Ferrés D, Marimon M, Saggion H, AbuRa'ed A. YATS: Yet Another Text Simplifier. In: Métais E, Meziane F, Saraee M, Sugumaran V, Vadera S, editors. Natural Language Processing and Information Systems. 21st International Conference on Applications of Natural Language to Information Systems, NLDB 2016; 2016 Jun 22-24; Salford, UK. Cham (Switzerland): Springer; 2016. 335-42. (LNCS, no. 9612).DOI: 10.1007/978-3-319-41754-7_32

YATS: Yet Another Text Simplifier

No Author Given

No Institute Given

Abstract. We present a text simplifier for English that has been built with open source software and has both lexical and syntactic simplification capabilities. The lexical simplifier uses a vector space model approach to obtain the most appropriate sense of a given word in a given context and word frequency simplicity measures to rank synonyms. The syntactic simplifier uses linguistically-motivated rule-based syntactic analysis and generation techniques that rely on part-of-speech tags and syntactic dependency information. Experimental results show good performance of the lexical simplification component when compared to a hard-to-beat baseline, good syntactic simplification accuracy, and according to human assessment, improvements over the best reported results in the literature for a system with same architecture as YATS.

1 Introduction

Automatic text simplification is a research field which studies methods and techniques to simplify textual content. Text simplification methods should facilitate or at least speed up the adaptation of available and future textual material, making accessible information *for all* a reality. Text simplification has also been suggested as a potential pre-processing step for making texts easier to handle by generic text processors such as parsers, or to be used in specific information access tasks such as information extraction, summarization, or question answering. The interest in automatic text simplification has grown in recent years and in spite of the many approaches and techniques proposed, there is still an incredible space for improvement. The growing interest in text simplification is evidenced by the number of languages and users which are targeted by researchers around the globe. Simplification systems and simplification studies do exist for many languages: English [7], Brazilian Portuguese [1], and Spanish [21] just to name a few. In automatic text simplification, the algorithms can involve either or both lexical and syntactic simplifications. Lexical simplification is the task of identifying and substituting complex words for simpler ones in given contexts. Syntactic simplification is the task of reducing the grammatical complexity of a sentence while keeping the original information and meaning. An example involving both lexical and syntactic simplifications is given in (1), where the passive voice is changed into active and complex words are substituted for simpler synonyms.

(1) a. The poem was **composed** by the **renowned** artist.
b. The **famous** artist **wrote** the poem.

In this paper we present YATS, a text simplifier for English that has been built with the aim to improve text readability and understandability in order

to help people with intellectual disabilities. The system, however, is highly configurable and adaptable and the resources used can easily be changed to meet the needs of people with other conditions and special demands.[1] As we will show in this paper, our system achieves state-of-the-art performance in several evaluations.

After this introduction, Section 2 presents an overview of related work on automatic simplification, Section 3 describes our system, Sections 4 discusses the evaluation and the results we achieved and, finally, Section 5 concludes.

2 Related work

While automation of text simplifications has just recently become an established field of NLP, there is an active research in this field and several approaches have been proposed for lexical and syntactic automatic simplifications.

There are two main differing approaches in automatic syntactic simplification: those that use manually created rules and those that apply data-driven methodologies.

Early work on text simplification relied on hand-crafted rules to perform syntactic simplifications. Chandrasekar et al. [8] developed a linear pattern-matching hand-crafted rule system that simplified a few specific constructions, namely relative clauses, appositions, and coordinated clauses. In a second approach, Chandrasekar and Srinivas [9] induced simplification rules from a comparison of the structures of the chunked parses of an aligned corpus of sentences and their hand-simplified forms. Then, Siddharthan [23] described a rule-based system that peformed syntactic simplifications in three phases: (1) analysis, which used chunking and PoS tagging, followed by pattern-matching, (2) transformation, which applied a few rules for dis-embedding relative clauses, splitting conjoined clauses and making new sentences out of appositives, and (3) regeneration, which fixed mistakes by generating referring expressions, selecting determiners, and preserving discourse structure with the goal of improving cohesion of the simplified text. More recent hand-crafted systems made use of transfer rules that operated on the output of a parser; De Belder and Moens [13], for instance, used the phrasal parse tree produced by the Stanford parser, and Siddharthan [24] presented a framework based on applying transformation rules to a typed dependency representation produced by the Stanford parser. This is the approach that has been mostly followed in non-English projects [2, 15, 5].

