

Wikipedia Simplifier

Stanford CS224N Custom Project, Option 3 (graded)

Daniel Kharitonov

Department of Computer Science

Stanford University

dkharito@stanford.edu

Abstract

This project examines the problem of synthesizing articles for Simple English Wikipedia. The latter is a project extending the power of Wikipedia to kids, students of foreign languages, and people with learning disabilities by means of using simplified vocabulary, punctuation, and sentence structures. The task of converting Wikipedia articles into Simple English is hard because it falls in the gap between translation and summarization: articles in Simple English are neither the shortened versions of main Wikipedia, nor they maintain the one-on-one sentence correspondence needed to train a translation system. To tackle this challenge, we employ a transformer-based text generation model directly constrained for the output embeddings, and demonstrate that it can be effective in simplifying the English text conditioned on context coverage acquired in training.

1 Introduction

Simple English Wikipedia ¹ strives to employ a vocabulary of just 1,000 most common words and reach the readability Flesch score over 70 (Fairly Easy). An automated translation from one readability level into another is generally difficult because the available training set (existing Simple Wikipedia articles) are neither summaries, nor close copies of their Wikipedia counterparts.

Generally speaking, for purposes of transforming main Wikipedia articles into Simple English, we would like to construct a system that can separate language attributes (choice of words, sentence structures, and paragraph formatting) from the content (Wikipedia statements). Unlike still images where content (lines and shapes) is easily separable from style (colors, strokes and spatial transformations), natural language does not exhibit an obvious style-content separability. We can discern four known approaches to deal with this issue in the context of our task and the known NLP techniques.

First, a pretraining step (sentence realignment) can be employed to reduce transformation task to translation [1]. An advantage of this method (contingent on realignment) is that a broad array of automated translation architectures can be applied to convert Wikipedia articles into Simple English. The main downside lies in the difficulty of finding the proper sentence pairs: common simplification techniques of splitting complex sentences or rephrasing the content can easily confuse the realignment processor.

Secondly, articles in Simple English can be seen as a special case of content summarization. Summarization is a well-researched topic with many available implementations, which mostly rely on extracted key-phrases, e.g. [2]. Unfortunately, Simple English

¹<http://simple.wikipedia.org>

Wikipedia is not intended to be an abridged version of main Wikipedia, and does not rely on compression of knowledge. Moreover, abstractive summarization draws from the same vocabulary as the main article, which means that even a well-formed summary does not always produce an improved readability score.

Next, there is active work on reductionist transformation techniques, such as delete-retrieve-generate (DRG) [3]. Being the opposite of abstractive summarization methods, DRG workflow intends to identify key phrases responsive to changes and perform interventions on them. In simplification context, this might help with atonement of uncommon words and phrases. However, DRG does not alter the sentence structures and effectively operates on the n-grams. This limitation is critical for Simple English, where average sentence length is an important factor in readability.

Finally, a growing number of publications take the holistic approach to language transformations and rely on pretrained generators to produce samples in target style. While training the generator itself is typically easy, the main difficulty with such methods lies in controlling the flow of output. Several architectures exist to control the output *styles* (e.g. [4]), but there appears to be no general solution for controlling the *content* of output. We intend to fill this gap and provide an architecture that can achieve exactly that.

2 Our Approach

In this work we use an original approach of repeated sampling from a pretrained generator constrained for the content. The main idea is to exploit the fact that a text generation model can produce samples aligned with the seed. This seed ‘activates’ associations acquired in training, and repeated sampling can produce a wide variety of responses in the target style (Simple English). A pipeline comparing these samples to target content in meaning can reject samples that fail to match the embedding score of the source, effectively manipulating the output by means of a rejection-pass filter (RPF).

Formally, our architecture relies on ability of generative models to produce samples from the training dataset with style z and content c via latent generative distribution:

$$\hat{x} \sim G(z, c) = \prod p(x_t | x_{t-1..0}, c)$$

With a generative model, we can formulate the problem of style transformation for source lexeme s as finding such sequence x_t that the score difference norm $\|e(s) - e(x_t | x_{t-1..0}, c)\|_1 \leq \tau$. More precisely, since a trained generative model already includes style z , we can control the generator output simply by means of repeated sampling admitted on the condition of satisfying a similarity threshold τ (a hyper-parameter).

This process of threshold-based filtering is recurring and self-adjusting: for as long as the generative model produces the output consistent with embeddings of input lexemes, samples will be admitted; a failure to match the threshold forces the output filter to accept a lexeme from the donor text unchanged (identity transformation) which resets the context. This way, the growing vector distances between the embedding vectors of content in donor text and generator output are periodically realigned.

