7-2018

THE USE OF RECOMMENDER SYSTEMS IN WEB APPLICATIONS – THE TROI CASE

Donjeta Mulaj
University for Business and Technology - UBT, Donjeta.mulaj@gmail.com

University for Business and Technology School of Computer Sciences and Engineering

THE USE OF RECOMMENDER SYSTEMS IN WEB APPLICATIONS – THE TROI CASE

Candidate - Donjeta Mulaj

July / 2018

Pristina

Universities for Business and Technology School of Computer Sciences and Engineering

Master Thesis

Academic Year (2013/2014)

Student: Donjeta Mulaj

Supervisor: Krenare Pireva Nuqi

May / 2018

ABSTRACT

Avoiding digital marketing, surveys, reviews and online users behavior approaches on digital age are the key elements for a powerful businesses to fail, there are some systems that should preceded some artificial intelligence techniques. In this direction, the use of data mining for recommending relevant items as a new state of the art technique is increasing user satisfaction as well as the business revenues. And other related information gathering approaches in order to our systems thing and acts like humans. To do so there is a Recommender System that will be elaborated in this thesis. How people interact, how to calculate accurately and identify what people like or dislike based on their online previous behaviors. The thesis includes also the methodologies recommender system uses, how math equations helps Recommender Systems to calculate user's behavior and similarities. The filters are important on Recommender System, explaining if similar users like the same product or item, which is the probability of neighbor user to like also. Here comes collaborative filters, neighborhood filters, hybrid recommender system with the use of various algorithms the Recommender Systems has the ability to predict whether a particular user would prefer an item or not, based on the user's profile and their activities. The use of Recommender Systems are beneficial to both service providers and users. Thesis cover also the strength and weaknesses of Recommender Systems and how involving Ontology can improve it. Ontology-based methods can be used to reduce problems that content-based recommender systems are known to suffer from. Based on Kosovar's GDP and youngsters job perspectives are desirable for improvements, the demand is greater than the offer. I thought of building an intelligence system that will be making easier for Kosovars to find the appropriate job that suits their profile, skills, knowledge, character and locations. And that system is called TROI Search engine that indexes and merge all local operating job seeking websites in one platform with intelligence features. Thesis will present the design, implementation, testing and evaluation of a TROI search engine. Testing is done by getting user experiments while using running environment of TROI search engine. Results show that the functionality of the recommender system is satisfactory and helpful.

TABLE OF CONTENTS

Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

Acknowledgements

First and foremost, I offer my sincerest gratitude to the person who supported me continuously during my Master Thesis journey with her knowledge, patience and motivation, so many thanks to my supervisor Dr. Krenare Pireva Nuci. I attribute the level of my Master's degree to her encouragement and effort and without her guidelines this thesis would not have been completed or written. In a simple way, I could not wish a better or friendlier supervisor.

I want to thank my husband and my parents too, for supporting me throughout all my studies at University for Business and Technology. I want to pronounced that this thesis is a dedication to my son called TROI, at the same time saying sorry to him, for the time I sacrificed on studies and I could spent with him. Hope so this thesis will make him proud with his mam and makes his life easier in the future.

And at last but not least thanks to GOD for giving me health and straightens to finish the studies.

1. Introduction

The explosive growth of the world-wide web and the emerging popularity of e-Commerce has caused a big mess on the way how to extract useful information. There is a trend to involve Recommender Systems technologies, which is developed to minimalize the gap between information collection and analysis, by applying particular filtering algorithms to existing information in order to rank and predict the relevant data of the user [1].

With the use of various algorithms, the Recommender Systems has the ability to predict whether a particular user would prefer an item or not, based on the user's profile and their activities. The use of Recommender System is beneficial to both service providers and users.

From the providers perspective, the companies gather information about the preferences of users, based on analysis of their activities, what they have liked so far and what they may prefer in the future, this approach of using technology for personalizing user activities had a positive impact on business revenues. Whereas, from user's perspective even when they experience the situation that they don't know what exactly to search, a Recommender System will take care for appropriate suggestions.

Therefore, the need to use efficient and accurate recommendation techniques within a system that will provide relevant and dependable recommendations of items for particular users cannot be over-emphasized [2]. In the popular web sites, the site employs a Recommender System to personalize the online store for each customer tending to adapt the "items" and the "environment" based on users characteristics. For example, if the e-commercial online store that provides songs (and CDs), that has integrated a Recommender System, it tends to recommend to the users the similar "Song" that the user has experienced in the past, or that showed interests while surfing in their online store. In this commercials domain, the use of Recommender System is applied to facilitate the decision-making process whether the recommended song is appropriate, and if so, it tends to pursue the user to "hear" or "buy" the product.

From the technical perspective, there are two main categories of Recommender Systems that the providers are integrating in their online shops, the Recommender Systems that are based on the data used for ranking and recommending the items [1]:

- Content based filtering and
- Collaborative filtering [1].

Content based filtering (CBF) approaches create relationships between items by analyzing and inheriting characteristics of the items based on their content. In contrast, collaborative filtering (CF) systems do not analyze the item content properties, but instead take advantage of information about users' habits and their activities to recommend potentially interesting items. The analysis of user behavior patterns, allows collaborative filtering systems to consider characteristics that would be difficult to acquire from content-based systems approach. CF approaches are also well suited to handle semantic heterogeneity, when different research fields use the same word to mean different things [1].

Let's describe the idea through the online store example that we mentioned previously, which applies a Recommender System to recommend and predict songs (Songs to users based on user profile or user activities on the system). The system in this online store have captured the user preferences based on a hybrid use of algorithms specified above, which facilitate the process of recommending appropriate items to the user, based on users' activities and initial interests[1]. The process flows as follows:

1. The user initiates the request
2. The system ranks the appropriate items of interests, as well as suggests new items based on user background and user activity history
3. The list of items is shown to user interface

To conclude, today the integration of recommender systems is used in various domains [1], so in this context, through this thesis we are proposing the use of recommender systems in an online search engine, which mainly deals with recruitment process of users, in a list of jobs that are provided from the local and regional companies, as well as government institution. In order to explain the idea behind our proposal, we first started with literature review which is going to be presented in the Chapter 2, in order to be able to review the existing technologies used so

far and be able to justify our new proposal in line with the findings that we have from this chapter. Then in chapter 3, is defined the problem declaration and the list of objectives which are specific and measurable toward dealing with the problem raised. Chapter 4, is described the methodology used within this thesis, further in Chapter 5, is described the process of design and implementation of the new proposal framework, named TROI Search Engine, followed by the explanation of technologies used in TROI search engine. Furthermore, in Chapter 7, are discussed the evaluation approaches followed in order to evaluate the proposed, TROI search engine. And finally, the thesis in concluded through chapter conclusion.

2. Literature Review

Recommender Systems are information processing systems that actively gather various data in order to generate their recommendations to particular users [3]. Recommender Systems often need to process large amount of data and produce useful information. The similar activities are followed in our everyday life, when someone is interested to buy a CD or watch a movie, they usually take into account the suggestions of their friends and more. These suggestions pursue the user whether to by the product or not. In the similar direction, the use of recommender systems could facilitate this process through personalization techniques, which takes into account the users activities as well as user's friend's activities. Based on the algorithms used the recommender systems ranks and recommends the appropriate items to the users based on users' background and/or their activities.

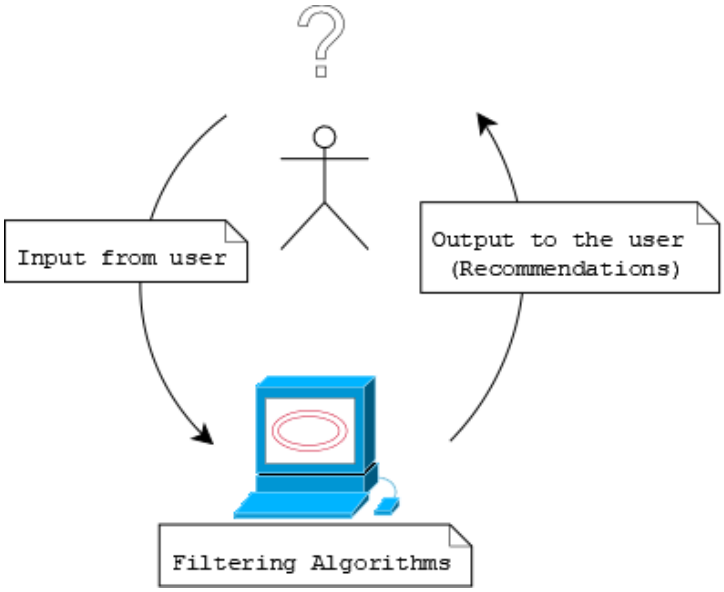

Figure 1: User interaction with Recommender System

In Figure 1 depicts the approach, how the recommender systems filter the input data, depending on the algorithms used, and then it recommends the list of desired "items" to the users, which tends to be those that could match users interest and users background.

Input data of the system is the most important part in order to feed the system with the appropriate indented desire of the users, and these data are going through various phases (discussed further in the next chapters) in order to end up with appropriate recommendation to the user. Further to the directed input data of the user, the system could encounter the user background, which could be the users' past activities, such as the ratings, number of visited pages, the times spent in particular link, to name a few. Following various processing techniques that will be discussed in following section, the output data from the system are the listed items that the system responds to the users, such as: the list of items that the system ranks and predict, and the related information of that item that the system will provide as part of item descriptions.

In order to describe the whole system, how the recommender systems work, and how these data are processed using various algorithms, we firstly start to explain what recommender systems are, and then describe the limited number of algorithms that are being used nowadays, in order to facilitate the process of recommendation.

2.1 Recommender System

In order to be able to predict the user activities, facilitate the decision making of the user, and make easier the process of hitting the appropriate desired items, many top providers have adapted the recommender technology to their customers, such providers as: Amazon, YouTube, Netflix, Yahoo, TripAdvisor, Last.fm, and IMDb 1 which have increased the user satisfaction as well as the business revenues what they are aiming for. Moreover, number of media companies are now developing and deploying Recommender Systems as part of the services they provide to their subscribers. A promotion of Recommender Systems and its application in various web application has experienced a popularization, when the Netflix competition was organized. Netflix hosted a competition for drafting the most appropriate and effective algorithms in order to rank and recommend the movies to their users, where the price for the best proposal exceeded a million dollar. As part of many proposals within this competition the

[1] https://www.amazon.com/ , https://www.youtube.com/ , https://www.netflix.com/ , https://www.yahoo.com/ , https://www.tripadvisor.com/ , https://www.last.fm/, https://www.imdb.com/

proposed hybrid approach has improved substantially the performance of Recommender System [4].

