University of Piraeus

Department Of Digital Systems

Post Graduate Program On
Digital Systems and Services

Master Thesis

Recommender Systems Comparison

Σύγκριση Συστημάτων Προτάσεων

Vasileios Simeonidis

This thesis was supervised by
Dr. Dimosthenis Kyriazis

August 15, 2017

This page was intentionally left blank.

1 Acknowledgements

This thesis couldn't be completed without the great support I received from so many people over this year. I would like to thank the following people.

My supervisor Dr. Dimosthenis Kyriazis for totally supporting me in the choices I made and giving me the freedom I was needing.

My friend and fellow student Dimitris Poulopoulos for helping me understand the field of recommender systems and supported me technically and theoretically.

My friends and colleagues George Adamopoulos and Nikos Silvestros for constantly teaching me high-level engineering and scientific thinking. Also, I would like to thank them for putting the right amount of pressure on me in order to complete this thesis.

Kronos, the development team I am part of and helping me to keep my spirit high. Thank you, Apostolos Chissas, Kostas Rigas, Christos Grivas, Peter Lengos, Vasiliki Giamarelou, Maria Karkeli, Eleni Karakizi, Spyros Argyroiliopoulos, Nikos Anagnostou and Ioannis Koutsileos.

Last but not least, I would like to thank my close friends and family for bearing me while I was anxious about the completion of this thesis.

This page was intentionally left blank.

2 Preface

The last decade Internet has been flooded with information. Information that no one can filter to find what he needs, raw data, videos, music or products. Large retail sites like Amazon developed recommenders systems in order to offer products to their users. The need although is not limited only in the retail area.

Web sites like Youtube or Vimeo need to recommend to each user of their, videos that may like to watch next. Facebook is another example of an application utilizing lots of data and offering recommendations on what you may want to read or who may be a friend of yours. Most of the times, a recommender system is not the core functionality of an application. It is through a very useful feature that gives a clear advantage in any business area needed.

This thesis aims to destiguise metrics on recommender systems that can be proved useful to compare them. Also, this thesis, performs a comparison between two algorithms of the collaborative filtering family. The content based with focus on items and the machine learning oriented alternating least square (als).

This page was intentionally left blank.

This page was intentionally left blank.

Part I. Master Thesis

3 Introduction

In the late 40s, a man called Alan Turing was investigating the case of a programmable computing machine. Of course, there were similar tries before. Babbage, for example, was famous for creating a computation machine. What made Turing's machine to differ? This machine was the first programmable one. I don't know how easily programmable you would call a machine in binary code, but at its time was the one. Turing prophetically enough said that we might need plenty of mathematicians of ability in order to program those machines.

The years passed by, and lots of those machines appeared. The time was approaching for a general purpose, commonly supportive program that would take care of the trivial tasks. Tasks like standard input and output, and common program execution management. That kind of programs is called operating systems.

Based on the previous advance the services provided by companies become more sophisticated. Services provided through software or provided using the software. To give an example, let us assume that we have a company A. This company specializes in delivering high-quality cars. After the operating systems introduction, this company has no need to make an operating system in order host an application that handles orders. This gave them the opportunity to low the cost of developing and maintaining a large part of their information system. Now, let's assume that we have a company B, this company specializes in consulting other companies on how to handle their orders. Either it does it through a software delivered to its customers or via a software for helping the company itself, an operating system truly changed their competitive advantage.

The user provides data and actions. The application enhances the raw data provided by the user and persist them in a structured way, this could be for example a database, relational or not. If we take into account the above process, this means that the system has information about each user, his actions and the data he provided. This information is structured. Of course, the structure was made to serve the business cases of the application.

The year is 1989, a British engineer working at CERN named Tim Berners-Lee invented the World Wide Web (W3). Of course, the first web site was on the world wide web which is somehow self-referential. W3 was mainly used by scientists in universities and institutes in order to share their work.

About a decade later, the RFC 1945: Hypertext Transfer Protocol - HTTP/1.0 [8], co-authored by Berners-Lee, was published, and W3 started to follow a more structured format.

The last decade Internet has been flooded with information. Information that no one can manage to find what he needs. This information contains from raw data to videos, music or products. Large retail sites like Amazon developed recommenders systems in order to offer products to their users. The need although is not limited only in the retail area.

Web sites like Youtube or Vimeo need to recommend to each user of their, videos that may like to watch next. Facebook is another example of an application utilizing lots of data and offering recommendations on what you may want to read or who may be a friend of yours. Most of the times, a recommender system is not the core functionality of an application. It is through a very useful feature that gives a clear advantage in any business area needed.

The way a recommender system has been built is very dependent on the business case which will be served. Even a specific case of recommendation, similar to one already existing, might need a different recommender system.

So far recommender systems have been a very interesting area of study. Netflix in 2009 declared a challenge, which can be found here [1]. Its reward was one million dollars for the task of improving the accuracy of predictions. The prize was granted to BellKor's Pragmatic Chaos team for their algorithm. You can come across this challenge on lots of papers published every year.

