Blueprint of the end-to-end pipeline of the Research Paper Simplifier

1. preprocessing steps used in a research paper

- **Named Entity Tagging:** Identify named entities like people, places, organizations, and more using tools like Stanford CoreNLP. These entities are crucial for understanding the text.

- **Anonymization of Named Entities:** Replace identified named entities with tokens like "PER@1" or "LOC@2" to protect privacy. For example, "John and Bob" becomes "PER@1 and PER@2."

- **De-Anonymization at Test Time:** During testing, reverse the anonymization by matching tokens to their original entities. This process may occasionally fail but is generally reliable.

- **Handling Low-Frequency Words:** Replace words that appear three times or less in the training data with a special token like "UNK" to manage vocabulary and improve efficiency.

2. Definition of simplification (feature-wise)

- **Lexical simplification (LS)** only focuses to simplify complex words of one sentence. LS needs to identify complex words and find the best candidate substitution for these complex words. The best substitution needs to be more simplistic while preserving the sentence grammatically and keeping its meaning as much as possible

3. Scope of simplification

Scope of simplification: Lexical simplification, which aids in our understanding of the scientific material, will be our main focus. LS is a useful tool for text simplification since research indicates that readers who are conversant with a document's vocabulary can frequently decipher its content even if they find some of the grammatical constructions to be unclear. Three processes typically comprise the LS framework: filtering and substitute ranking (SR), complex word identification (CWI), and complex word substitute generation (SG).

4.Problem formulation (i.e. precise formulation of Input, Output, Loss function)

Input:

Input Sentence: A sentence or a short text passage containing complex words that need to be simplified.

Complex Word Identification: A binary mask indicating the position of complex words in the input sentence (e.g., 1 for complex words, 0 for non-complex words).

Output:

Output Sentence: The simplified version of the input sentence where complex words are replaced with simpler words or phrases. This can be represented as a sequence of words.

Complex Word Substitution Mask: A binary mask indicating which words were replaced (e.g., 1 for replaced words, 0 for unchanged words).

Loss Function:

C is the set of complex words in the input sentence.

G is the set of ground truth simpler words or phrases to substitute for the complex words.

P is the set of predicted substituted words or phrases.

R is the set of rankings for the predicted substitutes.

Complex Word Identification Loss (Binary Cross-Entropy Loss):

This loss encourages the model to correctly identify complex words.

It's the average binary cross-entropy loss over all words in the input sentence:

$L_{cwi} = - (1/|C|) * \sum(c \in C) [g_c * \log(p_c) + (1 - g_c) * \log(1 - p_c)]$

Substitute Generation Loss (Cross-Entropy Loss):

This loss ensures that the model generates suitable substitutes for the identified complex words.It's the average cross-entropy loss over all words in the input sentence:

$L_{sgl} = - (1/|C|) * \sum(c \in C) \sum(g \in G) [g * \log(p_g)]$

Substitute Ranking Loss (RankNet Loss):

This loss encourages the model to rank the proposed substitutes correctly.It measures the difference between the predicted rankings and the correct rankings. RankNet loss can be used to optimize the ranking:

$L_{srl} = - (1/|C|) * \sum(c \in C) (1/2) * [(1 - R_c) * (s_c - s_c') - \log(1 + e^{(-(s_c - s_c'))})]$

s_c is the true score of the ground truth substitute.

s_c' is the predicted score of the substitute.

