Project Report: Wikipedia Simplifier

By Rishav Bikarwar (Team 4)
Project Mentors: Ananya Gupta and Rishabh Dugaye

Objective: to simplify the Wikipedia articles with simplified vocabulary and sentence structures to make them more readable to children and non-native speakers

We started by learning the basics of python.
Assignment 1: implemented Trie data structure in python
Assignment 2: implemented the Hangman game in python

Next, we learnt the basics of deep learning. We understood the working of neural networks and presented it to the mentors.
Assignment 3: implemented AND boolean function using neural networks in python from scratch

After that, to make training deep network models easier, we learnt the TensorFlow library. Alongside we also explored Natural Language Processing (NLP).
Assignment 4: implemented deep learning model for movie review sentiment analysis using TensorFlow framework; also tried using pre-trained word embeddings

We then read the Wikipedia Simplifier paper. The authors set up a pipeline to replace complex phrases with easy ones. They fine-tuned the GPT-2 transformer on Simple Wikipedia and used it for text generation.
Assignment 5: understood and wrote a summary of the Wikipedia Simplifier paper

Final Model:

I started by reading papers on '*Text Simplification*' and documented the various approaches, relevant datasets and evaluation metrics that have been used for simplifying texts. It can be found here: Text-Simplification-Survey

The main problem with Text Simplification is the unavailability of labelled parallel simplification data. Various approaches have made use of extracting parallel simple-complex sentence pairs from Wikipedia and Simple Wikipedia.

I then went on to see the state of the art approaches used for this task.
MUSS: Multilingual Unsupervised Sentence Simplification by Mining Paraphrases is one such approach that doesn't require simplification data. MUSS uses a novel approach to sentence simplification that trains strong models using sentence-level paraphrase data instead of proper simplification data. These models leverage unsupervised pretraining and controllable generation mechanisms to flexibly adjust attributes such as length and lexical complexity at inference time.

So, in this project, I have tried implementing this approach.

Approach

Step 1 - Mining Paraphrases:

a. Sequence Extraction:
A *Sequence* consists of multiple sentences: to allow sentence splitting or fusion operations.

i. *docs from the HEAD split in CCNet* (categorized by language)
ii. *doc* -> *sentences* using NLTK sentence tokenizer
iii. *adjacent sentences* -> *sequences* with max characters = 300
iv. *filter sequences*: remove those having >= 10% punctuation characters or low language model probability according to a 3-gram Kneser-Ney language model trained with kenlm on Wikipedia

b. Creating a Sequence Index Using Embeddings:
extracted sequence (1 billion) -> n-dimensional embeddings using LASER (n = 1024 reduced to 512 by using PCA followed by random rotation)

c. Mining Paraphrases:
i. *index these embeddings* (for use with faiss)
ii. *find nearest neighbours:* each sequence is used as query (q_i) against these 1 billion sequences to find the top 8 nearest neighbours using L2 distance (faiss). keep those with L2 distance < 0.05 and relative distance with other 7 neighbours < 0.6
iii. *paraphrase filtering* –
- remove almost identical pp with character level Levenshtein distance <= 20%
- remove pp coming from the same document
- remove pp where one sequence is contained in other

Step 2 - Simplifying with ACCESS:

ACCESS is a method to make any seq2seq model controllable by conditioning on simplification-specific control tokens. Apply this to seq2seq pre-trained transformer models based on the *BART*.

Training with Control Tokens:
During train time, control tokens that give info about the target sequence are provided to the model. Prepend the following control tokens to every source in the training set:

<NumChars_XX%> Character Length Ratio
<LevSim_YY%> replace-only Levenshtein similarity
<WordFreq_ZZ%> aggregated word frequency ratio
<DepTreeDepth_TT%> dependency tree depth ratio

Selecting Control values at Inference:
During inference time, control the generation by selecting a given target control value. Choose these hyperparameters based on the SARI score on the validation set or by using prior knowledge based on the target audience.

Step 3 - Leveraging Unsupervised Pretraining

Fine-tune the pre-trained generative model BART on the newly created training corpora.

