even though the neural approach is quite popular nowadays, a well-crafted rule-based architecture can outperform a neural-based system. Štajner and Glavaš [58] propose a system that performs transformations, not only at the lexical and syntactic levels but also on the discourse level. The system combines event-based simplification with lexical simplification and leads to significantly more content reduction within a sentence and within a text, managing to delete even whole sentences. Furthermore, Zhong et al. [68] observe that discourse-level factors contribute to the challenging task of predicting sentence deletion for simplification.

Bahdanau et al. [4] introduce an encoder–decoder recurrent neural network (RNN) architectures for text simplification. This model is improved by using the attention mechanism proposed by Luong et al. [34] and different types of word embeddings, i.e., Word2Vec. Nisioi et al. [44] utilize techniques from machine translation for text simplification and use a sequence-to-sequence encoder–decoder RNN-based architecture. Both the encoder and the decoder have two long short-term memory (LSTM) [23] layers of 500 hidden states of size 500. A global attention mechanism combined with input feeding for the decoder is also employed. Thus, the text simplification problem is viewed as a monolingual machine translation problem from a complex lexicon to a more simplified one. The authors are able to improve the original system with respect to the metrics proposed in the original paper. Surya et al. [61] propose a framework composed of a shared encoder and a pair of attentional decoders assisted by discrimination-based losses and denoising. The experimental results obtained on unlabeled texts collected from English Wikipedia show that the model performs lexical and syntactic text simplification. Other approaches use document embeddings for aligning sentences. Paun [46] proposes a new unsupervised method for aligning text based on Doc2Vec embeddings and a new alignment algorithm capable of aligning texts at different levels. Cumbicus-Pineda et al. [11] propose a new edit-based system that deals with syntactical simplifications by employing edit operations at word level and graph convolutional networks to minimize the dependency of the sentence's structure. Using this approach, the system manages to extract the exact representation of syntax.

For lexical simplification, Maddela and Xu [35] introduce a dataset of 15 000 words labeled with a ranking from 1 to 6 by 11 human volunteers. They also propose an architecture with both neural and non-neural elements to determine which words in a sentence are candidates for replacement, which candidates to generate, and, in the end, whether the new candidate sentences are indeed simpler than the original sentence. Moreover, Konkol [29] proposes a set of features for identifying complex words. Using information such as the unigram and bigram probability of a given word, the number of sentences in which a word appears, and the WordNet Synset size of a word, the author is able to train an SVM (Support Vector Machines) [10] model to predict if a word is complex or not. Zhao et al. [67] propose an asymmetric denoising autoencoder-based method for sentences with separate complexity to generate appropriate complex/simple pair. Alarcon et al. [2] propose a lexical simplification system for Spanish that (1) identifies complex words using an SVM model, (2) offers replacement candidates by employing a substitute generation approach using databases and a selection technique based on a pre-trained Word2Vec embedding model for Spanish, and (3) uses a pre-trained multilingual BERT model for word sense disambiguation. Qiang et al. [49] propose LSBert, a lexical simplification system based on pre-trained BERT to (1) determine complex polysemous words and (2) improve the candidate selection.

Garbacea et al. [18] propose a compact pipeline for the simplification task. Their work focuses on the first two subtasks of the pipeline. In the first step, they decide if a given text needs or not to be simplified by applying different traditional or deep classification models. The second step is the explanation, where they highlight the part of the text that needs to be simplified.

Lin and Wan [32] propose a new model for Semantic Dependency Information guided Sentence Simplification

(SDISS). SDISS uses a sentence encoder to obtain contextual representations and a graph encoder to extract semantic dependencies. Using these two encoders, the system manages to produce simplified sentences that are aware of the semantic dependencies between the words within a sentence.

Dehghan et al. [12] propose a new system (GRS) that combines generating and revision for unsupervised text simplification. GRS uses BART-based [30] paraphrasing and deletion as edit operations and a scoring function based on three metrics, i.e., Meaning Preservation, Linguistic Acceptability, and Simplicity, to search for the best simplification. Devaraj et al. [13] also use paraphrasing to simplify medical texts. The model is a transformer-based encoder–decoder that penalizes the decoder when producing 'jargon' terms using an unlikelihood loss function for augmentation. The system is further studied by Devaraj et al. [14] who observe that the system yields errors at insertion, deletion, and substitution. Zhang et al. [66] use a neural model for sentence deletion prediction to simplify text while maintaining the functional discourse structure. The neural model uses a Bi-LSTM encoder–decoder structure with an attention layer between the encoder and decoder. As input, the model receives BERT sentence representations.

3 Methodology

SimpLex's architecture is presented in Fig. 1 and it shows the general overview of the different modules that are used by our solution to achieve the task of text simplification. We describe in detail each of SimpLex's module in the following subsections.

SimpLex's input is a sentence, and the output is the simplified sentence.

The first module, i.e., complexity predictions, determines if the words contained in a sentence are complex or not. This module uses a neural network to predict the complexity of words within the input sentence. Section 3.1 presents in detail this module.

The second module, i.e., synonym generation, is used to generate synonyms for the complex words determined by the complexity predictions module. Section 3.4 presents in detail this synonym generation module.

After determining which type of embedding to use, i.e., selecting between a word embedding approach (Word embedding-based) or a transformer one (Transformer-based), the third module, i.e., Synonym Selection, is used to further refine the synonym list by choosing the synonyms that have the biggest similarity to the original word to be replaced. Section 3.3 presents the Synonym Selection module. The word embedding-based is presented in Sect. 3.3.1, while the transformer-based approach is presented in Sect. 3.3.1.

The fourth module, i.e., generate candidate sentences, is used to generate candidate sentences regardless of the embedding. Section 3.4 presents in detail the generate candidate sentences module.

The fifth module, i.e., sentence ranking, is used to rank sentences and select the best sentence that is simpler than the original one and also keeps the meaning intact. Section 3.5 presents the sentence ranking module. The perplexity ranking approach is presented in Sect. 3.5.1, while the cosine similarity ranking approach is presented in Sect. 3.5.2. The output of this module is the simplified sentence.

In Sect. 3.6, we present SimpLex's algorithm which puts all these modules together in order to obtain the best-simplified version for a given sentence.

Finally, in Sect. 3.7, we present the implementation details and user interface, and we provide the open source link to SimpLex's source code.

3.1 Complexity prediction

Given a sentence, the complexity prediction module decides which words are the best candidates for a potential replacement. Failure to select the right words or selecting too many words can lead to candidate sentences that remain the same or are more complex and obfuscated than the original ones. To determine whether a word is complex

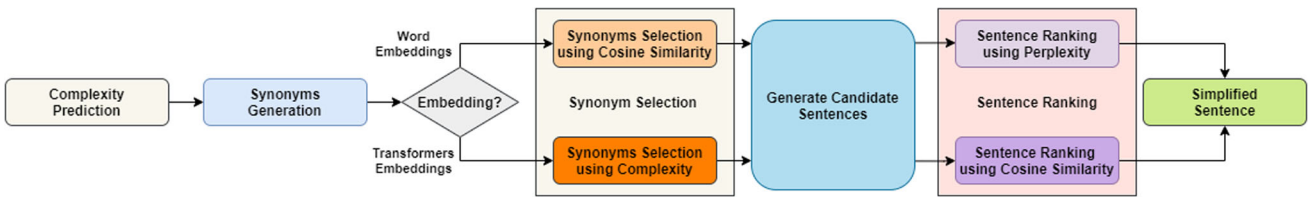

Fig. 1 SimpLex architecture

or not, we build a multilayer perceptron neural network that classifies the words into complex or simple classes. We use the dataset from [35] containing words ranked on a scale from 1 to 6. We take the complexity rankings of the words and set the threshold of complexity at 3 to create two classes with words ranked from 1 to 3 labeled as simple and words ranked from 3 to 6 labeled as complex. After re-labeling the words, we select the features for each word, basing our assessment on the work presented in [29]. Thus, we select the following five features:

(1) *Unigram probability of apparition in the selected language model* in a language model, words that appear more often are, in general, more common, and thus, less complex.
(2) *Number of sentences in which the word appears in the selected language model* this is an important feature, as a word that is not necessarily simple can appear many times in a single sentence because of the fact that is, for example, the subject of the sentence. If a word appears in just a few sentences, it may be that the word is not as common as other words that appear in many sentences.
(3) *Number of apparition of the word in the selected language model* the same reasoning as the unigram probability of apparition.
(4) *Word length* more of a heuristic, but small words tend to be simpler than long words.
(5) *WordNet Synset size of the word* in WordNet [40], a Synset of a word consists of words that express the same concept. The larger the Synset, the more meanings a word has. Thus, it can potentially be more complex.

