COMP20008 Elements of Data Processing
Assignment 1

Last updated: March 10, 2022

Marks and due date

The assignment is worth 20 marks (20% of the subject grade) and is due on at 6pm, Wednesday 13 April 2022 (Melbourne time). The estimated time commitment 20 – 30 hours.

Submission instructions

Please see Ed.

Background

It should be no surprise to us that while news sources provide most of the information we learn every day, it is questionable whether the news we read is credible and trustworthy. Differentiating legitimate and correct news articles from 'fake' or misleading ones is an important task, raising the interest of journalists, policy makers and researchers in many fields such as social science and data science. Specifically in the context of health, the consequences of delivering illegitimate news can be paramount and irreversible.

In this assignment, you will be working on a dataset comprising more than 1,000 health-related news articles, the reviews of these articles, and tweets about the new articles.

Using this dataset, you will have an opportunity to learn how to identify 'good' and 'bad' news by characterising some linguistic patterns of news, as well as comparing different news outlets based on their correctness of reporting.

Learning outcomes

- Be able to read and write data in different formats, and combine different data sources.
- Practice text processing techniques using Python.
- Practice data processing, exploration and visualisation techniques with Python.
- Practice visual analysis and written communication skills.

Dataset

The dataset consists of more than 1,000 health-related news articles, the reviews of these articles, and tweets about the new articles. They are located in the folder `/course/data/a1/`.

News articles

Each of the news articles is stored in a file in the `/course/data/a1/content` folder. The name of each file corresponds to the ID of the article and is in the format `story_reviews_xxxxx.json`, where each x is a digit; for example, `story_reviews_00001.json` contains the news article of ID `story_reviews_00001`.

News reviews

Each article was independently reviewed by at least one expert, based on a list of criteria. Against each criterion, the article receives a rating of satisfactory, not satisfactory or not applicable, and an explanation for the rating. Each article is also assigned an overall rating between 0 (least accurate) to 5 (most accurate).

All reviews are stored in a single file: `/course/data/a1/reviews/HealthStory.json`. If you read it using Python's `json`, you should see a list of reviews.

Tweets

News articles are shared on Twitter. The IDs of Tweets about the news articles have been recorded in `/course/data/a1/engagements/HealthStory.json`. You should get a dictionary with keys being article IDs, and values containing tweets. Each tweet is represented by a unique integer. Note that a tweet may appear multiple times, so if you count tweets, you need to avoid counting duplicates.

Your tasks (Total 20 marks)

Task 1. Loading data (1 mark)

Implement the function `task1()` in `task1.py` that outputs a json file called `task1.json` in the following format:

```
{
    "Total number of articles": X,
    "Total number of reviews": Y,
    "Total number of tweets": Z
}
```

where `X`, `Y` and `Z` are the number of news articles, number of reviews, and number of tweets respectively.

NOTE: You are free to write any helper functions (even in another file) if you like, but do not change the function definition of `task1()`. For example, do not add any argument to `task1()`. This applies to all tasks.

To run your solution, open the terminal and run `python main.py task1`

Task 2. Data aggregation (2 marks)

In each review, you will see a `news_id` field containing the ID of the article. This information allows you to match the review with the news article.

Implement the function `task2()` in `task2.py` which combines the articles with their reviews to work out how many "satisfactory" ratings that article receives, out of 10 criteria in total. Your function should save its output to a csv file called `task2.csv`, which contains the following headings: `news_id`, `news_title`, `review_title`, `rating`, `num_satisfactory`. Each row in the file should contain the details of one article, with

- `news_id` being the ID of the news article in the format `story_reviews_xxxxx`;
- `news_title` being the title of the news article;
- `review_title` being the title of the review article;
- `rating` being the overall rating of the article (between 0 and 5); and
- `num_satisfactory` being the total number of criteria (between 0 and 10) that are satisfactory.

The rows in `task2.csv` should be in ascending order of `news_id`.
To run your solution, open the terminal and run `python main.py task2`

Task 3. Extracting meta-data (2 marks)

Each news article comes with a `publish_date` field, which specifies when the article was published. The field is in the millisecond precision format, which is a floating point number. Convert it into a more readable format using the `datetime.datetime.fromtimestamp()` function. It should return you an object of type `datetime`, and you will need to consult the documentation to extract the year, month and date of the article.

