**Text Analytics and Entity Resolution Using Apache Spark**

**Executive Summary**

There is a vast and rapidly increasing quantity of online news content, each source publishing or covering almost the same news - content wise. Since these articles or entities do not share any common attribute but still have an underlying relationship, there is inherently the problem of identifying and linking or grouping different data content of the same real world entity. One such solution to this problem can be resolved using Entity Resolution.

Entity Resolution, or "Record linkage" refers to the process of joining records from one data source with another that describe the same entity. Entity Resolution (ER) refers to the task of finding records in a dataset that refer to the same entity across different data sources – in this case news articles from two major Indian news channel, Indian Express and The Hindu. The task of resolving entities and detecting relationships becomes easy with ER, particularly when combining two datasets may or may not share a common identifier.

The objective of this report is to present the result of how Apache Spark can be used to apply powerful and scalable text analysis techniques and perform entity resolution across two datasets of news articles. The report highlights the process of data collection and cleaning, using Entity Resolution technique display similar news articles as group and also display similar news articles as per the search query input from the user, and finally divide the results as per the sentiment carried by the respective news article.

**Keywords** – Apache Spark, Text analytics, Entity Resolution, Sentiment Analysis

**Business Problem**

The report attempts to highlight the following business problem

* Given two dataset of news articles from different sources, identify and display the same news articles which carry the same content or topic
* Identify the sentiment carried by the news articles and present the results as per their polarity – Positive, Negative or Neutral

**Methodology**

The following summarises the methodology used

1. **Web scraping in R**

We performed web scraping in R to procure data from two Indian news channel RSS feed – **The Hindu** and **The Indian Express** using ‘rvest’ package in R. The data was collected for around 40 days at an average interval of 10 hours. We have performed text cleaning and corpus building in R using ‘tm’ package, and extracted the following data

* Title – The main title of the news article
* Link – The web URL of the news article
* PubDate – The published date of the news article
* Desc – A short description of the news article to highlight what the article is about

1. **ER as Text Similarity - Weighted Bag-of-Words using TF-IDF in PySpark**

In a given corpus, some tokens have higher importance than the others, which add value overall. We have used ‘Term Frequency – Inverse Document Frequency’ or TF-IDF to assign the correct weight to each token in the entire corpus. These weights correctly specify which tokens are to be favoured and give better results when comparing different documents

* 1. **Term Frequency (TF)**

Term Frequency computes the frequency of a token in the same document. E.g. if document d contains 100 tokens and token t appears in d 20 times, then the TF weight of t in d is 20/100 = 1/5. Usually the intuition is that if a token appears more often in a document, then it is more important or carries more meaning in the document

* 1. **Inverse Document Frequency (IDF)**

IDF identifies all the tokens that are rare overall in the entire corpus. Rare tokens add more value than the common ones and it also helps in identifying the two documents sharing the rarer token as compared to the common token easily. IDF for a token t in a collection of documents D is calculated as follows

* D is the total number of documents in the entire corpus
* Compute n(t), the number of documents in D which contain the token t
* IDF(t) = D/n(t) (total documents in corpus) / (documents containing token t)
  1. **Term Frequency- Inverse Document Frequency (TF-IDF)**

The total TF-IDF weight for a token in a document is the product of its TF and IDF weights.