3 Evaluation criteria, baseline, and a worked example

Given our aspiration to supplement human-written Simple English articles by automation, we propose employing the readability score of the former as a baseline. According to [5], the average Flesch score of current Simple English Wikipedia is 61.46, which is markedly above the main Wikipedia, where the average score is only 51.8. While the Flesch-Kincaid formula is not perfect as it only accounts for the average length of sentences and words while ignoring the actual frequency distribution over the entire

vocabulary, it also indirectly culls rare words (that tend to be long) and is widely used as a measure of readability.

Let us draw an example to demonstrate how our architecture works. Consider the following Wikipedia fragment:

Cats are valued by humans for companionship and for their ability to hunt rodents.

In this sentence we observe two logically separable lexemes. Once the parser identifies the first lexeme *"Cats are valued by humans for companionship"*, it samples potential replacements from the text generator trained on the Simple English corpus. Let us imagine the best sample is **"There are many different types of domestic cats"**, with content similarity score (inner product of normalized phrase embeddings) of 0.8354. If we consider this score below the threshold τ , we reject the sample and take the original lexeme unchanged (no replacement provided). In the next step we use the first lexeme as a seed to generate samples for replacement of the second: *"for their ability to hunt rodents"*. Let us assume a text generator produces a candidate **"to hunt rodents"**. This phrase has a similarity score of 0.9332 with a second lexeme, and assuming it clears the threshold we can now accept the sample, so the fully transformed sentence becomes:

Cats are valued by humans for companionship and to hunt rodents.

After this transformation, we employ the resulting text as a new seed, and continue to process the first lexeme of the next sentence in the article.

Note the original sentence had a Flesch score of 57.27, while the transformed passage has a score of 68.77, an almost ten-point improvement in ease of reading. This improvement is a result of deleting the preposition ("for their ability"), which was not critical to the content of article. Therefore, the change proposed by the system is reasonable and fits our goal of language simplification.

4 Framework Architecture

We are using a GPT-2 transformer-based text generation model [6] and a Google Sentence Encoder [7] as the building blocks of the architecture. The pipelining layer that takes the source content, parses it into lexemes, feeds into the generator, and applies the rejection-pass filter is all-original (see Fig. 1)

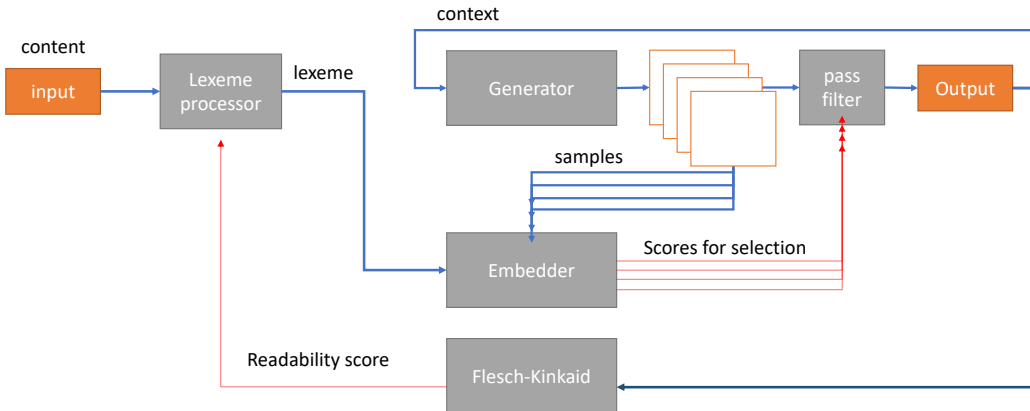


Figure 1: The architecture of a rejection-pass filter (RPF)

We assume the requisite hyperparameters to be adjusted in a separate feedback loop after a cycle of output evaluation is complete. These hyperparameters define the work

of the generators and lexeme processor and can be critical to the final output quality. The optimal hyperparameters acquired during validation run are shown in Table 1.

Table 1: RPF Hyperparameters

τ	min	minsoft	max	seed	nsamples	t°
0.88	3	7	16	10	10,000	0.92

Here, τ is a rejection threshold (minimum acceptable similarity), $\{MIN, MINSOFT, MAX\}$ are the unconditional minimum, punctuation-marked minimum, and the maximum lexeme size in words respectively, *SEED* is the number of lexemes provided as context, *NSAMPLES* stands for the number of candidates produced by the generator per each lexeme, and t° is the generation temperature.