In fact, there are various reasons as to why service providers may want to exploit this technology (Recommended Systems) [3]:

- Selling the product/service – Service providers are always looking for revenue growth. Applying Recommender Systems techniques is an important function, which creates recommendation for users and these recommendations has increased the process of sales.
- Selling more diverse items - Categorizing data and allowing users to select more than one category. It helps the user to see diverse items.
- User satisfaction - Building well designed Recommender Systems, with appropriate user interfaces makes the user more familiar with the services that are offered from providers, which indirectly increases the user satisfaction by recommending the appropriate items.
- User fidelity – User should have enough information on HOW TO use the system. Recommender System should identify the users and inform them using various kinds of user-system interaction approaches. Also rating system manage user's satisfaction.
- User understanding – Recommender Systems are built to know users' preferences, by applying different techniques. It is easier to calculate preferences if their valid actions are valid. With validity means, the process of clicking or visiting pages they really like. This could be measured with the times spent in the particular page/link.

All these aforementioned reasons had an impact to raise the importance, as well as reason "WHY" service providers should involve Recommender Systems in their existing systems.

2.2 Recommendation filtering techniques

In Recommender Systems, the calculation is based on two dimensions matrix, one represents the user dimension and the other one represents the item. The process of manipulation between the user and the item generates an activity named the transaction which reflects the preferences of the user for that particular item [3].

"Items" are the objects that are recommended, which are characterized by their complexity and their value or properties. There is evidence that when a user is acquiring an item, this will have a cost impact, which covers the cognitive cost of searching for the item and the real monetary cost eventually paid for the item [3]. The value of items is categorized as [5]:

- Items with low complexity are: news, web pages, books, CDs, movies
- Items with larger complexity are: digital cameras, mobile phones, PCs
- Items that have been considered are insurance policies, financial investments, travels, jobs

"Users" of a Recommender Systems, may have very diverse goals and characteristics. In order to personalize the recommendations and the human-computer interaction, Recommender Systems exploit a range of information about the users. This information can be structured in various ways and again the selection of what information to model depends on the recommendation techniques.

"Transactions" are those activities that we generically refer to a transaction as a recorded interaction between a user and the Recommender Systems. Transactions are log-like data that store important information generated during the human-computer interaction and which are useful for the recommendation generation algorithm that the system is using.

Furthermore, the recommenders may increase their "knowledge" based on previous user-item activities, and it increase the reputation of the "items" by analyzing the previous recommendation of items to a group of users, where the recommender may collect information, whether the previous recommendation were successfully recommended or not. For example, the recommender can categorize as "positive" recommendation to particular group of users if

the item is useful for the user, or “negative” if the item was not appropriate and the system has wrongly recommended the particular item.

2.2.1 Content-based filtering

The system tends to learn overtime, when dealing with recommend items that are similar to the ones that the user interacted in the past [3]. A content-based recommender approach works with data that the user provides either explicitly (rating) or implicitly (clicking on a link). Based on that data, a user profile generates, the system uses them for decision process in order to suggest the items to the user. As the user provides more inputs or takes actions on the recommendations, the engine matures over time, and is able to predict the items in more accurate manner.

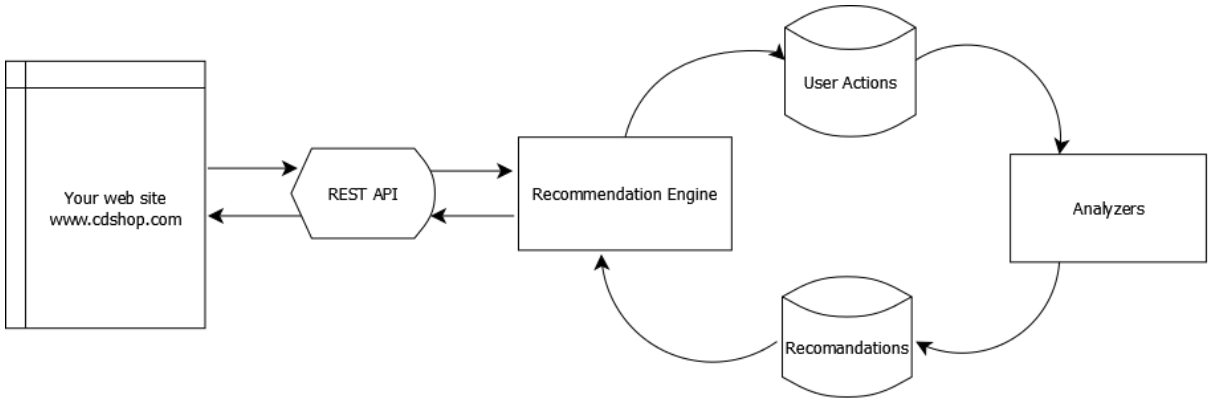

Figure 2 : Recommended System through content – based filtering

In Figure 2 is described a flow of connections between web site and recommendation engine. This connection is done by an API (Application programing interface). API get user data, user activates and send to the recommendation engine. Recommendation Engine analyzers user behaviors and create recommendations [3].

Furthermore, the concepts that are used within recommender systems that do apply content filtering approach are elaborated as follows. The concepts of Term Frequency (***TF***) and Inverse Document Frequency (***IDF***) are used in information retrieval system and also content-based filtering mechanisms (such as a content-based recommender). They are used to determine the relative importance of an item such as: a document, article, news item, and movie. TF is simply the frequency of a word/term in a document. IDF is the inverse of the document frequency among the whole corpus of documents. TF-IDF is used mainly because of two reasons: Suppose we search for ***"the rise of analytics"*** on Search Engine**.** It is certain that "***the"*** will occur more frequently than "***analytics"*** but the relative importance of analytics is higher than the search query point of view. In such cases, TF-IDF weighting negates the effect of high frequency words in determining the importance of an item (document). **Calculation of term frequency**, which measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length (the total number of terms in the document) as a way of normalization[2]:

TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).

Calculation of inverse document frequency, which measures how important a term is. While computing TF, all terms are considered equally important, even a set of words , such as "is", "of", and "that", may appear a lot of times but have no importance, which then will under the removal process, which tends to exclude these unimportant words from the rest of the list. Thus, we need to weigh down the frequent terms while scale up the rare ones, by computing the following formula:

IDF(t) = log_e(Total number of documents / Number of documents with term t in it).

[2] http://www.tfidf.com/
http://ethen8181.github.io/machine-learning/clustering_old/tf_idf/tf_idf.html

So, let's take an example in order to concrete the situation that we are explaining above, so, consider a document containing 100 words where in that document word "cat" appears 3 times. The term frequency for cat is then (3 / 100) = 0.03. Now, assume we have 10 million documents and the word "cat" appears in thousands of these. Then, the inverse document frequency is calculated as log (10,000,000 / 1,000) = 4. Thus, the TF-IDF weight is the product of these quantities: 0.03 * 4 = 0.12.

Calculation of TF-IDF, short for term frequency–inverse document frequency, is a numeric measure that is use to score the importance of a word in a document based on how often it appeared in that document and a given collection of documents. The intuition for this measure is: If a word appears frequently in a document, then it should be important and we should give that word a high score. But if a word appears in too many other documents, it's probably not a unique identifier, therefore we should assign a lower score to that word. The math formula for this measure:

Equation 1: Calculation of Term Frequency (***TF***) and Inverse Document Frequency (***IDF***)

$$TF\ IDF(t, d, D) = TF(t, d) \times IDF(t, D)$$

Where t denotes the terms; d denotes each document; D denotes the collection of documents. It can be seen that the effect of high frequency words is dampened and these values are more comparable to each other as opposed to the original raw term frequency. The numerator: D is inferring to our document space. It can also be seen as D = d1, d2, … , dn where n is the number of documents in your collection. The denominator: | {d ∈ D : t ∈ d} | implies the total number of times in which term t appeared in all of your document d (the d ∈ D restricts the document to be in your current document space). Note that this implies it doesn't matter if a term appeared 1 time or 100 times in a document, it will still be counted as 1, since it simply did appear in the document. As for the plus 1, it is there to avoid zero division[3].

[3] http://www.tfidf.com/
http://ethen8181.github.io/machine-learning/clustering_old/tf_idf/tf_idf.html

Equation 2: Term frequency – Inverse document frequency

$$idf(t, D) = log \quad {|D|}/{1 + |\{d \in D : t \in d\}|}$$

After calculating TF-IDF scores, how do we determine which items are closer to each other, rather closer to the user profile?

Then the similarity within intended search item, and the terms as part of specific corpus could be calculated using Vector Space Model [6].

Vector Space Model (VSM) computes the similarity of items based on the angle between the vectors. VSM represent documents in an n-dimensional space vector where "n" is the size of the vocabulary of terms present in the set of documents that we are trying to represent.

In VSM model, each item is stored as a vector of its attributes (which are also vectors) in an n-dimensional space and the angles between the vectors are calculated to determine the similarity between the vectors. Next, the user profile vectors are also created based on his actions on previous attributes of items and the similarity between an item and a user is also determined in a similar way.

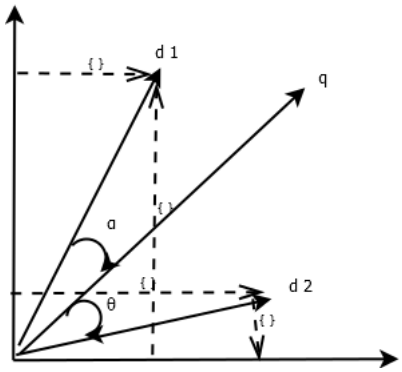

Figure 3 : User profile vectors [7]

In Figure 3 there are two items represented as vectors, d1 and d2. In order to calculate the similarity between these two items we will use vector space modeling which computes the similarity of item's based on the angle between the two vectors. Let's explain through an example, assume we the following documents contains the words as follows:

- Document 1: The boxer rebellion
- Document 2: The boxer
- Document *3:* The rebellion

These three documents can be represented as vectors (as x, y and z) through these words "rebellion", "the", "boxer" respectively. Then the documents numerically can be presented as:

- Document 1: [1 , 1, 1]
- Document 2: [0 , 1 ,1]
- Document 3: [0 , 1, 1]

The value 1 means that that word is present in the document and 0 means that term is not involved. In a 3D space our documents can be visualized as below:

