CA – Assignment 4: Argument Generation

- Group: `FakeNews`
- Group members:
  - Adnan Manzoor
  - Sajjad Pervaiz
  - Kevin Taylor
  - Christoph Schäfer

Structure

```
.
└── argument-generation-assignment
    ├── Documentation.pdf
    ├── README
    ├── requirements.txt
    ├── data
    │   ├── essay_prompt_corpus.json
    │   ├── sample_predictions.json
    │   ├── predictions.json
    │   └── train-test-split.csv
    └── code
        ├── evaluation.py
        ├── model.py
```

Scripts

`model.py`: PageRanking sentences and selecting top two for generating predictions. If enabled uses Google's T5(Text-To-Text Transfer Transformer)[1] to generate abstractive summary from those top two ranked sentences.

`evaluation.py`: Script to evaluate the `Rouge F` score.

How to run the scripts

- On a clean `venv` install the packages in the `requirements.txt` file (around 800 MB download required for first time):

```
pip install -r requirements.txt
```

- Make sure you have the same directory structure as above otherwise adjust the paths in the scripts accordingly.

- Run `model.py` to generate the predictions in `data/` directory with name `predictions.json`. (It is important to note that running this script for the first time will lead to a download of around 800MBs NLTK data [3] + glove word embeddings)

```
python model.py
```

- To generate abstractive summary enable the T5 transformer (around 250MBs will be downloaded when run for the first time):

```
python model.py --t5 true
```

- Run `evaluation` script with path to `data/` folder.

Explanation

We started to use random sentence selection, to figure out what the lowest baseline is, with results around `0.12 rouge-1 f score`. Based on that we realized, that to reach the given baseline marginally, a not too complex method of sentence selection should suffice.

We followed up by picking sentences based on basic features we read about, that indicate a good summarising sentence. These included the longest sentence and selecting sentences based on preselected keywords (e.g. In conclusion..., The best...). This however was not successful enough, as it was in the range of the random selection results.

Finally, we tried extractive summarisation using PageRank evaluation on text, inspired by the given paper from Alshomary et.al.[2]. Therefore we use pre-trained word-embeddings to create similarity matrix and selected the best sentences based on that. This lead to a `rouge-1 f score of 0.138`, with which we reached our goal. The steps in the process include:

- splitting the text into individual sentences
- finding vector representation (word embeddings) for each and every sentence. We used word embeddings because they capture the meaning of words in the vector form
- similarities between sentence vectors are then calculated and stored in a matrix
- the similarity matrix is then converted into a graph, with sentences as vertices and similarity scores as edges
- Finally, top two of the top-ranked sentences form the final summary

Additionally, we tested Google's T5 (Text-To-Text Transfer Transformer)[1] on the top two ranked sentences, which easily produced results higher than the baseline around `0.176 rouge-1 f score`. This was done because while analysing the prompts from the training data we observed that the prompts are `abstractive` in nature i.e they were different than any sentence in the text. The page-ranking approach is `extractive` in nature i.e it uses the sentences already present in text. Therefore we also included the T5 transformer that generates abstractive summary from the top ranked sentences and this can be enabled using the `--t5 true` argument while running the model.

[1] https://arxiv.org/pdf/1910.10683.pdf

[2] https://webis.de/downloads/publications/papers/alshomary_2020b.pdf

[3] https://www.nltk.org/data.html