1. **Compute Similarity between text using Cosine Similarity**

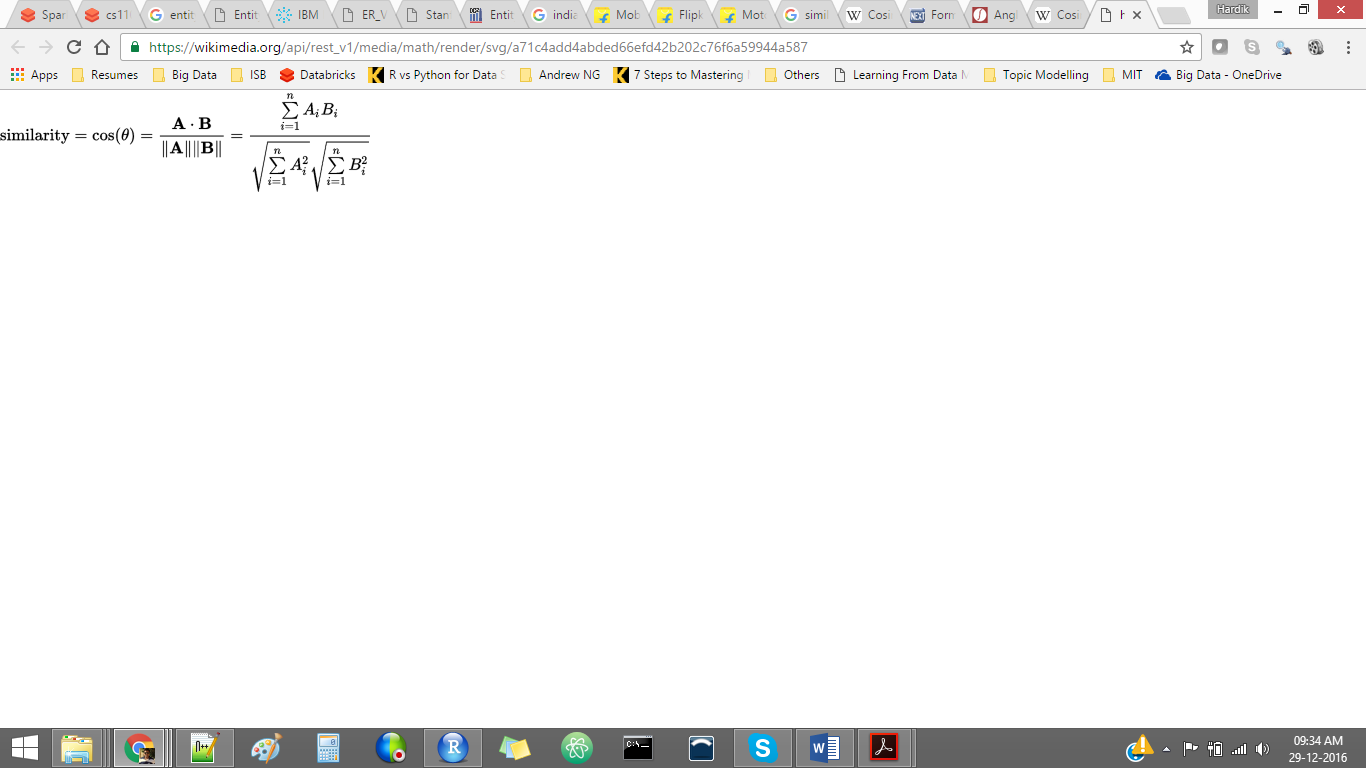
We have used Cosine Similarity as the index to compute distance between the two documents (strings). Here each document is treated as a vector of their TF-IDF weights. Then to compute similarity, the cosine angle between the two documents is calculated.

If *A* and *B* are two documents whose similarity is to be calculated, then we calculate token vectors as

A – Vector of its tokens represented by their TF-IDF weights

B – Vector of its tokens represented by their TF-IDF weights

Then determine their similarity using the formula



Where,

**A.B –** Dot product between the two vectors

**ǁAǁ** - Magnitude of vector A

**ǁBǁ** - Magnitude of vector B

The angle between the two documents is small if they share many common tokens and hence their similarity score is high (ideally = 1). This is because small angle means that they are pointing in the same direction. If the angle between them increases, it means they are pointing in different direction and hence their similarity score tends to 0.

1. **Grouping similar news articles using Python-Igraph package**

Python igraph is the graph package used to create and perform graph analysis. Here we have used this package to group all similar news articles and cluster them together