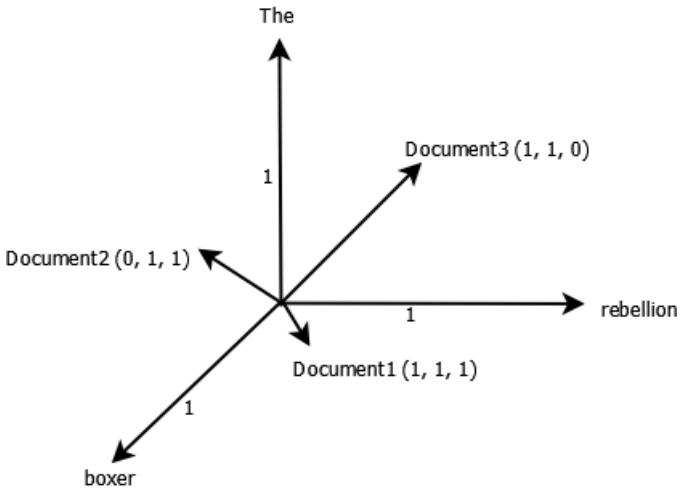

Figure 4 *:* 3D vector presentation for individual words [7]

In Figure 4 are represented these three words “The”, “Rebellion”, “boxer” in three 3D vector presentation. Now we need to calculate the angle between two vectors to find the similarity. To find the similarity between these documents we need to calculate the angle between two vectors. So here we will calculate the similarity for documents (D1, D2), (D1, D3) and (D2, D3) and calculate the similarity with each-other. The calculation cosine formula between two vectors is as follows [7]:

$\overrightarrow{D1=(1,1,1)}$ $\overrightarrow{D2=(1,1,0)}$ $\overrightarrow{D3=(0,1,1)}$

$$\left(\overrightarrow{D1}, \overrightarrow{D2}\right) = \cos\frac{\overrightarrow{D1} \times \overrightarrow{D2}}{\left\|\overrightarrow{D1}\right\| \left\|\overrightarrow{D2}\right\|} = \frac{\sqrt{6}}{3} \approx 0.81$$

$$\left(\overrightarrow{D1}, \overrightarrow{D3}\right) = \cos\frac{\overrightarrow{D1} \times \overrightarrow{D3}}{\left\|\overrightarrow{D1}\right\| \left\|\overrightarrow{D3}\right\|} = \frac{\sqrt{6}}{3} \approx 0.81$$

$$\left(\overrightarrow{D2}, \overrightarrow{D3}\right) = \cos\frac{\overrightarrow{D2} \times \overrightarrow{D3}}{\left\|\overrightarrow{D2}\right\| \left\|\overrightarrow{D3}\right\|} = 0.5$$

Now we can say that, D2 and D3 are little bit farther from each other, D1 and D2 are closer to each other also D1 and D3 are closer to each other. [4]

Cosine similarity: Cosine similarity [8] tends to be useful when trying to determine how similar two texts/documents are. Cosine similarity is suitable for those scenarios because it ignores magnitude and it get focused on orientation. In NLP (Natural Language Processing), this might help us still detect that a much longer document has the same “theme” as a much shorter document since we don’t worry about the magnitude or the “length” of the documents themselves. Intuitively, let’s say we have 2 vectors, each representing a sentence. If the vectors are close to parallel, maybe we assume that both sentences are “similar” in theme. Whereas if the vectors are orthogonal, then we assume the sentences are independent or NOT “similar”. Depending on your use case, maybe you want to find very similar documents or very different

[4] http://trigonaminima.github.io/recommender-systems/tfidf/nlp/2016/11/02/From-Vector-Space-Models-to-Recommender-Systems/

documents, so you compute the cosine similarity. First, we need the geometric definition of the "dot" product [8]:

$$\textit{Equation 3:} \quad A \times B = \|A\| \times \|B\| \cos \emptyset$$

Intuitively, the cos(angle) will be 0 when the angle between vectors are orthogonal. If the vectors are orthogonal, we think of them as linearly independent. If we're looking for "similar" vectors, we probably don't want them to be independent! Ideally, we want the cos(angle) to be as close as possible to 1. We think of the vectors as oriented in the same direction or if they're sentences, perhaps they have the same "meaning". Typically, we compute the cosine similarity by just rearranging the geometric equation for the dot product:

$$\textit{Equation 4:} \qquad \cos \theta = \frac{A \times B}{\|A\| \times \|B\|}$$

A naive implementation of cosine similarity: Let's assume we have 3 sentences that we want to determine the similarity between each other:

- sentence_m = "*Mason really loves food*"
- sentence_h = "*Hannah loves food too*"
- sentence_w = "*The whale is food*"

For this example we compute the process as follows:

- sentence_m: Mason=1, really=1, loves=1, food=1, too=0, Hannah=0, The=0, whale=0, is=0
- sentence_h: Mason=0, really=0, loves=1, food=1, too=1, Hannah=1, The=0, whale=0, is=0
- sentence_w: Mason=0, really=0, loves=0, food=1, too=0, Hannah=0, The=1, whale=1, is=1

Looking at our cosine similarity equation above, we need to compute the dot product between two sentences and the magnitude of each sentence we're comparing.

Table 1: Sentences and the magnitude of each sentence.

	Mason	**Really**	**Loves**	**Food**	**Too**	**Hannah**	**The**	**Whale**	**Is**
sentence_m	1	1	1	1	0	0	0	0	0
sentence_h	0	0	1	1	1	1	0	0	0
sentence_w	0	0	0	1	0	0	1	1	1

Now we just need to calculate the cosine [8]:

$$\cos(\text{sentence\_m}, \text{sentence\_h})$$

Beside the cosines similarity, in order to check the similarity between two items (either users, or content), we can use the **Euclidean Distance** [9, 10] which calculates the similarity. This makes sense if you think of users as points when there are many dimensions (as many dimensions as the items), whose coordinates are preference values. This similarity metric calculates the Euclidean Distance (d) between two such user points. The smaller is distance value then the users are more similar to each other. Similarity will be calculated using the formula $\frac{1}{1+d}$ where d is the distance. The Euclidean distance $r_2(x, y)$ between two 2-dimensional vectors $x = (x_1,\ x_2)^T$ and $y\ =\ (y_1,\ y_2)^T$ is given by:

Equation 5: $$r_2(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2} = \sqrt{\sum_{i=1}^{2}(x_i - y_i)^2)}$$

Table 2: Ratings given to six movies by six film critics (from http://www.metacritic.com).

	Australia	Body of Lies	Burn After Reading	Hancock	Milk	Revolutionary Road
David Denby (New Yorker)	3	7	4	9	9	7
Todd McCarthy (Variety)	7	5	5	3	8	8
Joe Morgenstern (Wall St Journal)	7	5	5	0	8	4
Claudia Puig (USA Today)	5	6	8	5	9	8
Peter Travers (Rolling Stone)	5	8	8	8	10	9
Kenneth Turan (LA Times)	7	7	8	4	7	8

Table 2 shows the ratings that a few well-known US film critics gave to a small group of movies. We shall use these data to develop a simple recommender system. Unlike a realistic system, in this case every person has rated every film. We can represent this data as vector per critic.

$$x_1 = (3, 7, 4, 9, 9, 7)^T$$

…

$$x_6 = (7, 7, 8, 4, 7, 8)^T$$

Where x_1 corresponds to David Denby, x_2 corresponds to Todd McCarthy, and so on. We can go ahead and use equation (3) to compute the Euclidean distances between six critics in the 6-

dimensional review space [9, 10]. Even though, there are various approaches that we could follow in order to find the similarities between interest of the user and the existing job lists, in our proposal, we have used cosines similarity.

Why cosine similarity: Cosine similarity is generally used when working with text data represented by word counts. We could assume that when a word (e.g. science) occurs more frequent in document 1 than it does in document 2, that document 1 is more related to the topic of science. However, it could also be the case that we are working with documents of uneven lengths (Wikipedia articles for example). Then, science probably occurred more in document 1 just because it was way longer than document 2. [5]

2.2.2 Collaborative filtering

In contrast, the collaborative filtering also referring as social filtering, filters information by using the recommendations of other people [11]. It is based on the idea that people who agreed in their evaluation of certain items in the past are likely to agree again in the future. For example, a user who wants to see a particular movie, might ask for recommendations from friends. The recommendations of some friends who have similar interests are trusted more than recommendations from others. This information is used in the decision on which movie to watch [11]. Following this section, there are listed various approaches that are used in order to predict the items using the collaborative filtering approach.

Neighborhood-based approach [11] - Most collaborative filtering systems apply the so-called neighborhood-based technique. In the neighborhood-based approach a number of users are selected based on their similarity to the active user. A prediction of the active user is made by calculating a weighted average of the ratings of the selected users. To illustrate how a

[5] https://cmry.github.io/notes/euclidean-v-cosine

collaborative filtering system makes recommendations consider the example in movie ratings table below. This shows the ratings of five movies by five people. To predict if Ken would like the movie "Fargo", Ken's ratings are compared to the ratings of the others. In this case the ratings of Ken and Mike are identical and because Mike liked Fargo, one might predict that Ken likes the movie as well.

Notice: A "+" indicates that the person liked the movie and a "-" indicates that the person don't like the movie

Table 3: Collaborative filtering system in movie ratings table below.

	Amy	Jef	Mike	Chris	Ken
The Piano	-	-	+		+
Pulp Fiction	-	+	+	-	+
Clueless	+		-	+	-
Cliffhanger	-	-	+	-	+
Fargo	-	+	+	-	?

Instead of just relying on the most similar person, a prediction is normally based on the weighted average of the recommendations of several people. The weight given to a person's ratings is determined by the correlation between that person and the person for whom to make a prediction. As a measure of correlation, the Pearson correlation coefficient can be used. In this example a positive rating has the value 1 while a negative rating has the value –1, but in other cases a rating could also be a continuous number.

Selecting Neighborhoods - Many collaborative filtering systems have to be able to handle a large number of users. Making a prediction based on the ratings of thousands of people has serious implications for run-time performance. Therefore, when the number of users reaches a certain amount a selection of the best neighbors has to be made. Two techniques, correlation-threshold and best-n-neighbor [11], can be used to determine which neighbors to select. The

first technique selects only those neighbors whose correlation is greater than a given threshold. The second technique selects the best n neighbors with the highest correlation [10].

Item -To-Item approach - Several existing collaborative-filtering-based recommendation systems have been designed and implemented since early 90's. Collaborative filtering techniques have been proven to provide satisfying recommendations to users in Shardanand & Maes, 1995. Ringo mechanism determines the similarity of users based on user rating profiles. Instead of measuring the similarities between people the ratings are used to measure the correlation between items. The Pearson correlation coefficient can again be used as a measure. For example, the ratings of the movies "Fargo" and "Pulp Fiction" have a perfect correlation. Based on this correlation one might predict that Alba likes "Fargo" given the fact that he liked "Pulp Fiction" [11].

Classification Approach - Collaborative filtering can also be formulated as a classification problem. To illustrate this, consider again the example in Movie rating table. In order to predict if Alba likes the movie 'Fargo' a learning method has to determine the class of this movie. The learning method can be trained by using the four movies that Alba has rated as training instances. The movies 'The Piano', 'Pulp Fiction' and 'Cliffhanger' are labeled positive while the movie 'Clueless' is labeled as a negative training instance. A movie can be directly represented as a vector, where each component of the vector corresponds to a rating of a different user. This would mean however that many values are missing because normally an item will not have been rated by all the users. For learning methods that cannot handle missing values the items have to be represented in a different way [11].