Using aligned sentences of English Wikipedia and Simple English Wikipedia [10], other works have been from a data-diven prespective, mostly building upon methodologies (and evaluation measures) traditionally used in machine translation. Zhu et al. [30] learned a simplification model inspired by syntax-based SMT, consisting of a translation model, a language model, and a decoder, which was able to perform four rewriting operations, namely substitution, reordering, splitting and deletion. Coster and Kauchak [10] and Wubben et al. [28] viewed

[1] If the paper is accepted links to the resources developed will be made available with the final version of the paper.

simplification as a monolingual translation task and they applied phrase based MT to the text simplification augmented with a phrasal deletion model [10] and a post-hoc reranking procedure that ranked the output [28]. Siddharthan and Angrosh [25] presented a hybrid system that combined manually written synchronous grammars for syntactic simplificatins with an automatically acquired syncronous grammar for lexicalised constructs.

Work on lexical simplification began in the PSET project [14]. The authors used WordNet to identify synonyms and calculated their relative difficulty using Kucera-Francis frequencies in the Oxford Psycholinguistic Database. De Belder and Moens [13] combined this methodology with a latent words language model which modeled both language in terms of word sequences and the contextual meaning of words. Wikipedia has also been used in lexical simplification studies. Biran et al. [3] used word frequencies in English Wikipedia and Simple English Wikipedia to calculate their difficulty, and Yatskar et al. [29] used Simple English Wikipedia edit histories to idenfify the simplify operations of the form $x \rightarrow y$ extracted when a page has just been changed from x in a version to y in the next version of the page. [2]

3 The YATS System

The YATS system has been built with open source software and has both lexical and syntactic simplification capabilities. The lexical simplifier uses a vector space model approach to obtain the most appropriate sense of a given word in a given context (similar to [3] or [5]) and word frequency simplicity measures to rank synonyms [7]. The syntactic simplifier uses rule-based analysis and generation techniques that rely on PoS tags and dependency trees, which allows a broad coverage of common syntactic simplifications with a small hand-crafted rules.

3.1 Lexical Simplification in YATS

The lexical simplifier is composed of the following phases, executed sequentially: (i) Document analysis, (ii) Complex words detection, (iii) Word sense disambiguation, (iv) Synonyms ranking, and (v) Language realization.

The document analysis phase extracts the linguistic features from the documents. It uses the GATE[3] [12] NLP API and some processing resources from its ANNIE pipeline [11] to perform: tokenization, sentence splitting, part-of-speech (PoS) tagging, lemmatization, named entity recognition and classification, and co-reference resolution.

The automatic complex word detection is done using frequency thresholding over psycholinguistic resources. The procedure identifies a word as complex when

[2] The information extracted by using thes approach is avalable at http://www.cs.cornell.edu/home/llee/data/simple/ and can be used to extract the actual simplification rules.

[3] http://gate.ac.uk

the frequency count of the word in a given psycholinguistic database is in a range determined by two threshold values (i.e. w is complex if $min \leq w_{frequency} \leq max$). The two psycholinguistic resources that can be used separately in our lexical simplification system are: Age-of-Acquisition norms[4] [19] and Kucera-Francis [5] frequency counts [18]. The second resource is a set of frequency counts extracted from the Brown Corpus (1,014,000 words). This corpus consists of five hundred samples of about two thousand words each, which are assigned to fifteen categories or genres.