With such a wide study of recommender systems, it is reasonable to start wondering "How are we going to compare recommender systems?". As we will discuss below, there are several papers suggesting ways of comparison. The majority of those papers are using the dataset given in the challenge above.

In this thesis, the first system to compare is a content-based recommendation system that provides predictions based on movies genre attributes. The second system is the based on the Alternating Least Squares (ALS) implementation of Apache Spark's MLlib.

The comparison metrics used are the Mean Absolute Error (MAE), the Root Mean Squared Error and the ratio between them (MAE/RMSE). Last but not least is the execution of time metric, measuring the training and estimation time.

4 Related Work

In this thesis's chapter, we will list numerous different approaches made in order to compare recommender systems.

[1] http://www.netflixprize.com/

4.1 RecBench: Benchmarks for Evaluating Performance of Recommender System Architectures [1]

The University of Minnesota, published in 2011 a paper stating a comparison between a recommender framework and a DBMS-based one. In that paper, they used the Movie Lens dataset 100k, from the Netflix Challenge. The benchmark had five areas of comparison. Those areas were initialization, pure recommendation, filtered recommendation, blended recommendation, item recommendation and item update.

The initialization task was about the preparation needed for the system to go live. The next area was the pure recommendation. By pure recommendation, the author means the home page recommendation, meaning the items that are going to be on the homepage. Moving forward, we find the filtered recommendation. This recommendation is constrained by variables specific to the item, like movie genre etc. Another area of this evaluation contains the blended recommendation. Those recommendations are based on free text provided by the user in order to search. Item prediction is another area of the evaluation, in this prediction the user is navigated to the items page and the system is trying to predict the user's rating on the item. Last but not least, the paper examines the case of a new item being added to the system and how this is going to be incorporated into it.

As a result, of those experiments, the paper concludes that "hand-build recommenders exhibit superior performance in model building and pure recommendation tasks, while DBMS-based recommenders are superior to more complex recommendations such as providing filtered recommendations and blending text-search with recommendation prediction issues.

4.2 Recommender Systems Evaluation: A 3D Benchmark [2]

In this paper, the authors recognize the need for a common benchmark formula for recommender systems. This need leads them to propose one. They named it the 3D recommendation evaluation because they evaluate a system in three axes. These axes are business models, user requirements, and technical constraints. In business model axis they state that a recommender system must be evaluated on how well it serves the business case it is used for. In their paper, they give the example of a video on demand service and evaluate it versus the pay per view business model and pay per subscription.

In the user requirements axis, the evaluate the system based on what needs it covers for the users. Is it, for example, going to reduce search time or decision-making time.

Last but not least is the technical constraints axis. In this axis, the system is being evaluated based on data or hardware constraints, scalability, and robustness.

4.3 RiVal: A New Benchmarking Toolkit for Recommender Systems [3]

RiVal, is an open source toolkit implemented in Java programming language. Rival is available via Maven repositories. It is used in order to measure the evaluate recommender systems. Its evaluation is based on three points. Those points are data spitting, item recommendation, candidate item generation and performance measurement.

5 Machine Learning and Collaborative filtering

5.1 Machine Learning

Learning is the complex process that contains among other knowledge acquisition and organization.

Moving forward lets examine the objectives and processes of machine learning [9]. The objectives of machine learning can be summarized in three major categories as shown below.

- Task Oriented Studies Those studies emphasize on creating and analyzing a predefined set of tasks in order to improve their performance. They are also called engineering approach.
- Cognitive Simulation In this area, the computer program tries to imitate the learning process of a human being.
- Theoretical Analisis is the last but not least of the machine learning objectives. Here is where scientists try to investigate, domain agnostic learning processes and algorithms.

It is stated that there are two types of learning. Those types are knowledge acquisition and skill refinement. To make things more clear let us give some examples. Let us assume that you are studying about a new dog breed. This type of learning will be based on the acquisition of the knowledge. What color usually this breed is, what is its sizes and so on. Now let us assume that is the second time you are riding a bike, you have the knowledge of what goes where but your skill is not refined. As you will continue to ride the bike you will follow the second type of learning which is the skill refinement.

Numerous processes of learning is known, from processes that someone may call naive, to very efficient and complex ones. Below is listed roughly some of those processes.

- Rote learning and direct implanting of knowledge process requires no knowledge transformation by the learner. The

learner deterministically responding the knowledge we put in. When a computer programmer, instructs its machine through the code he provided, and that code contains an if statement. This is a knowledge that has been directly implanted to it.

- Learning by instruction , for example, we provide a set of rules to a program in a given language. The receiving system or the learner if you prefer, will have to transform this knowledge in order to utilize it.

- Learning by analogy is the process that allows the learner to project an existing knowledge to the new one in order to understand it.