2.2.3 Hybrid-based filtering

In order to achieve better recommendation results, researchers combined both techniques content filtering and collaborative filtering to build Hybrid Recommender Systems, which seek to inherit advantages and eliminate disadvantages [12, 13].

Depending on the domain and data characteristics, different types of combinations might produce dissimilar outputs. The following list describes several hybridization techniques that come into consideration to merge CF and CBF recommenders.

Table 4: A list of hybridization [12, 13].

Hybridization method	Description
Weighted	The scores (or votes) of several recommendation techniques are combined together to produce a single recommendation.
Switching	The system switches between recommendation techniques depending on the current situation.
Mixed	Recommendations from several different recommenders are presented at the same time
Feature combination	Features from different recommendation data sources are thrown together into a single recommendation algorithm.
Cascade	One recommender refines the recommendations given by another.
Feature augmentation	Output from one technique is used as an input feature to another.
Meta-level	The model learned by one recommender is used as input to another.

Weighted - Perhaps the most straightforward architecture for a hybrid system is a weighted one. Given items are scored separately by both incorporated recommender, whereas the final output is a linear combination of the intermediate results. Typically, empiric means are used to determine the best weights for each component. Note that content-based recommenders are able to make prediction on any item, but collaborative recommender can only score an item if there are peer users who have rated it.

Mixed - In many domains it is infeasible to receive an item score by both recommenders, because either rating matrix or content space are too sparse. Mixed hybridization techniques generate an independent set of recommendations for each component, and join the ranked candidates before being shown to the user. However, merging the predicted items of both recommenders makes it difficult to evaluate the improvement about the individual components.

Switching - Some hybrid systems consist of more than two recommendation components with different underlying CF and/or CBF approaches. Often recommenders are ordered, and if the first one cannot produce a recommendation with high confidence, then the next one is tried, and so on. Otherwise switching hybrids might select single recommenders according to the type of user. However, this method assumes that some reliable switching criterion is available.

Feature Combination - Systems that follow the feature combination approach involve one recommendation component, which is supported by a second passive component. Instead of processing the features of the contributing component separately, they are injected into the algorithm of the actual recommender.

Feature Augmentation - The strategy of feature augmentation is similar in some manner to feature combination. But instead of using raw features from the contributing domain, feature augmentation hybrids support their actual recommender with features passed through the contributing recommender. Usually, feature augmentation recommender is employed when there is a well-engineered primary component that require additional knowledge sources. Due to the fact that most applications expect recommendations in real time, argumentation is usually done offline. In general, feature augmentation hybrids are superior to feature combination methods, because they add a smaller number of features to the primary recommender. Melville, Mooney and Nagarajan developed a feature argumentation recommender that incorporates

content-based and collaborative methods to predict new items of interest for a user. Their Content-Boosted Collaborative Filtering (CBCF) algorithm learns a content-based model over the training data to generate ratings for unrated items. The generated dense rating matrix is then used for collaborative recommendation by the actual recommender. CBCF overcomes disadvantages of both CF and CBF methods, and significantly improved the predictions of the recommender system.

Cascade- The concept of a cascade hybrids is similar to feature augmentation techniques. However, cascade models make candidate selection exclusively with the primary recommender and employ the secondary recommender simply to refine item scores. For example, items that were equally scored by the main component might be re-ranked employing the secondary component.

Meta-Level - This kind of hybrids employ a model learned by the contributing recommender as input for the actual one. Although the general schematic of meta-level hybrids reminds on feature augmentation techniques, there exists a significant difference between both approaches. Instead of supplying the actual recommender with additional features, a meta-level contributing recommender provides a completely new recommendation space. However, it is not always necessarily feasible to produce a model that fits the recommendation logic of the primary component [12, 13].

So, reviewing the recommender techniques, especially the approaches that are used in order to offer the recommendation, he have naturally come to a proposal of creating the named TROI search engine, which aim and learning objectives are proposed in the following Chapter 3.

2.3 Ontology and ontology-based Recommender Systems

Ontology is known as the philosophical study of the nature of being, becoming, existence or reality as well as the basic categories of being and their relations. Traditionally listed as a part of the major branch of philosophy known as metaphysics, ontology often deals with questions concerning what entities exist or may be said to exist and how such entities may be grouped, related within a hierarchy, and subdivided according to similarities and differences. Although ontology as a philosophical enterprise is highly theoretical. Parmenides of Elea was a pre-Socratic Greek philosopher and he was among the first to propose an ontological characterization of the fundamental nature of reality. Etymologically the term ontology, comes from Greek and means essentially "the study or theory of being". Yet, historically the first known written use of the word comes from the Latin ontologies in the early 17th century. The term ontology is an apt example of such a word. We know the accepted history of its use and its etymology, we know how it has been commonly used throughout history and we can study its changes with the advent of AI and computer science in the mid-1970s.

The modern history of ontology really beings within Artificial Intelligence (AI) research around 1970s and 1980s. In computer science, ontology is a technical term presenting an artifact that is designed for a purpose, which is to enable the modeling of knowledge about some domain, real or imagined. Thus, an ontology as defined and used within modern computer science (and now many other fields), in simple terms, a system for the formal organization of information. This relates to the ancient philosophy on the nature of existence in that both systems classify "being/that which exists" whether they are subjects/objects in a domain, conceptual models for automated reasoning, or categories of individual identity. There are literally thousands of existing ontologies in the world today in virtually every industry from software engineering to medical research, e-commerce to banking, linguistic processing to document publishing and so forth. Even in the Data Management industry alone there are too many to easily discuss [14, 15, 16, and 17].

The rapid growth of Unstructured Data over the past few years produced an explosion of Big Data products meant to aid in the "structuring" of such massive amounts of data created from blogs, social networking, video and a host of other "unstructured" elements.

Enterprises worldwide are hurrying to collect, analyze, and translate into appreciable information petabytes and even Exabyte's of data so they can achieve a competitive edge in the marketplace. The past few years have been a focused-on ontologies and the development of tools using it, as well as standards for their representation and access via the World Wide Web. Most current work involves the rapid construction of ontologies for specific domains by a small team of "ontology engineers", who typically identify the relevant concepts and relations apriority and then impose it on the system. The resulting ontologies are typically static, representing a set of objects/concepts, relations, and properties that remain constant [14, 15, 16, 17].

However, in some cases, the domain modeled by an ontology changes over time, and, depending on the time perspective, a different ontology is relevant for retrieval. For example, in order to retrieve all relevant information for a query about Germany, it may be necessary to recognize that instances of East Germany and West Germany in an ontology representing geo-political entities prior to 1989 are each a part of a single entity, Germany, in a later ontology. This demands that information represented in the ontology is both temporally contextualized and that relations among entities that are relevant during different temporal intervals are available to support user queries.

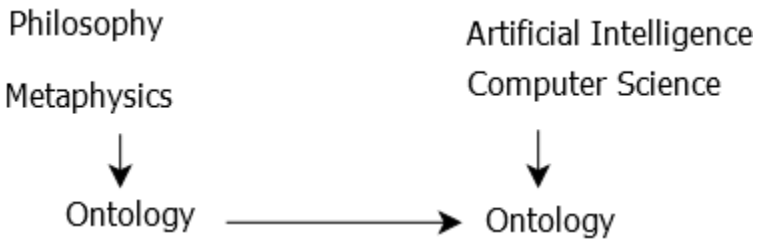

Figure 5 : Ontology originated

Furthermore, it is necessary to count for the fact that the course of the ontology's evolution and the processes that have affected it are a part of the knowledge that should be brought to bear on the analysis of information at any given time [14, 15, 16, and 17].

Different ontology definitions can be found in the computer and information science literature. But we can conclude that ontology is used for:

- sharing of the domain knowledge
- reusing of the domain knowledge
- analyzing domain knowledge
- separation of domain knowledge from operational knowledge
- making domain assumptions explicit

The most prevalent ontology definition is the following [14]:

"A body of formally represented knowledge e is based on a conceptualization… A conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose. Every knowledge base, knowledge-based system, or knowledge-level agent is committed to some conceptualization, explicitly or implicitly. An ontology is an explicit specification of a conceptualization."

We analyze Gruber's ontology definition in more detail. Three main concepts are central to this definition: "formal", "domain conceptualization" and "explicit". "Formal" refers to knowledge representation that is mathematically described and machine readable. A "domain conceptualization" is an abstract model of a phenomenon, i.e. an abstract view of domain concepts and relationships among them. "Explicit" expresses clear and precise definitions of concepts and their relationships [14].

In artificial intelligence, the knowledge in computer systems is thought of as something that is explicitly represented and operated on by inference processes. Any software that does anything useful cannot be written without a commitment to a model of the relevant world- entities, properties, and relations in that world. It is common to ask whether a payroll system "knows" about the new tax law, or whether a database system "knows" about employee salaries. Information-retrieval systems, digital libraries, integration of heterogeneous information sources, and internet search engines need domain ontologies to organize information and direct the search processes. For example, a search engine has categories and subcategories that help organize the search. The search-engine community commonly refers to these categories and

subcategories as ontologies. Object-oriented design of software systems similarly depends on an appropriate domain ontology. Objects, their attributes, and their procedures more or less mirror aspects of the domain that are relevant to the application. Object systems representing a useful analysis of a domain can often be reused for a different application program. As information systems model large knowledge domains, domain ontologies will become as important in general software systems as in many areas of artificial intelligence. In artificial intelligence, while knowledge representation pervades the entire field, two application areas in particular have depended on a rich body of knowledge. One of them is natural-language understanding. Ontologies are useful in NLU (natural language understanding) in two ways. First, domain knowledge often plays a crucial role in a well-designed domain ontology provides the basis for domain knowledge representation. In addition, ontology of a domain helps identify the semantic categories that are involved in understanding discourse in that domain. Knowledge-based problem solving is the second area in AI that is a big consumer of knowledge.

Ontology specification in knowledge systems has two dimensions:

- Domain factual knowledge provides knowledge about the objective realities in the domain of interest (objects, relations, events, states, causal relations, and so forth).
- Problem-solving knowledge provides knowledge about how to achieve various goals. A piece of this knowledge might be in the form of a problem-solving method specifying—in a domain-independent manner—how to accomplish a class of goals.

With few exceptions the domain factual knowledge dimension drives the focus of most of the AI investigations on ontologies. This is because applications to language understanding motivates much of the work on ontologies [14].

