ФИЗИКО-МАТЕМАТИЧЕСКИЕ И ТЕХНИЧЕСКИЕ НАУКИ

УДК 004.891
МРНТИ 28.23.35

COMPARING BIG DATA ANALYTIC TOOLS USING MUSIC DATASET

R.I. BEKTEMIROV, U.T. NURKEY

Suleyman Demirel University

Abstract: *A huge repository of petabytes of data is generated each day from modern information systems and digital technologies such as scientific data analysis, social media data mining, recommendation systems, and analysis on web service logs.The data has a huge power to directly guide us to knowledge detection. Big data in turn requires whole new approach and tools to handle it. Analysing these massive data requires a lot of efforts to extract knowledge for decision making. Huge volumes of data and its unstructured nature raise new challenges and issues regarding its management and processing. This paper covers some of the most popular tools for analyzing big data. Hadoop, Spark and Pig are major and modern tools in big data analytics. Thus and so these tools were chosen for comparison. Results of this research show that various tasks require different tools and there is no all-in-one solution. Any big data problems stand in need developers to use proper tool to make job done in a way better and quicker.*

Keywords: *big data, Hadoop, Spark, Pig, comparison of big data platforms*

ҮЛКЕН ДЕРЕКТЕРДІ ТАЛДАУ ҚҰРАЛДАРЫН ӘНДЕР ЖИЫНТЫҒЫН ҚОЛДАНА САЛЫСТЫРУ

Аңдатпа: *Деректердің петабайттарының үлкен репозиторийі күн сайын заманауи ақпараттық жүйелерден және ғылыми деректерді талдаудан, әлеуметтік медиа деректерін өңдеуден, ұсыныс жүйесінен және веб-қызмет журналдарынан талдау сияқты цифрлық технологиялардан жасалады. Деректер білімді анықтауға тікелей бағыттайтын зор күшке ие. Үлкен деректер, өз кезегінде, жаңа тәсілмен өңделуге арналған құралдарды қажет етеді. Бұл массивтік деректерді талдап маңызды деректерді табу және соған сәйкес шешімдер қабылдау көп күш жұмсауды талап етеді. Деректердің үлкен көлемі және оның құрылымдық емес сипаты оны басқару мен өңдеуге қатысты жаңа мәселелерді тудырады. Бұл мақала үлкен деректерді талдаудың ең танымал құралдарының кейбірін қамтиды. Hadoop, Spark және Pig деректерді талдауға бағытталған негізгі және заманауи құрал болып табылады, сондықтан бұл құралдар салыстыру үшін таңдалды. Осы зерттеудің нәтижелері әртүрлі тапсырмалардың әрқилы құралдарды талап ететінін көрсетеді және барлығы бірдей бір платформамен шешілмейді. Үлкен деректермен байланысты кез келген мәселелер бағдарламашылардың сапалы және жылдам жұмыс жасаулары үшін тиісті құралды пайдалануларын қажет етеді.*

Түйінді сөздер*: үлкен деректер, Hadoop, Spark, Pig, үлкен деректерді талдау құралдарын салыстыру*

СРАВНЕНИЕ АНАЛИТИЧЕСКИХ ИНСТРУМЕНТОВ ДЛЯ БОЛЬШИХ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ НАБОРА ТЕКСТА ПЕСЕН

Аннотация: *Огромное хранилище размерами в петабайты данных генерируется каждый день из современных информационных систем и цифровых технологий, таких как анализ научных данных, анализ данных в социальных сетях, системы рекомендаций и анализ журналов веб-служб. Данные обладают огромной силой, чтобы напрямую направлять нас к обнаружению знаний. Большие данные, в свою очередь, требуют совершенно нового подхода и инструментов для их обработки. Анализ этих массивных данных требует много усилий на разных уровнях для извлечения знаний и дальнейшего принятия решений. Огромные объемы данных и их неструктурированный характер порождают новые проблемы и вопросы, связанные с их управлением и обработкой. В этой статье рассматриваются некоторые из самых популярных инструментов для анализа больших данных – Hadoop, Spark и Pig являются основными и современными инструментами для анализа больших данных, в связи с чем эти инструменты были выбраны для сравнения. Результаты этого исследования показывают, что для различных задач требуются разные инструменты и нет единого решения. Любые проблемы с большими данными нуждаются в том, чтобы разработчики использовали соответствующий инструмент, чтобы сделать работу более качественной и быстрой.*

Ключевые слова: *большие данные, Hadoop, Spark, Pig, сравнение платформ для больших данных*