  For example, let us imagine ourselves in an auditorium, and we having a class about lions. In order to help the example, let us assume that the only knowledge we have about the animal kingdom is dogs and cats. The professor knows or at least suspecting the specific knowledge we have. Then he says, a lion has the body structure of a cat and its size is twice as much the dog. We projected the new knowledge on a given one.

- Learning by example. In this area we give to the learner a set of examples which are notated as true or false. The learner will have to make assumptions and create patterns between the examples in order to answer a request that is not notated.

- Learning from observation and discovery is the last process we are going to investigate. This process is also called *unsupervised learning*. In this type of learning the learner tries to classify the observations without the help of a supervisor. Unsupervised learning has two main aspects. The first aspect is the passive observation. In this aspect, the learner classifies the observations made to the environment he exists. The second and more interesting part of is the active experimentation. This is where the learner tries to perturb its environment and observe the results of it.

  Let's give an example of active experimentation. Assuming we have a smart virtual machine hypervisor. The hypervisor follows the unsupervised active experimentation learning paradigm. This hypervisor hosts a virtual machine of 5 cores and 2GB of memory. The hypervisor has information about the number of processors used, the amount of memory available to the system and the traffic network.

  The hypervisor decides to experiment on the virtual machine and reduces the amount of available memory to 128MB. Instantly the hypervisor observes a dramatic reduction in the network traffic and the memory usage to be 100%.

On the other hand, the 5 cores are underutilized. Those metrics makes the learner classify this situation as a nonregular and returns the resources to the virtual machine.

5.1.1 Latest advancements in machine learning

Nowadays, deep neural networks can win games using strategies that a player could not expect. Like a biological brain, it consists of layers of neurons, but this time is figments of memory. Lower level neurons analyze pixels of a picture then they send data to higher level neurons which are trained on higher level concepts like a dog or a cat. Deep neural networks have shown that they can recognize items on a picture as accurately as humans do.

In a conference in Berlin has presented evidence in support of a new theory on how deep learning works by a computer scientist and neuroscientist from the Hebrew University of Jerusalem. This man called Naftaly Tishby argues that deep neural networks learn by a process called *information bottle neck* [10]. The theory states that the network gets rid of the noise in order to pass information to higher level neurons. A Google researcher, Alexander A. Alemi, states that this theory could serve "not only as a theoretical tool for understanding why our neural networks work as well as the way they do currently, but also as a tool for constructing new objects and architectures".

5.2 Collaborative filtering

In order to better understand the term, collaborative filtering lets take a look in a dictionary.

collaborative: adjective, characterized or accomplished by collaboration [11].

filter: noun, something that works like a filter, as by removing, blocking, or separating out certain elements [11].

As we can see by the given definitions, collaborative filtering is the technique that allows us to select an object from a given set based on the set itself.

The need of having recommender systems lies between the need of obtaining recommendations from trustworthy sources and the availability of a large amount of user data.

Like on any demand and supply system, on the one hand, lies the demand of accurate and trustworthy recommendations. On the other hand are the tons of user data that can serve this demand.

Over the years have been developed techniques that can utilize those data, in order to provide good recommendations. Those techniques are highly dependent on the volume of data they use.

The more the data the more accurate the recommendation. But its system's training phase is largely impacted

by the volume mentioned before. Thus, any algorithm or system is will provide good recommendations as long as it is trained with the right dataset.

Those algorithms were initially based only on statistical models that were available at the time. Whereas the data were growing rapidly, and the sample started to approach the population.

5.3 Content based

Content-based, it seems a very attractive term but let us take a look at the very definition of those words.

content: noun, something that is contained [11].

base: noun, a fundamental principle or groundwork; foundation; basis [11].

Content based collaborative filtering is the technique that allows us to select objects from a given set based on the actual values of its items.

This could be a rote or information implant learning processes.

The most common and easy to interpret the way of recommendation is content-based collaborative filtering. In this area of algorithms, you are trying to utilize data from other users in order to come to a recommendation. Those data might be attributes that characterize the item of interest.

In the case of users, those attributes may be their age or occupation, whereas for a product might be its color, prize or weight in kilograms. In order to put this to a mathematical expression, we could write the following.

$$w = R^{-1}M^T \tag{1}$$

Raw data though are not always clear or normalized. Due to this fact, we would consider to normalizing the approach we used above. If we were about to add a normalization factor to that expression we will end up with the one below.

$$w = (\lambda I + R^T R)^{-1} R^T M \tag{2}$$

That kind of algorithms is easier to interpret. They also can handle well a cold-start problem. But they are computational heavy, meaning that the scaling of them is limited.

5.3.1 Latent Factors

Finally let's take a look at the terms latent and factor in the dictionary.

latent: adjective present but not visible, apparent, or actualized; existing as potential [11].

factor: noun one of the elements contributing to a particular result or situation [11].

Based on the definitions above we can assume that the latent factor col-

laborative filtering, allows us to select objects from a given set based on factors that are not clearly depicted in the actual values of each item.