Since words may have more than one meaning, a Word Sense Disambiguation (WSD) phase is applied in order to select the most appropriate word replacement out of a list of "synonyms". The WSD algorithm used is based on the Vector Space Model [27] approach for lexical semantics which has been previously used in Lexical Simplification [3]. The WSD algorithm uses a word vector model derived from a large text collection from which a word vector for each word in WordNet-3.1[6] is created by collecting co-occurring word lemmas of the word in N-window contexts (only nouns, verbs, adjectives, and adverbs) together with their frequencies. Then, a common vector is computed for each of the word senses of a given target word (lemma and PoS). These word sense vectors are created by adding the vectors of all words (e.g. synonyms, hypernyms) in each sense in WordNet. When a complex word is detected the WSD algorithm computes the cosine distance between the context vector computed from the words of the complex word context (at sentence or document level) and the word vectors of each sense from the model. The word sense selected is the one with the lowest cosine distance (i.e. greater cosine value) between its word vector in the model and the context vector of the complex word in the sentence or document to simplify. Two data structures were produced following this procedure: 1) one that contains 81,242 target words and 135,769 entries, 2) another version that uses only synonyms to create the word sense vectors and has 63,649 target words and 87,792 entries. The plain text of the Simple English Wikipedia[7] (which had 99,943 documents in the dump we used[8]) has been extracted using the WikiExtractor[9] tool. The FreeLing 3.1[10] [20] NLP API has been used to analyze and extract the lemmas and the PoS tags. These lemmas were extracted from a 11-word window (5 lemmas are extracted to each side of the target words). Where synonym ranking is concerned, we rank synonyms in the selected sense by their simplicity and find the simplest and most appropriate synonym word for the given context [26]. The simplicity measure implemented is word frequency (i.e. more frequent is simpler) [7]. Several frequency lists were compiled for YATS, however for the experiments to be described here the Simple English Wikipedia

[4] http://crr.ugent.be/archives/806
[5] http://www.psych.rl.ac.uk/kf.wds
[6] http://wordnet.princeton.edu/
[7] http://simple.wikipedia.org
[8] simplewiki-20140204 dump version.
[9] http://medialab.di.unipi.it/wiki/Wikipedia_Extractor
[10] http://nlp.cs.upc.edu/freeling/

frequency list was used.
The final step, language realization, generates the correct inflected forms of the final selected synonym word substitutes in the contexts. The SimpleNLG [16] Java API is used to perform this task considering the context and the PoS tag of the complex word. The default lexicon of the SimpleNLG is currently used for the morphological realisation.

3.2 Intrinsic Evaluation of the YATS Lexical Simplification Component

Horn et al. [17] have produced a dataset for evaluation of lexical simplification system containing 500 examples. Each example contains a sentence and a target word, randomly sampled from alignments between sentences in pairs of articles from English Wikipedia and Simple English Wikipedia produced by [10]. Fifty Mechanical Turkers[11] provided simplifications (i.e. lexical substitutes) for each sentence in the dataset. Moreover, counts for the lexical substitutes proposed were obtained so as to produce a frequency-based rank for the set of replacements. One example of the evaluation dataset is shown below:

> **Sentence:** A haunted house is defined as a house that is believed to be a center for supernatural occurrences or paranormal phenomena.
> **Replacements:** events (24); happenings (12); activities (2); things (2); accidents (1); activity (1); acts (1); beings (1); event (1); happening (1); instances (1); times (1); situations (1)

The example shows: (i) a sentence where the *target* word to simplify is underlined (i.e. occurrences) and (ii) its possible replacements, together with the number of annotators selecting the replacement (e.g. the word *events* was chosen 24 times as simpler synonym for *occurrences*). We have carried out a hard intrinsic evaluation of the lexical simplification system using the above dataset. We have used the YATS lexical simplifier to select the most appropriate and simpler synonym of each target word in the dataset. Note that this is not a real application scenario of our lexical simplifier, since we already know which target word to simplify and therefore the task is somehow simpler. As a baseline for comparison purposes we have decided to use two approaches: (i) the system proposed by [6] and replicated in [22] which simply selects the most frequent synonym of the target word ignoring the possible polysemy of the target word (i.e. no WSD), and (ii) the rules induced by the system proposed by [29] which are freely available (see Section 2). The frequency-based baseline uses the same resources as YATS, that is, WordNet for finding synonyms, and same file for lemma frequencies used in YATS.