1. Introduction

Crucial changes in traditional data analyzing platforms are being made by Big data in information era. 4V's of Big data (Volume, Velocity, Variety, Variability) is increased to 5V: value is added. Since we have general knowledge about big data's other characteristics, Value is the main point which we need to consider when any kind of data is analyzed. Thus, to play out any sort of analysis on such voluminous and complex information, scaling up the hardware stages winds up fast approaching and picking the correct tool turns into a significant choice if the client's prerequisites should be fulfilled in a sensible measure of time. There are a few major big data analysing platforms accessible with various attributes and picking the correct tool requires an inside and out learning about the abilities of each [1]. Choosing the right platform that can handle expanding amount of data and analyse them by satisfying client's demand is the main aim of this paper. In this paper we will focus on Apache Hadoop, Pig and Spark; will be provided some characteristics on each tool, highlighting advantages and disadvantages of each, followed by practical experiments with music dataset by comparing mentioned tools on:

- Iterative task support;
- Computing time, how fast results will be computed;
- Data access by I/O performance;
- Real time processing and fault tolerance

Song lyrics dataset for experiment will be used from [2], which consists from 57650 items of song and were gained from LyricsFreak. Dataset consists from only 4 columns: artist name, song name, URL reference as a link and unmodified lyrics of this song.

In this paper, we will compare mentioned tools by analysing results for each of the following tasks that will be done on Hadoop Mapreduce, Pig and Spark:

1. Finding the most popular word for each artist; (multiple transformations)

2. Finding similar songs, consists of 2 subtasks:

1) Shingling documents by length of 5 words (iteration)

2) Reducing with Jaccard Similarity (computation)

2. Literature Review

A. Apache Hadoop

Apache Hadoop is an open-source software framework for parallel processing huge sets of data on large clusters (consisting from thousands of nodes) , built from reliable commodity hardware, by using simple programming model called MapReduce. Hadoop divides files into indepen-

dent blocks, then distributes them to nodes, where they are summarized separately. Data is stored in HDFS, which is a distributed file system, where all files are neighboring set of bytes. Hadoop first implements map operations to each block of data from HDFS by sorting and redistributing results depending on key values, then second part (reducer) consists from collecting data items with same keys. Reducers can aggregate the intermediate results from mapper to generate final result and write them again to HDFS. Thousands of map and reduce tasks run across different nodes in each cluster parallelly. YARN is one more component of Hadoop which is responsible for coordinating applications runtime. Survey about parallel data processing with MapReduce is accessible in [3].

There are diverse coding approaches besides Hadoop MapReduce, like Pig script and Hive interface, but both works on top of Hadoop Mapreduce.

B. Apache Pig

Apache Pig is a platform that was built on top of MapReduce by Yahoo Company, providing developers better control on MapReduce. As java has jvm, pig uses pig runtime as an execution environment. Fact says that 10 lines of code in pig will do the same as MapReduce with 200 lines java code. However, compiler converts pig latin into MapReduce, creating consecutive set of jobs. Developers can write functions in Python, Java, JavaScript and Ruby, also in Groovy to extend Pig latin code, if needed. Storing, manipulating and executing data is allowed in Pig.

C. Apache Spark

Apaches product, which also parallelly process data across clusters, was invented in 2012 at the AMPLab. Spark uses system memory, which makes it a way faster than Hadoop MapReduce, which reads and writes data to HDFS. Process of writing data to RAM is completed by using RDD (Resilient Distributed dataset). Spark Core plays significant role in coordinating scheduling, applying abstraction of RDD, optimizing and connecting Spark to proper file system. However, data transfer (I/O) between nodes still creates some network congestion.

D. Apache Hive

Hive was created by Facebook for programmers who are fluent with SQL to work in Hadoop environment. It can process structured data in tables. Hive uses HiveQL, which is SQL-like query language, that supports basic operations like select, join, project, aggregate and union all. Users can load data from external sources and insert data operators (DML), respectively.

According to paper by Dr. Urmila R. Pol, author explains working principles of Hadoop Mapreduce, Pig and Hive by comparing them between each other. Author mentions that writing Mapreduce code with Java would require writing several lines of code, and would take more time if user is not good familiar with Java. In this case, using another approach to write Mapreduce tasks like by writing them in Pig Latin or Hive SQL language would reduce development and testing time overall. As stated in the paper, even if Pig or Hive do not run as fast as MapReduce's native Java, using them boosted data analysts productivity. It is because writing Pig script takes only 5 percent of that time which is needed to write in Java, however decreasing runtime performance. Thus, Hive and Pig are considered performance boosting tools for data analysts, according to this paper [4]. Writing join functionality in java will be very complicated while using Hive SQL with several levels of nested FROM clauses.