This type of learning is learning by example.

The group of latent factors algorithms does not take into consideration the meta-data we have for any user or item.

In that area, we are trying to determine relationships between a user and an item based only on the rating. Those relationships may not be the age or the color.

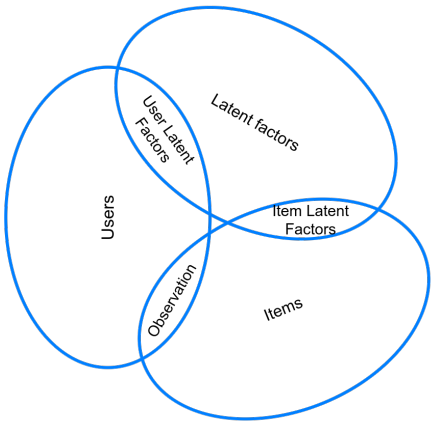

Fig. 1: **LatentFactors**

Alternating least squares (ALS) algorithm belongs to the group above. In this case, we assign initially random values of rating between user and items. Then it takes the error between the actual value and the one assigned to it.

Then the algorithm runs again using as input the errors and tries to minimize them. Below we can see a how this algorithm is defined.

Algorithm 1 ALS for Matrix Completion

1: Initialize X,Y
2: **repeat**
3: **for** `u=1...n` **do**
4: $x_u = (\sum_{r_{ui}} y_i y_i^T + \lambda I_k)^{-1} \sum_{r_{ui}} r_{ui} y_i, \in r_{u*}$
5: **for** `i=1...m` **do**
6: $y_i = (\sum_{r_{ui}} x_u x_u^T + \lambda I_k)^{-1} \sum_{r_{ui}} r_{ui} x_u, \in r_{*i}$
7: **until** convergence

As we can see above, this algorithm has a λ parameter used for normalization during the process. We can see the difference below where we have both the expression with and without normalization factor.

ALS is a very efficient recommender algorithm. Due to its nature, it can be easily parallelized reducing the execution time needed [12]. It also requires no meta-data about any user or item. Although, ALS suffers from the cold start problem.

$$\min_{X,Y} \sum_{r_{ui} observed} (r_{ui} - x_u^T y_i)^2 \tag{3}$$

$$\min_{X,Y} \sum_{r_{ui} observed} (r_{ui} - x_u^T y_i)^2 + \lambda (\sum_u ||x_u||^2 + \sum_i ||y_i||^2) \tag{4}$$

Moving forward this thesis, we are going to discuss how those two algorithms were implemented and validate the results they gave.

6 Experiment

6.1 Infrastructure

As the experiment's infrastructure, we will describe the frameworks used during the implementation. The framework that has been used to implement the item-based algorithm was Apache Spark. The ALS algorithm was used from the Apache Spark's MLLib library. The framework is common to infrastructure because Apache Spark can orchestrate the work on multiple machines as well as in one. So the framework is as close as we can get to the infrastructure.

6.1.1 Apache Spark

The last decade, analyzing big data is at its peak. Lots of data are produced on daily basis. This means that the need of extracting information from them is raised. Lots of frameworks have been used in order to manage and analyze this amount of data. One of the analysis reasons is the need for accurate item recommendations to users. Those items could be movies (e.g. Netflix), music (e.g. Spotify) or products in general(e.g. Amazon). One of the most popular frameworks that could enable this in a distributed way was Apache's Hadoop MapReduce.

Apache Hadoop has discrete jobs of processing data. The most common jobs are the map and reduce but it has two more jobs, combine and partition. Hadoop has a master node and N worker nodes. The first is responsible to distribute the work, and the second for the work to be done. Each worker usually is called after the job is executing. Hence we have the mapper, the reducer, the partitioner and the combiner. In order to put this to a schema, you can see the figure 2 below.

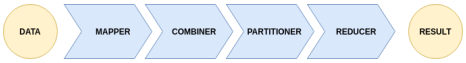

Fig. 2: **Hadoop Jobs Order**

Hadoop map reduce, is a distributed map-reduce system, this means that it has a mechanism to distribute work on nodes and a common interface for handling data. In Hadoop's case, this was able to happen due to Apache Hadoop Yarn and the Hadoop Distributed File System or as commonly used HDFS. When a job was scheduled, data were loaded by the HDFS to a worker, then the worker was done, he was putting the result back to the HDFS.

As mentioned in [13], "The term MapReduce actually refers to two separate and distinct tasks that Hadoop programs perform. The first is the map job, which takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). The reduce job takes

the output from a map as input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce job is always performed after the map job."

So Hadoop has two basic processes, Map which is responsible for turning the data into key-value pairs, and Reduce which takes those pairs and turns them into valuable data.

If we would like to see where in the DIKW (Data Information Knowledge Wisdom) stack. The Map process would start with data while the reduce one will end up with information. Of course, this is not always the case, lots of algorithms require lots of cycles in order to complete.

