##### **Team Members:**

1. Karan Patel - kxp173930
2. Pandit Joshi - pxj171830
3. Prashanth - sxp178231

# Large Scale Data Collection and preprocessing in Spark

### **Objective:**

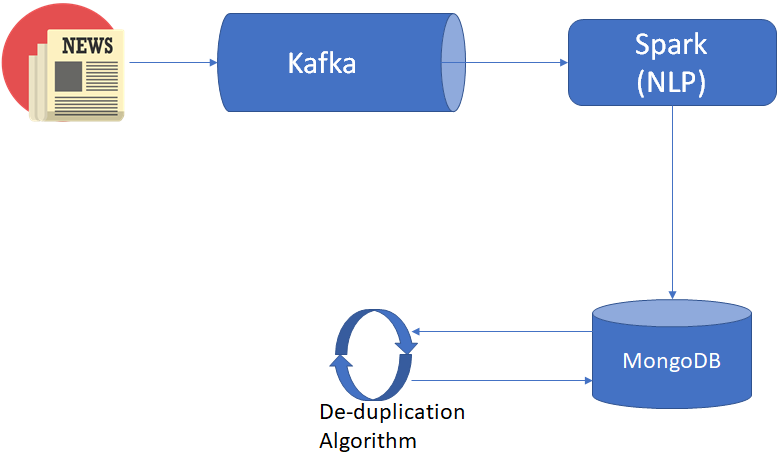
To summarize the news articles and possibly figure out duplicate articles using event encoders in a near real-time manner running the processes in Spark based system.

### **Our Approach:**

Stage 1: Web Scraping & Article Storage

Stage 2: UD-parse

Stage 3: Preprocessing and running DBSCAN algorithm to find the duplicate articles



### **Stage 1: Web Scraping**

1. We feed a list of *spanish news website* URLs as input.
2. We use **Scrapy’s** spider module to get all links from these URLs.
3. We filter the above links to only those which are articles.
4. The links obtained in step 2 are then fed to***Newsplease*** one at a time, which retrieves the article and its metadata such as:
   1. Title
   2. Authors
   3. Description
   4. Language
   5. Source domain
   6. Text
   7. Main image URL
   8. Published date
   9. Article URL
5. The articles are stored in json format in MongoDB.
6. We then proceed to the next stage - NLP processing

### Stage 2: Reading raw news data from MongoDB and processing with NLP tools to generate UD parse for each statement.

We are using the stanfordnlp pipeline for processing each sentence in the news. Data will be fetched from MongoDB and passed through the pipeline to generate tokens and dependency parse of the sentences.

Dependency parse will be stored back to MongoDB to enable further processing.

### Stage 3: Preprocessing and running DBSCAN algorithm to find the duplicate articles

We are using the DBSCAN algorithm on the output data after processing the raw data by the UD-parse and PETRARCH2. Since our purpose is to cluster duplicate documents, however, clustering can be applied to numerical data only. Hence, we have to devise a method by which we could quantify the textual information present in the documents in a certain way such that we can measure the similarity or dissimilarity between documents. In this project, we have tried to do just that. We have first quantified the text data in documents and obtained a range of corresponding numerical data, from which we used with a popular clustering algorithm to cluster the documents.

The steps performed can be listed sequentially as follows.

Step 1: Getting the data from Mongo DB and doing initial Text Preprocessing

Step 2: Term Weighting

Step 3: Computation of similarity between documents based on their term weights.

Step 4: Clustering

Step 5: Cluster performance evaluation

### In Step 1 we do some basic pre-processing on our initial text data. Here we remove punctuations, stopwords, numbers etc from the initial data.

### In Step 2 we find the term weights for each document i.e. we weight each word in accordance with their relevance to the document.

### In Step 3, we compute the similarity between texts based on the computed weights on the previous stage.

### In Step 4, we perform the main task of the project i.e. the clustering of the text.

### In Step 5, we evaluate the performance of the clusters formed. We designed the system to automatically find the best parameters for clustering resulting in the best performance in clustering.