To train our model, we need to choose a dataset to train our complexity prediction model. The choice of the dataset is especially important, as it must reflect in a realistic manner the common words used in day-to-day speech. For example, a dataset that contains academic discourse might not be a very inspired choice, as the words that are used frequently in this type of dataset might not be necessary simple or common words. We choose the News Crawl dataset [7] with news articles from 2017 as mass media employs easy-to-understand words in their writings.

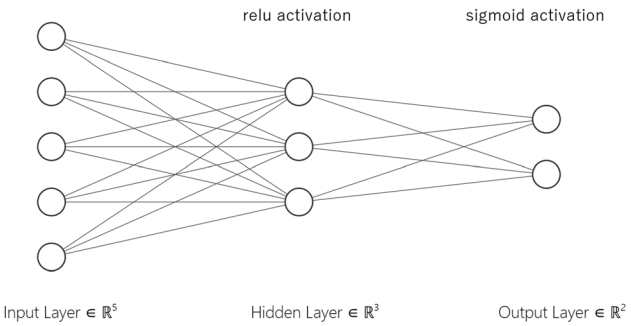

Fig. 2 The neural network for complexity prediction

We train a multilayer perceptron (MLP) neural network with one hidden layer of size 3 (Fig. 2). The neural network's input layer contains 5 units. Each unit corresponds to one of the five features identified above. The output is a one-hot-encoder vector that determines the probability of a word to be complex or simple, i.e., the vector $[1\ 0]$ means the word is simple, while the vector $[0\ 1]$ means the word is complex. The hidden layer employs ReLU as the activation function. The ReLU activation function used is defined as the positive part of its argument (Eq. (1)). We choose this activation function as it solves the vanishing gradient problem [20]. The final layer contains 2 units for classifying the words into complex or simple. These units employ the sigmoid activation function (Eq. (2)). The sigmoid function is used to predict probabilities that exist in the range [0, 1], mapping the output of the model to the on-hot-encoder class representation. The proposed neural network uses ADAM [28] as the solver for weight optimization, along with a constant learning rate of 0.001. The ADAM optimizer is a good choice for sparse gradients as it is invariant to their diagonal rescale [28].

$$f(x) = \max(0, x) \tag{1}$$

$$f(x) = \frac{1}{1 + e^{-1}} \tag{2}$$

3.2 Synonym generation

The synonym generation module builds a list of potential replacements for each candidate complex word. At this stage of the simplification pipeline, it is not necessary to select the best candidates from among the synonyms, as there will be a specialized module for this task. In the literature, two main directions have been explored regarding synonym generation: the thesaurus-based approach and the automatic approach. In the automatic approach, pairs of substitutions are generated from aligned corpora of texts [54]. By far, the most popular dataset has proven to be the aligned Wikipedia-Simplified dataset. Yatskar et al. [64] explored an approach in which they extracted pairs of simplifications from Simple Wikipedia edits made by users. The thesaurus approach is much simpler, as it simply requires querying a source and extracting the result. This has the added advantage that

many reputable thesauri have been checked manually, and miss-matches are not very frequent. The main problem with a thesaurus-based approach is that the context of the word is not known. Thus, a word that manifests a high degree of polysemy can pose a problem for such a system, resulting in irrelevant matches. Furthermore, the part of speech that the word has in a sentence must be taken into consideration: take, for example, the word 'looks' which can be used either as a verb (to look) or as a noun (appearance).

In the SimpLex architecture, we employ a thesaurus-based search, querying https://www.synonym.com/. Prior to querying the thesaurus, we transform the word to its base form, the so-called lemmatization step. In this way, we make sure that we have the best chance of obtaining the ideal candidate among the synonym list. We also perform the following steps:

(1) Select only those words that have the same part of speech as the original word, and
(2) Transform the remaining candidates to the original form required by the sentence.

The transformation operations include: declination of the verbs to the correct tense and person, pluralization for the nouns, morphological agreement, and comparative and superlative modes for the adjectives and adverbs.

After these steps are completed, we obtain a list of synonyms that can be immediately plugged in the original sentence in place of the original word and still create grammatically correct sentences. In the following modules, the synonyms can be further refined by taking into consideration more complex aspects of the sentence.

3.3 Synonym selection

In this module, we aim to further refine the synonym list by choosing the synonyms that have the biggest similarity to the original word to be replaced. We use two distinct approaches, one based on word embeddings cosine similarity (word embedding-based) and one based on transformer embeddings (Transformer-based).

3.3.1 Synonym selection using cosine similarity

For the word embedding-based approach, we compute the cosine similarities (Eq. (3)) between the embedding $\mathbf{s} = \{s_1, \ldots, s_n\}$ of synonym *syn* and the embedding $\mathbf{w} = \{w_1, \ldots, w_n\}$ of the target word w. We keep only the synonyms that have a cosine similarity above the average of all the similarities.

$$\cos(\mathbf{w}, \mathbf{s}) = \frac{\sum_{i=1}^{n} w_i s_i}{\sqrt{\sum_{i=1}^{n} w_i^2} \sqrt{\sum_{i=1}^{n} s_i^2}} \tag{3}$$

Thus, we make sure that the new candidate words are indeed suitable replacements for the target word. We can now remark that using word embeddings, the initial thesaurus search can be bypassed if we consider the top-k best words with respect to their cosine similarities to the original word. As it turns out, the thesaurus search is indeed necessary. Embedding models are able to determine correct relations between words, such as synonymy, but are prone to labeling antonyms as related words [39] (for example, the words 'king' and 'queen' will be branded as similar, even though they are not real synonyms). Thus, we can end up in a situation in which the words are not really synonyms but related words with similar meanings. Using the thesaurus search, we ensure that the selected words are already good matches and just keep those that are the most similar to the candidate word.

3.3.2 Synonym selection using complexity

For the transformers-based approach, we do not need to check if the candidate synonym is similar to the target word because this is automatically done when a sentence embedding is created. Thus, we only check if the candidate synonym is indeed simpler than the word we want to replace. Because the target word is already marked as complex, it is sufficient for the synonym to be predicted as a simple word. Furthermore, we keep only those synonyms that are being predicted by the complexity prediction multilayer perceptron neural network as simple.

3.4 Generate candidate sentences

Regardless of the embedding, this module generates candidate sentences. Given a sentence $S = \{w_1, \ldots, w_n\}$, a word w at position $k \in \overline{1, n}$, and a list of synonyms *syns*, we obtain a *candidate* sentence by replacing w with a synonym $syn \in syns$ without changing any of the existing words that are on the left $(w_l, 0 \leq l < k)$ or write $(w_r, k < r \leq n)$ of word w. The candidate sentences consist of the original sentence in which the complex word detected is replaced by a synonym. We also perform the transform step from the Synonym Selection module.

3.5 Sentence ranking

After all the candidate sentences are generated, this module chooses the best sentence that is both simpler than the original one and also keeps the meaning intact.

