A comparison of different clustering techniques

Open Library case study

Attenni Giulio, Monaco Marco

attenni.1756298@studenti.uniroma1.it

monaco.1822895@studenti.uniroma1.it

Big Data Computing – Project presentation
MSc Computer Science
Università di Roma "La Sapienza"

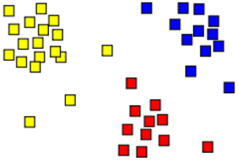
July 9, 2021

Outline

Objective

SAPIENZA
UNIVERSITÀ DI ROMA

- Our main objective is to compare two different techniques to perform cluster analysis.
- Clustering is an NP-complete problem, thus we manage to find just approximate solutions.
  - Centroid Model: the well-known K-means randomized algorithm.
  - Graph-based Model: our idea is to reduce the clustering problem to the problem of computing the connected components of a graph, which has a polynomial solution. Of course, the input instance will be an approximation of the clustering instance, therefore the solution will be an approximation too. ([HS00] is another example of graph's connectivity-based approach)



Case Study

- Open Library is an online project of the Internet Archive intended to create "one web page for every book ever published" [Opea].
- They make available more than 20.000.000 books, and meta-data are freely provided to developers.

Project Overview

Data extraction

- JSON to CSV
- Data cleaning
- Feature Engineering

Models' construction

- Graph definition
- Compute graph's connected components
- Compute k-means clusters

Evaluation

- Comparison of Silhouette coefficient and execution time of the results obtained with the two techniques

Outline

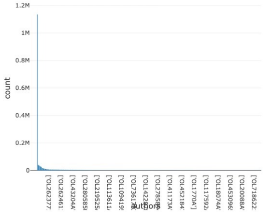
SAPIENZA
UNIVERSITÀ DI ROMA

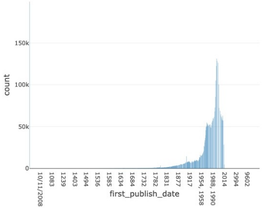
JSON to CSV

- Open Library provides data dumps downloadable from [Opeb].
- We have downloaded "works dumps", a 11.8 GB txt file in which json records are stored.
- Json records have been parsed into a csv file keeping only the properties listed here, resulting in more than 22.264.179 entries for a total of 4.86 GB of data.

Preprocessing

- Preprocessing Analisys:





- Data Preprocessing:
  - Drop features with more than 0.7 null entries
    - Remaining features: ["key", "title", "subjects", "authors"]
  - Drop null and duplicate entries
    - Number of remaining entries: 17.970.902

Feature Engineering

SAPIENZA
UNIVERSITÀ DI ROMA

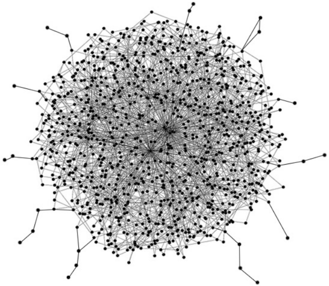
- We decided to use the standard NLP preprocessing pipeline.
- Steps:
  - Text cleaning; i.e., case normalization, trimming, filter out punctuation symbols and extra whitespace.
  - Tokenization; i.e., split text into tokens
  - Stopwords removal.
  - Stemming (Snowball stemmer), optional.
- Word2Vec for text features, which resulted in 70-dimensional vectors for each feature, except for 'authors' whose vector has 10-dimensions.
- Single vector feature resulting from the assembling of all features.