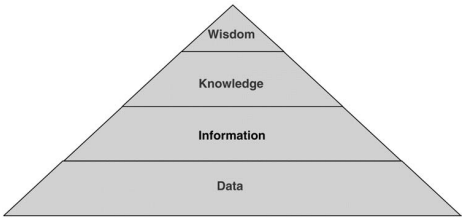

Fig. 3: **Data Information Knowledge Wisdom Pyramid [4]**

But let's take a step back and take a look at Hadoop's architecture. As it is described on its official website [14], and shown in the figure 4 Hadoop uses Hadoop yarn in order to coordinate which process will run on which machine. Also, it uses its file system, the HDFS in order to have a common reference for the files over the network. Last but not least, Hadoop ecosystem is supported by the Hadoop Commons library.

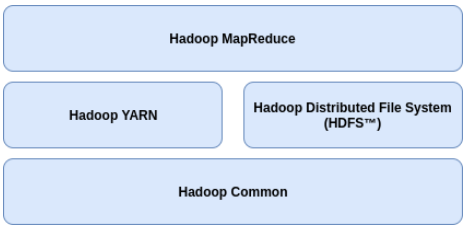

Fig. 4: **Hadoup Software Stack**

In 2009, University of California, Berkley, proposed a new framework for cluster computing in their paper, Spark: Cluster Computing with Working Sets [15]. They wanted to tackle two major Hadoop issues.

The first was the iterative jobs. Each Hadoop job reads from the disk to load data. This means that having iterative jobs, on any given algorithm, you were going to get a large time penalty on reading and of course writing to the disk.

The second issue was the interactive analytics. Each Hadoop SQL interface was running as a separate job, and as we mentioned previously we have a big impact on execution time.

In order to break the acyclic nature of Hadoop, they introduced the Spark's major abstraction, the RDDs. The name RDD stands for Resilient Distributed

Datasets. Those datasets are a read-only collection of objects distributed across machines. If a machine fails, the lost part of the RDD can be recalculated. This notion is called lineage.

Spark is implemented in Scala. Scala is a high-level statically typed programming language. At the time that paper was published, it was believed that Spark was the only system available in a general-purpose programming language to make clusters process a large amount of data. As it was mentioned in [15] "We believe that Spark is the first system to allow an efficient, general-purpose programming language to be used interactively to process large datasets on clusters"

Back to RDDs, an RDD can be created with four different operations as it is described in [15].

- The first operation is loading data from any shared file system.

  That file system could be HDFS or even an Amazon S3.

- The second way to create an RDD is by parallelizing any Scala collection.

  Spark will slice the collection into pieces and distribute it among the nodes.

- The third way is via transforming an RDD to another one.

  Because RDDs are immutable, any transformation operation on an RDD, filter, map, flatmap, will generate a new RDD. The last but not least method is by changing an RDDs persistence using save or cache operations.

Spark also give us the power to do a lot of different distributed operations. Some of them were mentioned before, but we also have operations that would return data to the driver program like collect or reduce.

Another important feature of Sparks spine are the shared variables. Spark at its first appearance introduced two of them.

- The first, shared variable is the broadcasted variables.

  Those variable, RDDs or not, are variables that are commonly used in an algorithm, like a look-up table. By broadcasting a variable, each node gets a copy of the variable in order to access it quickly.

- The second shared variable that was introduced in that paper was the Accumulators.

  Those variable live on the spark context, but they can only be increased by any worker and be read from the driver program only. That paper concludes that Spark can outperform Hadoop in some machine learning algorithms and more specific on logistic regression.

Coming back to today, Spark's current architecture is depicted below in

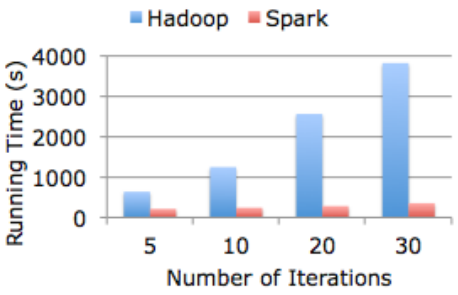

Fig. 5: **Logistic regression, Hadoop vs Spark [5]**

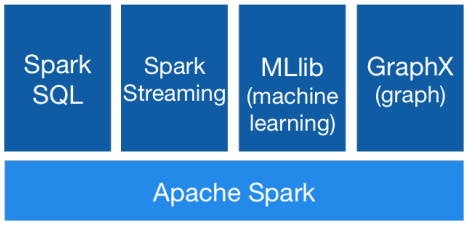

Fig. 6: **Apache spark stack [6]**

fig. 6. Spark nowadays has an SQL interface in order to search in RDDs within a query language. Also, Spark support a streaming API to make available real-time analytics. Most of the core machine learning algorithms like ALS, which I used in order to complete this thesis, are the Spark's MLlib component. Finally, the Spark has the component GraphX that is used for handling graphs and graph computation.