The algorithm implemented by our RPF architecture works as follows:

Algorithm 1: RPF

Input: content source s
Output: simplified text $st = ""$
repeat
 Identify next lexeme $l = process(s)$
 for $i = 1$ **to** $nsamples$ **do**
 Generate next sample $x_i = generate(st)$
 Save embedding score $e(x_i)$
 end for
 if $max(|e(x_i) - e(l)|) > \tau$ **then**
 $st = argmax(|e(x_i) - e(l)|) + st$
 else
 $st = l + st$
 end if
until source s is done

5 Dataset and Implementation details

We used the existing Simple Wikipedia as our data corpus, scraping the articles and cleaning them of formatting, references and XML artifacts. We fine-tuned our generator on 75K articles (55MB of text) which is one-half of the corpus (the other half for searching the hyperparameter space and the final testing). Since GPT-2 comes pretrained for the general language features, fine-tuning stage was relatively short (about 8,000 steps).

Our lexeme processor is a heuristics engine implemented in python. It splits the incoming sentence into lexemes preferring terminal punctuation or semicolons, and falling back to other punctuation markers (comma, em dash, etc) if no terminal boundaries are found within a reasonable span. The configurable hyperparameters (Table 1) provide the compromise between the content preservation and simplification: if we make the lexemes too short, they cannot be effectively simplified. If we make lexemes too long, the generator will have trouble producing samples that will match them well. Our rejection filter is a simple gate driven by distance threshold τ . Empirically, we observed that Wikipedia simplification works best when this threshold is rather conservative, and the generator operates at a lowered temperature.

6 Experiments

In the testing part of the project, the RPF framework was applied to twenty Wikipedia articles with their Simple English counterparts unseen in training. In total, 556 lexemes were processed and 210 replacements were found (37.7% transformation rate). The average improvement in readability score was +4.712, with standard deviation 3.32. Lexeme replacement rates along with readability changes in the Flesch-Kinkaid score for select Wikipedia articles are shown in Table 2.

Table 2: Experimental Results in Select Wikipedia Articles

<i>article</i>	"AI"	"Bible"	"Christ"	"Cat"	"Mouse"	"Omelette"
replaced	6/27	12/37	24/54	63/256	11/31	5/31
Flesch	24.78/12.25	59.43/54.86	58.61/48.84	73.68/66.47	64.51/59.53	76.11/74.93
change	+12.53	+4.57	+9.77	+7.21	+4.98	+1.18

This table merits some discussion.

The first observation is that lexeme replacement rates can vary based on the article. In our experiments, we saw the rates as low as 3% and as high as 80%. These rates seem to be driven by the amount of material available for training (Simple Wikipedia corpus) adjacent to the article chosen for transformation. For a specific example, the lowest replacement rates are seen in the "Personal" category that describe biographies. Since there is little training material that would fit such context well, replacement rates tend to remain low. By way of contrast, the highest replacement rates are seen when there is rich and fertile context to draw from – such as in articles on popular religions or cultural phenomena.

We can also observe that changes in readability scores are invariably positive, with a qualifier that such a change might be trivial or achieved with means simpler than a neural net. As an example, a Wikipedia article on AI includes the following passage *"In the twenty-first century, AI techniques have experienced a resurgence following concurrent advances in computer power, large amounts of data, and theoretical understanding; and AI techniques have become an essential part of the technology industry, helping to solve many challenging problems in computer science, software engineering and operations research."*

It is not difficult to split this sentence into several pieces mechanically to improve its readability. Such a trivial change can lead to large differences in the Flesch-Kinkaid score. A more complicated intervention would involve a restructuring of the whole phrase.

For instance, consider this Wikipedia excerpt:

Heracles' very existence proved at least one of Zeus' many illicit affairs, and Hera often conspired against Zeus' mortal offspring as revenge for her husband's infidelities.

Our RPF algorithm provides replacement for this sentence in the following form:

Heracles' very existence proved at least one of Zeus' many affairs, and Hera was crushed by the wrath when she had harboured this offspring of Zeus.

The latter text certainly reads easier, yet the main reason for simplification is not a shortened sentence, but the shorter (and thus more common) words. This would be a cherry-picked example of an intervention that other methods (like DRG) would not be able to achieve.

In our experiments, we observed RPF to produce a mixture of simple and complex interventions. Not every replacement results in a coherent modification, and we will discuss some artifacts in the next section.

7 Error analysis

Insofar we have only focused on the formal features of transformed text, namely the average sentence and word length that are the two factors in the Flesch-Kincaid readability score. These factors alone cannot adequately describe the transformation because they do not estimate the qualitative language features – preservation of content and text fluidity (conformance with rules of grammar, syntax, and sentence structuring). In this section, we will focus on accepted replacements and describe some issues seen after processing.