Some cases where pure Hadoop MapReduce is preferable than Pig or Hive is discussed in article [5], like:

- job where complicated form of distributed cache is required;
- Job where optimization is needed in mapper or reducer level ;
- Job where is needed some form of partitioning;
- Performance of Hadoop is much better than other two, even if they enhancing their functionality kit and etc.

Other technologies (*Hadoop, Spark,* Graphics Processing Unit (*GPU*) , High Performance Computing Clusters (*HPC*), Multicore processors and Field Programmable

Gate Arrays (*FPGA*)) that can handle big

data is reviewed in paper [6], where authors first explained vertical and horizontal scaling, types of platforms/technologies along with their strength and weaknesses. Practical applications with general idea when scaling up and scaling out should be preferred is given along with explanation of improving systems by selecting appropriate platform. Authors claim that cases where expandable platform, which can support huge amount of data, Hadoop MapReduce and Spark is perfect, especially if there is no need for real-time responses; whereas cases where queries are processed in real-time - vertical scaling is needed. We can add that if application needs to be processed not in real-time with high velocity, Spark is the best option; but if volume matters, Hadoop can handle far more than that of Spark, since it stores data in disk memory. Which of them should be used in exact situation is given in [7], highlighting each's dignity.

Performance comparison between Hadoop and Spark was observed in article [8], which states that Spark runs in-memory 100 times faster and 10 times on disk; for 100TB of data Spark has been recorded 3 times quicker than MapReduce; also for k-means and Naive Bayes algorithms has been found to be faster. Advantage of Spark by its cyclic connection is also mentioned along with prices of each tool: Hadoop is less costly than Spark. In terms of security and fault tolerance, Hadoop is recommended to be more secure and reliable.

3. Proposed Method

Comparing tools should be built on platform dependent and algorithm dependent, giving general view about strengths and weaknesses of big data systems based on tendency of each. However, it also should be counted that application based comparing strongly depends on problem that needs to be solved.

Table below represents theoretical comparison of each big data analyzing tool by its characteristics.

	Language	*Abstraction level*	*Development effort*	*Code efficiency*	*Data access/ data I/O performance*	*fault tolerance*	*real-time processing*	*iterative task support*
MapReduce	Compiled	Low	Challenging	High	External memory	5	2	3
Spark	Compiled	Medium	Hard	High	In-memory cache	5	5	5
Pig	scripting	High	Easy	Less	External memory	5	2	3
Hive	SQL-like	High	Easier than Pig	Less	External memory	5	2	3

Table 1 Theoretical comparison of Big data tools

Table 1 shows more qualitative relation between tools than quantitative, comparing characteristics like *I/O performance for iterative processing, real-time processing, language, abstraction level, code efficiency* and *development effort.* MapReduce, Pig and Hive are not basically used for real-time processing tasks, because they are slow in transferring data Input/output between nodes. Therefore they receive only 2 out of 5.

All of the frameworks can be scaled up to tens of thousands of nodes and fault tolerant.

MapReduce, Pig and Hive are not designed for iterative processing, because data has to be written onto disk after each iteration, which creates a huge bottleneck while disk I/O. However MapReduce has developed tools and frameworks, one of it is improved by Hadoop development, which is for improving iterative tasks, therefore they received 3 out of 5. Decision between Hadoop and Spark lays between whether user needs a off-the-shelf instruments or he needs optimization of cluster performance.

Data has been processed in Amazon EMR cloud cluster, which provides managed frameworks like Hadoop, Spark, Presto and HBase

with Jupiter based notebook. Amazon EMR is used due to its reliability, easiness, flexibility and elasticity, also to set same conditions for compared platforms. Instances that where applied for master- m4.large, has 4 cores with 16 GB RAM , and for workers-c4.xlarge with 4 cores and 8 GB RAM each. Furthermore no tuning was done on MapReduce, Spark and Pig, so that they have default settings. However, since Spark is not using all available cores that were given, it needs to be set manually to use them; therefore some additional improvements on clusters were done.