In order to carry out the evaluation some transformations have to be applied to the dataset: (i) all replacements have to be lemmatized and counts merged for replacements with the same lemma (e.g. in the example above *activities* and *activity* have to be collapsed under *activity* with count $1 + 2 = 3$); (ii) for evaluating both YATS and the frequency-based baseline only replacements (i.e.

[11] https://www.mturk.com

System	Oracle	YATS	Frequency	Rules
Lex.Simp.Acc.	0.81	**0.21**	0.19	0.11

Table 1. Average lexical simplification accuracy of three systems and an oracle on the Horn et al.'s lexical simplification dataset [17].

lemmas) appearing as synonyms of the target word are considered (e.g. in the example above *occurrence* has only 3 possible synonyms in our lexical resource: *happening, presence, and event*, therefore all other listed replacements will not be considered since they can not be produced by the considered system). We use *lexical simplification accuracy* as a metric defined for a sentence S, target word T, a set of weighthed replacements $Replacements(S,T)$ of T in S, and $Syno$: the synonym chosen by the system, as follows:

$$Lex\_Simp\_Acc(S,T,Syno) = \frac{\sum_{R \in Replacements(S,T)} Match(R, Syno, R_n)}{\sum_{R \in Replacements(S,T)} R_n}$$

where R_n is the number of annotators who have chosen R as replacement, $Match(R, Syno, R_n)$ is R_n if $R = Syno$ and 0 otherwise. That is, the system wins as many points as annotators have chosen $Syno$ as replacement. The denominator of the formula is a normalization factor which indicates how many annotators have provided replacements for the target word. As an example, if for the instance above a system selects "event" as a subtitute of "occurrences", then it will obtain 24 (=23+1)points. Given k pairs of sentences and target words $< S, T >$, the overall lexical simplification accuracy of a system will be its average lexical simplification accuracy. The results of evaluating the three systems: YATS, frequency-based baseline, and rules are shown in Table 1. We also include the maximum possible lexical simplification accuracy a system could obtain (i.e. an oracle), that is, the accuracy of a system which always selects the replacement which gives the maximum benefit.

Results indicate a good performance of the YATS system, overtaking both a hard to beat baseline and rule-based system which rules were induced from corpora. Morover, YATS was able to provide valid replacements for 146 sentences while the frequency-based baseline provided 141 replacements and the rule-based system only 71 (half YATS's coverage).

3.3 Syntactic Simplification in YATS

In syntactic simplification, complicated structures are detected in the sentence and replaced by other linguistic constructions that are easier to read. YATS simplifies the following syntactic structures: appositive phrases, adverbial clauses, coordinated clauses, coordinated correlatives, passive constructions, relative clauses, and subordinated clauses. The syntactic simplification is organized as a two-phase process: *document analysis*, which identifies the syntactic structures to be simplified, and *sentence generation*, which produces the simplified structures.

The document analysis phase uses three main resources to identify complex syntactic structures: (i) the GATE/ANNIE analysis pipeline used for lexical simplification; this step performs tokenization, sentence splitting and NE recognition, (ii) the Mate Tools dependency parser [4], which adds a dependency labels to sentence tokens, and (iii) a set of GATE JAPE (Java Annotation Patterns Engine) grammars which detect and label the different kind of syntactic phenomena appearing in the sentences.
Rules were manually developed in an iterative process by using dependency-parsed sentences from Wikipedia which were indexed using the ANNIC system available in GATE. The process resulted in a set of JAPE rules able to recognize and analyze the different kinds of syntactic phenomena appearing in the sentences (those stated above). Each rule is composed of a left-hand-side (LHS) and a right-hand-side (RHS). The LHS of the rules consist of an annotation pattern description while the RHS consists of annotation manipulation statements. Annotations matched on the LHS of a rule may be referred to on the RHS by means of labels that are attached to pattern elements. These rules rely on dependency trees, which allows a broad coverage of common syntactic simplifications with a small hand-crafted rules. Given the complex problem at hand, it is not enough to perform pattern matching and annotation of the matched elements, also the different annotations matched instantiating the patterm have to be properly annotated and related to each other. This process is carried with Java code in RHS of each rule. JAPE rules can be organized in grammars which when compiled can be used to perform transduction over annotations based on regular expressions. Each syntactic phenomena dealt with in the system has a dedicated grammar (i.e., set of rules). These grammars are organized in a main file which specifies the order in which the grammars have to be applied to the text. The complete rule-based system is composed of: one rule for appositive constructions, 17 rules for relative clause identification, 10 rules for coordination of sentences and verb phrases, 4 rules for coordinated correlatives, 8 rules for subordination (concession, cause, etc.), 12 rules for adverbial clauses, and 14 rules for passive constructions. Figure 1 shows one of the 17 JAPE rules dealing with relative clauses. Only the regular pattern (LHS) is shown which will match a dependency-parsed sentence.