7.1 Phrase encoder errors

This project employs Google Sentence Encoder for calculation of candidate acceptance scores. This is a state-of-the-art encoder, but sometimes it fails to indicate a mismatch.

For example, consider the following two phrase pairs with RPF output shown in bold:

Development of cat breeds started in the mid 19th century.
The domestication of cats started in the late 19th century.

Muslims believe Jesus was born of a virgin.
The Christians believe that Jesus was born of a virgin.

According to Google Encoder, the sentences within pairs are very similar (score 0.8867 and 0.9015), yet they communicate the entirely different meaning. The first output suggests that cats were wild prior to the 19th century – which is not true. The second statement is true but shifts the context from Muslims to Christians, which leads to more transformation errors downstream. To combat such blunders we can raise the acceptance threshold τ , but this may also lower the effective replacement rates.

7.2 Suboptimal treatment of name entities and rare words

In the following sentence pairs, the framework encounters names and words that were not in the training set. This prompts RPF to produce credible-looking replacements with some "hard to spot" errors. For the first pair, it takes a good knowledge of biology to reject the notion that a mouse can be an ape in a fictitious genus. In a second example, an even harder mental effort is required to realize that "catlóg" is not Old English:

The best known mouse species is the common house mouse ("Mus musculus").
Mouse is one of the species in the genus "Cyanopithecus".

The origin of the English word 'cat', Old English catt, is thought to be the Late Latin word cattus, which was first used at the beginning of the 6th cent.

The origin of the English word 'cat' derives from the Latin "catum", and the Old English "catlóg", which was first used in the 7th century.

In all these cases the network does a good (maybe even "too good") effort to comply with a seemingly impossible requirement to produce words not seen in training.

7.3 Oversimplification

It seems like some of the Simple English constructs picked from the training set appear to be "too simple". Consider the following sentence pairs:

Cats are known for spending considerable amounts of time licking their coats to keep them clean.

Cats are also known to lick their fur, which helps them clean them.

The second sentence is an effective reduction, but also involves a colloquialism "helps them clean them" that would not pass copyediting. This example highlights the fact that RPF may pick unwanted language biases from the training set.

7.4 Non-verbatim quotes

Another interesting artifact is seen when rendering quotes. Consider this sentence pair:

According to the March 2007 edition of Time, the Bible "has done more to shape literature, history, entertainment, and culture than any book ever written".

According to the March 2007 edition of Time, the Bible has become a "remarkably influential and enduring" book.

The RPF was successful in rephrasing the original, but it also altered the quote. The preservation of verbatim quotes is not learned in training, so this artifact highlights one gap in our framework.

8 Future work

There are several limitations to the proposed framework that came up during experimentation. First and foremost, it appears the training set *must* have sufficient context for the model to be effective. Attempts to generate lexemes in unknown contexts result in the low replacement rates. This can be partially alleviated by doing a second pass over the generated text going in the opposite direction, which would require a second Simple English generation model trained on the sentences read right-to-left.

Second, our Wikipedia simplifier seems to have hard time dealing with Name Entities, rare words and their coreferences. This can be partially alleviated with a better training method that penalizes dropped Name Entities and hanging coreferences.

Finally, finding a matching sample requires a large number of trials and slows the generation process. Our analysis was effectively limited by the amount of time we could spend digesting the Wikipedia articles, where an average lexeme processing time is on the order of a minute. A brute-force solution to this problem would require a faster GPU, but a more intelligent solution might require stopping sampling early if there are enough high-quality candidates.

Code

The code for this project is available at <https://github.com/volkfox/224n-final/>

References

- [1] William Coster and David Kauchak. Simple english wikipedia: A new text simplification task. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, page 665–669, USA, 2011. Association for Computational Linguistics.
- [2] Tian Shi, Yaser Keneshloo, Naren Ramakrishnan, and Chandan K. Reddy. Neural abstractive text summarization with sequence-to-sequence models. *CoRR*, abs/1812.02303, 2018.
- [3] Akhilesh Sudhakar, Bhargav Upadhyay, and Arjun Maheswaran. Transforming delete, retrieve, generate approach for controlled text style transfer, 2019.
- [4] Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation, 2019.
- [5] Anderson Kate. Wikipedia’s writing — tests show it’s too sophisticated for its audience. *The Scholarly Kitchen*, 2012.
- [6] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2006.
- [7] D. Cer, Y. Yang, S. Kong, N. Hua, N. Limtiaco, R. St. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, Y. Sung, B. Strope, and R. Kurzweil. Universal sentence encoder. *CoRR*, abs/1803.11175, 2018.