1. **Sentiment Analysis using TextBlob package**

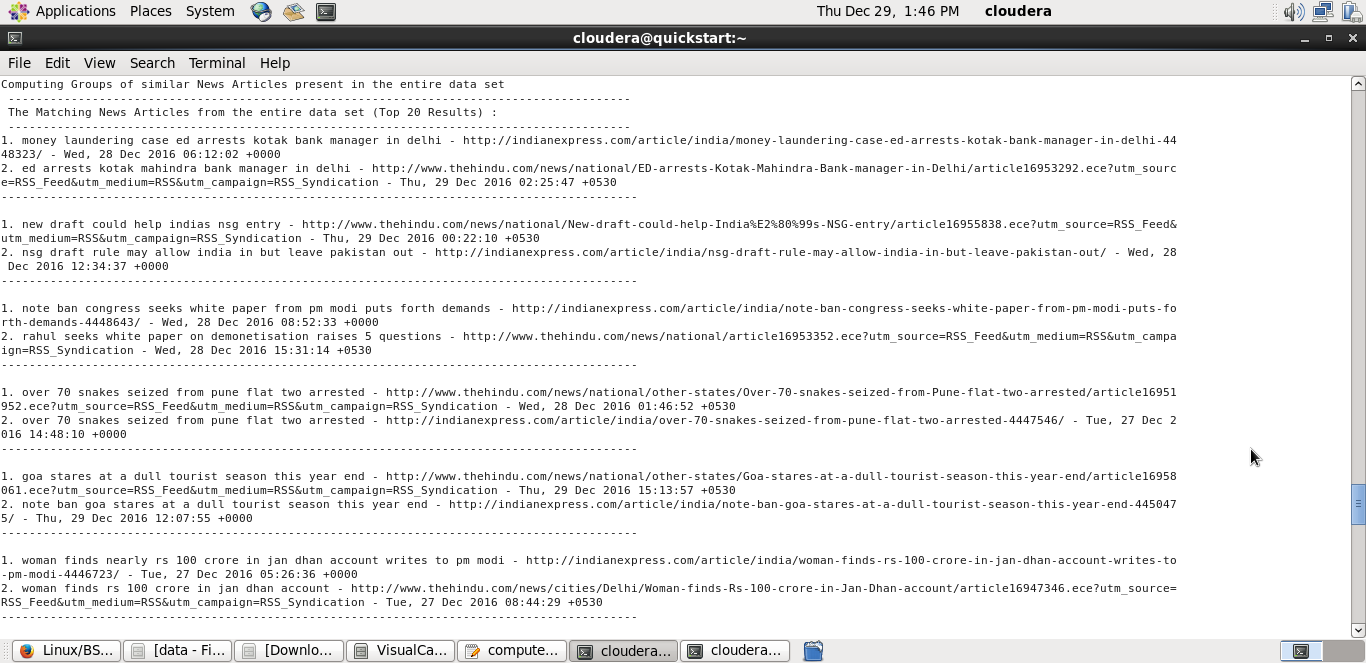
TextBlob is a python library used to process textual data. As part of its API, it provides many common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.