3.5.1 Sentence ranking using perplexity

For the word embedding-based, we use perplexity as the metric for ranking sentences. We define the perplexity of a sentence $S = \{w_1, \ldots, w_n\}$ with respect to a given language model as $PP(S)$ (Eq. (4)). The rewritten form of perplexity for unigrams (under the assumption that all words are independent) is $PP_1(S)$ (Eq. (5)). While for a bigram, under the first-order Markov assumption, the perplexity of a sentence S is $PP_2(S)$ (Eq. (6)). For both formulas, the probability of a word w is $p(w) = \frac{f_w}{|V|}$ and the probability of a word w given the word v is $p(w|v) = \frac{f_{v,w}}{f_v}$, where $f_w, f_v, f_{v,w}$ are the frequency of w, v, and the co-occurrence of w and v, respectively, and $|V|$ is the size of the vocabulary.

$$PP(S) = p(w_1, w_2, ..., w_n)^{\frac{1}{n}} \tag{4}$$

$$PP_1(S) = (p(w_1)p(w_2)\ldots p(w_n))^{\frac{1}{n}} \tag{5}$$

$$PP_2(S) = (p(w_1)p(w_2|w_1)\ldots p(w_n|w_{n-1}))^{\frac{1}{n}} \tag{6}$$

Because of the small occurrence probabilities and their multiplication, the values can become very small, and numerical stability problems can occur. Thus, we can rewrite the relations for perplexity using Eqs. (7) and (8) for unigrams and bigrams, respectively.

$$PP_1(S) = 2^{-\frac{1}{n}\sum_{i=1}^{n} log(p(w_i))} \tag{7}$$

$$PP_2(S) = 2^{-\frac{1}{n}(log(p(w_1))+\sum_{i=2}^{n} log(p(w_i|w_{i-1})))} \tag{8}$$

In our sentence ranking, we use a linear combination of the two perplexity metrics. Equation (9) presents this score, which uses φ to minimize the impact of the bigrams. Rewritten with numerical stability in mind, the final scoring function is presented in Eq. (10).

$$PP_{\text{Score}}(S) = (1-\varphi)\cdot PP_1(S) + \varphi \cdot PP_2(S) \tag{9}$$

$$\begin{aligned} PP_{\text{Score}}(S) = (1-\varphi)\cdot 2^{-\frac{1}{n}\sum_{i=1}^{n}\log(p(w_i))} \\ + \varphi \cdot 2^{-\frac{1}{n}(\log(p(w_1))+\sum_{i=2}^{n}\log(p(w_i|w_{i-1})))} \end{aligned} \tag{10}$$

It should be noted that this approach ensures that sentences that are more probable with respect to the chosen language model are selected. This is generally desirable in a text simplification tool, but one major drawback of this approach is that it does not have any clue about the context of the words. This can become problematic in the case of polysemy. Take, for example, the noun 'spider,' which can mean the arachnid or, more rarely, an additional rest cue used in billiards. A simplification system that does not know context might interpret the word 'spider' in the sentence 'The English player used the spider to execute the shot' as referring to the arachnid. This is why it is desirable to introduce embeddings that are context-aware.

3.5.2 Sentence ranking using cosine similarity

For the transformer-based approach, we exploit the *sentence-level embeddings* created by the transformer embedding. The advantage of transformers over word embeddings is that the embeddings for a given word can change depending on the context in which that particular word is used in a sentence, while word embeddings are static embeddings.

After we have sentence embeddings, we can select the candidate sentence that has the biggest cosine similarity to the original sentence. These cosine similarities have, in general, high values above 0.9. This is because only a word has been replaced, and the other words remain the same. Thus, the sentence remains mainly unchanged. This is not an issue, as differences are still visible between the words that fit the sentence and keep the context, and words that change the context or meaning. It is interesting to note that the approach using transformers will not necessarily produce a new sentence with the lowest perplexity, but this is not always a bad thing, as in some cases, the meaning cannot be fully preserved by just looking for the smallest perplexity.

3.6 SimpLex algorithm

Algorithm 1 presents the SimpLex pseudocode. Given a sentence S (Line 1), we test if each word w is complex or not (Line 2) using the methodology presented in Sect. 3.1. If the word w is complex, then we generate the list of its synonyms *syns* (Line 3) using the methodology presented in Sect. 3.2. We check the synonym complexity and update the synonyms list removing all the words that are not deem complex by the proposed multilayer perceptron complexity prediction classifier (Sect. 3.1) (Lines 4 to 6).

Algorithm 1: SIMPLEX algorithm

Input: A complex sentence S
Output: A simplified sentence S'
1 **foreach** *word* w *in* S **do**
  // Determine w complexity
2 **if** w *is a complex word* **then**
    // Get all synonyms syn for the word w
3    $syns \leftarrow \{syn \mid syn \sim w\}$
    // Check the synonym complexity
4   **for** $syn \in syns$ **do**
5   **if** syn *is not a simple word* **then**
6        $syns \leftarrow syns \setminus \{syn\}$
7   **if** *use* WORD EMBEDDINGS **then**
      // Compute the average cos
8      $cos_{avg} \leftarrow 0$
9      $\mathbf{w} \leftarrow WordEmbedding(w)$
10   **foreach** $syn \in syns$ **do**
11       $\mathbf{s} \leftarrow WordEmbedding(syn)$
12       $cos_{avg} \leftarrow cos_{avg} + cos(\mathbf{w}, \mathbf{s})$
13     $cos_{avg} \leftarrow \frac{cos_{avg}}{|syns|}$
      // Keep only synonyms with $cos \geq cos_{avg}$
14     $syns' \leftarrow \emptyset$
15   **foreach** $syn \in syns$ **do**
16       $\mathbf{s} \leftarrow WordEmbedding(syn)$
17   **if** $cos_{avg} \geq cos(\mathbf{w}, \mathbf{s})$ **then**
18         $syns \leftarrow syns \setminus \{syn\}$
      // Generate candidate sentences
19     $candidates \leftarrow \emptyset$
20   **foreach** $syn \in syns$ **do**
        // Replace word w with syn in S
21       $candidate \leftarrow \{w_l | w_l \in S \wedge 0 \leq l < k\} \cup \{syn\} \cup \{w_r | w_r \in S \wedge k < r \leq n\}$
22       $candidates \leftarrow candidates \cup \{candidate\}$
      // Get the candidate with the lowest perplexity
23     $PP_{min} \leftarrow \infty$
24   **foreach** $candidate \in candidates$ **do**
25   **if** $PP_{Score}(candidate) < PP_{min}$ **then**
26         $PP_{min} \leftarrow PP_{Score}(candidate)$
27         $S' \leftarrow candidate$
28   **else if** *use* TRANSFORMERS EMBEDDINGS **then**
      // Generate candidate sentences
29     $candidates \leftarrow \emptyset$
30   **foreach** $syn \in syns$ **do**
        // Replace word w with syn in S
31       $candidate \leftarrow \{w_l | w_l \in S \wedge 0 \leq l < k\} \cup \{syn\} \cup \{w_r | w_r \in S \wedge k < r \leq n\}$
32       $candidates \leftarrow candidates \cup \{candidate\}$
      // Get candidate with the highest cos
33     $cos_{max} \leftarrow -\infty$
34     $\mathbf{S} \leftarrow TransformerEmbedding(S)$
35   **foreach** $candidate \in candidates$ **do**
36       $\mathbf{C} \leftarrow TransformerEmbedding(candidate)$
37   **if** $cos(\mathbf{S}, \mathbf{C}) > cos_{max}$ **then**
38         $cos_{max} \leftarrow cos(\mathbf{S}, \mathbf{C})$
39         $S' \leftarrow candidate$
      // Compute the perplexity for S'
40     $PP_{S'} \leftarrow PP_{Score}(S')$
41 **return** S'

For the word embedding-based approach (Line 7), we perform the following steps as discussed in Sect. 3.3.1:

(1) Compute the average cosine similarity between word embedding **w** of word w and each embedding **s** of synonyms $syn \in syns$ (Lines 8 to 13);
(2) Keep only the synonyms that have a cosine similarity that is larger or equal to the average (Lines 14 to 18);
(3) Generate all candidate sentences *candidates* by replacing w with its synonyms syn without changing any of the existing words position on the left (w_l) or right (w_r) of w (Lines 19 to 22) as discussed in Sect. 3.4;
(4) Extract the candidate sentence S' with the lowest perplexity score (Lines 23 to 27) using Eq. (10) from Sect. 3.5.1.

