Unstructured Data - Assignment 2

28 February 2024

Assignment 2

In this assignment, you will use various transformer models for semantic search and for language generation. We will be using the `transformers` python package from huggingface; **note** that this package will automatically download language models as required the first time the code is run, and they can be quite large. (The entire assignment might download a few GB.) You might want to do this on campus, depending on your internet situation.

This assignment is to be done individually. You may discuss the project with your classmates, but the work you turn in should be your own.

Part 1 - Comparing and Using Embeddings

Goal

The main goal of this part of the assignment is to experiment with different embedding techniques in an information retrieval context. It also reinforces the definition and use of the `json` format.

Setup

This assignment will use python. We will use the `sentence-transformers` python package and its dependencies.

https://huggingface.co/sentence-transformers

If you have python installed on your machine, you can use

```
pip3 install -U sentence-transformers
```

If you do not have python, you can install conda from https://anaconda.org/ and then install sentence-transformers with

```
conda install -c conda-forge sentence-transformers
```

If you are already familiar with python and conda environments, you can work however you wish.

Use the provided `A2Part1.py` file as a template, and the attached file `tweets-utf-8.json.zip`, which contains tweets geolocated to London and Ottawa over a period of time in 2017. Unzip this prior to starting.

Note

In Part 1, you should consider each tweet to be a "document."

Questions

Coding (40 pts)

1. Write a function `get_tweets()` that uses the `json` python package to read the tweets from `tweets-utf-8.json` and produces a list of strings that contain the text of each tweet. (Each line of `tweets-utf-8.json` contains one json object.)

2. Write a function `sort_by_sim(query_embedding,document_embeddings,documents)` that takes the embedding of a query document, a list of document embeddings, and a list of the corresponding documents, and returns a list of pairs of the form `(similarity,document)`, sorted in decreasing order according to cosine similarity between each document and the query. You can use any packages you like; note that `numpy` has a `dot` function. If a similarity computation would involve a divide by zero, define the similarity to be `0` instead (This is not correct, but is OK for our purposes.)

3. Write a function top25_glove() that returns the top 25 most similar tweets (as (similarity,document) pairs) to the query "I am looking for a job." using the glove-based sentence embedding defined here: https://huggingface.co/sentence-transformers/average_word_embeddings_glove.840B.300d

4. Write a function top25_minilm() that returns the top 25 most similar tweets (as (similarity,document) pairs) to the query "I am looking for a job." using the MiniLM-based (derived from BERT) sentence embedding defined here: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2 **FYI** - this model takes quite a bit longer to run (almost 10 minutes).

Intepreting (10 pts)

Answer the following questions in a file called `A2Part1.txt`.

Examine the output from both models given our test query, “I am looking for a job.”

1. Identify two differences in the overall results of the two methods and explain why these differences might be occurring.
2. Try out the query in Twitter’s own search on their website. (Note you don’t need an account to try it.) Do you think Twitter might be using a semantic search technique like the ones you tried? Why or why not?

Part 1 Deliverables

Submit your `A2Part1.py` and `A2Part1.txt` files as an attachment on OWL. **Submissions will only be accepted through OWL.**

Part 2 - Using Generative Language Models

Goal

To learn about how generative language models can be used in practice, focusing on GPT-2.

Setup

This part uses the `transformers` package which can be installed with conda or pip.

Questions (25 pts)

1. Write a script that generates a “story” using a local GPT-2 model. Your story should: 1) be at least 100 words long; 2) not have repeated phrases; and 3) be the same every time your script is run. It might be nonsensical and/or hilarious. Use the skeleton code provided in `A2Part2.py` as a starting point, and https://huggingface.co/blog/how-to-generate as a reference document. Record your story in a file called `A2Part2.txt`.

Note that the provided `A2Part2.py` uses pytorch rather than the older TensorFlow which is used in the reference document. The syntax is pretty much identical except for the setup, which we provided. If you really want to use TensorFlow instead, that’ s fine too.

Part 2 Deliverables

Submit your `A2Part2.py` as an attachment on OWL along with your story in a file `A2Part2.txt`. **Submissions will only be accepted through OWL.**

Checklist

Your owl submission should include the following attachments and no additional files:

```
A2Part1.py
A2Part1.txt
A2Part2.py
A2Part2.txt
```