WIKIPEDIA SIMPLIFIER (SUMMARY)

1. INTRODUCTION:

The simple English wikipedia is an English language version of wikipedia. An automated translation from one readability level into another is generally difficult. Transforming main wikipedia articles into simple English includes a system that can separate language attributes from the content. 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 can be employed to reduce transformation task to translation. Secondly, articles in simple English can be seen as a special case of Content Summarization. It does not rely on Compression of knowledge. Next, delete-retrieve-generate (DRG) technique. 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. Finally, a growing number of Publications take the holistic approach to language transformations and rely on pretrained generators to produce samples in target style

2. APPROACH

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 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).

Latent generative distribution with style z and content c:

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

Here we can formulate the problem of style transformation for source lexeme s as finding such sequence x_t that the score difference norm $\|e(s) - e(x_t | x_{t-1} \ldots 0, c)\|_1 \leq T$. More Precisely, since a trained generative model already includes style z, we can control generator output simply by means of repeated sampling admitted on the condition of satisfying a similarity threshold T. This process of threshold-based filtering is recurring and self-adjusting.

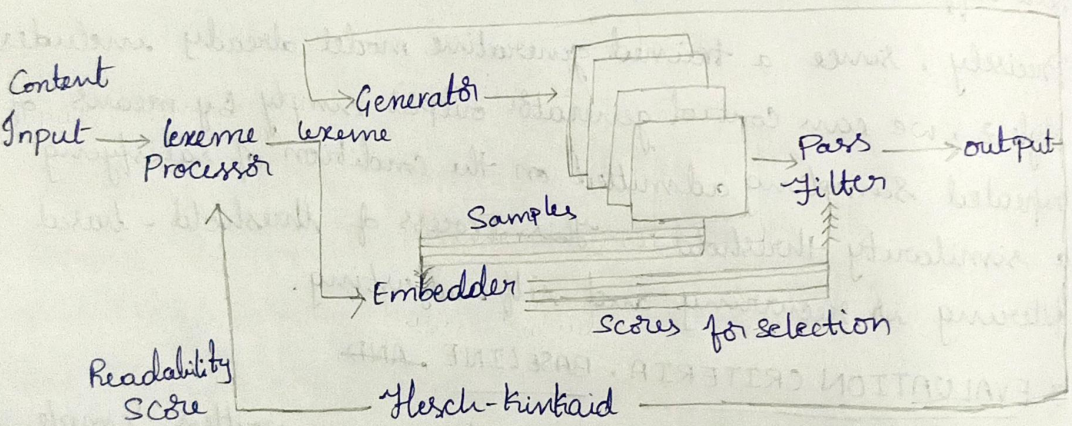
3. EVALUATION CRITERIA, BASELINE ~~AND~~

Given our aspiration to supplement human-written Simple English articles by automation, we propose employing the readability score of the former as a baseline. The Average Flesch Score of current simple English wikipedia is 61.46, which is more than the flesch score of main wikipedia. The Flesch-Kincaid formula is not perfect as it only accounts for average length of sentences and words while ignoring the actual frequency distribution over the entire vocabulary. Here lexemes of sentences are subjected to potential replacements from the text generator trained on the simple English corpus. After this, if the content similarity score is

Less than the threshold T, it will be rejected and original lexeme is taken. And in the next step we use the lexeme as a seed to generate samples for replacement of the other

4 FRAME WORK ARCHITECTURE

We are using a GPT-2 transformer-based text generation model and a Google Sentence Encoder as the building blocks of the architecture. The pipelining layer that takes the source content, Pareses it into lexemes, feeds into the generator, and applies the rejection-Pass filter is all-original.



The hyperparameters define the work of the generators and lexeme processor. and can be critical to the final output quality.

RPF Hyperparameters: T is a rejection threshold, {MIN, MINSOFT, MAN} 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 stand for the number of candidates produced by the generator per each lexeme, and t° is the generation temparature.

RPF architecture algorithm:

Input: Content Source S

output: Simplified text St = " "

repeat

Identify next lexeme l = process(s)

for i = 1 to n Samples do

Generate next Sample x_i = generate(st)

Save embedding Score e(x_i)

end for

if max(le(x_i) - e(l)l) > T then

St = arg max(le(x_i) - e(l)l) + St

else

St = l + St

end if

until source s is done

5. DATASET AND IMPLEMENTATION DETAILS.

Our lexeme Processor is a heuristics engine implemented in python. It splits the incoming sentence into lexemes preferring terminal punctuations, and falling back to other punctuation markers if no terminal boundaries are found with in a reasonable span. The Configurable hyperParameters Provide the Compromise between the Content preservation and simplication. If we wikipedia Simplification works best when the threshold is conservative, and the generator operates at a lowered temparature.

6. EXPERIMENTS

The lexeme replacement rates can vary based on the article. The rates are driven by the amount of material available for training (Simple Wikipedia corpus) adjacent to the articles

for transformation. "Personal" Category that describe biographies are seen to have lowest replacement rates where as the highest replacement rates are seen when there is rich and factual context to draw from. Such as in articles on popular religions or cultural phenomena.

The 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. A trivial change can lead to large differences in the Flesch-Kincaid score. A more complicated intervention would involve a restructuring of the whole phrase. Not every replacement results in a coherent modification.

7. ERROR ANALYSIS

The formal features of transformed text alone can not adequately describe the transformation because they do not estimate the qualitative language features such as preservation of content and text fluidity.

7.1 Phrase encoder errors: This project employs Google sentence Encoder for calculation of Candidate acceptance score. This is a state-of-the art encoder, but sometimes it fails to indicate a mismatch. Although the sentences are very similar according to their scores they can communicate entirely different meaning. To combact such blunders we can raise the acceptance threshold τ, but this may lower the effective replacement rates.

7.2 Suboptimal treatment of name entities and rare words:

The framework with names and words that were not in the training set which promots RPF to produce Credible-Looking replacements with Some "hard to Spot" errors. In all these cases the network does a good effort to Comply with a seemingly impossible requirement to Produce words not seen in training.

7.3 Over simplification:

It seems like some of the simple English constructs picked from the training set appear to be too simple. In these Cases the RPF may pick unwanted language biases from training set.

7.4 Non-Verbation quotes:

Another interesting Artifact is seen when rendering quotes. The RPF may alter the quote in some cases. 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. By doing a second pass over the generated text going in the opposite direction Can alleviate low replacement rates. Better training method Can be used when dealing with Name Entities and rare words. A brute-force solution to a large number of trails for finding a match which ~~resp~~ slows the generation process would require faster GPU, but a more intelligent solution might require stoping Sampling early if there are enough high-quality Candidates.