Apache Spark was by design meant to work within Hadoop ecosystem, and most importantly with the HDFS. Apache Spark does not have a file system by itself. You can load data from almost any database, cloud-based or not, even from a local file system. But most will agree that Hadoop and Spark work together just perfect.

To conclude, Spark has dominated the big data field the last years, Amazon and other cloud providers give you the option to deploy an Apache Spark cluster on their infrastructure. Also, large companies like Google and Intel are actively contributing to projects like Apache Spark On Kubernetes which can be found at its Github repository [2]

7 Dataset

Selecting a clear and reliable data set is very important for every experiment in the field. Due to that need several papers, like [1], are using the Movie lens data set. This dataset contains users, movies, and ratings. Each user may rate more than a movie and each movie can be rated by more than one user. The dataset split to multiple subsets of 80000 training sets and respective 20000 test set. It also provides scripts that can be used to create more sets.

In order to better visualize the above dataset, we created an entity relationship(ER) diagram below.

[2] https://github.com/apache-spark-on-k8s/spark

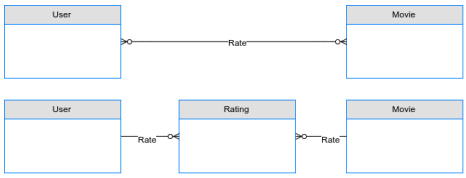

Fig. 7: **MovieLens ER diagram [7]**

8 Implementation and assumptions

During the implementation, I had to make some assumptions and choices. The first of choices was the framework and the programming language that the implementation would take place. The framework that has been chosen, as you may have already figured out, is apache spark due to its trend and the high scalability it offers. The language of choice was scala, due to its functional nature and its compatibility with apache spark.

9 Results

In this chapter, we will examine the results taken from the experiment described in the previous chapter. In order to measure the performance of each recommender system, we used three different metrics. Those metrics are the Mean Absolute Error (MAE), the Root Mean Square Error (RMSE), and the execution time of each system.

The Mean Absolute Error is defined as the sum of each error's absolute value divided by the number of them. Let's take a look at error's definition. It is common in statistics to symbolize the error with e_i. The i index shows which measurement's error is. An error can be calculated using the following formula $e_i = \widehat{y_i} - y_i$, referring to $\widehat{y_i}$ as the estimated value for the i-th item whereas to y_i as the actual value of the i-th item. For example if we estimated that the $\widehat{y_i} = 5$ and its actual value is $y_i = 5.2$ and its error could be the following.

$$e_i = \widehat{y_i} - y_i => e_i = 5 - 5.2 => e_i = -0.2 \tag{5}$$

Mean absolute error is easier to interpret than other statistical metrics like RMSE. We will examine RMSE later in this chapter. If we want to express mathematically the MAE we could write the following.

$$MAE = \frac{\sum_{i=1}^{n} |e_i|}{n} = \frac{\sum_{i=1}^{n} |\widehat{y_i} - y_i|}{n} = \frac{\sum_{i=1}^{n} \sqrt{(\widehat{y_i} - y_i)^2}}{n} \tag{6}$$

During the experiment we took the MAE metric for each system. The two following tables can show the results in detail.

Tab. 1: **Content Based Algorithm Results vs Mean Absolute Error**

Training Dataset | Testing Dataset	Mean Absolute Err
u1.base | u1.test	1.64674314
u2.base | u2.test	1.60552221
u3.base | u3.test	1.60892590
u4.base | u4.test	1.62591920
u5.base | u5.test	1.62846586
ua.base | ua.test	1.6425364
ub.base | ub.test	1.63571965

Tab. 2: **Latent Factors Algorithm Results vs Mean Absolute Error**

Training Dataset | Testing Dataset	Mean Absolute Error
u1.base | u1.test	1.1818684937
u2.base | u2.test	1.1800652808
u3.base | u3.test	1.1785360748
u4.base | u4.test	1.1780543877
u5.base | u5.test	1.1686585292
ua.base | ua.test	1.2008035301
ub.base | ub.test	1.2134460078

As we can see, comparing the two systems on this metric we realize that the latent factors system outperformed the content based one. That is clearly depicted on the graph below.

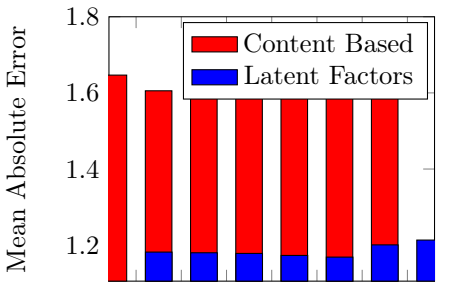

Fig. 8: **Latent Factors vs Content Based on Mean Absolute Error**

In bibliography the most common statistical metric for recommender systems is the root mean square error(RMSE). This metric is considered to give a greater estimation of error magnitude due to the fact that it uses the squared value of each error. In order to better understand this metric, we can have a look at its mathematical type below.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} e_i^2}{n}} = \sqrt{\frac{\sum_{i=1}^{n} (\widehat{y}_i - y_i)^2}{n}} \quad (7)$$

Each error is measured as before but now due to the structure of that metric the larger the error is, the greatest the impact it has on RMSE.