Before starting comparison mentioned tools, lyric dataset was preprocessed by:

- deleting new lines in each row of song, so that each song takes one row;
- All stop words and common words were deleted, like *chorus, bridge, etc.*

First task is to find the most repeated (beloved) word for each singer, was done by simple word count example in Picture 1 with Pig Latin language:

```
REGISTER bigdata.jar;
all_data = LOAD 'songdata_tabbed.csv' USING PigStorage('\t') as (singer:chararray, song:chararray, lyrics:chararray);
tokenized = FOREACH all_data GENERATE singer, kz.sdu.bdt.pig.Tokenizer(lyrics) as tokens;
flat = foreach tokenized generate singer, flatten(tokens);
grp = group flat by ($0, $1);
cnt = foreach grp generate flatten(group) as (singer, word), COUNT($1);

g = group cnt by singer;
result = foreach g {
    prods = order cnt by $2 desc;
    top_prods = limit prods 1;
    generate flatten(top_prods);
};
--sorted = order result by $0;

dump result;
```

Pic.1 Popular word among singers with Pig

As second task checking lyrics for plagiarism was chosen. That is necessity is to find pairs of song that have similar lyrics. Main concern of this problem is how to compare two documents efficiently, as direct comparison is not an option, due to possible small changes in the song. There are some approaches to solve this issue, such as MinHash[11], SimHash[12], Word2Vec[13]. However purpose of this research is to compare tools and implementation of such advanced algorithms might interfere with results. This is the reason why simple approach with Jaccard Similarity was chosen.

Jaccard similarity [10] is equal to division of the size of the intersection of A and B sets to the size of their union:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

where elements of the set are words in songs.

Despite that Jaccard similarity works perfect for most of the time, this approach cannot be used alone while comparing text, because it does not take into consideration position of words in songs. For example: sentences “Alex loves Beth” and “Beth loves Alex” are considered exactly similar by plain Jaccard similarity algorithm, but are not. To address this issue Shingling of lyrics is used[9]. Song is divided in shingles of 5 consecutive words. However it is easy to see that final size of shingled song is much bigger, so to slightly reduce memory usage all sentences are hashed to some Integer. For purposes of this research simple java String.hashCode() is used.

Thereby second task is spitted into two different subtasks: Shingling of lyrics and finding Jaccard similarity of song regarding all other songs in dataset.

Shingling songs

Algorithm for shingling is simple:

First, lyrics converted to lowercase. Then all non-letter characters are removed. Third, sub-sentences of 5 words are created. Finally, these sentences are hashed and outputted.

Code below is in Pig script, which allows hashing shingles despite its amount.

```
REGISTER bigdata.jar;
all_data = LOAD 'songdata_tabbed.csv' USING PigStorage('\t') as (singer:chararray, song:chararray, lyrics:chararray);
min_hashed = FOREACH all_data GENERATE CONCAT(singer, '|', song), kz.sdu.bdt.pig.MinHasher(lyrics) as hashes;

dump min_hashed;
```

Pic. 2 Hashing shingles

Jaccard similarity

To calculate jaccard similarity, *Cartesian product* was used for all hashed shingles in joiner, and only those lyrics that got jaccard similarity higher than 0.7 have been taken into account as the most similar pairs. Pig and Spark has predefined function for finding Cartesian product, whereas in MapReduce it needs to be implemented from scratch.

```
REGISTER bigdata.jar;
all_data = LOAD 'song_hashes' USING PigStorage('\t') as (song:chararray, hashes:chararray);
all_data2 = FOREACH all_data GENERATE song as song2, hashes as hashes2;
crossed = CROSS all_data, all_data2;
jac_sim = FOREACH crossed GENERATE song, song2, kz.sdu.bdt.pig.JaccardSimilarity(hashes, hashes2) as jaccard_similarity:double;
result = FILTER jac_sim BY song != song2 AND jaccard_similarity > 0.7;
dump result;
```

Pic. 3 Jaccard similarities with Pig

4. Experimental results

A. Popular word among singers with Pig

Fig. 1 shows time distribution in seconds. As it can be seen launching distributed jobs on small dataset provides no advantages over local mode. This might happen due to overhead of running task on distributed systems. Very likely that situation changes drastically on huge datasets.

Table 2 shows the sample of the result of running first task, which lists the most repeated word in all songs for every singer. There are totally 643 singers in the dataset.