Outline

SAPIENZA UNIVERSITÀ DI ROMA

Graph formulation

- At this point every book is represented by a vector in a (2x70+10)-dimensional space.
- We can define the graph as follows:
  - Nodes:
    $V = \{v | v \text{ is a vector representing a book}\}$
  - Edges:
    $E = \{(u, v) | sim(u, v) > \varepsilon\}$
    where:
    - $sim(u, v)$ is the cosine similarity
    - ε is a predefined threshold



Graph Implementation

- To build the graph, we have used GraphFrames which allows us to easily define the graph and compute the connected components.
- We had to tune the similarity threshold ε, the optimal value resulted to be 0,95.

```
from graphframes import *
sim_udf = udf(lambda x,y: float(x.dot(y))/float(x.norm(2)*y.norm(2)), DoubleType())
def build_graph(df, thr=SIM_THRESHOLD):
  nodes_df = df.select(["id"]).cache()
  sim_df = engineered_books_df .alias("src")\
    .join(engineered_books_df .alias("dst"), col("src.id") != col("dst.id"))\
    .select(
        col("src.ID").alias("src"),
        col("dst.ID").alias("dst"),
        sim_udf("src.features", "dst.features").alias("cos_sim"))\
    .sort(desc("cos_sim")).cache()
  edges_df = sim_df.filter(sim_df.cos_sim>thr).cache()
  return GraphFrame(nodes_df, edges_df)
```

K-means Implementation

- To build the cluster, we set K equal to the number of connected components in the graph computation.
- We had to tune different hyper-parameters:
  - the tolerance, whose optimal value resulted to be 0.000001
  - the maximum number of iterations, whose optimal value resulted to be 100

Outline

SAPIENZA
UNIVERSITÀ DI ROMA

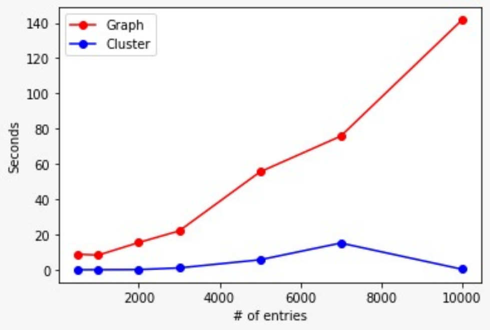
Metrics

SAPIENZA
UNIVERSITÀ DI ROMA

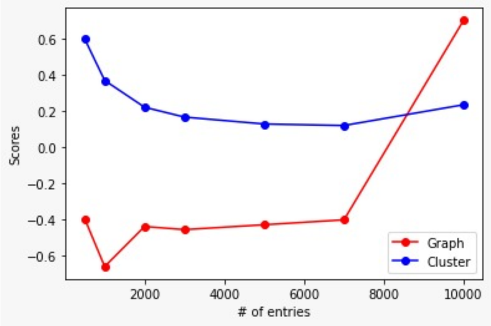
- Evaluation aims at comparing the goodness of the two methods' solutions and the execution time.
- Metrics:
  - Silhouette Coefficient: used to evaluate the goodness of the clusters; this metric takes into account both intra-cluster similarity and inter-cluster similarity.
  - CPU-time: to avoid to keep into account idle time due to the databricks platform.

Experimental Results

- Dataset Size vs CPU time



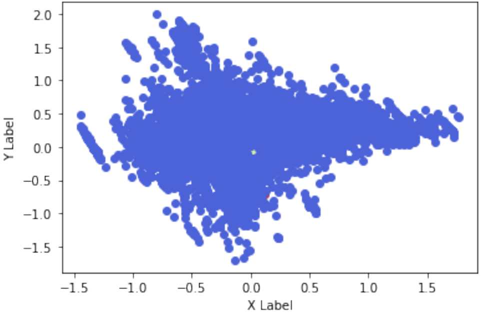
- Dataset Size vs Silhouette Coefficient