The results we got during the experiment on that metric was that latent factors system outperformed the content base again. You can see the details of the result on the two following tables.

Tab. 3: **Content Based Algorithm Results vs Root Mean Square Error**

Training Dataset | Testing Dataset	Root Mean Square
u1.base | u1.test	2.1040
u2.base | u2.test	2.0594
u3.base | u3.test	2.0914
u4.base | u4.test	2.0862
u5.base | u5.test	2.0990
ua.base | ua.test	2.1224
ub.base | ub.test	2.0994

Tab. 4: **Latent Factors Algorithm Results vs Root Mean Square Error**

Training Dataset | Testing Dataset	Root Mean Square
u1.base | u1.test	1.3793
u2.base | u2.test	1.3784
u3.base | u3.test	1.3754
u4.base | u4.test	1.3706
u5.base | u5.test	1.3668
ua.base | ua.test	1.3968
ub.base | ub.test	1.4119

The results above can be shown clearly on the graph following, depicting the

performance of latent factors and content based systems, on each dataset, on RMSE metric.

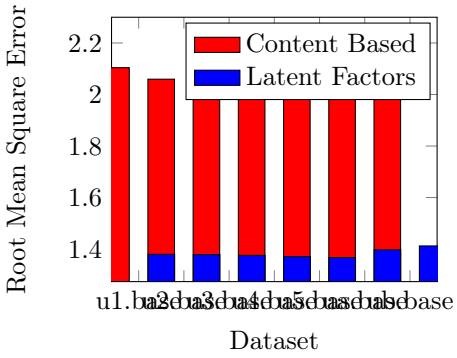

Fig. 9: **Latent Factors vs Content Based on Root Mean Squeare Error**

Even if RMSE is considered a better metric when large errors are undesired, it is useful to check those systems on the $\frac{MAE}{RMSE}$ metric too. It is easily proven that $MAE \leq RMSE$. Those two are equal when the magnitude of all errors is the same. Examining this ratio we can see if the magnitude of the errors has close values.

Tab. 5: **Content Based Algorithm Results vs MAE over RMSE**

Training Dataset | Testing Dataset	MAE \ RMSE
u1.base | u1.test	0.7826436654
u2.base | u2.test	0.7705881338
u3.base | u3.test	0.769284562
u4.base | u4.test	0.779348957
u5.base | u5.test	0.775823798
ua.base | ua.test	0.7738870097
ub.base | ub.test	0.7791162099

As it was expected and this area of examination, latent factors system has ten percent (10%) fewer error spikes than the content based on every dataset. The graph below depicts the results.

Tab. 6: **Latent Factors Algorithm Results vs MAE over RMSE**

Training Dataset | Testing Dataset	MAE \ RMSE
u1.base | u1.test	0.8568332404
u2.base | u2.test	0.856103107
u3.base | u3.test	0.8566826659
u4.base | u4.test	0.8558442062
u5.base | u5.test	0.8550276992
ua.base | ua.test	0.8596460429
ub.base | ub.test	0.8593943861

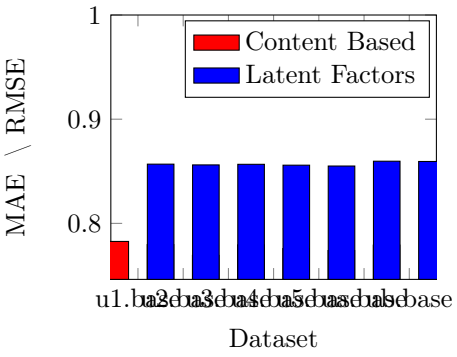

Fig. 10: **Latent Factors vs Content Based on MAE over RMSE**

The last metric we took in order to compare those two systems was the execution time. Our execution time metric we included the time need to train the system against a data set and the time needed to calculate the metrics. We extracted the methods used on metrics calculation in order to be commonly used and impact each execution time result on the same level. The results can be found in the tables below and on the graph that visualizes them, also

below.

