CMSC 438 / 691 – Fall 2024

Homework Assignment 5

Announced: 11/21

Due: Friday, 12/6, 5pm

The problem

Write a function
`def find_most_relevant(query, text_list)`
that on input is given:
a string “query”, and a list of strings “text_list”,
and does the following:

1. calculates similarity scores of the query to each string form “text_list”
2. finds the string from “text_list” that is has highest score, i.e., that is semantically most related to “query”

The function should return two variables:

1. idx: a single integer, the index of the highest similarity string in the list (return lower index in case of a tie)
2. scores: a numpy array of length n, where n is the length of the ”text_list”, with float values representing similarity scores

The problem

Write a function

```
idx, scores = find_most_relevant(query, text_list)
```

- Use encoder Transformer architecture to get a semantic embedding vector for a string.
- Use cosine similarity applied to embeddings to measure "relatedness"
- The function is essentially a simple version of the basic Retrieval-augmented generation (RAG) search subsystem, see Lecture 24, slides 44-49 (especially slide 46)

The problem - example

- Example:

```
idx, scores = find_most_relevant('city',
    ['undergraduate curriculum committee',
     'townhall planning group',
     'ski resort management',
    ])
```

should return:

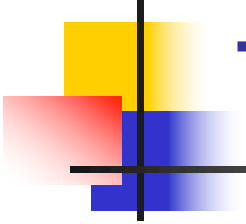
```
idx: 1
scores: [0.0819, 0.3624, 0.1637]
(scores above were rounded to 4 decimal digits)
```

These results indicate that "townhall planning group" is semantically most similar/relevant to the query "city"

Embedding model

- Use class SentenceTransformer from sentence_transformers library (https://www.sbert.net/docs/package_reference/sentence_transformer/SentenceTransformer.html) to obtain the embeddings for the query and for each string in text_list
  - Use 'all-MiniLM-L6-v2', a small but effective model
- Useful functions:
  - torch.cosine_similarity or
  - SentenceTransformer.similarity
- text_list will have no more than 100 elements, it is ok to use simple search (e.g. torch.argmax) for finding highest similarity

Testing your code



- See H05_datagen.py in Canvas for an example of a code that can automatically generate a large number of strings based on a given topic

- This can be used to generate a collection of strings on several topics, and then test your code by calling it with one of the topics as "query"

- In your submission, do not include any code for generating test strings, we will use our own

Returning the Assignment

- Solution code should be written by you and you only (no web/book/friend/etc. code)

- Upload through Canvas/Gradescope
  - Similar to Homework 3 & 4
  - A single file with your `find_most_relevant` function
  - Do not include string generating code from H05_datagen.py in your submission

    - Make sure your file has all the necessary imports
    - If your code doesn't "compile" or throws an exception, gradescope will fail, with 0 points
    - It is advisable to either delete any of your testing code, or "guard" it with:
      ```
      if __name__ == "__main__":
      ```