Implementation

I also learnt the *PyTorch* framework (while exploring) and created various deep learning models using it. So, I have used this framework to fine-tune the model and the "*transformers*" library, which contains the pre-trained *BART* model. Given below are other few notable libraries that I used:

- cc_net: includes data from thousands of websites
- wikipedia: to fetch Wikipedia articles
- laser: to compute the sentence embeddings
- nltk: to tokenize sentences
- faiss: to find nearest neighbours
- torch: to generate a dataset and to train the model

The CC_Net corpus requires around 7 TB storage which was not available. So I could not reproduce the results as obtained by the authors of the paper. Instead, to check if all the functions work correctly, I have used some sample text and tried to find pairs of similar meaning sentences. After that, I trained the model using that data.

For evaluation purposes, I have used the pre-trained MUSS model from the GitHub repository of the official implementation of this paper with few modifications. I then fetched Wikipedia articles and tried simplifying them. I also used the summarization pipeline to summarize the simplified text. Then, I compared the complex, simple and summarized text based on the readability indices.

Result

The functions I implemented for fine-tuning the model are working as expected. They can be tested with some more extensive data as well. The training was slow, though, as it was using just the CPU. The pipeline can be modified to use GPU as well, which will be much faster.

Using the pre-trained MUSS model to simplify Wikipedia articles gives excellent results. Given below are few pairs of complex and simplified sentences:

Sentence from Wikipedia article	Simplified sentence by MUSS
The Messi–Ronaldo rivalry is an association football rivalry between Argentine forward Lionel Messi and Portuguese forward Cristiano Ronaldo.	The Messi-Ronaldo rivalry is a football rivalry between two very good football players: Cristiano Ronaldo and Lionel Messi.
The average lifespan of pet cats has risen in recent decades.	In recent years, the average age of pet cats has increased.
Regardless of preference, football critics generally agree that both are the greatest players of their generation, outperforming their peers numerically, largely by a significant margin.	No matter what people think, football people generally agree that both are the greatest players of their time. Many people think they are the best players of their time.
It was long thought that cat domestication was initiated in ancient Egypt, as since around 3100 BC veneration was given to cats in Egypt.	It was long thought that cat domestication started in ancient Egypt. Since around 3100 BC, cats were given special treatment in Egypt.
A juvenile cat is referred to as a kitten.	When they are young, cats are called kittens.
As agricultural practices spread, so did tame and domesticated cats.	When people started farming, they kept domesticated cats.

There was a very significant improvement in the readability indices as well. The table below shows the difference (the article used was “Messi-Ronaldo rivalry”):

Readability Index (article on Messi-Ronaldo)	Source sentence	Simplified Sentence
Flesch Reading Ease(+)	42.58	68.3 (+25.72)
Flesch Kincaid Grade(-)	16.5	8.7 (-7.8)
Automated Readability Index(-)	20.6	10.8 (-9.8)
Smog Index(-)	14.4	10.1 (-4.3)

Readability Index (article on cat)	Source sentence	Simplified Sentence
Flesch Reading Ease(+)	54.76	82.85 (+28.09)
Flesch Kincaid Grade(-)	11.8	5.1 (-6.7)
Automated Readability Index(-)	14.8	6.9 (-7.9)
Smog Index(-)	13.5	8.9 (-4.6)

Note: (+): higher => better (-): lower => better

Final Words

Through MUSS, the authors propose a sentence simplification approach that does not rely on labelled parallel simplification data, thanks to controllable generation, pretraining and large-scale mining of paraphrases from the web. This approach matches or outperforms previous state-of-the-art results.

Through this project, I have learnt a lot related to deep learning and especially NLP. It was fascinating. Further, to learn even more things, I would like to use web development to deploy this project with a straightforward UI to navigate Wikipedia articles with simplified sentences. Also, I will try to develop a way to transform any writing so that only the text gets replaced by simplified text while retaining the tables and images, maybe in real-time, like a scanner. 😋