The result of the rule application can be appreciated in Figure 2 (the GATE Graphical User Interface) where the relative clause has been identified together with the antecedent(*Eva*) of the relative pronoun (*whose*). Such rule together with the Java text-to-text generation programs would produce the simplification *Eva is the daughter of Augustine St. Clare. Eva' s real name is Evangeline St. Clare.* The following simplification priority order is applied when several syntactic phenomena of these types are detected in the sentence: Apposition - Relative Clauses - Coordination - Coordinated Correlatives - Passive Constructions - Adverbial Clauses - Subordinated Clauses. The system recursively simplifies sentences until no more simplifications can be applied.

The *sentence generation phase* uses the information provided by the analysis stage to generate simple structures. It applies a set of rules which are specific for

```
Rule: NonRestrRC_SbjPossWh
( ({!Token.category ==~ "(WDT|WP)"})*
  ({Token}):antecedent_end
):antecedent
(
 ({Token.string == ","}):rc_start
 (({Token.lemma == "whose"}):rprn
  ( (({Token.category == "RB"})?
{Token.category == "VBN", Token.func == "NMOD"})
   | (({Token.category == "RB"} | {Token.category == "JJS"})?
 {Token.category == "JJ", Token.func == "NMOD"})
   | (({Token.category == "JJ"})? {Token.category == "NN", Token.func == "NMOD"})
  )?
  {Token.func == "SBJ"})
 (({Token.string == ","})?({Token.category == "RB"}|(({Token.category == "IN"} |
 {Token.category == "TO"})({Token})+))?({Token.string == ","})?)
 ({Token.category ==~ "VB([DPZ])?|MD", Token.func == "NMOD"}):rc_hd
):rc
-->
{
// Java Code
....
}
```

Fig. 1. JAPE regular pattern over dependencies to identify a non-restrictive relative clause with possesive subject.