2.2.3.1 Role of Ontology in Recommended Systems

While storing digital information has become possible, retrieving and accessing resources in the growing collections is far from trivial. We are facing a mixture of information originating from professionally managed collections such as image or text databases to individually or collaboratively created content such as personal image collections, online encyclopedias or even the World Wide Web itself [15].

Digital information is mainly accessed using information retrieval (IR) systems. IR systems assume that the users are able to express their information need in the form of a query. In its most common form, a user enters a set of keywords which summarize the user's information need. Given the query, the goal of an IR system is to retrieve information which is relevant to the information need of the user. Recommender systems form a specific type of information filtering (IF) technique that attempts to present information objects that are likely of interest to the user. Instead of users actively searching for information, recommender systems provide advice to users about objects they might wish to examine. Recommendations can be based on the content of the objects or observations of user behavior [15].

In computer science, an ontology can be defined as a specification of a representational vocabulary consisting of definitions of classes, relations, functions, and other objects for a shared domain of discourse. Ontology-based methods can be used to reduce problems that content-based recommender systems are known to suffer from. These problems concern the way the systems analyze the content they recommend, the way they retrieve the content, and the way they treat heterogeneously represented content [14].

Ontologies have been applied to a variety of recommender systems to reduce content heterogeneity and improve content retrieval. For example, in good results to cope with content heterogeneity have been obtained by using sub summation hierarchies to generalize user profiles. Ontologies are used on a specific domain of product descriptions, and a hand crafted ontology is built just for this purpose. Similar approach is adopted for television program domain, and in for e-tourism domain complemented with mining the user behavior [15].

Ontology represents the concepts of a certain domain and their interrelations. In ontology-based recommender system, ontology represents both the user profile and the recommendable items. Preferences are richer and more detailed than the standard keywords-based ones. Ontology interprets terms with other terms according to their semantic relations. Its hierarchical structure allows an analysis of preferences at different abstraction levels. It is used to store and exploit the personal preferences of a user and has powerful modelling and reasoning capabilities. It permits a high degree of knowledge sharing and reuse. There are some ontology-based recommender systems. [16]

2.4 The role of recommended systems in Job Seeking and Recruiting Website

Recommendation system are applicable in job seeking and recruiting websites too. Recommendation systems helps to find the right candidate with the right job [18]. For this recommendation techniques can be applied such as content-based filtering, collaborative filtering and hybrid approaches can be applied. The content-based approach matches candidate profiles with employer profiles and job requirements. Job recommendation has played an important role on the online recruiting website. In particular, job recommender system are designed to retrieve a list of job positions and provide this suggestions to job candidates based on their profile and preferences [18]. Jobs are published automatically on the job portal as soon as they are entered into the system. On the other hand, the applicant creates a profile to apply it for one of the listed job positions. The user profile is stored in the system, letting the applicant reuse it for other job position. There are major requirements that should be derived when recommending candidates for a specific job [19].

- The matching of individuals to job depends on skills and abilities that individuals should have.
- Recommending people is a bidirectional process that needs to take into account the preferences not only of the recruiter but also of the candidate.
- Recommendations should be based on the candidate attributes, as well as the relational aspects that determine the fit between the person and the team members with whom the person will be collaborated.

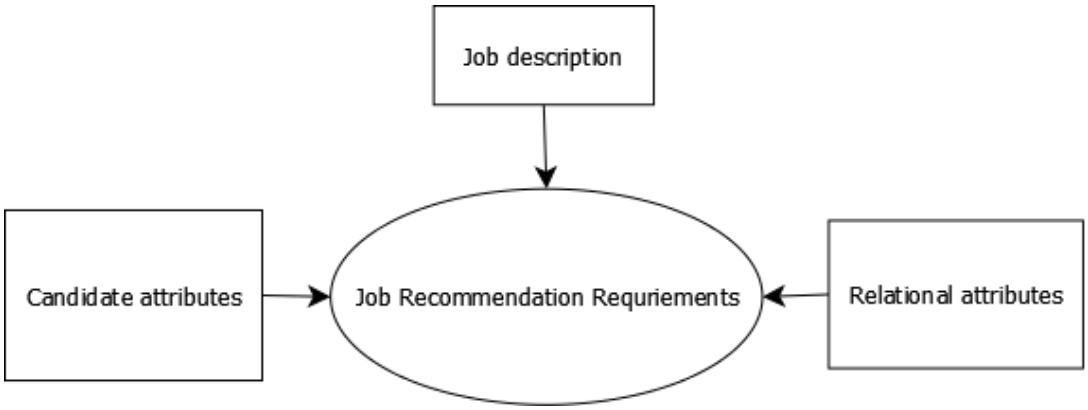

Figure 6: Model of system requirements for candidates/job recommendation [19]

As we can see on Figure 6 a job recommendation retrieve information's form candidate attributes job description and relational attributes in order to suggest the proper job to the candidates.

3. Problem Declaration

As elaborated in the previous chapters, the recommender systems are used mainly to offer a personalize services for the users of various online platforms in various domain, such as: online movie provider, online commercial shops, to name a few. However, in our case we are interested to see whether the idea of personalized services can be applied in job seeking platforms, and in order to offer personalized services which factors needs to be encounter. Is it enough to analyses the user profile, users' activities, or the algorithms used from the system perspective?

3.1 Aim and Objectives

The aim of this thesis is to analyze such factors and their impact in the overall performance of the recommender systems in jobs findings platforms. Furthermore, a list of algorithms used in recommender systems will be elaborated, and their performance will be simulated in a new proposed recommender system.

Objectives:

O1: A review of exiting recommender systems

O2: A review of techniques used in recommender systems

O3: A new proposed recommender systems and the integration of a particular algorithm in the new proposed TROI search engine

4. Methodology

Within this thesis the various methods are used in order to represent the new proposed TROI Search Engine, a new innovative Recommender System used locally. The Technology behind TROI search engine is described further in next chapters.

For reviewing the exiting recommender systems and the use of ontologies within RS, a **literature review method** has been used. Continuing with the **analysis and comprehensive method**, which enabled to list a number of algorithms and analyze their performance based on lower and upper bound approach.

Further, is used a **software development process** to develop the new proposed TROI search engine, which has integrated the **Apache Lucene, Apache Mahout** algorithms, for being able to personalize the item/product based on user profile.

5. Design and implementation of TROI Search Engine

Developing the proposed TROI search engine, we have followed the agile methodology, specifically the Agile Modeling. Agile Modeling a practice-based methodology for effective modeling and documentation of software-based systems, it contains a collection of values, principles, and practices for modeling software that can be applied on a software development project in an effective and light-weight manner [20]. Agile breaks down larger projects into small, manageable chunks called iterations. At the end of each iteration (which generally takes place over a consistent time interval) something of value is produced. The product produced during each iteration is able to be put into the world to gain feedback from users or stakeholders. While building TROI search engine we have followed two iteration to make the first release version. During first iterations is implemented, user interface design which includes this stories:

Design and implementation for:

- Home page
- Login
- Logout
- User profile

In the second iteration is done the logic of TROI search engine. Indexing process which collect documents which contains job positions. List this jobs on home page. And provide suggestions to the user.

- Index job pages
- List jobs
- Provide suggestions

Release	Iterations	Stories	Size
Version 1.0	1	Home page	3
		Login	5
		Log out	5
		User profile	13
	2	Index job pages List jobs Provide suggestions	21 13 21

There was a need in Kosovo market to have a search engine for finding jobs. Plan is started while drawn it in notebook paper. Then we end up with specific use cases and work flows. Developing was separated in small tasks and is implemented step by step.

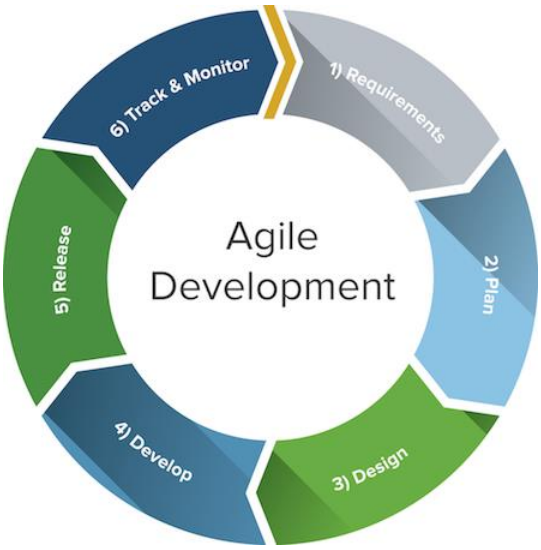

Figure 7: Agile Development Steps [20]

The proposed recommended system TROI is a search engine that facilitate the process of seeking jobs in Kosovo region. TROI search engine besides that it lists the number of jobs that are directly requested from users, it includes also a number of recommended jobs that the system has proposed to the users based on their profile.

The diagram depicted in Figure 8, shows three key processes that TROI search engine goes through in order to display a list of jobs to the users.

Figure 8 : TROI search engine flow of generating recommendations.

User requirements - While checking user activities in TROI search engine and we analyze user profile, user activities in system.

Processing requirements - We attempt to use ontological concepts or structures to address the content the semantics of documents and queries. Based on ontological concepts we made a job structures or job categorizations, user requirements will be process throw this structure, and also display recommendations.

Display recommendations

System lists the recommendations, which are processed by Apache Lucene and Apache Mahout Algorithms.

5.1 TROI search engine

In TROI search engine we have use the ontology as a set of hierarchy of concepts in order to be able to generalize and specialize the user queries [21]. So, ontology is a formal naming and definition of the types, properties, and interrelationships of the entities that really exist in a particular domain of discourse. For simplicity reason, there has been developed a simple ontology which tended to categorize the jobs in a hierarchical ordering. As a consistent and well-defined set of concepts is used both to index and retrieve jobs. Reasoning with ontological structures may also be applied in cases where there is no direct match between the query and the jobs. In practice, ontology-driven or semantic search can be achieved with semantic indexes, semantic annotations, query interpretation, or a combination of the above [17]. Many applications use ontological structures to reformulate queries with more generalized or specialized terms, producing results that are comparable to thesaurus- or dictionary-supported search solutions [17, 21].

TROI search engine, analyzes different pages that were used to provide information about seeking jobs in Kosovo Region based on user's interest. Recommendations (Objects) are issued based on comparison between their content and a user profile. As much detailed information is available about the user, system will able to build a more intelligent prediction.

We can formulate recommendations by the following formula, expressed in equation 6. A recommender system maps each user profile- object pair to a particular rating value by estimating the rating [14].