Tab. 7: **Content Based Algorithm Results vs Execution Time**

Training Dataset	Testing Dataset	Execution time (ms)
u1.base	u1.test	30114
u2.base	u2.test	21414
u3.base	u3.test	21764
u4.base	u4.test	21887
u5.base	u5.test	24124
ua.base	ua.test	26640
ub.base	ub.test	26861

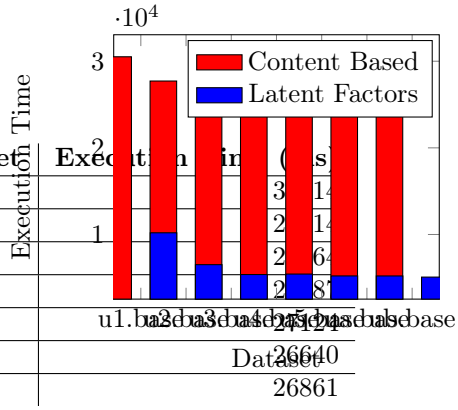

Fig. 11: **Latent Factors vs Content Based on Execution Time**

Tab. 8: **Latent Factors Algorithm Results vs Execution Time**

Training Dataset	Testing Dataset	Execution time (ms)
u1.base	u1.test	10195
u2.base	u2.test	6517
u3.base	u3.test	5377
u4.base	u4.test	5433
u5.base	u5.test	5217
ua.base	ua.test	5214
ub.base	ub.test	5083

Comparing those systems in the execution time metric we can also see that latent factors system outperformed the content base again by taking one-third of the time in its worst case.

10 Conclusion

Recommender systems have a short but intense history. It started from simple statistical models and nowadays it is a great field of study. Recommender systems are now used widely in the online market. Every day you are using them without even noticing it. While you are browsing videos or the web, getting a message from a friend or even listening to music a recommender system might serve you at the time.

In this last chapter of this thesis, we will summarize the experiment and the result we got. This thesis is the attempt of the author to compare two recommender algorithms. Those algorithms were the classic content based and the alternative least squares. The first one is in the area of collaborative filtering while the other is in the latent factors area.

Those two algorithms were implemented or used in Apache Spark. The first, the content based algorithm was implemented, the second one the alternating least squares was used via Apache Spark's MLlib library.

Then those systems were put to test. As metrics were used the mean absolute error(MAE), the root mean square error (RMSE), the ratio between them (MAE/RMSE) and the execution time. Execution time is composed of two parts,

the training time and the time taken to make the metrics. Because the metrics are common, on the same platform and they were using the same code, we can assume that the execution time difference has the training part and the prediction part for every rate in the test dataset.

During the results examination, as it was shown in the previous chapter, we found that the ALS system outperformed the content based on every metric we used. It showed low error metrics, MAE and RMSE, while execution time was low. The ratio MAE/RMSE was high. The last showed us that ALS has fewer data spikes comparing to the content based.

Recommender systems will be around for quite a long time, it is important to know how to compare them. Even more important is to identify which recommender algorithm to use for each business case.

This thesis was a very important milestone for the author. This milestone couldn't be true without the help of those mentioned in the acknowledgment page.

Vasileios Simeonidis,
August 15, 2017

References

[1] J. J. Levandoski, M. D. Ekstrand, M. J. Ludwig, A. Eldawy, M. F. Mokbel, and J. T. Riedl, "Recbench: benchmarks for evaluating performance of recommender system architectures," *Proceedings of the VLDB Endowment*, vol. 4, no. 11, 2011.

[2] A. Said, D. Tikk, K. Stumpf, Y. Shi, M. Larson, and P. Cremonesi, "Recommender systems evaluation: A 3d benchmark.," in *RUE@ RecSys*, pp. 21–23, 2012.

[3] A. Said and A. Bellogín, "Rival: a toolkit to foster reproducibility in recommender system evaluation," in *Proceedings of the 8th ACM Conference on Recommender systems*, pp. 371–372, ACM, 2014.

[4] J. Rowley, "The wisdom hierarchy: representations of the dikw hierarchy jennifer rowley," *Journal of Information Science*, pp. 163–180, 2007.

[5] "Hadoop Vs Spark." `https://amplab.cs.berkeley.edu/projects/spark-lightning-fast-cluster-computing/`. Accessed: 2017-06-24.

[6] "Apache Spark lightning-fast cluster computing." `https://spark.apache.org/`. Accessed: 2017-05-21.

[7] "Netflix prize." `http://www.netflixprize.com/`. Accessed: 2017-08-04.

[8] T. Berners-Lee, R. Fielding, and H. Frystyk, "Rfc 1945: Hypertext transfer protocol—http/1.0, may 1996," *Status: INFORMATIONAL*, vol. 61, 1997.

[9] J. G. Carbonell, R. S. Michalski, and T. M. Mitchell, "An overview of machine learning," in *Machine learning*, pp. 3–23, Springer, 1983.

[10] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method," *arXiv preprint physics/0004057*, 2000.

[11] "Dictionary.com unabridged," Oct 2017.

[12] B. H. Haoming Li, M. Lublin, and Y. Perez, "Cme 323: Distributed algorithms and optimization, spring 2015." `http://stanford.edu/~rezab/dao`, 2015. Lecture 14, 5/13/2015.

[13] "IBM what is map-reduce." `https://www.ibm.com/analytics/us/en/technology/hadoop/mapreduce/`. Accessed: 2017-06-24.

[14] "Apache Hadoop." `http://hadoop.apache.org/`. Accessed: 2017-06-24.

[15] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, (Berkeley, CA, USA), pp. 10–10, USENIX Association, 2010.