Data Science Capstone Milestone Report

1. Problem

The primary purpose of the project is to find similarities between articles about terrorist attacks as reported by The Guardian. Using document clusterization and topical analysis, the goal is to identify common expressions and language used to report on the attacks and find out if there are differences in how different attacks are being covered. The initial assumptions is that it should be possible to distinguish meaningful groups of articles based on the place of an attack as mentioned in the Paris, Beirut, and the Language Used to Describe Terrorism article.

The secondary purpose of the project is to create a reusable pipeline for document cluseristation, which could be reused to power bespoke data visualisations.

2. Approach

We're evaluating feasibility of all the following steps of the clusterization pipeline

a. Tokenizing and stemming each article
b. Calculating term frequency using TF-IDF
c. Calculating distance between different articles
d. Clustering articles using k-means, or similar
e. Potentially other steps to improve results of the clustering (e.g. LDA, WARD clustering)
f. Custom visualisation of the results

3. Challenges and issues

2.1 Fetching the articles

2.1.1 Fields of interest

We'll be mainly interested in the content of the articles which is stored under the 'fields.body' object. For further filtering, the 'sectionId' property of the document will also be of interest.

2.1.2 Paginated API

Since the Guardian API is paginated, it is necessary to re-architecture the fetchers module so that after receiving the response, we can check the ‘page’ and ‘total’ properties to see whether we have need to fetch another page of the results.

2.1.3 Minimizing requests

To minimize the amount of request we have to make (there is a per-hour and per-day limit on the Guardian API), we raise the number of returned articles per response to 50, the maximum allowed.

2.1.4 Filtering sections

Based on the previous data exploration, we’ll be working only with articles from the ‘World’, ‘UK News’ and ‘US News’ sections. We want to avoid book reviews, opinion pieces and others to interfere with our analysis. We’re interested only in the language of the reporting on the terrorist attacks.

2.2 Cleaning data

2.2.1 Removing HTML

The returned articles contain HTML markup, e.g. links. This content ends up confusing the tokenizer process, since the same word with and without hyperlink will be considered a different token.

To mitigate this, we’ll have to make sure to strip the content of all the HTML markup.

2.2.2 Removing proper nouns

Based on the first clusterization results, it’s obvious that names of places and people play a significant in assessing similarity of articles. To avoid this effect, we’re interested in shared language after all, we’ll need to find a way how to remove all proper nouns before we start with the clusterization

3. Clusterization

With the first test run on the full dataset (over 40,000 articles), it’s obvious that the K-Means clusterization algorithm from sklearn is taking far too long (over 4 hours). After some research, the Mini-batch K-means from the same library has been tested and results confirmed, that this algorithm performs much better on bigger datasets.

4. Performance

It quickly became obvious that to run the entire clusterization pipeline consumes almost 100% RAM and CPU power of the development machine for up to few hours. There will be some critical steps needed to reduce the processing time.

4.1 Debugging

To be able to accurately evaluate which parts of the pipeline are taking the longest to execute. The performance module has been implemented to provide a convenient way of tracking time lapsed between different parts of the pipeline.

4.2 Caching results

To reduce the processing time, we want to avoid repeating unnecessary heavy operations as much as possible.

To save processing time, a result of each operation is stored into a file, using sklearn joblib library. To be able to retrieve the cached results, a document is stored in the cache collection of the MongoDB database. This document contains a path to the object with stored results of the operation, and also query objects with parameters used to run particular operation. This way we make sure, we don't retrieved an incorrect cached results computed with different parameters.

For each module, we firstly check whether there's a cached result for the same operation and parameters. If there is, it loads the file with serialized results and returns it, instead of rerunning the computation. Thus saving a substantial amount of time.

This caching mechanism is implemented in the caching module.

4.3 Cloud computing

It might not be possible to run the clusterization pipeline on a development machine as many as time as necessary. One solution is to run all the scripts on a different machine, preferable rented virtual machine from one of the cloud providers.

Amazon EC2 is one of the most popular solutions and it does offer a wide range of configuration, which should allow to run the clusterization pipeline efficiently many times without blocking the development machine.