Equation 6 : $UserPrifiles \times Objects \rightarrow Ratings$

The rating function can be estimated in a way that the highest rated object O′ (or a number of highest rated objects) are selected:

Equation 7: $O' = arg\,max\, U \epsilon\, User\ Profiles, O \in Objects\ \ R(U,O)$ *[14]*

In content-based recommendation methods, the rating R (U, O) of object O for the user profile U is typically estimated based on the ratings assigned in the user profile U to other objects that

are relevant to object O in terms of their content. The definition of the rating function requires to measure the similarity between the user profile and the objects. The content of the objects O is characterized using a set of features, here defined as$Content\ (O)$. In addition, profile of a user U needs to be defined. The user profiles are also defined in terms of features that characterize the objects, here $ContentBasedProfile(U)$. The rating function can now be written as a score function of the content-based profile and a content object:

Equation 8: $R(U,O) = score(ContentBasedProfile(U), Content(O))$

In the case of content-based recommender systems, where the scoring is based on the content descriptions available in text or structured annotation, the score function can be implemented using methods developed in IR research [14].

Ontology-based Information Retrieval Methods

Light-weight ontologies provide controlled vocabularies that can be used in annotation of objects. This approach has brought improvements over classic keyword-based search through e.g. query expansion based on class hierarchies and other relationships. In the case of content-based recommender systems the profiles can be larger and it is not likely that all of the features that appear in an individual profile appear in an individual object. This emphasizes the importance of ranking. Recently, ranking of the ontology-based search has been enabled by extending Vector Space Model to combine text-based features and ontology-based features. The VSM enables straightforward representation of the objects and fast computation of the score function. Furthermore, this thesis extends the retrieval model by retrieval result clustering, where the initially highest ranked objects are clustered based on the annotation structure to avoid over-specialization. More about Vector Space Model is described in chapter two.

5.2 TROI design and implementation

TROI is a Recommender System which suggest jobs based on users' profiles and their activities. The backend of the TROI search engine has connected number of platforms that are operating in this domain, here in Kosovo. Such platforms as:

- https://ofertasuksesi.com
- http://www.peshkuiarte.com/
- https://www.duapune.com/
- http://ofertapune.net/
- https://telegrafi.com/

So, when users are tending to search for a specific job, they will probably do it by visiting websites one by one. So we were thinking whether we can create a search engine for seeking jobs, so it can operate like a hub to all these websites, and centralize the accessing opportunities for our users. So, we ended up with the proposal of TROI search engine which is built using Java technologies, in a maven project. For searching and to retrieve job information is used Apache Lucene, and to build recommendations is used the Apache Mahout, which are furthers explained in Chapter 6. Figure 9 shows the registration screenshot of TROI Search Recommender Systems. The form and the whole system is in Albania language since the users that will be using it are Albanians. The system is Model-View-Controller (MVC) and easily programmable for integrating other language packs in the future.

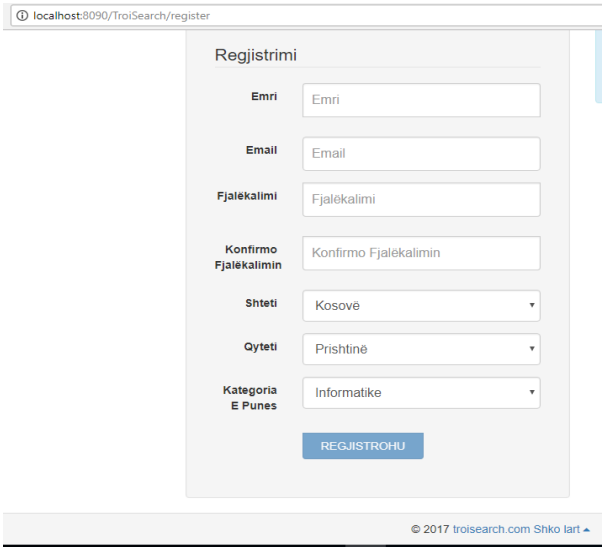

Figure 9: TROI search engine Registration form.

During the registration phase, the user is required to fulfill the following information: *name, email, password, password confirmation, country, and city and job category*. After the user has created successfully a profile, the users will be redirected to home page, which can contain job details and related information through which they can surf for different job categories.

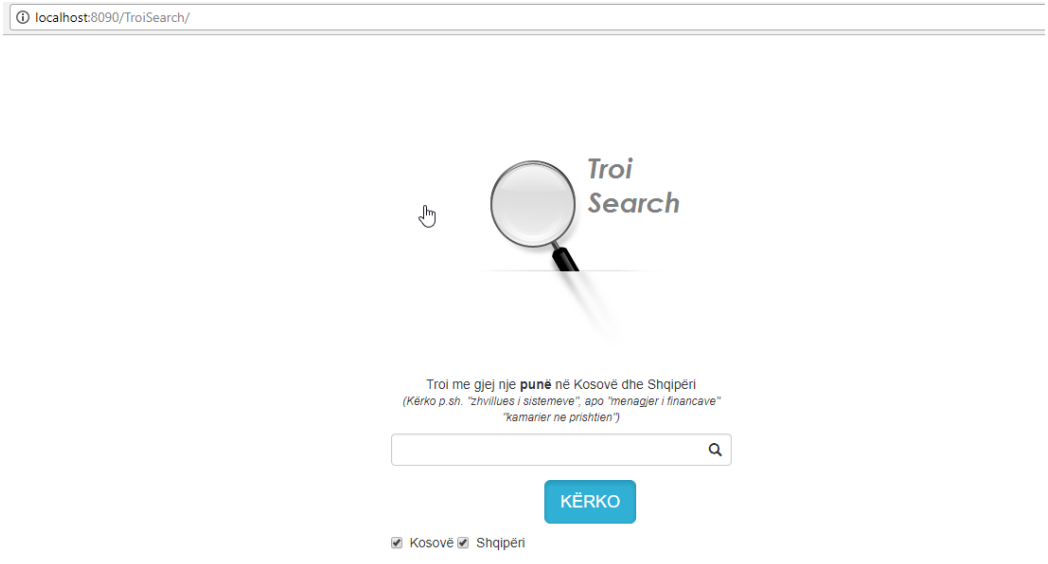

Figure 10: TROI search engine home page.

5.3 Use Case Diagram for TROI Search

A use case diagram is a graphic description of the interactions among the elements of a system [23]. In TROI search engine case, the use case diagram is depicted in Figure 11:

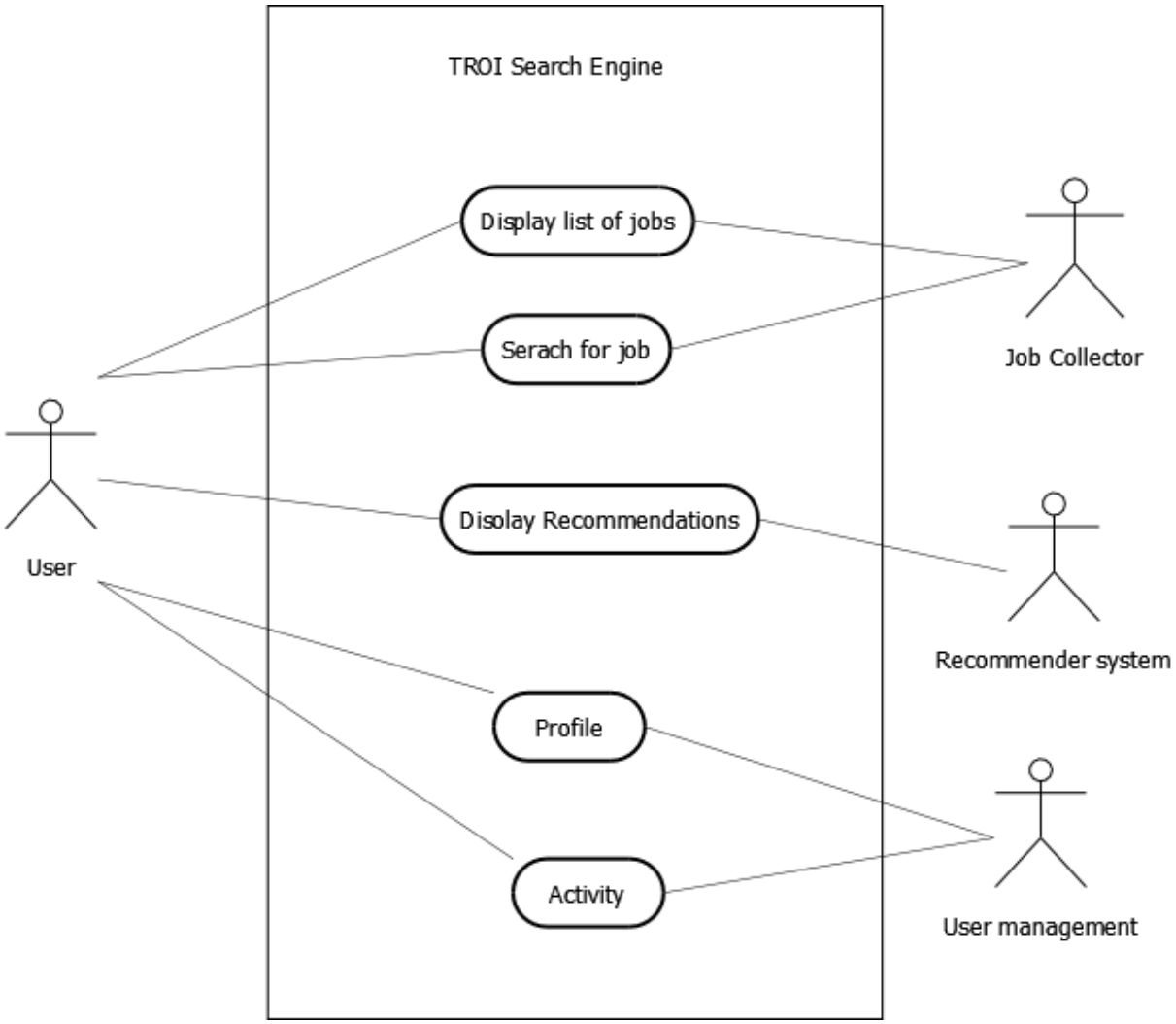

Figure 11: Use case diagram TROI search engine

System manage users and they requirements, based on their profiles, likes and preferences, and based on this information it tends to recommend jobs that are related to his interest. User create profile, search for jobs, give likes for those jobs, give ratings and read the recommendations for jobs.

5.3.1. User profile creation workflow

Work flow diagrams presents steps or actions in different situations. Workflow diagrams are used in documentation's and during the implementation of software.

In order for the users to be able to interact with TROI search engine, they firstly need to be registered, so the system will be able to know more about the user's background. In the work flow depicted in Figure 12 is described the procedure how the user creates a profile in TROI search engine.