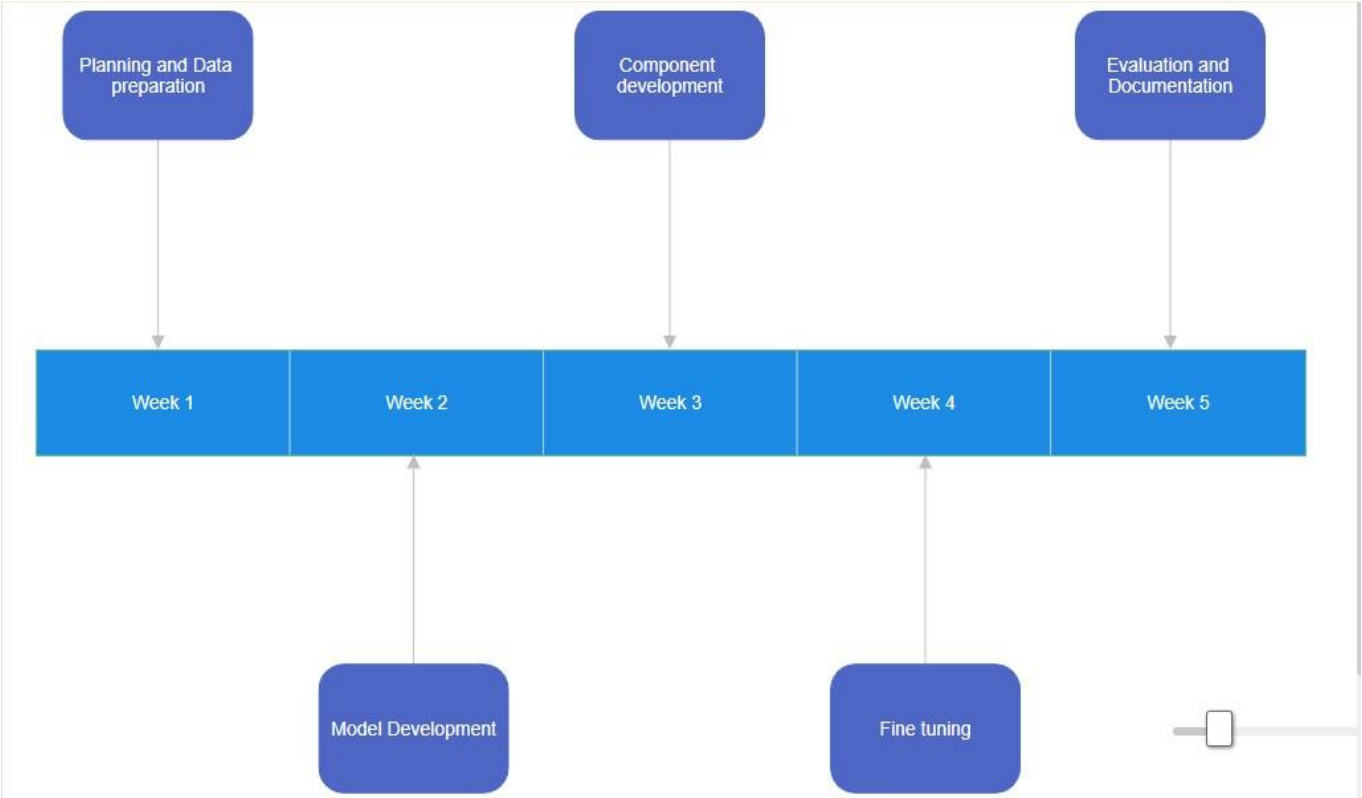
5. Methodology

- **Complex Word Identification (CWI):** We assign a complexity score to each word in a sentence. If a word's complexity score is above a certain threshold (e.g., 0.5), it's considered complex. We don't simplify entity words.
    -
- **Substitute Generation (SG):** We replace complex words with simpler alternatives. We use a language model (like BERT) to consider context while generating substitutes.
    -
- **Filtering and Substitute Ranking (SR):** We rank the substitute words based on how similar they are to the complex word, using vector representations and cosine similarity. Higher similarity means a higher ranking. This helps us choose the best substitutes.

Algorithm:

1.Identify complex words in the sentence (excluding the ones in the ignore list).
2.If there are complex words in the sentence:
2.1. Choose the top complex word (let's call it "w").
2.2. Generate substitute candidates for "w."
2.3. Rank these substitutes.
2.4. Select the top-ranked substitute.
2.5. If the top substitute is better (higher frequency or lower loss), perform the simplification.
2.6. Replace "w" with the top substitute and add it to the ignore list.
2.7. Repeat the process iteratively for other complex words.
3. Continue the iteration.
4. If there are no more complex words in the sentence, stop.

6. Timeline for completion of Project:



7.Task delegation among 3 members:

Team Member 1:

Data Preprocessing: Responsible for cleaning and preparing the Wikilarge dataset for use with LSBERT, including data cleaning, tokenization, and formatting.