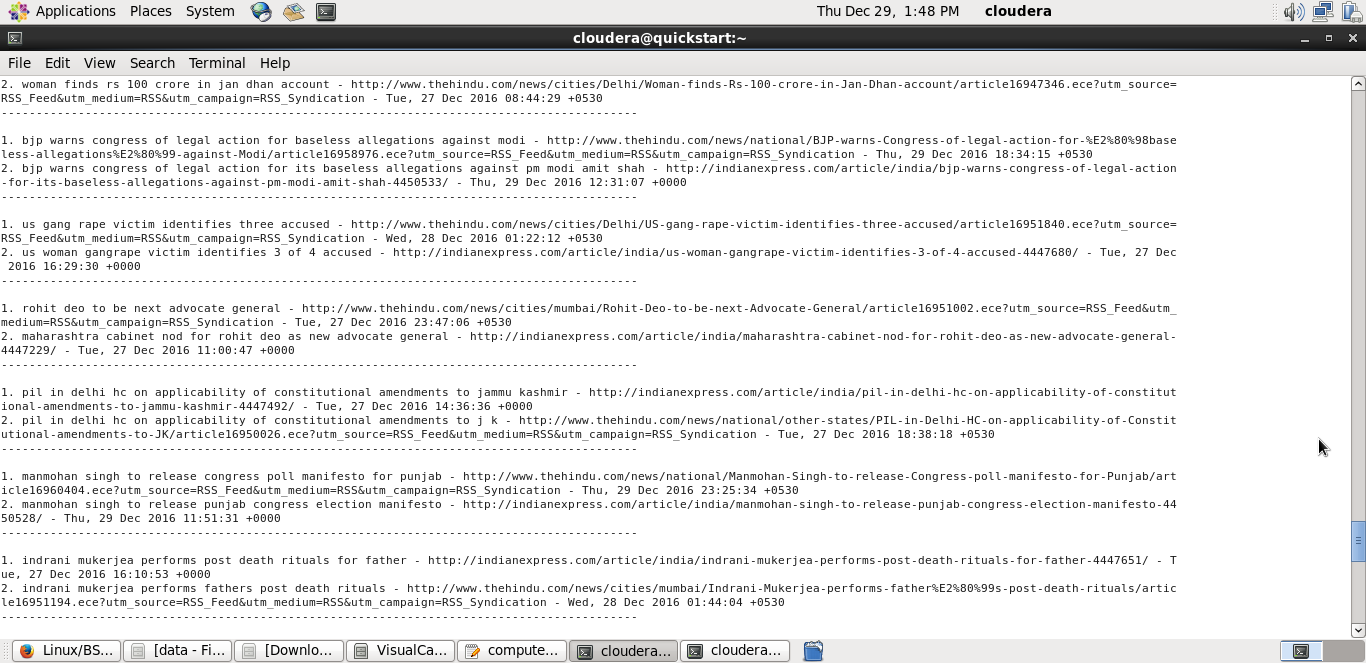
**Data and Code**

* https://github.com/HardikLGupta/BigDataSparkProject

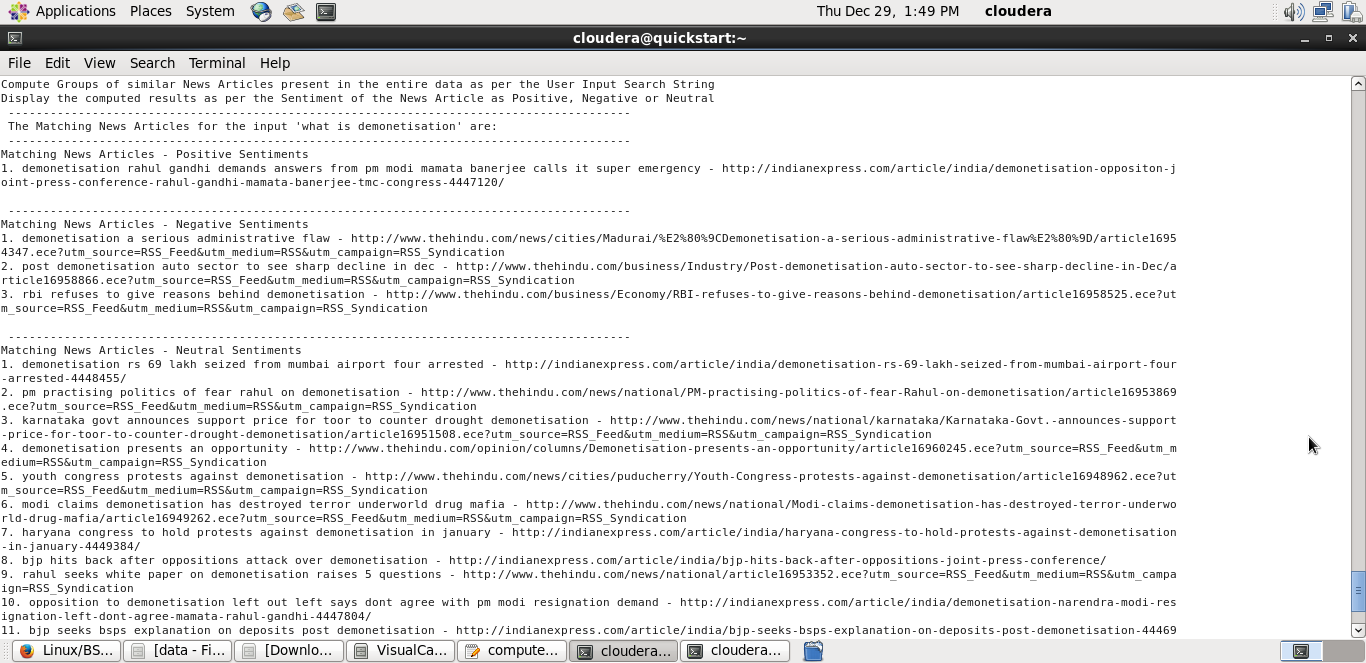
**Analysis**

1. Grouped Similar News Articles

****

****

1. Grouped News Articles as per User input classified by its sentiment

****

**Conclusion**

**Future Scope**

**References**

* <http://www.csd.uoc.gr/~vefthym/er/>
* <http://www.ibm.com/support/knowledgecenter/SS2HSB_8.1.0/com.ibm.iis.ii.overview.doc/topics/eas_con_entityresolution.html>
* <http://infolab.stanford.edu/serf/>
* <https://docs.cloud.databricks.com/docs/latest/courses/Introduction%20to%20Big%20Data%20with%20Apache%20Spark%20(CS100-1x)/Solutions/Module%204:%20Text%20Analysis%20and%20Entity%20Resolution%20Lab%20Solutions.html>
* <http://www.datacommunitydc.org/blog/2013/08/entity-resolution-for-big-data>
* <http://stackoverflow.com/questions/41332988/merge-tuples-having-atleast-one-common-element-to-form-a-common-tuple>