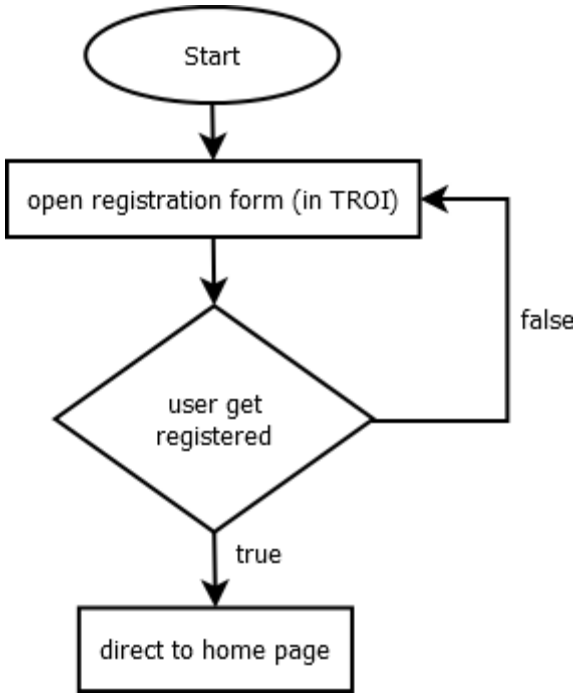

Figure 12 : TROI search engine user create profile work flow

First open TROI system, then user need to click on login menu, fill the form, if something went wrong while filling the form user will be informed by system, if the provided data are in correct, user will be redirected to login page with a list of recommendations.

5.3.2 Generate recommendation's workflow

Making easier for users to find the appropriate job that suits their profile TROI search engine generates the recommendations for them. And the workflow how our system recommends for jobs is displayed on Figure 13.

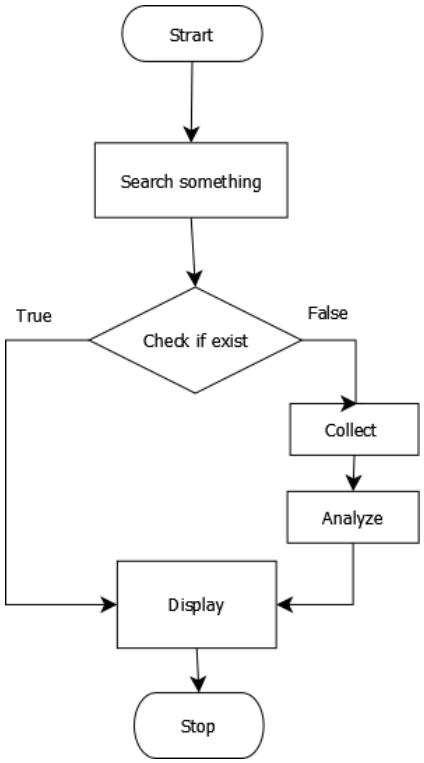

Figure 13: Display recommendations work flow.

TROI search engine read user's country, city, and job category. Based on those three fields build query to search in the collected multi documents. These documents will be indexed and analyzed, and then after display recommendations.

5.4 TROI Search Engine User Interface

In user interface we used bootstrap framework. The reason why we chose bootstrap is because: Is powerful front-end framework for faster and easier web development. It includes HTML and CSS based design templates for common user interface components like Typography, Forms, Buttons, Tables, Navigations, Dropdowns, Alerts, Modals, Tabs, Accordion, Carousel and many other as well as optional JavaScript extensions. Bootstrap also gives your ability to create responsive layout with much less efforts. Advantages of Bootstrap the biggest advantage of using Bootstrap is that it comes with free set of tools for creating flexible and responsive web layouts as well as common interface components. Additionally, using the Bootstrap data APIs you can create advanced interface components like Scrollspy and Typeaheads without writing a single line of JavaScript. Here are some more advantages, why one should opt for Bootstrap: Save lots of time — you can save lots of time and efforts using the Bootstrap predefined design templates and classes and concentrate on other development work.

The Figure 14, shows a piece of code using bootstrap framework.

Figure 14: Bootstrap usage in TROI search engine.

The key characteristics of using bootstrap framework are as follows [24]:

Responsive features — Using Bootstrap you can easily create responsive designs. Bootstrap responsive features make your web pages to appear more appropriately on different devices and screen resolutions without any change in markup.

Consistent design — All Bootstrap components share the same design templates and styles through a central library, so that the designs and layouts of your web pages are consistent throughout your development.

Easy to use — Bootstrap is very easy to use. Anybody with the basic working knowledge of HTML and CSS can start development with Bootstrap.

Compatible with browsers — Bootstrap is created with modern browsers in mind and it is compatible with all modern browsers such as Mozilla Firefox, Google Chrome, Safari, Internet Explorer, and Opera.

Open Source — and the best part is, it is completely free to download and use [24].

6. The technology used to build TROI Search Engine

Commonly, there are plenty of algorithms used in order to increase the efficacy of recommended items. In the following section, there are listed a short number of algorithms used. In this chapter we will elaborate algorithms of Apache Software Foundation, which can used to build a recommender system.
Here we will be focused to elaborate these three open source libraries which are under Apache Software Foundation:

- Apache Lucene
- Apache Mahout
- Apache OpenNLP

6.1 Apache Lucene

Apache Lucene™ [25, 26, 27] is a high-performance, full-featured written entirely in Java. It is a technology suitable for nearly any application that requires full-text search, especially cross-platform. Lucene is a library that allows the user to index textual data (Word & PDF documents, emails, webpages, tweets). It allows you to add search capabilities to your application.

There are two main steps that Lucene performs:

- Create an index of documents you want to search.
- Parse query, search index, return results.

Indexing -Lucene uses an inverted index (mapping of a term to its metadata). This metadata includes information about which files contain this term, number of occurrences to name a few. The fundamental units of indexing in Lucene are the Document and Field classes:

A Document is a container that contains one or more Fields. A Field stores the terms we want to index and search on. It stores a mapping of a key (name of the field) and a value (value of the field that we find in the content).

Lucene Architecture is shown in Figure 15:

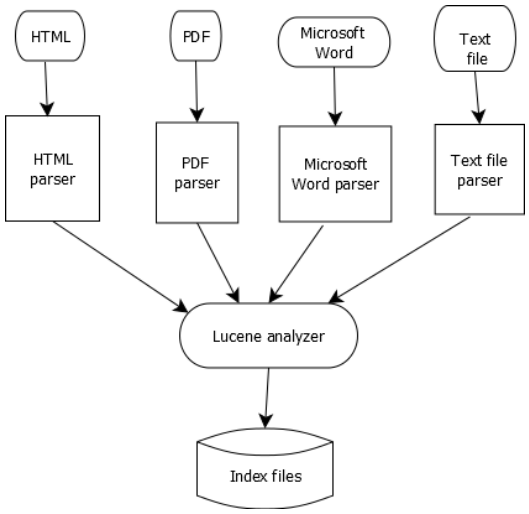

Figure 15: Overview of Lucene Architecture

In Figure 15 apache Lucene accept documents as html, pdf, Microsoft word, and text file. These job descriptions will be caught by their parser which are: html parser, pdf parser, Microsoft word parser, text file parser. The parser will index files and will assigned a unique id for each file. Analyzer which filter this document's, clean and get only necessary data. For indexing files Lucene uses an inverted index data structure for storing the Fields we want to search on.

Here is a short description for Lucene Architecture through three steps:

- Extracting Text: In order to be able to index documents of various types, Lucene needs to be able to extract the test from the given document into a format that it can parse.
- Analyze: This process filters and cleans up the text data. The text data goes through several steps (for example: extracting words, removing common (stop) words, make words lowercase, to name a few) and converts the text into tokens that can be added to the index. The picture to the right shows the indexing process which results in an

inverted index being stored on the underlying filesystem. See below for an example of an inverted index.

- Write Index: Lucene uses an inverted index data structure for storing the Fields we want to search on. An inverted index uses the tokens as the lookup key to find the documents which contains that token. It maps the content to its location. The index can be physically stored as part of a Directory (either in a file system or in memory). Below is an example of an inverted index. Logically, this represents the result of the indexing process.

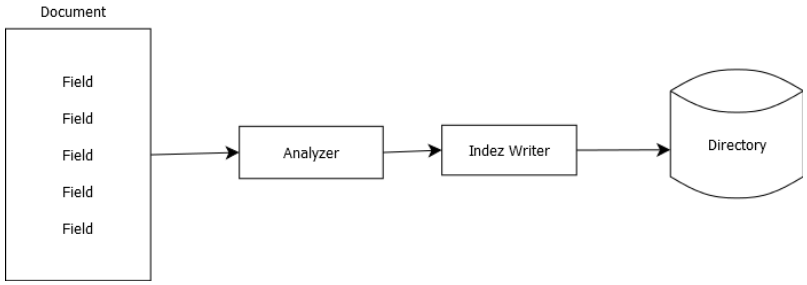

Figure 16: Indexing process

In Figure 16 are the steps which are done in indexing process. A document which is parse to analyzer. Analyzer create indexes for that document and saved those to a directory.

Following diagram illustrates the indexing process and use of classes. Index Writer is the most important and core component of the indexing process. We add Document(s) containing Field(s) to Index Writer which analyzes the Document(s) using the Analyzer and then creates/open/edit indexes as required and store/update them in a Directory. Index Writer is used to update or create indexes. It is not used to read indexes.

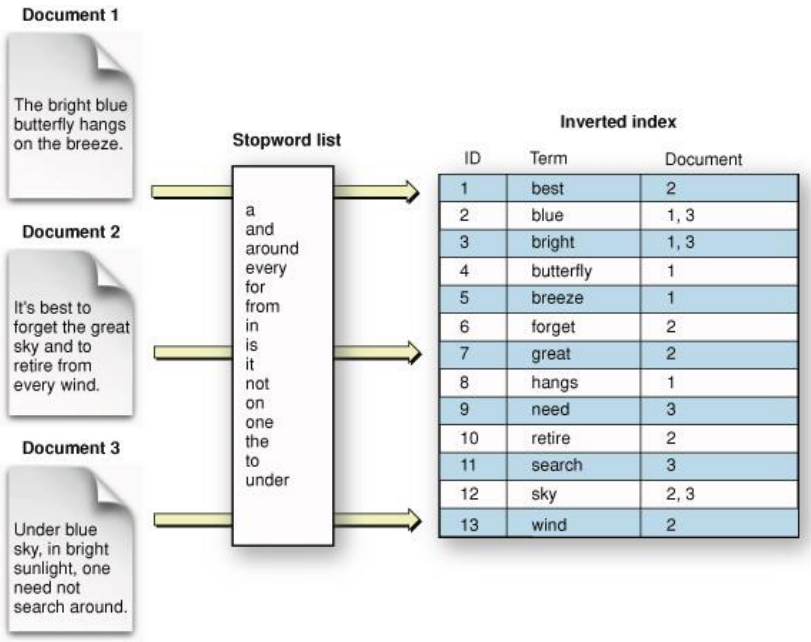

Figure 17: Inverted index process

Figure 17 is inverted index process in which Lucene instead of searching the text directly, it searches an index instead.