When using transformer-based approach (Line 28), the following steps are performed as discussed in Sect. 3.3.2:

(1) Generate all candidate sentences *candidates* using the same approach as when using word embeddings (Lines 29 to 32) as discussed in Sect. 3.4;.
(2) Use sentence embedding and extract the candidate sentence S' with the highest cosine similarity to S (Lines 33 to 39) as presented in Sect. 3.5.2.
(3) Compute the perplexity of S' (Lines 33 to 40) using Eq. (10) from Sect. 3.5.1.

Regardless of the used embedding, the algorithm returns the new simplified sentence S' (Line 41). We note that in our initial experiments, we used for the word embedding-based approach the cosine similarity as a metric for extracting the simplified sentence, but the results were worse than when using the perplexity score. Furthermore, in our transformer-based approach, we used the perplexity score to extract the best-simplified sentence, and we observed that this approach yields worse results than when using the cosine similarity, as can be seen in Sec. 4.

3.7 Implementation and user interface

SIMPLEX is implemented in *Python v3.7*. We use *NLTK*[1] [6] to preprocess the texts. For the complexity prediction module, we employ the multilayer perceptron model from *Scikit-Learn*[2] [47]. To generate candidate synonyms for the complex words, we use the *PyDictionary*[3] [8] library. The morphological changes are made using *pyinflect*[4] [25] and *pattern* [56] libraries. For loading word embeddings, we

[1] https://www.nltk.org/.
[2] https://scikit-learn.org/stable/.
[3] https://github.com/geekpradd/PyDictionary.
[4] https://github.com/bjascob/pyInflect.

use *Gensim*[5] [52], while for the transformer embedding, we employ *simpletransformers*[6] [51].

SimpLex also provides a simple-to-use and friendly user interface and full docker containerization [37]. In this way, a docker image is created from the Python official image, and then, the necessary third-party libraries are installed, along with the source code. After this, a basic REST API server is launched in the new container. SimpLex presents the user with a screen in which the sentence is introduced, and the simplification parameters are chosen (Fig. 3). The interface provides a quick and easy way of visualizing lexical changes made by our system.

The code is publicly available on GitHub at https://github.com/elena-apostol/SimpLex.

4 Results

In this section, we present the experimental evaluation of SimpLex and discuss the results.

4.1 Datasets

The dataset used for the testing of the simplification tool is WikiNet [24] and consists of a set of 100 English sentences taken from the Wikipedia corpus. The dataset contains alignments considered 'good' and 'partial good.' To evaluate the complexity prediction module, we employ other two datasets:

(1) the complexity ranking dataset [35], and
(2) the news corpus from News Crawl dataset [7] with news articles from 2017.

The complexity ranking dataset is a human-rated word complexity lexicon of 15 000 English words. The News Crawl dataset contains about 3.7 million sentences from all types of news, giving us a balanced and realistic look at the common words used in day-to-day speech.

4.2 Evaluation metrics

For the text simplification task, the evaluation is not as straightforward as for other machine learning tasks. This is because what is considered simple text is a very subjective topic, as it depends on the reader's experience with the language, education, etc. Nevertheless, researchers have been able to come up with a few interesting and efficient solutions to quantify the simplicity of a piece of text.

The two metrics that we choose for the evaluation of the simplification system are: SARI [63] and Perplexity Decrease [67]. SARI quantifies not only replacement correctness but also syntactical simplifications. It rewards both word replacement and eliminations, resulting in a more flexible evaluation for a simplification system. The Perplexity Decrease metric tracks how the perplexity changes when lexical simplifications are applied to the original text. It is desirable to generate sentences with a smaller perplexity so that they are more likely to appear, given the chosen language model. Although some studies also use BLEU [45] as an evaluation metric, the current literature shows that this metric is not adequate for evaluating Text Simplification [60].

We should note that there is a certain bias for particular types of simplifications for the SARI metric. High SARI scores are given to simplifications that have the most number of changes [44].

4.3 Setup

We test eight models in this setup: *(1)* five word embedding-based models, where the factor φ in which the bigram perplexity affects the sentence ranking score takes the values 0, 0.25, 0.5, 0.75, and 1, and *(2)* three transformer-based models, using the pre-trained BERT [15] (bert-base-uncased), RoBERTa [33] (roberta-base) and GPT2 [50] (gpt2) models from *Hugging Face* [62].

For the word embedding-based approach, we use Word2Vec [38] embeddings trained on the Wikipedia English corpus with a size of 300 [31].

4.4 Complexity prediction module results

Table 1 presents the evaluation of the complexity prediction module. The train and test datasets are split in a 95:5 ratio, resulting 14 250 train words and 750 test words.

The system manages to obtain good results for predicting word complexity (Table 1), even though the two datasets used (i.e., the complexity ranking dataset from [35] and the news corpus from News Crawl) have never been used together. Words that are ranked as simple by the human volunteers have a high probability of being detected by the classification model and remain unchanged in the text. Thus, the complexity prediction module turns out to be conservative, as the evaluation shows. In other words, the model determines with a higher accuracy simple words than complex words, i.e., a precision score of 0.79 when determining simple words versus a precision of 0.69 when determining complex words. This is desirable, as a complexity prediction system that classifies too many words as being complex can lead to simplifications that either obfuscate the initial sentence or make it lose its meaning.

To showcase the performance of our selected multilayer perceptron model, we compare it with three classic

[5] https://radimrehurek.com/gensim/.
[6] https://simpletransformers.ai/.

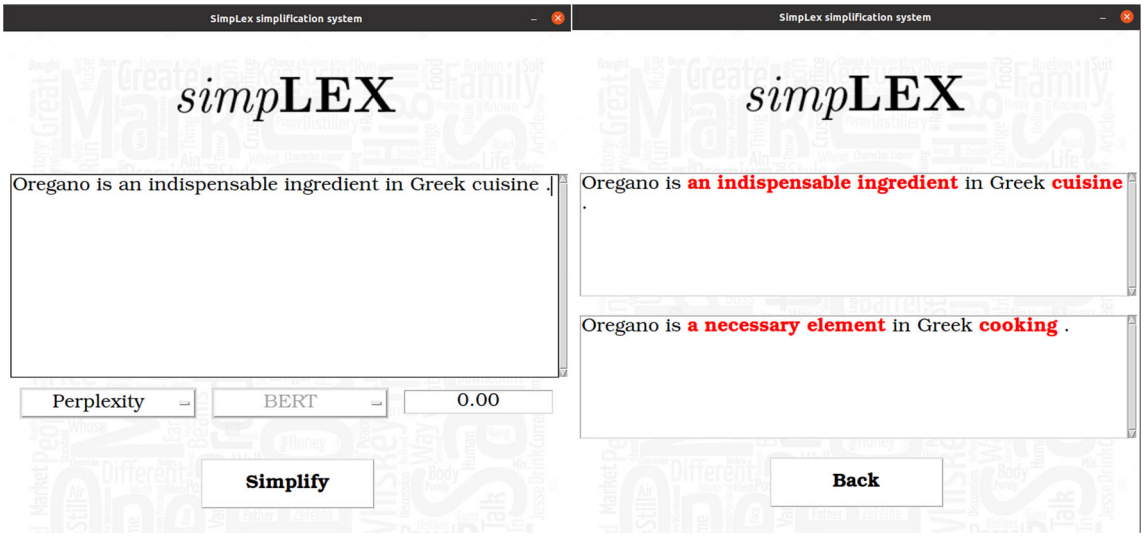

Fig. 3 The interface for the SimpLex system

Table 1 Complexity prediction system evaluation

Class	Precision	Recall	F1-score
0 (simple word)	0.79	0.85	0.80
1 (complex word)	0.69	0.64	0.66

