

COMP 598 Homework 8 – Using TF-IDF

30 pts

Assigned Nov 11, 2021

Due Nov 19, 2021 @ 11:59 PM

Non-standard (i.e., not built-in) python libraries you can use:

- pandas

In this assignment, we're going back to homework 3 and computing each pony's most frequent words using TF-IDF. Note that, throughout this assignment, we refer to "pony names" – use the canonical names we used for each of the main character ponies in HW3.

Task 1: Compute word counts (15 pts)

Write a script that computes word counts for each pony from all episodes of MLP. Your script, `compile_word_counts.py` should run as follows:

```
python compile_word_counts.py -o <word_counts_json> -d <clean_dialog.csv file>
```

Remember that `-o` and `-d` should refer to absolute paths, for example:

```
python compile_word_counts.py -o /path/to/word_counts.json -d path/to/clean_dialog.csv
```

For the output file, you should create directories if they do not exist.

The output file should be a dictionary with the following form:

```
{
    "twilight sparkle": {
        "<word1>": <# of times the word1 is used by twilight sparkle>,
        "<word2>": <# of times the word2 is used by twilight sparkle>,
        ...
    },
    "pinkie pie": {
        ...
    }
    ...
}
```

Make sure you have exactly the following keys for the pony names: "twilight sparkle", "applejack", "rarity", "pinkie pie", "rainbow dash", "fluttershy".

Indentation won't matter in this exercise, you can have a one-line JSON file or a pretty-printed one.

For your analysis:

- Some of the words are going to be rare and will have a very small frequency. These words are not going to be very useful for your analysis. You should only keep words with a frequency higher than a specific threshold. For this homework, only keep words that occur at least 5 times across ALL valid speech acts.
- Also, to avoid boring results (like, the most frequent word being "*the*"), remove all the *stopwords*. Use the attached list of words. This file will also be provided in the HW assignment on mycourses. You should remove any words present in this file from your analysis.

<https://gist.githubusercontent.com/larsyencken/1440509/raw/53273c6c202b35ef00194d06751d8ef630e53df2/stopwords.txt>

- Use the same dialog file we used in HW3. (clean_dialog.csv from <https://www.kaggle.com/liury123/my-little-pony-transcript>). You must submit your generated file, which must be named **word_counts.json** and placed at the root of the submission_template folder. Please check the README.md file for HW8 for further instructions.

Other details and reminders:

- Valid speech acts - only consider speech acts where the speaker is an exact match for **one** of the main character ponies. Ignore any others. Also lines which involve multiple characters, i.e. "Twilight and Fluttershy" or inexact matches, such as "future Twilight Sparkle" should be ignored.
- Treat each word encountered as case insensitive. Store words in all lowercase form.
- Before processing text, replace punctuation characters with a space – a punctuation character is one of these: () [] , - . ? ! : ; # &
- A word must only include alphabetic characters. All other words should be ignored.
- Remove the stopwords (listed here - <https://gist.githubusercontent.com/larsyencken/1440509/raw/53273c6c202b35ef00194d06751d8ef630e53df2/stopwords.txt>)
- Tip: to keep your script performant, store your word counts in dictionaries.

Task 2: Compute most frequent & distinctive pony language (10 pts)

Write the script compute_pony_lang.py which is run as follows:

```
python compute_pony_lang.py -c <pony_counts.json> -n <num_words>
```

The <pony_counts.json> file should have the same format output by your compile_word_counts.py script in Task 1. It should compute the <num_words> for each pony that has the highest TF-IDF score. Note that to compute the inverse document frequency, you should use the number of times the words were used by all 6 ponies (i.e., only use the counts in the pony_counts.json, not all speakers from the original script). The specific definition of TF-IDF you should implement is:

$$\text{tf-idf}(w, \text{pony}, \text{script}) = \text{tf}(w, \text{pony}) \times \text{idf}(w, \text{script})$$

$\text{tf}(w, \text{pony})$ = the number of times pony uses the word w (which we compute in task1)

$$\text{idf}(w, \text{script}) = \log \left[\frac{(\text{total number of ponies})}{(\text{number of ponies that use the word } w)} \right]$$

Output should be written in JSON format **to stdout** with the following structure:

```
{
  "<pony name>": [ "highest-tfidf-word", "second-highest-tfidf-word", ... ],
  "<pony name>": ...
}
```

Each pony word list should have <num_words> entries.

Use the same keys from Task 1 for the pony names: "twilight sparkle", "applejack", "rarity", "pinkie pie", "rainbow dash", "fluttershy".

As usual, the -c argument refers to an absolute path (and not just a file name). Indentation won't matter in this exercise, you can have a one-line JSON file or a pretty-printed one.

Task 3: Write unit tests for your methods (5 pts)

Write two unit tests for your code – one test for task 1, one for task 2. For the first task, your unit test should pickup a mini script file and produce counts that are checked against a JSON file containing ground truth counts. For the second task, your unit test should pickup the ground truth counts and compute TF-IDF scores that are checked against a JSON file containing ground truth TF-IDF scores.

In both cases, we will provide the mini script file, ground truth counts, and ground truth TF-IDF counts – so you will only write the inputs and the assertions to match your code. **Crucially, your tests should import your python code and run it, *NOT* invoke a new python3 process.**

Follow these instructions carefully:

- You will find three files under the **fixtures** folder: `mock_dialog.csv` `word_counts.true.json` `tf_idfs.true.json`.
- `mock_dialog.csv` is populated with an example. That means you can modify its contents accordingly. It should contain just a small dialog example, so you can manually count words and calculate the TF-IDF.
- Manually create contents for `word_counts.true.json` `tf_idfs.true.json` (i.e. calculate them manually)
- DO NOT MODIFY THE NAME OF THESE FILES
- You'll also find a test file named `test_tasks.py` under the **test/** folder. We already load the fixtures files for you.
- Replace the assertions on `test_task1` and `test_task2` with any meaningful assertion. Just make sure you use the fixtures we provided; they will be under the variables `self.mock_dialog`, `self.true_word_counts`, `self.true_tf_idfs`. **DELETE** the `self.assertTrue(True)` you spot.
- Your test suite **MUST** pass.

Please check hw8 submission template for more information. Please read the instructions carefully -

<https://github.com/druths/comp598-2021/tree/main/hw8>