Implement the function `task3()` in `task3.py` that performs the following two sub tasks:

- Extract the year, month, and day components from the `publish_date` of each article, and output a csv file called task3a.csv, which contains the following headings: `news_id`, `year`, `month`, `day`. Each row should contain the ID of an article in the format `story_reviews_xxxxx`, and the year, month, day on which it was published. The formats for year, month, and day are 4-digit year, 2-digit month, and 2-digit day of the month. For example, for the date 17 April 2022, the year, month, and day components are 2022, 04, and 17 respectively.

  The rows in `task3a.csv` should be in ascending order of `news_id`.

- Count the number of articles in each calendar year in the dataset, and output a file called `task3b.png`, describing the yearly article counts using an appropriately chosen graph.

There might be articles for which the publish date is unspecified; exclude those articles in your output. Also note that you will likely write helper functions for this task; however, as always, do not change the definition of `task3()`.

To run your solution, open the terminal and run `python main.py task3`

Task 4. Assessing the credibility of news agencies (2 marks)

Are news article published by The New York Times more credible than those by Fox News, according to the judgement of experts? Recall that each article is given an overall rating, which is a score between 0 and 5 (inclusive) indicating the credibility of the information presented in the article. In this task, you will compare the average rating of articles published by each news source.

In each review, you will see a `news_source` field, which tells you the where the article was published. Some reviews leave this field as blank (an empty string), in which case you can exclude them.

Implement the function `task4()` in `task4.py` that performs two sub tasks:

- Output a csv file called `task4a.csv`, which contains the following headings: `news_source`, `num_articles`, `avg_rating`. Each row should contain details about one news source.

  The rows in `task4a.csv` in ascending order of `news_source`.

- Output a file called `task4b.png` comparing the average ratings of all news sources, that have at least 5 articles. Choose an appropriate plot for this task. Also, sort the axes so that the most and least credible news sources are easily detected.

To run your solution, open the terminal and run `python main.py task4`

Task 5. Assessing credibility against popularity (1 mark)

Before doing this task, ask yourself some of these questions. Do people tend to share more accurate, credible news than less credible ones on social media? Why does "fake news" propagate much faster online (that is, why do people share much more fake news)? In Tasks 5, 6 and 7, you will perform analyses that give you some insights to these questions.

Implement the function `task5()` in `task5.py` which outputs a file called `task5.png` comparing the average number of tweets for each article-rating group. For example, for an article with a rating of 0, how many tweets does it have on average? Note that the tweets here include both original tweets and retweets.

To run your solution, open the terminal and run `python main.py task5`

Task 6. Text processing (2 marks)

News articles often have some characteristics that make them difficult to process; for example, they might contain unicode characters and even emojis. To make our analysis easier, in this task you will perform some preprocessing on the articles.

Implement the function `task6()` in `task6.py` that performs the following pre-processing steps on the content of the news articles: The content of an article is in the `text` field.

1. Convert all non-alphabetic characters (for example, numbers, apostrophes and punctuation), except for spacing characters (for example, whitespaces, tabs and newlines) to single-space characters. For example, `'&'` should be converted to `' '`.

2. Convert all spacing characters such as tabs and newlines into single-space characters, and ensure that only one whitespace character exists between each token.

3. Change all uppercase characters to lowercase.

4. Tokenise the resulting string into a list of words, using the space delimiter.

5. Remove all stop words in `nltk`'s list of English stop words from the resulting list. You may find the function `filter` in Python useful, but this is not necessary.

6. Remove all remaining words that are only a single character long from the resulting list.

7. Once steps 1 – 6 are done, build a vocabulary of distinct words as a JSON object and output it to a file called `task6.json`. Each key is a word and the value is an array of the `news_id` of articles containing that word. The entries in `task6.json` should be in ascending order of word and the list of articles for each word should also be in ascending order of `news_id`. The format of each key-value pair of the JSON object should be

   ```
   "word": ["story_reviews_xxxxx", "story_reviews_yyyyy"]
   ```

8. The creation of vocabulary should be implemented reasonably efficiently. The run time of task 6 should be no more than 45 seconds. Excessively long execution time for this task will result in a deduction of 1 mark. For example, using the `re` package is likely to be more efficient than writing for loops.

Note that you might want to use an output from a previous task for convenience, but this is not required.

To run your solution, open the terminal and run `python main.py task6`

Task 7. Detecting most indicative words of fake news (4 marks)

What is fake news? What is real news?