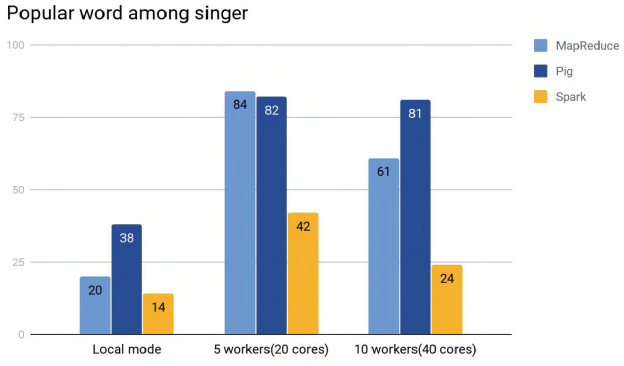

Fig. 1 Time for "Popular word among singer" task

n Sync	love(298)
ABBA	love(189)
Ace Of Base	love(112)
Adam Sandler	like(74)
Adele	love(161)
Aerosmith	yeah(401)
Air Supply	love(514)
Aiza Seguerra	love(60)
Alabama	love(367)
Alan Parsons Project	know(78)

Table 2. Result of "Popular word among singer" task

B. Shingling songs

Figure 2 shows that due to overhead of running jobs on distributed systems all tools perform better on local mode. However it can be seen that with increase of worker cores processing time decreases for all tools except Pig.

While previous two tasks took seconds for calculation, finding Jaccard similarity takes hours because each job needs to compare each hashed shingled sets with others, by applying [10] formula.

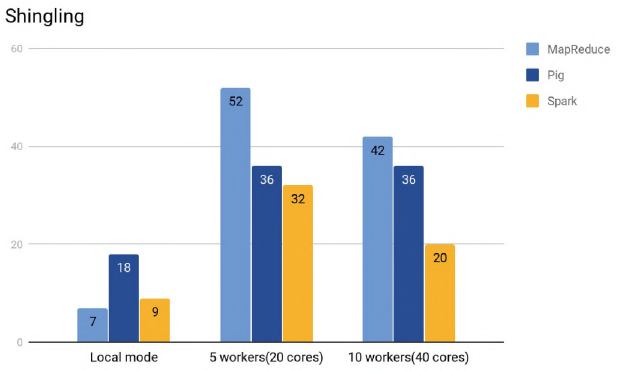

Fig. 2 Time for Shingling task

C. Jaccard similarity

It means that this task requires a lot of calculations. This is where power of distributed systems shines. Fig. 3 shows that distributed systems perform more that 2 times better for MapReduce and Spark. More processing power added, less time required to process task. First surprise here is that MapReduce performs better on 10 workers than Spark, but Spark outperforms MapReduce on lesser workers. Reason might be that Spark limits processing power to some percent of available cores, whereas MapReduce takes all free processors.

Second interesting outcome is that Pig takes same time to process the task regardless cluster size. Problem is in default configuration where Pig launches only two reducers to complete job. This nullifies all advantages of distributed processing. Therefore Pig requires additional configuration to run smoothly.

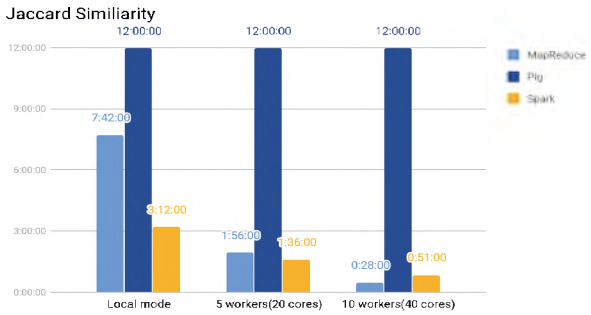

Fig. 3 Time for Jaccard Similarity

Sample of results of finding plagiarism between singers is given in Table 4.

ABBA - We Wish You A Merry Christmas	Harry Belafonte - We Wish You A Merry Christmas
Kenny Chesney - Pretty Paper	Randy Travis - Pretty Paper
James Taylor - I Didn't Know What Time It Was	Ella Fitzgerald - I Didn't Know What Time It Was
Robbie Williams - Do Nothing Till You Hear From Me	Ella Fitzgerald - Do Nothing' Till You Hear From Me
Rihanna - Golden Girl	Chris Brown - Golden Girl
Lady Gaga - Nature Boy	Natalie Cole - Nature Boy
Lady Gaga - White Christmas	Vince Gill - White Christmas
Lady Gaga - Winter Wonderland	Faith Hill - Winter Wonderland
Johnny Cash - Cindy	Nick Cave - Cindy
Avril Lavigne - Imagine	The Beatles - Imagine
Backstreet Boys - Cinderella	Lionel Richie - Cinderella
James Taylor - Santa Claus Is Coming To Town	Harry Connick, Jr. - Santa Claus Is Coming To Town
Rihanna - We Found Love	Coldplay - We Found Love

Table 4. Result of running Jaccard similarity task

All three tasks with amount of workers with time consumed are written in table below. Green labels indicate the fastest tool for each task. So in first's simple word count task Spark calculates faster than other two. In constructing hashes from shingles Spark works faster while using 5-10 cores due to its iterative processing ability. In the last task where computing is needed Spark performed better in local mode and with 20 processing cores, whereas with 40 cores Hadoop MapReduce processed data 2 times faster than Spark. Red labels indicate the worst tool in each situation.

