# C4 – Assignment #2 – Web-based Data Analysis

The `ea-thesaurus-lower.json` word associations file (discussed in class) contains a dictionary of word associations for 8,210 individual words (as a single JSON file). When a word in the word associations dictionary is looked up, the associations are returned as a list of individual dictionaries which themselves associate another word with a score.

You are to build a webapp which allows its users to upload any file of textual data for analysis. There are two analyses to perform: basic and advanced.

### Basic Analysis

Upon receiving the text file, your webapp processes the file as follows:

- The text in the file is converted to lowercase.
- Each individual word (i.e., token) in the text is looked-up in the word associations file.
- For each unique word found, the top three scoring associations are to be remembered.
- If a word is **not** found in the word associations file, this fact is to be remembered, too.
- For every word found (or not found), a frequency count for the word is to be provided.

See overleaf for an example analysis.

### Advanced Analysis

You are further asked to "score" each unique word appearing in the uploaded text file based on it appearing in the word associations file. Specifically, your user wants to be able to process the body of text and score (i.e., rank) each of the unique words as follows:

- A frequency count for each word is to be determined (`freq`).
- Each unique word in the text is to looked-up in the word associations file, and either the first, second, or third word association and score is to be retrieved, with which one to use controlled by a user-configurable setting (`association_score`).
- The `freq` and `association_score` values are to be multiplied together (`rank`)

Once the analyses are complete, your webapp needs to provide a mechanism which allows a user to look up any previous results. It is not enough to provide the results of the most-recent analysis: you must provide a lookup mechanism which allows the user of your webapp to lookup the results of any previous analysis. Additionally, your system is to only preform an analysis on a text file when required: if a text file has already been processed by your webapp, its remembered results are to be looked-up and displayed (i.e., the analysis is **not** to be repeated needlessly).

## Assignment Specification & Notes

1. Use Python 3 as your programming language on your web server (built with Flask), and use MongoDB as your backend database system. You are **not** required to store the word associations JSON file in MongoDB (unless you want to).

2. You are **not** required to host your solution on the cloud (on your computer is fine).

3. Be sure to test your solution.

4. E-mail your solution code (as a ZIP file) to `paul.barry@itcarlow.ie` by the due date/time.

5. This CA and is worth 10% of your final mark (up to 5 marks for the basic, and 5 marks for the advanced). While this CA is active, no new material will be presented in class.

6. You may be required to demonstrate your system running on your computer (so be ready).

7. Due date: **Friday, February 10th 2017**. Due time: end-of-day.

## Example Analysis

Given the following text in a file:

```
There are examples, and there is The Correct Way to do something.
```

Start by converting the text to lowercase:

```
there are examples, and there is the correct way to do something.
```

**Gotcha**: how are you going to handle that comma and full-stop? How are you going to handle punctuation in general?

Convert the text to a list of tokens:

```
['there', 'are', 'examples,', 'and', 'there', 'is', 'the',
 'correct', 'way', 'to', 'do', 'something.']
```

Lookup each token in the word associations JSON dictionary and pull out the top three scores, noting any words which aren't found (note the frequency counts for each word provided in parentheses):

```
and (1) -> [{'but': '32'}, {'or': '15'}, {'so': '7'}]
```

```
are (1) -> [{'you': '27'}, {'is': '15'}, {'not': '13'}]
correct (1) -> [{'right': '38'}, {'wrong': '14'}, {'incorrect': '11'}]
do (1) -> [{"don't": '22'}, {'not': '18'}, {'you': '6'}]
is (1) -> [{'not': '25'}, {'was': '18'}, {'are': '8'}]
something (1) -> [{'nothing': '28'}, {'else': '9'}, {'good': '8'}]
the (1) -> [{'a': '16'}, {'end': '15'}, {'cat': '4'}]
there (2) -> [{'here': '48'}, {'where': '11'}, {'then': '7'}]
to (1) -> [{'from': '17'}, {'go': '12'}, {'be': '7'}]
way (1) -> [{'out': '27'}, {'path': '14'}, {'road': '11'}]

examples (1) was not found in the associations list
```

**Hints**:

- Be sure to check the Flask documentation for how to upload a text file to a web server.
- Don't forget about the `Counter` module in the `collections` library.
- If you have a dictionary (or a `Counter`) called `data`, you can process an ordered version of it using `for k, v in sorted(data.items()): print(k, v)`
- The `PyMongo` documentation (especially the tutorial) is **required reading**.
- Advanced (and optional): the more adventurous of you may which to consider using some of the facilities provided by NLTK (`nltk.org`).