**Text Analytics documentation**

Github main link : https://github.com/serendio-labs

Github stage link : https://github.com/serendio-labs-stage

Detailed information about the different workspaces and relevant documentation are provided in the respective ReadME files.

**Possible future improvements :**

**Adding TextRank based keyword extraction utilizing GenSim into Text Analytics package:**

Recently we have utilized a keyword extraction algorithm called TextRank for Synapsify that might be useful to add it alongside RAKE in our Text analytics package. This algorithm works by creating a graph of co-occuring words and extracts the words/phrases which are linked more number of times similar to Page Rank algorithm

**Adding customized stop words feature for RAKE based keyword extraction :**

The RAKE keyword extraction algorithm extracts the keywords based on word phrase frequencies and Stop word lists. Mutiple stop words lists according to different domains could be utilized in improving the RAKE based keyword extraction.

**Improving the performance of Sentiment extraction module:**

In the sentiment calculation two modules namely Vectorizer and Chi-square-transformation takes up 95% of the sentiment calculation time.

* Vectorizer takes up 25% time whereas Chi-square transformation takes up 70% of the time.
* The main functionality of Chi-square transformation is dimensionality reduction.
* For first step we can probably test if removing the Chi-square transformation has any significant effect on accuracy. If it doesn't affect accuracy very much then the transformation could be removed.
* An alternate method is to use more faster libraries / distributed computing to perform chi-square transformations. Spark could be a good choice for distributed computing, its mllib package seems to have an implementation of chi-square transformation.

**Providing a single Thrift based call to extract all Text analytics parameters:**

Currently for clients such as PATHAR each of the text analytics components such as Sentiment, Topics, etc are called one by one resulting in multiple Thrift calls. This could be improved by creating a separate Thrift API which automatically invokes all the Text analytics modules on a given piece of Text.

**Improving the Rest API quality by performing thorough validations and providing good quality documentation with examples:**

The current supported REST api could be improved by handling File input, storage of output Data into Databases/Cloud, Performance testing with Jmeter and accuracy testing with Tagged input and output data.

**Integrating logging framework into the Diskoverer platform :**

Logging frameworks such as log4j and slf4j could be added to the Diskoveror platform with multiple levels such as INFO, WARN, DEBUG, CRITICAL and ERROR at appropriate places.

**Designing and creating appropriate UI for proper data visualization:**

The different Text analytics components could be visualized effectively in a web applications with Java script libraries such as D3.js, Bootstrap.js etc.