	Popular word among singer	Min hash	Jaccard similarity
Local mode			
MapReduce	20s	7s	7.7h
Pig	38s	18s	12h
Spark	14s	9s	3.2h
5 workers (20 cores)			
MapReduce	84s	52s	1.8h
Pig	82s	36s	12h
Spark	42s	32s	1.6h
10 workers (40 cores)			
MapReduce	61s	42s	28m
Pig	81s	36s	12h
Spark	24s	20s	51m

Table 3. Comparison by time taken for processing on each tool

Table 4 shows code amount for each mentioned tool for every task. MapReduce takes a lot effort from data analyzer, it requires writing more than twice of code of Spark and around 18 times more lines of code than it could be written in Pig for current task. This could be because Pig is scripting language, and needs minimum number of code lines.

Code amount	Popular word among singer	Min hash	Jaccard similarity
MapReduce	148	88	128
Pig	16	5	7
Spark	56	34	43

Table 4. Amount of code needed for each tool

5. Conclusion and future works

Pig looks as total outsider on these tasks. The only advantage of Pig over other tool is abstraction. Pig latin very concise language and needs a few lines of code for all tasks. Pig can be considered only in cases when there is no programmer available.

Spark is considered best all-around tool to process big data due to its in-memory caching capabilities. However results of research show that in some cases MapReduce can outperform Spark. Spark provides greater abstraction level than MapReduce, thus requires much less code to complete tasks. Even though

Spark showed worse results in some cases, for most tasks it performed the best. Adding here abstraction level and amount of languages available for working with Spark, makes Spark most preferable tool for processing such tasks out of compared ones.

However, all experiments were done on tools without any tuning. Adjusting launching parameters can radically change whole picture. Optimizing settings for tools and comparing them for performance with new conditions could be the next future work.

REFERENCES

1. Agneeswaran VS, Tonpay P, Tiwary J (2013) Paradigms for realizing machine learning algorithms. Big Data 1(4):207–214
2. https://www.kaggle.com/mousehead/songlyrics
3. Lee K-H, Lee Y-J, Choi H, Chung YD, Moon B (2012) Parallel data processing with MapReduce: a survey. ACM SIGMOD Record 40(4):11–20
4. *Big Data Analysis: Comparison of Hadoop MapReduce, Pig and Hive*. Available from: https://www.researchgate.net/publication/308074477_Big_Data_Analysis_Comparision_of_Hadoop_MapReduce_Pig_and_Hive
5. MapReduce vs. Pig vs. Hive - Comparison between the key tools of Hadoop, Available article from: https://www.dezyre.com/article/mapreduce-vs-pig-vs-hive/163
6. Dilpreet Singh and Chandan K. Reddy, "A Survey on Platforms for Big Data Analytics", Journal of Big Data, 1:1, 8, 2014.
7. https://www.scnsoft.com/blog/spark-vs-hadoop-mapreduce
8. https://dzone.com/articles/hadoop-vs-spark-a-head-to-head-comparison
9. https://www.todaysoftmag.com/article/1553/finding-similar-entities-in-bigdata-models
10. https://neo4j.com/docs/graph-algorithms/current/algorithms/similarity-jaccard/
11. Szmit R. (2013) Locality Sensitive Hashing for Similarity Search Using MapReduce on Large Scale Data. In: Kłopotek M.A., Koronacki J., Marciniak M., Mykowiecka A., Wierzchoń S.T. (eds) Language Processing and Intelligent Information Systems. IIS 2013. Lecture Notes in Computer Science, vol 7912. Springer, Berlin, Heidelberg
12. C. Sadowski and G. Levin. Simhash: Hash-based Similarity Detection. Technical report, Technical report, Google, 2007.
13. Tom Kenter , Maarten de Rijke, Short Text Similarity with Word Embeddings, Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, October 18-23, 2015, Melbourne, Australia