Project Report

Topic: Large Scale Data Collection and Preprocessing in Spark

CS 6350.001

Spring 2019

Team Members

Sri Chaitra Kareddy	**sxk180037**
Kiranmai Seemakurthi	**kns180002**
Nayen Atluri	**nxa180039**
Sreeram Chittela	**sxc180025**
Sukruti Hosundurg	**sxp180014**

May 08, 2019

Introduction:

Political event data are records of interactions among political actors which can be used to analyze political behavior. All these event data are collected and is used to quantify the propagation of events through space and time. This helps us in capturing complex interactions of the actors involved. The objective of the project is to extract data from Spanish news channels and generate details about the article using universal dependence parse trees for each sentence within the content. The extracted details will be used to compare two different articles to determine if they cover the same story.

Related Literature:

It is evident from the objective that it is necessary to crawl news channels for gathering data and extract relevant content. Natural language processing allows us to extract accurate content from each sentence in the article. It also supports multiple languages content extraction without performing word-to-word translation. All the data extracted from the news source will be stored in MongoDB, since it an open source NoSQL document database which stores data in form of key value pairs, makes integration of data easier and faster. This data will then be pipelined to get tokens of word. Detecting near duplicates on the web is challenging due to its volume and variety. A universal and parameter-free similarity metric, the normalized compression distance or NCD, has been employed effectively in diverse applications. To make this parameter-free method feasible on a large corpus of web documents, we propose a new method called SigNCD which measures NCD based on lightweight signatures instead of full documents, leading to improved efficiency and stability. We derive various lower bounds of NCD and propose pruning policies to further reduce computational complexity.

Requirements:

The set of software that we would be using in this project would be as follows:

Newsplease

MongoDB

PyCharm

Ufal-Udpipe package

Implementation:

For each document, we crawled over URLs using NewsPlease. Once metadata are obtained, a MongoDB object for that document is created using that list. The map function gives us the list of MongoDB objects. Each of them is a JSON-like object that contains all the metadata for that document. Processing this data and generating relevant dictionaries for actors that capture the actions or the statements involved in the news article is a challenging task in this scenario. It is laborious to manually create these dictionaries. The udfal - udpipe model of *udpipe.models.ud/spanish-ancora-ud-2.1-20180111* python package's trained model was used to create token of words. Later, parameter-free methods with a special similarity metric called normalized compression distance or NCD was used. NCD is measured by exploiting the off-the-shelf compressors to estimate the amount of information shared by any two documents. It has been proven to be universal and can naturally be applied to a variety of domains such as genomics, languages, music, and images. To deal with large collections of documents of very wide range of lengths, a new near duplicate algorithm called SigNCD which combines signature extraction process with normalized compression distance was implemented. Specifically, a punctuation-spot signature extraction method, which is robust and can be applied to different languages.

The normalized compression distance (NCD), formally defined as

$$\mathrm{NCD}(x, y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}}.$$

Here, C(xy) denotes the compressed size of the concatenation of x and y, and C(x) and C(y) denote the compressed size of x and y, respectively. NCD can be explicitly computed between any two strings or files x and y. Experimental evaluation shows that SigNCD outperforms NCD with an improvement in runtime. Signature extraction is the key part of SigNCD, aiming to capture the core contents for similarity matching. A simple choice is using punctuation as spots, which are likely to occur in every document and whose occurrences are widely and uniformly spread out in the documents. Hence punctuation-spot signatures extract the words around a subset of punctuations to construct a signature for each document. We refer to the signature extracted from a document x as sig(x) , which would be compressed by off-the-shelf compressors. Then the size of the compressed signature, denoted as C(sig(x)), would be used as NCD input. Besides, to measure the similarity between documents x and y, the concatenation of sig(x) and sig(y) would also need to be compressed, and the size is denoted as C(sig(x)sig(y)) . As compression is generally time-consuming, compressing signatures instead of full documents can significantly reduce computational complexity. Given the compression sizes as inputs, the normalized compression distance of a pair <x,y> based on signatures, denoted as SigNCD(x,y), can be simply obtained through:

$$\mathrm{SigNCD}(x, y) = \mathrm{NCD}(\mathrm{sig}(x), \mathrm{sig}(y)) = \frac{C(\mathrm{sig}(x)\,\mathrm{sig}(y)) - \min\{C(\mathrm{sig}(x)), C(\mathrm{sig}(y))\}}{\max\{C(\mathrm{sig}(x)), C(\mathrm{sig}(y))\}}.$$

<x,y> is detected as a near duplicate if SigNCD(x,y)<=Ƭ, where Ƭ is the similarity threshold, we used a threshold of 0.1.

Pruning was implemented to filter unnecessary comparisons. We used pruning with lower bound.

Lemma 3. When comparing a pair of objects <x,y> , if C(y)>=C(x) , then (1-C(x)/C(y)) is a lower bound of NCD(x,y).

Observations: The retrieved results are covering very similar stories in terms of context using the deduplication algorithm.

Challenges: We had to face multiple challenges while trying to implement this project. For Example, when we tried using Stanford NLP instead of Ufal-Udpipe package our system crashed. This happened because our system's RAM was insufficient to process/handle the implementation of Stanford NLP.

Schedule:

Task	Start Date	Time Allotted
Crawl News Channels	04/17	1 week
Pipeline data to get parse trees	04/24	1 week
Deduplication was implemented	05/01	5 days
Store data in MongoDB	05/06	1 day