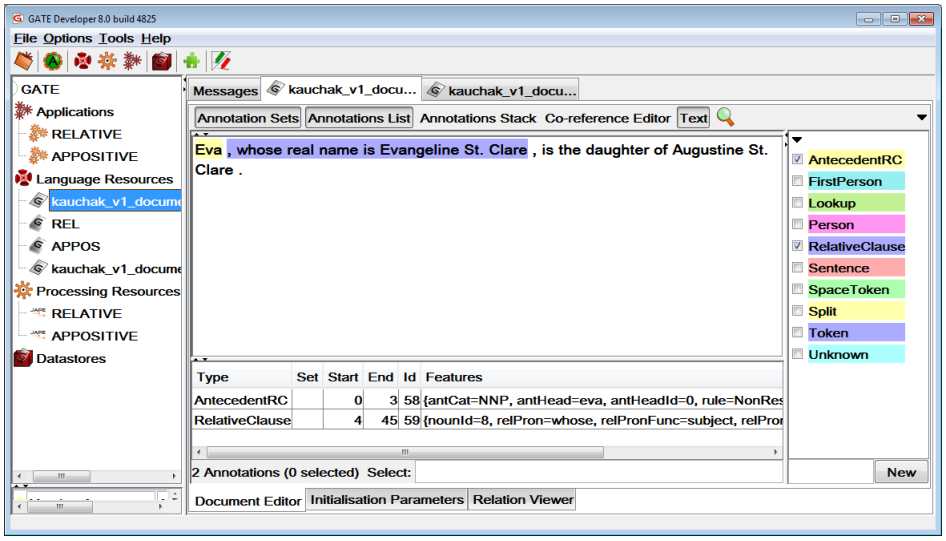

Fig. 2. JAPE relative clause grammar applied to the sentence *Eva, whose real name is Evangeline St. Clare, is the daughter of Augustine St. Clare.*

each phenomenon. These rules rules perform the common simplification operations, namely sentence splitting, reordering of words or phrases, word susbtitution, verbal tense adaption, personal pronouns transformation, and capitalization and de-capitalization of some words.

Grammar	*num.sents*	*fired*	*right*	*wrong*	*ignored*
Appositions	100	100%	79%	21%	0%
RCs	100	93%	79%	14%	7%
Coordination	100	62%	56%	6%	38%
Subordination	100	97%	72%	25%	3%
Passives	100	91%	85%	6%	9%
Total	500	89%	74.2%	14.4%	11.4%

Table 2. Evaluation of the JAPE grammars. The first column lists the syntactic phenomena, the second column indicates the number of sentences used which contained the sought syntactic phenomena, the third column indicates the number of times the grammar fired, the fourth column indicates the precision of the rule, the fifh column is the percent of wrong rule applications, and finally the last column is the percent of times the grammar did not fire.

System	Fluency	Adequacy	Simplicity
SW (Human)	4.58	3.76	3.93
S&A (Automatic)	3.73	3.70	**2.86**
YATS (Automatic)	**3.98**	**4.02**	**2.86**

Table 3. Average results of the human evaluation with eight human judges.

3.4 Intrinsic Evaluation of the YATS Syntactic Simplification Component

We have carried out an intrinsic evaluation of the rule-based systems in terms of precision. We have collected 100 sentence examples (not used for system development) per syntactic phenomena we target. Each grammar was then applied to the set of sentences the grammar was covering so as to analyze the performance of the rules on unseen examples. Results are presented on Table 2. In the Table, *right* means that the rule has recognized and annotated all necessary information for generation, *wrong* means that some the annotations required or part only part of them have not been produced, and *ignored* means that the rules were not fired for the sentece (i.e. a miss).

Most rules are rather precise, except perhaps those dealing with coordination which is a very difficult phenomena to recognize given its ambiguity. An analysis of the errors showed us that the JAPE rules produced errors due to the lack of coverage of certain structures, e.g. coordination of anchors, coordination of antecedents, or coordination of main verbs taking a subordinated clause. Some of the errors produced by the dependency parser were: the parser assigned a wrong PoS tag, the parser assigned a wrong function (e.g. the anchor was identified with a wrong token, the relative pronoun got a wrong function) or a wrong dependency (e.g. the subordinated clause depended on a wrong head, the parser did not identify all the dependents of a head or it analysed as dependents tokens which were not), the parser analysed a syntactic structure wrongly (e.g. the second conjunct in coordinated NPs was analysed as an apposition).

4 Human Evaluation of YATS

Current evaluation of automatic text simplification reported in the literature tend to be based on human judgments, specially when the system is targeted to specific populations. We performed manual evaluation relying on eight human judges,[12] who assessed our system w.r.t. fluency, adequacy, and simplicity, using the evaluation set used by Siddharthan and Angrosh [25], from which we randomly selected 25 sentences. Participants were presented with the source sentence from the English Wikipedia followed by three simplified versions from Simple Wikipedia (SW) (i.e. a sentence from Simple Wikipedia aligned to the sentence in English Wikipedia), the system developed by Siddharthan and Angrosh [25] (S&A), and YATS, in a randomisd order, and they were asked to rate each of the simplified version w.r.t. the extend to which it was grammatical (fluency), the extend to which it had the same meaning as the complex sentence (adequacy), and the extend to which it was simpler than the complex and thus easier to understand (simplicity). We used a five point rating scale where high numbers indicated better performance.