machine learning models (Table 2), i.e., support vector machine, random forest, and extra randomize trees. For this set of experiments, we perform 10 separate runs. For each run, we split the dataset into 75% training set and 25% test set and then train and test all the considered machine learning algorithms, resulting in 11 250 words for training and 3 750 words for testing. Each dataset split uses the same label ratio as the original dataset. After performing the 10 runs, we compute the average and standard deviation for each metric, i.e., accuracy, precision, and recall, w.r.t. the tested model. Among the three models, random forest obtains slightly better results in terms of accuracy, precision, and recall than extra randomize trees. The support vector machine obtains the overall worse results. Our proposed model outperforms the other models, obtaining an average accuracy of 0.79.

4.5 SimpLex system results

The system obtains good results, even though the two datasets used (the complexity ranking dataset from [35] and the news corpus from News Crawl) have never been used together. Thus, we have no guarantees that, for example, words that are ranked as simple by the human volunteers will have a high probability of apparition in the language model. The model turns out to be a conservative one that is more confident regarding simple words than complex words. This is desirable, as a complexity prediction system that classifies too many words as being complex can lead to simplifications that either obfuscate the initial sentence or make it lose its meaning.

Table 3 presents quantitative results for the task of text simplification. Thus, we evaluate both approaches, i.e., the word embedding-based and transformer-based, using SARI and Perplexity Decreases metrics. The SARI metric quantifies both replacement correctness and syntactical simplifications for the task of text simplification. The Perplexity Decrease shows how well the meaning is preserved while applying lexical simplifications to the original text.

In the experiments with word embedding-based approach, we use different φ values to determine the scores change when minimizing the impact of the bigrams in the scoring function, i.e., PP_{Score} (Eq. (10)). We observe that after a given threshold for φ, i.e., 0.25, the SARI score does not change, while, by increasing the bigrams impact, the Perplexity Decrease lowers, indicating a better generalization performance.

For the transformer-based approach, we use three different models: BERT, RoBERTa, and GPT3. We observe that BERT obtains the highest SARI score, i.e., 0.350, and Perplexity Decrease, i.e., 8.4%. This indicates that the BERT transformer manages to better quantify both replacement correctness and syntactical simplifications as well as preserve the meaning, as opposed to the other two transformer models.

The results in Table 3 show that the transformer-based approach outperforms the word embedding-based approach in terms of the SARI score, but in terms of Perplexity

Table 2 Complexity prediction model comparison

Model	Accuracy	Precision	Recall
Multilayer perceptron	**0.79 ± 0.04**	0.85 ± 0.01	0.79 ± 0.04
Support vector machine	0.66 ± 0.01	0.64 ± 0.01	0.66 ± 0.01
Random forest	0.71 ± 0.01	0.71 ± 0.01	0.71 ± 0.01
Extra randomize trees	0.69 ± 0.01	0.69 ± 0.01	0.69 ± 0.01

Bold value indicates the highest accuracy

Decrease, the word embedding-based approach achieves the biggest decrease. This is because the word embedding-based approach actively tries to find the sentence with the lowest perplexity. We observe that the GPT2 model achieves both a good SARI score and a significant decrease in the average Perplexity of the sentences. The word embedding-based models that have a high bigram factor start to suffer because the language model is not nearly big enough to capture all the bigram combinations. In this comparison, we do not observe any important difference between the transformer-based models.

We compare our work with NTS-w2v [44] and LightLS [19] (Table 3). In terms of SARI score, we observe that all the word embedding-based models obtain similar scores as NTS-w2v and slightly lower scores than the LightLS. Thus, the SARI obtained with NTS-w2v is very similar to the score obtained with our word embedding-based approach, i.e., between 0.300 and 0.310, when varying the bigram factor φ in range [0.00, 1.00]. The word embedding-based approaches are outperformed by LightLS, which obtains a SARI score of 0.349 in comparison with the 0.310 obtained by the word embedding-based approach with a bigram factor $\varphi = 0.00$.

The transformer-based models outperform NTS-w2v and obtain similar scores as LightLS. With SARI scores of 0.350, 0.349, and 0.347 for the transformer-based approaches that employ BERT, RoBERTa, and GP2, respectively, we observe that BERT obtains the best SARI score. The Transformer-based approach that employs RoBERTa obtains the same SARI score as LightLS, i.e., 0.349, while the transformer-based approach that employs GPT2 slightly lags behind, with a difference between the SARI scores with the best-performing approach being only 0.002.

4.6 Examples and discussion

In this subsection, a few examples of simplification results of sentences from the employed dataset are presented. We discuss the patterns that emerge and try to assess the simplification quality of our architecture by comparing the results obtained by SimpLex with the results obtained by LightLS [19] and NTS-w2v [44]. We present three examples in Table 4 ; examples that are discussed in detail in the following paragraphs.

For the first example (Example 1 from Table 4), the simplification system has selected three words as being candidates for simplification, i.e., *indispensable*, *ingredient*, and *cooking*. Both the transformer and the word embedding-based models perform well, finding suitable replacements for the given words. As the bigram factor increases in the word embedding-based models, fewer words are replaced. Interestingly, there are some visible differences between the chosen words among the transformer models. Among those, a visual examination tends to suggest that the BERT model achieves the most reasonable result, while the other two models tend to judge the context slightly poorly. We observe that LightLS [19] changes *indispensable* with *essential*. For *indispensable ingredient*, TS-w2v [44] does the same replacement as our transformer-based model that employs BERT. Both LightLS [19] and TS-w2v [44] do not manage to replace the complex word *cuisine*.

For the second example (Example 2 from Table 4), we observe the improvements made by the transformer-based

Table 3 Text simplification performance results

Simplification model	SARI	Perplexity Decrease
Word embedding-based, bigram factor $\varphi = 0.00$	0.310	9.8%
Word embedding-based, bigram factor $\varphi = 0.25$	0.300	8.4%
Word embedding-based, bigram factor $\varphi = 0.50$	0.300	8.2%
Word embedding-based, bigram factor $\varphi = 0.75$	0.300	7.1%
Word embedding-based, bigram factor $\varphi = 1.00$	0.300	5.8%
Transformer-based, BERT model	**0.350**	8.4%
Transformer-based, RoBERTa model	0.349	8.8%
Transformer-based, GPT2 model	0.347	9.3%
NTS-w2v [44]	0.311	N/A
LightLS [19]	0.349	N/A

