IRRS Lab 7: Map Reduce

Report

Date: December 24, 2025

Students:
Daniel Weroński Falcó
Adrian Hagen

1 Introduction

This report details the implementation and evaluation of a document clustering system using the K-means algorithm within the MapReduce framework. The primary objective was to categorise a subset of the ArXiv dataset into eight distinct scientific domains based on textual similarity. Using the `mrjob` Python library, we simulated a distributed environment to handle the computational intensity of iterative clustering. The laboratory explores the critical relationship between vocabulary selection, distance metrics(Jaccard similarity) and the overall efficiency of parallel processing in a MapReduce architecture.

2 Methodology

The experimental pipeline was divided into four primary stages: indexing, feature extraction, prototype initialisation, and iterative MapReduce execution.

1. **Indexing and Pre-processing:** Documents were indexed in Elasticsearch using a `letter` tokeniser and a Snowball stemmer to reduce words to their linguistic roots. We applied a length filter to discard tokens with fewer than 2 or more than 10 characters to eliminate noise.

2. **Feature Extraction:** Binary term vectors were generated by selecting a vocabulary of 250 words. To ensure cluster discriminability, we applied frequency filters: words appearing in more than 20% of documents (potential stop words) or fewer than 5% (outliers) were excluded.

3. **Similarity Metric:** We implemented the Generalised Jaccard similarity to compare binary document vectors against weighted prototype vectors. This was calculated as:

$$J(doc, prot) = \frac{doc \cdot prot}{||doc||_2^2 + ||prot||_2^2 - doc \cdot prot}$$

   Our implementation used an efficient two-pointer approach, thereby using the alphabetical ordering of the vocabulary to reduce computational cost.

4. **MapReduce Execution:** The `MRKmeansStep` class defined the Mapper for document assignment and the Reducer for prototype re-computation. The control script, `MRKmeans.py`, performed 20 iterations, passing the output of each step as the input for the next.

3 Results

The algorithm was executed for 20 iterations using four parallel processes (`--ncores 4`). The final prototypes exhibited clear thematic specialisation.

The computational performance showed significant variance across iterations. While the first iteration took only 2.88 seconds, the second one already took more than 8 seconds and subsequent iterations' execution times steadily increased to a maximum of 11.99 seconds.

The sharp increase in running time from the first to the second iteration is primarily due to the algorithm's initialization. Initially, prototypes are single documents chosen

Cluster	Top Attributes (Stemmed)	Inferred Topic
CLASS0	how , inform, import, understand, understand	General Research
CLASS1	equat, theori, solut, deriv, space	Mathematics
CLASS2	quantum, phase, physic, experiment, transit	Quantum Physics
CLASS3	increas, temperatur, investig, valu, found	Thermodynamics
CLASS4	prove, ani, given, bound, some, known	Theoretical CS/Math
CLASS5	learn, network, train, deep, dataset	Machine Learning
CLASS6	mass, star, galaxi, stellar, emiss	Astrophysics
CLASS7	optim, effici, bound, linear, complex	Optimisation

Table 1: Top attributes and thematic labels for selected clusters.

at random. These are ”sparse,” containing only a small subset of the vocabulary, which makes the initial Jaccard similarity computations in the mapper highly efficient.

In subsequent iterations, the ”Maximization” step recomputes each prototype as the average of all documents in its cluster. This creates ”dense” prototypes containing a much larger variety of tokens and real-number frequencies. Comparing a document against these dense probability distributions requires significantly more computations of the Jaccard Similarity.

This complexity growth follows a power law consistent with Heaps’ Law, which states that the rate of discovering new words decreases as the total amount of text increases. While the prototypes rapidly gain complexity during early iterations as they ”absorb” the cluster’s vocabulary, the rate of growth slows down as they saturate the 250-word vocabulary, eventually leading to more stable computation times.

Another reason for this increase can be attributed to thermal throttling on the host hardware, which lacked active cooling and reached temperatures in excess of 100°C during the workload.

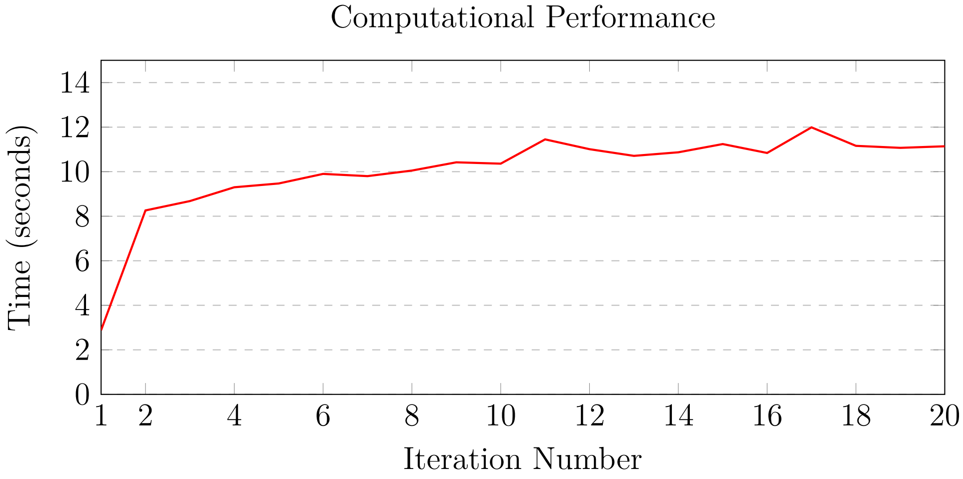

Figure 1: Processing time for each of the 20 K-means iterations. The increase in execution time observed indicates that from iteration 2 onwards, the CPU temperature reaches its 100–101°C limit, triggering thermal throttling.

As illustrated in Fig. 1, the execution time per iteration did not follow the expected pattern of initial overhead followed by stability. Instead, a sharp increase in running time was observed starting at Iteration 2, coinciding with the CPU reaching its thermal limit of 100–101°C. The computation time steadily increases as the primary cooling thermal mass (laptop's cooling chamber) is saturated, but the secondary thermal mass (laptop chassis) begins to absorb heat, leading to a further gradual performance degradation as the CPU increasingly throttles further to manage temperature.

4 Discussion

The results demonstrate the high impact of vocabulary pruning on cluster quality. Initially, using default parameters resulted in clusters dominated by common terms like "the" and "is." By restricting the maximum frequency to 20%, we successfully isolated technical vocabularies that closely align with the original ArXiv folder structure.

However, CLASS0 remained a "generic" cluster, containing terms like "how" and "understand." This indicates that even with frequency filtering, some high-frequency research-meta terms persist and can dilute cluster purity. Regarding performance, the lack of a linear speed-up when using multiple cores is explained by the overhead of inter-process communication in a local simulation and the hardware-induced frequency scaling observed during the mid-iterations.

5 Conclusion

This laboratory successfully implemented a scalable K-means clustering solution using MapReduce. We conclude that the choice of vocabulary—specifically the exclusion of high-frequency terms—is the most critical factor in achieving interpretable document clusters. Despite hardware limitations causing thermal throttling, the system effectively categorised over 58,000 documents into meaningful scientific domains. Future work could involve implementing TF-IDF weighting within the MapReduce steps to further reduce the influence of generic research terminology.