Table 3 shows the results for the complete data set. As can be observed, our system achieved the same mean score for simplicity as Siddharthan and Angrosh's [25], and is slightly better at fluency and adequacy, though not statistically significant. Both automatic systems are comparable to the Simple Wikipedia version.

5 Conclusion

Automatic text simplification is a key enabler in making texts more accessible in the information society. Here, we have presented YATS, a text simplifier for English with lexical and syntactic simplification capabilities. The lexical simplifier uses a vector space model approach to obtain the most appropriate sense of a given word in a given context and word frequency simplicity measures to rank synonyms. We have shown that our lexical simplifier outperforms a hard-to-beat baseline procedure based on frequency and a rule-based system highly cited in the literature. The syntactic simplifier, which is linguistically motivated and based on peer-reviewed work, uses rule-based syntactic analysis and generation techniques that rely on part-of-speech tags and dependency trees. Experimental results of human assessment of the system output showed improvements over the best reported results in the literature. Future research includes experiments to better assest the performance of the system (e.g, lexical simplification in other available datasets), improve the coverage of the syntactic simplifier by re-training the parser, and extending the scope of the lexical simplifier relying on more advanced vertor representations (e.g. embeddings). Finally we intend to port our systems to other languages such as Spanish.

[12] All of them were proficient in English. None of them had participated in the development of the simplifier.

References

1. Aluísio, S.M., Gasperin, C.: Fostering Digital Inclusion and Accessibility: The PorSimples Project for Simplification of Portuguese Texts. In: Proceedings of the NAACL HLT 2010 Young Investigators Workshop on Computational Approaches to Languages of the Americas. pp. 46–53. YIWCALA '10, Association for Computational Linguistics, Stroudsburg, PA, USA (2010)
2. Barlacchi, G., Tonelli, S.: Ernesta: A sentence simplification tool for childrens stories in Italian. In: A. Gelbukh (Ed.) Computational Linguistics and Intelligent Text Processing, series Lecture Notes in Computer Science, vol. 7817. pp. 476–487. Springer Berlin Heidelberg (2013)
3. Biran, O., Brody, S., Elhadad, N.: Putting it simply: A context-aware approach to lexical simplification. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2. pp. 496–501. HLT '11, Association for Computational Linguistics, Stroudsburg, PA, USA (2011)
4. Bohnet, B.: Very High Accuracy and Fast Dependency Parsing Is Not a Contradiction. In: Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010). pp. 89–97. Stroudsburg, PA, USA (2010)
5. Bott, S., Rello, L., Drndarevic, B., Saggion, H.: Can Spanish Be Simpler? LexSiS: Lexical Simplification for Spanish. In: Proceedings of 24th International Conference on Computational Linguistics (COLING 2012). pp. 357–374. Mumbai, India (2012)
6. Carroll, J., Minnen, G., Canning, Y., Devlin, S., Tait, J.: Practical simplification of English newspaper text to assist aphasic readers. In: Proceedings of the AAAI98 Workshop on Integrating AI and Assistive Technology. pp. 7–10 (1998)
7. Carroll, J., Minnen, G.M., Pearce, D., Canning, Y., Devlin, S., Tait, J.: Simplifying Text for Language-Impaired Readers. In: Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics (EACL 1999). pp. 269–270. Bergen, Norway (1999)
8. Chandrasekar, R., Doran, C., Srinivas, B.: Motivations and methods for text simplification. In: Proceedings of the 16th International conference on Computational linguistics (COLING 1996). pp. 1041–1044 (1996)
9. Chandrasekar, R., Srinivas, B.: Automatic induction of rules for text simplification. Knowledge-Based Systems 10, 183–190 (1997)
10. Coster, W., Kauchak, D.: Learning to simplify sentences using Wikipedia. In: Proceedings of the Workshop on Monolingual Text-To.Text Generation, 49th Annual Meeting of the Association for Computational Linguistics. pp. 1–9. Portland, Oregon, USA (2011)
11. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In: Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL 2002). Philadelphia, PA, USA (2002)
12. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V., Aswani, N., Roberts, I., Gorrell, G., Funk, A., Roberts, A., Damljanovic, D., Heitz, T., Greenwood, M.A., Saggion, H., Petrak, J., Li, Y., Peters, W.: Text Processing with GATE (Version 6) (2011)
13. De Belder, J., Moens, M.F.: Text simplification for children. In: Proceedings of the SIGIR Workshop on Accessible Search Systems, SIGIR workshop on accessible search systems, Geneva. pp. 19–26 (2010)