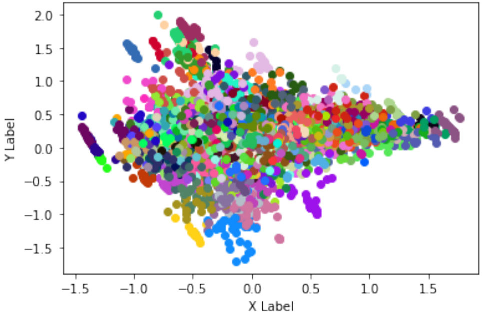
Experimental Results - 2D

SAPIENZA UNIVERSITÀ DI ROMA

- Graph connected componetnts' plot with PCA 2D vectors

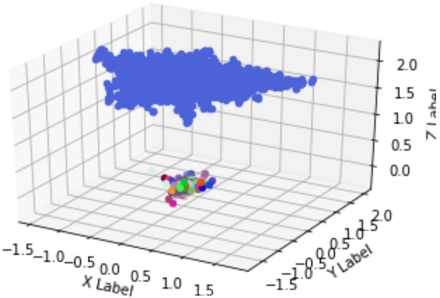


- Clusters' plot with PCA 2D vectors

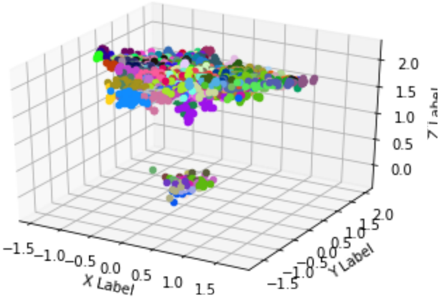


Experimental Results - 3D

- Graph connected componetnts' plot with PCA 3D vectors



- Clusters' plot with PCA 3D vectors

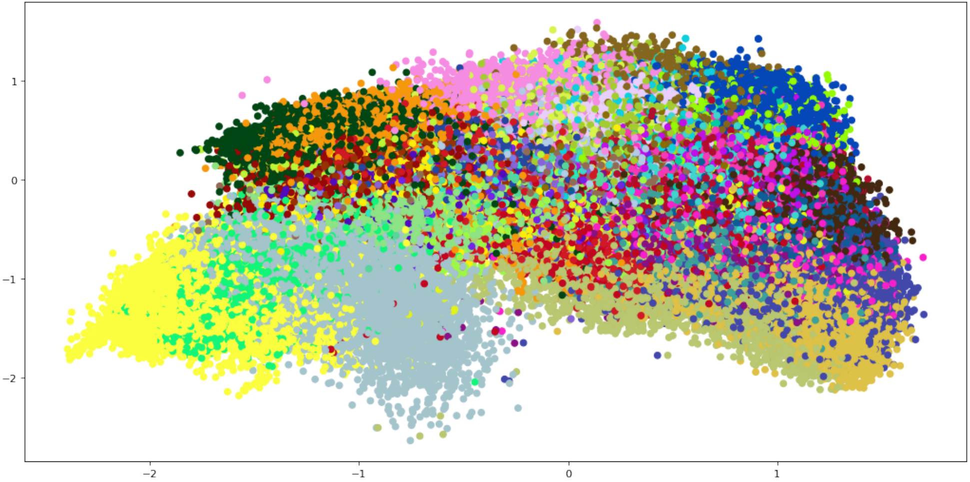


Limitations

- The main limitation we encountered is due to databricks platform
  - The total execution time was much grater than the CPU time
    - In particular, for computing the graph's connected components, the platform overhead resulted in the inability to perform such a task on a big portion of the dataframe. For this reason, we stopped to gather results at 10.000 entries.
    - The computation of K-means suffered less than the one of graph's connected component, thus we managed to perform k-means up to 1.000.000 entries.

K-means Additional Results

- Clusters' plot with PCA 2D vectors



Outline

Sapienza Università di Roma

Conclusions and Future works

SAPIENZA
UNIVERSITÀ DI ROMA

- Conclusions
  - Based on our experiments we can observe that k-means the most of the times provides better results. Moreover, it is also faster than computing connected components.
  - Both methods might be used to solve the cold start issue of recommender systems, finding the cluster that better fits the user's initial preferences.
- Future works
  - More detailed experiments could be pursued to obtain results with general validity.
  - Exploit different kind of similarity metrics other than cosine similarity.

References I

Erez Hartuv and Ron Shamir.
A clustering algorithm based on graph connectivity.
Information Processing Letters, 76(4):175–181, 2000.

Open Library Wikipedia.

Open Library Data Dumps.

Thank you for your attention!