In the context of this dataset, an article is considered real or legitimate if its rating is at least 3. On the other hand, a fake news article has a rating below 3. Based on this, you should be able to separate the articles into two collections:

- Fake news: articles with a rating below 3; and
- Real news: articles with a rating of 3 or higher.

Odds ratio

In this task we will use the vocabulary from Task 6 to discover which words are the most indicative of fake news.

First, define the probability that a word w in our vocabulary appears in a real news article as

$$p_r(w) = \frac{\#\text{ real articles containing word } w}{\#\text{ real articles}}.$$

Further, the odds that a word w appears in a real article is defined as

$$o_r(w) = \frac{p_r(w)}{1 - p_r(w)}.$$

If you have done some probability, odds are an equivalent way of presenting a probability. Odds essentially compare the number of events that produce an outcome against the number that do not: the higher the probability, the higher the odds. An odds value greater than 1 indicates that the word w is more likely to appear in a real news article than it is absent.

Similarly, we define the $p_f(w)$ and $o_f(w)$ for each w in fake news articles.

Note that in your calculation, to avoid odds of 0 or infinity, you can exclude words w such that $p_r(w) \in \{0.0, 1.0\}$ or $p_f(w) \in \{0.0, 1.0\}$.

Now, for each word w in the vocabulary, we have two values: $o_r(w)$ and $o_f(w)$. Define the odds ratio for fake news to be

$$or(w) = \frac{o_f(w)}{o_r(w)}.$$

This ratio compares the odds that w appears in a fake article against the odds that w appears in a real article. A fraction less than 1 indicates w is more likely to appear in a real article; a value greater than 1 indicates w is more likely to appear in a fake article; and a value of exactly 1 indicates w appears equally frequently.

To make it easier to compare odds ratios (for fake news), we transform it into $lor(w)$, or `log_odds_ratio`, for fake news using the logarithmic function:

$$lor(w) = \log_{10}(or(w)).$$

This way, the `log_odds_ratio` (for fake news) is positive if $or(w) > 1$ and is negative if $or(w) < 1$. The `log_odds_ratio` is 0 if $or(w) = 1$, which indicates that w is not more representative of fake nor of real news .

Your tasks

Implement the function `task7()` in `task7.py` that does the following:

- Calculate the `log_odds_ratio`, for fake news, for each word in the vocabulary.
  - Exclude the words where p_r or p_f is exactly 0 or 1 (words that appear exclusively in real news or fake news).
  - Exclude the words which appear in fewer than 10 articles and words that appear in all articles.

  With the remaining words in the vocabulary:

- Output a csv file called `task7a.csv` with the following headings: `word`, `log_odds_ratio`. Each row should represent a word in the vocabulary and the log odds ratio for fake news of that word. The value of `log_odds_ratio` is to be rounded to 5 digit from the decimal point. The entries in `task7a.csv` should be in ascending order of word.
- Output a file called `task7b.png` which contains an appropriately chosen graph, showing the distribution of the log odds ratios for all words.
- Output a file called `task7c.png` which contains an appropriately chosen graph, showing the top 15 words with the highest odds ratios for fake news, and the top 15 words with the lowest odds ratios.

To run your solution, open the terminal and run `python main.py task7`

Task 8. Analysis Report (6 marks)

Write a brief report of no more than 500 words to summarise your analytical findings. You should incorporate the plots in tasks 4, 5, 6 and 7 in your analysis and also include the following:

- A brief descriptions of the data sources and of the plots from tasks 4, 5, 6 and 7. (0.5 mark)
- Comment on the credibility of news sources based on task4b.png. (0.5 mark)
- Your observations on the average number of tweets for each rating group based on task5.png. Specifically, is it true that the more credible a news article is, the more it is retweeted? (1 mark)
- Interpretation of the distribution of the log odds ratios for all words as shown in the plot in task7b (task7.png). (1 mark)
- The extent to which you agree or disagree with the "most indicative" words of fake news and words of real news produced from Task 7c, as shown in task7c.png. Relate this to your personal experience reading health-related news and whether it could help you detect an illegitimate article effectively. (1 mark)
- A brief discussion of the limitations of this dataset, the processing techniques and what can be done in the future.' (1 mark)

The report should be coherent, clear, and concise, and written in an appropriate language. (1 mark)

Submit your report by uploading the pdf file called `task8.pdf`

To check if you have uploaded the file correctly, open the terminal and run `python main.py task8`