Bold value indicates the highest SARI score

Table 4 Output comparison of the simplification system

Example 1	
Oregano is **an indispensable ingredient** in Greek **cuisine**	Original
Oregano is **a necessary element** in Greek **cooking**	Word embedding-based, bigram factor $\varphi = 0.00$
Oregano is **a vital** ingredient in Greek **cooking**	Word embedding-based, bigram factor $\varphi = 0.25$
Oregano is **a vital** ingredient in Greek **cooking**	Word embedding-based, bigram factor $\varphi = 0.50$
Oregano is **a vital** ingredient in Greek **cooking**	Word embedding-based, bigram factor $\varphi = 0.75$
Oregano is **a vital** ingredient in Greek cuisine	Word embedding-based, bigram factor $\varphi = 1.00$
Oregano is **a vital element** in Greek **cooking**	Transformer-based, BERT model
Oregano is **a critical base** in Greek **cooking**	Transformer-based, RoBERTa model
Oregano is **a vital element** in Greek **preparation**	Transformer-based, GPT2 model
Oregano is **an essential** ingredient in Greek cuisine	LightLS [19]
Example 2	
It is **situated at** the coast of the Baltic Sea, where it **encloses** the city of Stralsund	Original
It is **find out at** the coast of the Baltic Sea, where it **tubes** the city of Stralsund	Word embedding-based, bigram factor $\varphi = 0.00$
It is **find out at** the coast of the Baltic Sea, where it **wraps** the city of Stralsund	Word embedding-based, bigram factor $\varphi = 0.25$
It is **find out at** the coast of the Baltic Sea, where it **wraps** the city of Stralsund	Word embedding-based, bigram factor $\varphi = 0.50$
It is **based at** the coast of the Baltic Sea, where it **wraps** the city of Stralsund	Word embedding-based, bigram factor $\varphi = 0.75$
It is **find out at** the coast of the Baltic Sea, where it **wraps** the city of Stralsund	Word embedding-based, bigram factor $\varphi = 1.00$
It is **located at** the coast of the Baltic Sea, where it **covers** the city of Stralsund	Transformer-based, BERT model
It is **determined at** the coast of the Baltic Sea, where it **bathes** the city of Stralsund	Transformer-based, RoBERTa model
It is **determined at** the coast of the Baltic Sea, where it **tubes** the city of Stralsund	Transformer-based, GPT2 model
It is **near at** the coast of the Baltic Sea, where it **surrounds** the city of Stralsund	LightLS [19]
It is **located on** the coast of the Baltic Sea, where it **surrounds** the town of Stralsund	NTS-w2v [44]
Example 3	
Since 2000, the **recipient** of the Kate Greenaway Medal has also been **awarded** the £5,000 Colin Mears Award	Original
Since 2000, the **host** of the Kate Greenaway Medal has also been awarded the £5,000 Colin Mears Award	Word embedding-based, bigram factor $\varphi = 0.00$
Since 2000, the **host** of the Kate Greenaway Medal has also been awarded the £5,000 Colin Mears Award .	Word embedding-based, bigram factor $\varphi = 0.25$
Since 2000, the **host** of the Kate Greenaway Medal has also been awarded the £5,000 Colin Mears Award	Word embedding-based, bigram factor $\varphi = 0.50$
Since 2000, the **host** of the Kate Greenaway Medal has also been awarded the £5,000 Colin Mears Award	Word embedding-based, bigram factor $\varphi = 0.75$
Since 2000, the **host** of the Kate Greenaway Medal has also been awarded the £5,000 Colin Mears Award	Word embedding-based, bigram factor $\varphi = 1.00$
Since 2000, the **heir** of the Kate Greenaway Medal has also been awarded the £5,000 Colin Mears Award	Transformer-based, BERT model
Since 2000, the **dependent** of the Kate Greenaway Medal has also been awarded the £5,000 Colin Mears Award	Transformer-based, RoBERTa model
Since 2000, the **receiver** of the Kate Greenaway Medal has also been awarded the £5,000 Colin Mears Award	Transformer-based, GPT2 model
Since 2000, the recipient of the Kate Greenaway Medal has also been **received** the £5,000 Colin Mears Award	LightLS [19]

Table 4 (continued)

Example 3	
Since 2000, the **host** of the Kate Greenaway Medal has also been **made** with the £5,000 Colin Mears Award	NTS-w2v [44]

Bold indicates the original complex words and their corresponding simplification

models in comparison with the word embedding-based models. In the word embedding-based models, the words *situated* and *encloses* are replaced by sub-optimal candidates, as the model has no knowledge of the context in which the words are used. The context refers to a geographical location, so the most appropriate words to be used are those that refer to spatial placement. The transformer models do not suffer from this drawback and manage to find more suitable and even natural replacements for the selected words. There are, again, differences between the models. RoBERTa and GPT2 choose strange candidates for the word *encloses*, while BERT seems to choose the most natural replacement among the tested models. Both LightLS [19] and NTS-w2v [44] replace *encloses* with *surrounds*, while for *situated*, LightLS [19] uses *near* with the wrong preposition *at* and NTS-w2v [44] uses *located on*.

In the last example (Example 3 from Table 4), we present a single word replacement, i.e., *recipient*, in the context of a larger simple sentence. The larger context refers to a medal nomination, so we anticipate the perfect candidate to be along the line of *receiver* or *winner*. As it can be seen, the word embedding-based models fail to capture the context, as expected and produce a sub-par simplification, choosing the word *host*. The choice is not satisfactory, as it changes the meaning of the sentence. Now, the subject of the sentence refers no more to the recipient of the medal but the host that awards the medal. This is a perfect example in which the lack of context can significantly hurt the performance of the simplification architecture. The transformer-based models benefit from context awareness and have a better chance of finding the right candidate. Thus, among the three pre-trained models tested, BERT and RoBERTa find sub-par, strange candidates, but the GPT2 model finds a good match in the word *receiver*. For this example, LightLS [19] does not replace *recipient*, while NTS-w2v [44] replaces it with *host*. Although SimpLex classifies *awarded* as a simple word, both LightLS [19] and NTS-w2v [44] replace it with *received* and *made*, respectively.

5 Conclusions

In this paper, we present SimpLex, a novel lexical simplification architecture that employs both word and transformers embeddings—achieving objective O_1. SimpLex uses either a word embedding-based or a transformer-based approach to generate simplified sentences—answering the research question Q_1. The word embedding-based approach uses Word2Vec and perplexity, while the transformer-based approach uses three transformers, i.e., BERT, RoBERTa, and GPT2, and cosine similarity. We perform ample experiments to show the feasibility of our architecture. For evaluation, we use two metrics, i.e., SARI and Perplexity Decrease. We compare our solution with two state-of-the-art models, i.e., LightLS [19] and NTS-w2v [44]—achieving objective O_1. We conclude that the transformer-based approach is more suited for the task of text simplification as transformer word and sentence embeddings better preserve the context improving the task of synonym detection and should be used together.

Furthermore, SimpLex provides a simple-to-use and friendly user interface—answering the research question Q_2. It can be run either from the command line or as a docker. We also provide the code for further development for interested users and researchers in the field of text simplification.

The current research identifies a series of shortcomings in the task of designing and developing text simplification systems [57], such as (1) publicly available datasets for low resource languages, (2) evaluation metrics that are focused on the final user, (3) quality of preserving grammaticality and meaning by the automatic text simplification systems, and (4) simple-to-use systems by non-specialized users.

The research community is trying to address some of these shortcomings by proposing new research tasks and directions [17]: *(1)* word or sentence ranking, *(2)* background knowledge searching, *(3)* text simplification for scientific datasets With SimpLex, our proposed novel text simplification system, we try to address some of these shortcomings. To summarize our contributions:

(1) We propose a new algorithm for text simplification that determines the complexity of words and utilizes word embedding-based or transformer-based approach to generate simplified sentences;

(2) We present a novel text simplification system called SimpLex that offers an intuitive user interface and a modular design that can help future development as well as provide a baseline for further research;
(3) We perform an in-depth analysis of our solution and compare our results with two state-of-the-art models, i.e., LightLS [19] and NTS-w2v [44].

In future work, we aim to add a syntactic simplification module. We also plan to use graph embedding and word representation formalism, e.g., Abstract Meaning Representation (AMR) [5] or MRS (Minimal Recursion Semantics) [9], for text simplification.

Declarations

Conflict of interest The authors declare that they have no conflict of interests.