14. Devlin, S., Tait, J.: The Use of a Psycholinguistic Database in the Simplification of Text for Aphasic Readers. In: Linguistic Databases. pp. 161–173 (1998)
15. Gasperin, C., Maziero, E., Alusio, S.M.: Challenging choices for text simplification. In: T.A.S. Pardo et al. (Rds.) PROPOR 2010 LNAI 6001. pp. pages 40–50. Springer Berlin Heidelberg (2010)
16. Gatt, A., Reiter, E.: SimpleNLG: A Realisation Engine for Practical Applications. In: Proceedings of the 12th European Workshop on Natural Language Generation. pp. 90–93. ENLG '09, Association for Computational Linguistics, Stroudsburg, PA, USA (2009)
17. Horn, C., Manduca, C., Kauchak, D.: Learning a lexical simplifier using Wikipedia. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference. pp. 458–463 (2014)
18. Kucera, H., Francis, W.N.: Computational analysis of present-day American English. Brown University Press, Providence, RI (1967)
19. Kuperman, V., Stadthagen-Gonzalez, H., Brysbaert, M.: Age-of-acquisition ratings for 30,000 English words. Behavior Research Methods 44(4), 978–990 (2012)
20. Padró, L., Stanilovsky, E.: FreeLing 3.0: Towards Wider Multilinguality. In: Proceedings of the Language Resources and Evaluation Conference (LREC 2012). Istanbul, Turkey (2012)
21. Saggion, H., Gómez-Martínez, E., Etayo, E., Anula, A., Bourg, L.: Text simplification in Simplext: Making text more accessible. Revista de la Sociedad Española para el Procesamiento del Lenguaje Natural 47 (2011)
22. Shardlow, M.: Out in the open: Finding and categorising errors in the lexical simplification pipeline. In: Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14). Reykjavik, Iceland (may 2014)
23. Siddharthan, A.: Syntactic simplification and text cohesion. In: Proceedings of Language Engineering Conference (LEC 2002). pp. 64–71 (2002)
24. Siddharthan, A.: Text Simplification using Typed Dependencies: A Comparision of the Robustness of Different Generation Strategies. In: Proceedings of the 13th European Workshop on Natural Language Generation. 13th European Workshop on Natural Language Generation. Nancy, France (2011)
25. Siddharthan, A., Angrosh, M.: Hybrid text simplification using synchronous dependency grammars with hand-written and automatically harvested rules. In: Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2014). Gothenburg, Sweden (2014)
26. Specia, L., Jauhar, S.K., Mihalcea, R.: SemEval-2012 Task 1: English Lexical Simplification. In: First Joint Conference on Lexical and Computational Semantics. pp. 347–355. *SEM 2012, Montréal, Canada (2012)
27. Turney, P.D., Pantel, P.: From frequency to meaning: Vector space models of semantics. J. Artif. Int. Res. 37(1), 141–188 (2010)
28. Wubben, S., van den Bosch, A., Krahmer, E.: Sentence Simplification by Monolingual Machine Translation. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012): Long Papers - Volume 1. pp. 1015–1024. Jeju Island, Korea (2012)
29. Yatskar, M., Pang, B., Danescu-Niculescu-Mizil, C., Lee, L.: 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 (2010)
30. Zhu, Z., Bernhard, D., Gurevych, I.: A Monolingual Tree-based Translation Model for Sentence Simplification. In: Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010). pp. 1353–1361. Beijing, China (2010)