**Dataset**: HOTPOT QA Dataset.

**First Phase:**

I have created the following dataframe from the HOTPOT QA dataset,

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **question-id** | **passage-id** | **question** | **passage\_title** | **sentence** | **passage\_length** | **sia\_score** |

Question-id: unique id for each question

Passage-id: candidate passage id of each question

Question: question+exact answer

Passage\_title: candidate passage title

Sentence: each sentence from candidate passage

Passage\_length: length of the candidate passage, required for calculating the average SIA score for the candidate passage

Sia\_score - sia score between question and each sentence of the candidate passage

**Second phase:**

I have computed the average SIA score for each candidate passage by representing them in the following column format.

|  |  |  |
| --- | --- | --- |
| **question** | **passage\_title** | **sia\_score** |

Question: question+exact answer

Passage\_title: candidate passage title

sia\_score: average SIA score of the candidate passage

Task Assigned:

Handled data cleaning, data parsing, data preprocessing for HOTPOT QA

* **Data Preprocessing & Cleaning phase:** Key columns from the json data are extracted. When extracting the sentences from candidate passage, all the sentences with empty string or less than two characters are considered as special cases and are not included as they are producing invalid scores in the score prediction phase.
* **Data parsing Phase**: converted the .json data format to .csv format having all the necessary key columns.
* **Score Prediction Phase**: researched, tested, verified various pretrained & trained models like Bert, Roberta etc Among all, pretrained, fine-tuned Web BERT-based semantic text similarity model trained on STS-B dataset is producing good results for HOTPOT QA dataset, this model modifies PyTorch transformers by abstracting away all the research benchmarking code for ease of real-world applicability.
* For the STSbenchmark, we have sentence pairs annotated together with a score range [0-5] indicating the similarity.
* **Data Processing Phase:** After generating the scores for each sentence of every candidate passage, processed the data to generate an average SIA score for candidate passage, saved all the results in output\_data.csv format

Consider the following sample data:

[{"supporting\_facts": [["Arthur's Magazine", 0], ["First for Women", 0]],

"level": "medium",

"question": "Which magazine was started first Arthur's Magazine or First for Women?",

"context": [ ["Radio City (Indian radio station)", ["Radio City is India's first private FM radio station and was started on 3 July 2001.", " It broadcasts on 91.1 (earlier 91.0 in most cities) megahertz from Mumbai (where it was started in 2004), Bengaluru (started first in 2001), Lucknow and New Delhi (since 2003).", " It plays Hindi, English and regional songs.", " It was launched in Hyderabad in March 2006, in Chennai on 7 July 2006 and in Visakhapatnam October 2007.", " Radio City recently forayed into New Media in May 2008 with the launch of a music portal - PlanetRadiocity.com that offers music related news, videos, songs, and other music-related features.", " The Radio station currently plays a mix of Hindi and Regional music.", " Abraham Thomas is the CEO of the company."] ],

………………..

……………….

"answer": "Arthur's Magazine",

"\_id": "5a7a06935542990198eaf050",

"type": "comparison"}

From this, I have extracted question, answer, set of candidate passages from the context, again from the context, i have extracted sentences and computed STS scores between <Q+A> and the every <sentence> of each candidate passage , finally calculated the average sia score for each candidate passage in the context.

**Instructions to run the code:**

All the executions are aligned sequentially in jupyter notebook

1. Run the Necessary Installations section
2. Run the data preprocessing section
3. Run the scores generation section
4. Run the data processing section
5. Please ignore the commented code

**User Controls:**

* Look for the variable ‘FOLDER’, update the value to the main folder path where the input\_data.json file exists
* Change the variables ‘start’ and ‘end’ to control data size
* The output csv files will be generated in the same ‘FOLDER’

Also uploaded train, test, dev data and the models used while experimenting and training, experimental code is commented