References

1. Al-Thanyyan SS, Azmi AM (2022) Automated text simplification: a survey. ACM Comput Surv 54(2):1–36. https://doi.org/10.1145/3442695
2. Alarcon R, Moreno L, Martinez P (2021) Lexical simplification system to improve web accessibility. IEEE Access 9:58755–58767. https://doi.org/10.1109/access.2021.3072697
3. Alva-Manchego F, Scarton C, Specia L (2020) Data-driven sentence simplification: survey and benchmark. Comput Linguist 46(1):135–187. https://doi.org/10.1162/coli_a_00370
4. Bahdanau D, Cho K, Bengio Y (2015) Neural machine translation by jointly learning to align and translate. In: Bengio Y, LeCun Y (eds) 3rd International conference on learning representations ICLR
5. Banarescu L, Bonial C, Cai S, Georgescu M, Griffitt K, Hermjakob U, Knight K, Koehn P, Palmer M, Schneider N (2013) Abstract meaning representation for sembanking. In: Proceedings of the 7th linguistic annotation workshop and interoperability with discourse. Association for Computational Linguistics, Sofia, Bulgaria, pp 178–186. https://aclanthology.org/W13-2322
6. Bird S, Klein E, Loper E (2009) Natural language processing with Python: analyzing text with the natural language toolkit. O'Reilly Media, Inc
7. Bojar O, Chatterjee R, Federmann C, Graham Y, Haddow B, Huck M, Yepes A.J, Koehn P, Logacheva V, Monz C, Negri M, Neveol A, Neves M, Popel M, Post M, Rubino R, Scarton C, Specia L, Turchi M, Verspoor K, Zampieri M (2016) Findings of the 2016 conference on machine translation. In: Proceedings of the first conference on machine translation, vol 2, shared task papers. Association for computational linguistics. https://doi.org/10.18653/v1/W16-2301
8. Bora P (2020) Pydictionary . https://github.com/geekpradd/PyDictionary
9. Copestake A, Flickinger D, Pollard C, Sag IA (2005) Minimal recursion semantics: an introduction. Res Lang Comput 3(2–3):281–332. https://doi.org/10.1007/s11168-006-6327-9
10. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297. https://doi.org/10.1007/bf00994018
11. Cumbicus-Pineda O.M, Gonzalez-Dios I, Soroa A (2021) A syntax-aware edit-based system for text simplification. In: Proceedings of the international conference on recent advances in natural language processing (RANLP 2021), pp 324–334. INCOMA Ltd. https://aclanthology.org/2021.ranlp-1.38
12. Dehghan M, Kumar D, Golab L (2022) GRS: Combining generation and revision in unsupervised sentence simplification. In: Findings of the association for computational linguistics: ACL 2022, pp 949–960. Association for computational linguistics. https://doi.org/10.18653/v1/2022.findings-acl.77
13. Devaraj A, Marshall I, Wallace B, Li J.J (2021) Paragraph-level simplification of medical texts. In: Proceedings of the 2021 conference of the North American chapter of the association for computational linguistics: human language technologies, pp 4972–4984. Association for computational linguistics. https://doi.org/10.18653/v1/2021.naacl-main.395
14. Devaraj A, Sheffield W, Wallace B, Li J.J (2022) Evaluating factuality in text simplification. In: Proceedings of the 60th annual meeting of the association for computational linguistics, vol 1: Long Papers, pp 7331–7345. Association for computational linguistics. https://doi.org/10.18653/v1/2022.acl-long.506. https://aclanthology.org/2022.acl-long.506
15. Devlin J, Chang M.W, Lee K, Toutanova K (2019) Bert: pre-training of deep bidirectional transformers for language understanding. In: Conference of the North American chapter of the association for computational linguistics, pp 4171–4186. ACL
16. Erdem E, Kuyu M, Yagcioglu S, Frank A, Parcalabescu L, Plank B, Babii A, Turuta O, Erdem A, Calixto I, Lloret E, Apostol ES, Truică CO, Šandrih B, Martinčić-Ipšić S, Berend G, Gatt A, Korvel G (2022) Neural natural language generation: a survey on multilinguality, multimodality, controllability and learning. J Artif Intell Res 73:1131–1207. https://doi.org/10.1613/jair.1.12918
17. Ermakova L, Bellot P, Braslavski P, Kamps J, Mothe J, Nurbakova D, Ovchinnikova I, San-Juan E (2021) Text simplification for scientific information access. In: Lecture notes in computer science. Springer International Publishing, pp 583–592. https://doi.org/10.1007/978-3-030-72240-1_68
18. Garbacea C, Guo M, Carton S, Mei Q (2021) Explainable prediction of text complexity: the missing preliminaries for text simplification. In: Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing vol 1 Long Papers, pp 1086–1097. Association for computational linguistics. https://doi.org/10.18653/v1/2021.acl-long.88
19. Glavaš G, 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, vol 2 Short Papers. Association for computational linguistics. https://doi.org/10.3115/v1/p15-2011
20. Glorot X, Bordes A, Bengio Y (2011) Deep sparse rectifier neural networks. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics, pp 315–323. PMLR. https://proceedings.mlr.press/v15/glorot11a.html
21. Gooding S (2022) On the ethical considerations of text simplification. In: Ninth workshop on speech and language processing for assistive technologies (SLPAT-2022), pp 50–57. Association for computational linguistics. https://doi.org/10.18653/v1/2022.slpat-1.7
22. Grubišić A, Žitko B, Gašpar A, Vasić D, Dodaj A (2022) Evaluation of split-and-rephrase output of the knowledge extraction tool in the intelligent tutoring system. Expert Syst Appl 187:115900. https://doi.org/10.1016/j.eswa.2021.115900

23. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735
24. Hwang W, Hajishirzi H, Ostendorf M, Wu W (2015) Aligning sentences from standard Wikipedia to Simple Wikipedia. In: Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: human language technologies. Association for Computational Linguistics, Denver, Colorado, pp 211–217. https://doi.org/10.3115/v1/N15-1022. http://ssli.ee.washington.edu/tial/projects/simplification/
25. Jascob B (2020) Pyinflect. https://github.com/bjascob/pyInflect
26. Jin X, Lin B.Y, Rostami M, Ren X (2021) Learn continually, generalize rapidly: lifelong knowledge accumulation for few-shot learning. In: Findings of the association for computational linguistics: EMNLP 2021, pp 714–729. Association for computational linguistics . https://doi.org/10.18653/v1/2021.findings-emnlp.62
27. Kajiwara T, Komachi M (2016) Building a monolingual parallel corpus for text simplification using sentence similarity based on alignment between word embeddings. In: Proceedings of COLING 2016, the 26th international conference on computational linguistics: technical papers, pp 1147–1158. https://www.aclweb.org/anthology/C16-1109
28. Kingma D.P, Ba J (2015) Adam: a method for stochastic optimization. In: The 3rd international conference on learning representations (ICLR2015)
29. Konkol M (2016) Uwb at semeval-2016 task 11: exploring features for complex word identification. In: Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016), pp 1038–1041
30. Lewis M, Liu Y, Goyal N, Ghazvininejad M, Mohamed A, Levy O, Stoyanov V, Zettlemoyer L (2016) BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In: Annual meeting of the association for computational linguistics, pp 7871–7880. https://doi.org/10.18653/v1/2020.acl-main.703
31. Lin TJ (2020) Word2Vec embeddings trained on wikipedia https://github.com/lintseju/word_embedding
32. Lin Z, Wan X (2021) Neural sentence simplification with semantic dependency information. In: Proceedings of the AAAI conference on artificial intelligence, pp 13371–13379. https://ojs.aaai.org/index.php/AAAI/article/view/17578
33. Liu Y, Ott M, Goyal N, Du J, Joshi M, Chen D, Levy O, Lewis M, Zettlemoyer L, Stoyanov V (2019) Roberta: a robustly optimized bert pretraining approach
34. Luong T, Pham H, Manning CD (2015) Effective approaches to attention-based neural machine translation. In: Proceedings of the 2015 conference on empirical methods in natural language processing, Association for computational linguistics, Lisbon, Portugal, pp 1412–1421. https://doi.org/10.18653/v1/D15-1166. https://www.aclweb.org/anthology/D15-1166
35. Maddela M, Xu W (2018) A word-complexity lexicon and a neural readability ranking model for lexical simplification. In: Proceedings of the 2018 conference on empirical methods in natural language processing, Association for Computational Linguistics, Brussels, Belgium, pp 3749–3760. https://doi.org/10.18653/v1/D18-1410. https://www.aclweb.org/anthology/D18-1410
36. Martin L, Éric de la Clergerie Sagot B, Bordes A (2020) Controllable sentence simplification. In: Conference on language resources and evaluation, pp 4689–4698
37. Merkel D (2014) Docker: lightweight linux containers for consistent development and deployment. Linux J 239:2
38. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. In: International conference on learning representations
39. Mikolov T, Yih W.t, Zweig G (2013) Linguistic regularities in continuous space word representations. In: Proceedings of the 2013 conference of the North American chapter of the association for computational linguistics: human language technologies, pp 746–751
40. Miller GA (1995) Wordnet: a lexical database for english. Commun ACM 38(11):39–41. https://doi.org/10.1145/219717.219748
41. Nassar I, Ananda-Rajah M, Haffari G (2019) Neural versus non-neural text simplification: a case study. In: Australasian language technology association, pp 172–177
42. Nassar I, Ananda-Rajah M, Haffari G (2019) Neural versus non-neural text simplification: a case study. In: Proceedings of the 17th annual workshop of the australasian language technology association, pp 172–177
43. Nishihara D, Kajiwara T, Arase Y (2019) Controllable text simplification with lexical constraint loss. In: Proceedings of the 57th annual meeting of the association for computational linguistics: student research workshop, pp 260–266. https://doi.org/10.18653/v1/P19-2036
44. Nisioi S, Štajner S, Ponzetto S.P, Dinu L.P (2017) Exploring neural text simplification models. In: Proceedings of the 55th annual meeting of the association for computational linguistics, vol 2: Short papers, pp 85–91
45. Papineni K, Roukos S, Ward T, Zhu WJ (2002) Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting of the association for computational linguistics, pp 311–318
46. Paun S (2021) Parallel text alignment and monolingual parallel corpus creation from philosophical texts for text simplification. In: Proceedings of the 2021 conference of the North American chapter of the association for computational linguistics: student research workshop, pp 40–46. Association for computational linguistics, Online
47. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M (2011) Édouard Duchesnay: Scikit-learn: machine learning in Python. J Mach Learn Res 12:2825–2830
48. Pennington J, Socher R, Manning C (2014) GloVe: global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), ACL, Doha, Qatar, pp 1532–1543. https://doi.org/10.3115/v1/D14-1162
49. Qiang J, Li Y, Zhu Y, Yuan Y, Shi Y, Wu X (2021) LSBert: lexical simplification based on BERT. IEEE/ACM Trans Audio Speech Lang Process 29:3064–3076. https://doi.org/10.1109/taslp.2021.3111589
50. Radford A, Wu J, Child R, Luan D, Amodei D, Sutskever I (2019) Language models are unsupervised multitask learners. OpenAI Blog 1(8):9
51. Rajapakse T (2020) Simpletransformers. https://simpletransformers.ai/
52. Řehůřek R, Sojka P (2010) Software framework for topic modelling with large corpora. In: Proceedings of the LREC 2010 workshop on new challenges for NLP frameworks. ELRA, Valletta, Malta, pp 45–50
53. Säuberli A, Ebling S, Volk M (2020) Benchmarking data-driven automatic text simplification for German. In: Proceedings of the 1st workshop on tools and resources to empower people with reading dIfficulties (READI), pp 41–48. European language resources association. https://aclanthology.org/2020.readi-1.7

54. Sikka P, Singh M, Pink A, Mago V (2020) A survey on text simplification. arXiv preprint arXiv:2008.08612
55. Sjöblom E, Creutz M, Aulamo M (2018) Paraphrase detection on noisy subtitles in six languages. In: Proceedings of the 2018 EMNLP workshop W-NUT: the 4th workshop on noisy user-generated text, pp 64–73. https://doi.org/10.18653/v1/W18-6109
56. Smedt TD, Daelemans W (2012) Pattern for Python. J Mach Learn Res 13(66):2063–2067
57. Štajner S (2021) Automatic text simplification for social good: progress and challenges. Find Assoc Comput Linguist ACL-IJCNLP 2021:2637–2652
58. Štajner S, Glavaš G (2017) Leveraging event-based semantics for automated text simplification. Expert Syst Appl 82:383–395. https://doi.org/10.1016/j.eswa.2017.04.005
59. Stodden R, Kallmeyer L (2022) TS-ANNO: an annotation tool to build, annotate and evaluate text simplification corpora. In: Proceedings of the 60th annual meeting of the association for computational linguistics: system demonstrations, pp 145–155. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.acl-demo.14
60. Sulem E, Abend O, Rappoport A (2018) BLEU is not suitable for the evaluation of text simplification. In: Proceedings of the 2018 conference on empirical methods in natural language processing. Association for computational linguistics. https://doi.org/10.18653/v1/d18-1081
61. Surya S, Mishra A, Laha A, Jain P, Sankaranarayanan K (2019) Unsupervised neural text simplification. In: Proceedings of the 57th annual meeting of the association for computational linguistics, Association for computational linguistics, Florence, Italy, pp 2058–2068. https://doi.org/10.18653/v1/P19-1198
62. Wolf T, Debut L, Sanh V, Chaumond J, Delangue C, Moi A, Cistac P, Rault T, Louf R, Funtowicz M, Davison J, Shleifer S, von Platen P, Ma C, Jernite Y, Plu J, Xu C, Scao TL, Gugger S, Drame M, Lhoest Q, Rush A.M (2020) Transformers: state-of-the-art natural language processing. In: Conference on empirical methods in natural language processing. ACL
63. Xu W, Napoles C, Pavlick E, Chen Q, Callison-Burch C (2016) Optimizing statistical machine translation for text simplification. Trans Assoc Comput Linguist 4:401–415
64. Yatskar M, Pang B, Danescu-Niculescu-Mizil C, Lee L (2010) For the sake of simplicity: unsupervised extraction of lexical simplifications from Wikipedia. In: human language technologies: the 2010 annual conference of the North American chapter of the association for computational linguistics. Association for Computational Linguistics, Los Angeles, California, pp 365–368
65. Ye Q, Lin B.Y, Ren X (2021) CrossFit: a few-shot learning challenge for cross-task generalization in NLP. In: Proceedings of the 2021 conference on empirical methods in natural language processing, pp 7163–7189. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.emnlp-main.572
66. Zhang B, Choubey P.K, Huang R (2022) Predicting sentence deletions for text simplification using a functional discourse structure. In: Proceedings of the 60th annual meeting of the association for computational linguistics vol 2: Short Papers, pp 255–261. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.acl-short.28
67. Zhao Y, Chen L, Chen Z, Yu K (2020) Semi-supervised text simplification with back-translation and asymmetric denoising autoencoders. In: AAAI conference on artificial intelligence, pp 9668–9675. Association for the advancement of artificial intelligence (AAAI). https://doi.org/10.1609/aaai.v34i05.6515
68. Zhong Y, Jiang C, Xu W, Li J.J (2020) Discourse level factors for sentence deletion in text simplification. In: Proceedings of the AAAI conference on artificial intelligence, pp 9709–9716. Association for the advancement of artificial intelligence (AAAI). https://doi.org/10.1609/aaai.v34i05.6520