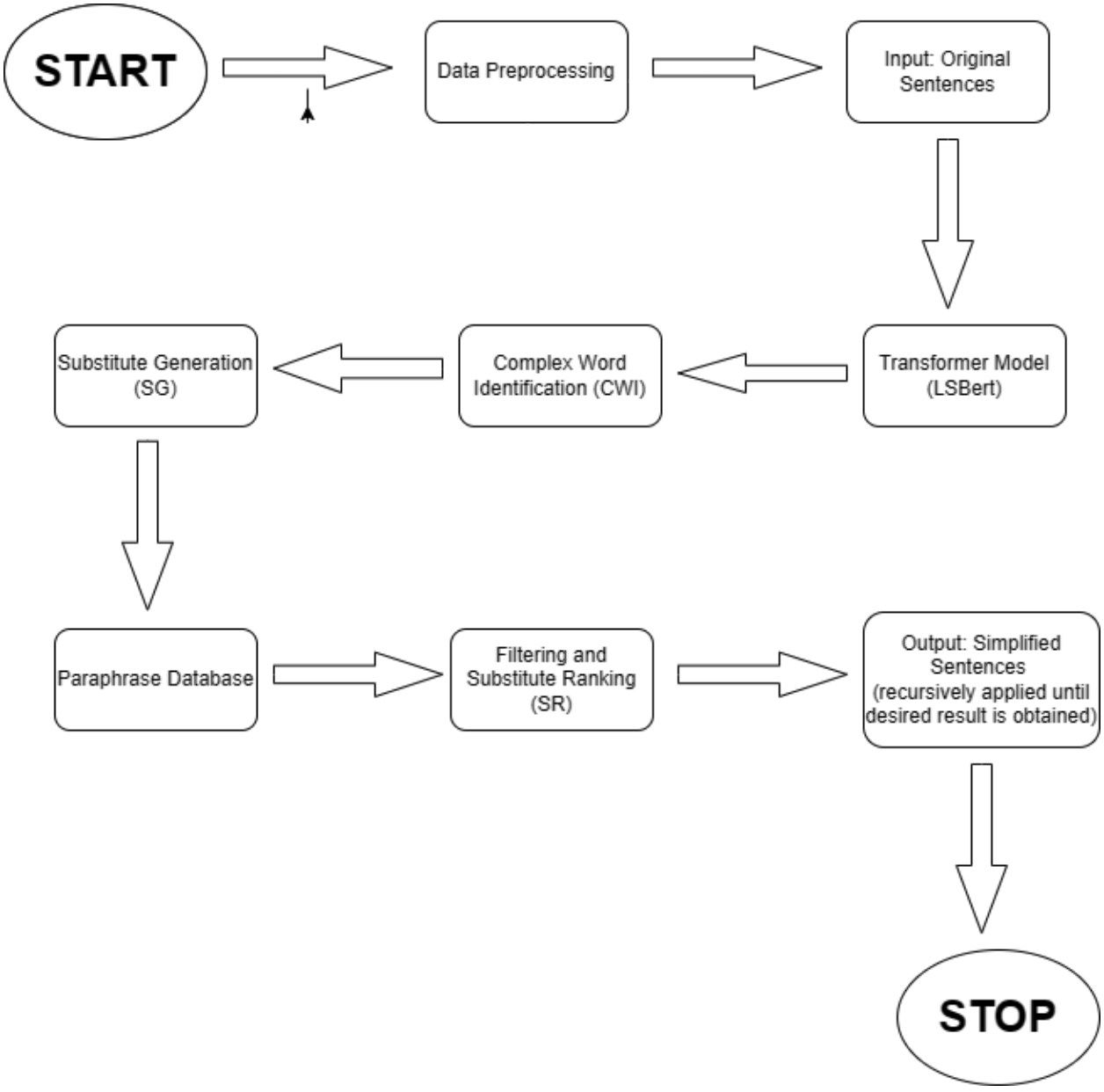
Team Member 2:

Model Implementation and Fine-Tuning: Focus on implementing the LSBERT model and fine-tuning it for lexical simplification, including setting up the model architecture, training, and hyperparameter optimization.

Team Member 3:

Complex Word Identification: Develop a complex word identification algorithm to flag complex words within a given text. This may involve creating a list of complex words, using heuristics, or leveraging pre-trained language models.

Flowchart


START
Data Preprocessing
Input: Original Sentences
Transformer Model (LSBert)
Complex Word Identification (CWI)
Substitute Generation (SG)
Paraphrase Database
Filtering and Substitute Ranking (SR)
Output: Simplified Sentences (recursively applied until desired result is obtained)
STOP