Screen 6 – NodeJS Test Case

Task

Show the top 10 most occurring words in a set of documents retrieved from a list of URLs.

The application is divided to three parts, the **Launcher** (*app.js*), **DataFetch** class (*data-fetch.js*) and **TextAnalyzer** class (*text-analyse.js*).

The Launcher:

Create an instance of **DataFetch** class, initiate it with the list of URLs of the text-files, set the configurations of request-retry,.

I have tried to make it flexible, so the user can change the strategy of dealing with errors, what means which kind of errors the application should retry fetch data from text-file source - Network errors, User side errors or others -, maximum number of attempts in case of failure, delay between retrying.

And create an instance of **TextAnalyzer** class, initiate it with the text fetched via the **DataFetch** object, then call the function responsible for finding the most occurred words.

The DataFetch class:

I used *requestretry* package, before I came across it I was thinking to use time-intervals and array of attempts for each URL.

I used *async* package to run the requests parallel, before I came across it I was thinking to use events to call the data handler after fetching all files. I have faced difficulties with accessing the DataFetch object using this but it was referring to the process, not the object, so I assigned the object to temporary variable to use it inside the callbacks.

The TextAnalyzer class:

This class is responsible of analyzing the text and figuring out the mot occurred words in the text, I made it more flexible for dealing with common words, propositions, to-be verbs, adverbs… and how many desired words, and how to deal with words have the same number of occurrences.

How to use:

Initiate an instanse object of DataFetch class with list of URLs

```
var dataFetchObj = new DataFetch( listOfUrls );
```

Set the configurations of retry-attempts

```
dataFetch.setConfigurations({
  maxAttempts: num,  retryDelay: num_ms
});
```

Start request data from URLs and pass the results to the data-processing function

```
dataFetchObj.startFetch( processDataFunction );
```

Inside the processDataFunction initiate new instanse object of TextAnalyzer

```
var textAnalyzerObj = new TextAnalyzer( textToAnalyze );
```

Set the configurations of analyzing text-file

```
textAnalyzerObj.setConfigurations({
  topWordsCount: num,
  // the number of top words wanted
  excludeCommonWords: booleanValue,
  // strategy of dealing with common words
  excludedWords: exludedWordsString,
  // excluded words from compariion
  excludeNumbers: booleanValue,
  // strategy of dealing with numbers
  fairStrategy: booleanValue,
  // how to deal with words have the same number of occurrences
});
```

Get the top-N-words

```
var results = textAnalyzerObj.getTopNWords();
// the result is array of pairs word/count
```