Lucene is able to achieve fast search responses because, instead of searching the text directly, it searches an index instead. This would be the equivalent of retrieving pages in a book related to a keyword by searching the index at the back of a book, as opposed to searching the words in each page of the book.

This type of index is called an inverted index, because it inverts a page-centric data structure (page->words) to a keyword-centric data structure (word->pages). The index stores statistics about terms in order to make term-based search more efficient [25, 26, 27].

6.1.1 Searching

Once our documents are indexed, we will need to add search functionality. All queries of the index are done through the Index Searcher. Given a search expression, we parse the query, create a Query Parser and search the index for results. The results are returned as Top Docs which contain Score Docs, which contain the document IDs and the confidence scores of the results that match the query. The fundamental classes for searching are:

- **Index Searcher** - Provides "read-only" access to the index. Exposes several search methods that take in a Query object and return the top n "best"
- **Top Docs** as the result. This class is the counter part to the Index Writer class used for creating/updating indexes.
- **Term** - Basic unit for searching. Counterpart to the Field object used in indexing. We create a certain Field when indexing (for ex: "Name": "Chuck Norris") and we use Terms in a Term Query when searching. It contains the same mapping from the name of the field to the value
- **Query**: Lucene provides several types of Queries, including Term Query, Boolean Query, Prefix Query, Wildcard Query, Phrase Query, and Fuzzy Query. Each type of query provides a unique way of searching the index.
- **Query Parser**: Parses a human-readable query (for ex: "opower AND arlington") into Query object that can be used for searching.
- **Top Docs - Container** for pointers to N search results. Each Top Doc contains a document ID and a confidence score.

Here is an overview of the process described above [25, 26, 27]:

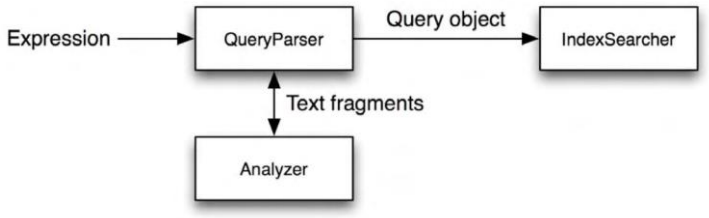

Figure 18: Searching process

In Figure 18 is searching process from begin. Expression which is passed to Query Parser →Analyzer and in the end, it is searched in the directory of indexes.

Lucene, a very popular open source search library from Apache, provides powerful indexing and searching capabilities for applications. It provides a simple and easy-to-use API that requires minimal understanding of the internals of indexing and searching. In this article, you learned about Lucene architecture and its core APIs.

Lucene has powered various search applications being used by many well-known Web sites and organizations. It has been ported to many other programming languages. Lucene has a large and active technical user community.

Applications and web applications using Lucene:

- JIRA - issue tracker,
- Hotels and Accommodation - A hotel & accommodation comparison engine. We use Lucene for product/price groupings and for our front end search.
- IBM
- IBM Omni Find Personal e-mail Search - A powerful semantic search engine for your e-mail
- IBM Omni Find Yahoo! Edition - a no-charge, entry-level enterprise search software solution
- AOL
- LinkedIn
- Hi5

To name a few.

Lucene, a very popular open source search library from Apache, provides powerful indexing and searching capabilities for applications. It provides a simple and easy-to-use API that requires minimal understanding of the internals of indexing and searching. Lucene has powered various search applications being used by many well-known Web sites and organizations. It has been ported to many other programming languages. Lucene has a large and active technical user community [25, 26, 27].

6.2 Apache Mahout

Apache Mahout is a new open source project by the Apache Software Foundation (ASF) with the primary goal of creating scalable machine-learning algorithms that are free to use under the Apache license. Machine learning is a discipline of artificial intelligence that enables systems to learn based on data alone, continuously improving performance as more data is processed. Machine learning is the basis for many technologies that are part of our everyday lives [28].

Some examples of applied machine learning algorithms include:

Recommendation engines: Numerous web sites today are able to make recommendations to users based on past behavior, and the behavior of others [28].

Spam filtering: Nearly every modern email provider is able to automatically detect the difference between a spam message and a legitimate one, only presenting the latter ones to the user. These filtering engines use machine-learning algorithms such as clustering and classification [28].

Natural Language Processing: Many of us have smartphones that understand what we mean when we ask "When are the niners playing next?" Making a computer understand this phrase is no simple task - it has to know that "niners" is slang for the San Francisco 49ers, which is an American football team, so it needs to consult with the National Football League's schedule to

provide the answer. All of this was made possible by applying machine-learning algorithms to vast sets of language data to make these connections [28].

Until recently, data scientists had to implement and customize machine-learning algorithms manually to the computing framework that they were using, resulting in a significant amount of work. Now, with Mahout, data scientists can write MapReduce jobs that reference a number of predefined algorithms to build these kinds of applications easily.

Below is a current list of machine learning algorithms exposed by Mahout.

- Collaborative Filtering
    - Item-based Collaborative Filtering
    - Matrix Factorization with Alternating Least Squares
    - Matrix Factorization with Alternating Least Squares on Implicit Feedback
- Classification
    - Naive Bayes
    - Complementary Naive Bayes
    - Random Forest
- Clustering
    - Canopy Clustering
    - k-Means Clustering
    - Fuzzy k-Means
    - Streaming k-Means
    - Spectral Clustering
- Dimensionality Reduction
    - Lanczos Algorithm
    - Stochastic SVD
    - Principal Component Analysis
- Topic Models
    - Latent Dirichlet Allocation
- Miscellaneous

- Frequent Pattern Matching
- RowSimilarityJob
- ConcatMatrices
- Colocations

These components and their implementations make it possible to build out complex recommendation systems for either real-time-based recommendations or offline recommendations. Real-time-based recommendations often can handle only a few thousand users, whereas offline recommendations can scale much higher. In many cases, this is a reasonable approach that allows you to meet the demands of a large system with a lot of users, items, and preferences.

Architecture of Mahout - Mahout currently provides tools for building a recommendation engine through the Taste library — a fast and flexible engine for CF (Collaborative filtering). Taste supports both user-based and item-based recommendations and comes with many choices for making recommendations, as well as interfaces for you to define your own. Taste consists of five primary components that work with Users, Items and Preferences:

- Data Model: Storage for Users, Items, and Preferences
- User Similarity: Interface defining the similarity between two users
- Item Similarity: Interface defining the similarity between two items
- Recommender: Interface for providing recommendations
- User Neighborhood: Interface for computing a neighborhood of similar users that can then be used by the Recommenders

These components and their implementations make it possible to build out complex recommendation systems for either real-time-based recommendations or offline recommendations. Real-time-based recommendations often can handle only a few thousand users, whereas offline recommendations can scale much higher. Taste even comes with tools for leveraging Hadoop to calculate recommendations offline. In many cases, this is a reasonable approach that allows you to meet the demands of a large system with a lot of users, items, and preferences [28].

Applications of Mahout - Companies such as Adobe, Facebook, LinkedIn, Foursquare, Twitter, and Yahoo use Mahout internally.

Foursquare helps you in finding out places, food, and entertainment available in a particular area. It uses the recommender engine of Mahout.

- Twitter uses Mahout for user interest modelling.
- Yahoo! uses Mahout for pattern mining.

Mahout has quietly undergone huge transformation from Map Reduce / Java based Machine Learning to Mathematically Expressive Scala / Engine Neutral / GPU Accelerated [28]

6.3 Apache OpenNLP

Apache OpenNLP [29] is an open source project that is cross platform and written in Java. It is a toolkit, for NLP (Natural Language Processing), based on machine learning. Natural Language Processing is all about the interaction between computer and human. Generally, humans interact with each other using vocabulary. And the language they are using (say English, Spanish, Hindi, etc.,) has some set of rules. It does not happen all the time that all people speaking these languages to communicate use the grammar of the language alike. Different people might use different words for conveying the same information. But as people around them have already known them or used to such kind, can understand or get the summary or inference from what they are saying. Humans perceive information like context, inference etc., from the sentences formed using vocabulary and grammar. And when a machine or computer is expected to understand the context, inference or summary or useful information from the data it gets from a human, there are some gaps that needs to be filled. These gaps are the tasks that Natural Language Processing deals with, to make a machine understand a human language or speak to human in natural language [29].

Apache OpenNLP is an open-source library that provides solutions to some of the Natural Language Processing tasks through its APIs and command line tools. Apache OpenNLP uses machine learning approach for the tasks of processing natural language. It also provides some of the pre-built models for some of the tasks. Following are the tasks to which Apache OpenNLP provides APIs, and those we deal with examples in this OpenNLP Tutorial.

Apache OpenNLP Tutorial – APIs:

Named Entity Recognition is to find named entities like person, place, organization or a thing in a given sentence. OpenNLP has built models for NER which can be directly used and also helps in training a model for the custom data we have. Named Entity Recognition Example with existing model

Document Categorizer Categorizing or Classifying: A given document to one of the pre-defined categories is what a Document Categorizer does. OpenNLP provides an API that helps in categorizing or classifying documents. As categorizing documents cannot be generalized like NER, there are no pre-built models available, but anyone can build a model by his/her own requirements. Document classification using Maximum Entropy (Maxent) Document classification using Naive Bayes.

Sentence Detection: The process of identifying sentences in a paragraph or a document or a text file is called Sentence Detection. OpenNLP supports Sentence Detection through its API. It provide pre-built models for sentence detection, and also a means to build a model for requirement specific data.

Parts of Speech Tagging: Understanding grammar is an important task in NLP. Identifying Parts of Speech in a given sentence is a stepping block to understand grammar. Apache OpenNLP provides APIs to train a model that can identify Parts of Speech or use a pre-built model and identify Parts of Speech in a sentence.

Table 5: POS Tagger Example in Apache OpenNLP marks each word in a sentence with the word type.

Input to POS Tagger	John is 27 years old.
Output of POS Tagger	John_NNP is_VBZ 27_CD years_NNS old_JJ ._.

The word types are the tags attached to each word. These Parts Of Speech tags used are from Penn Treebank.

Table 6: In this table is just a short list of complete list of Parts Of Speech tags from Penn Treebank.

NNP	**Proper Noun, Singular**
VPZ	Verb, 2rd person singular present
CD	Cardinal Number
NNS	Noun, Plural
JJ	Adjective
……	

Tokenization : Tokenization is a process of breaking down the given sentence into smaller pieces like words, punctuation marks, numbers to name a few. Apache OpenNLP provides APIs to train a model or use a pre-built model and break a sentence into smaller pieces. Tokenization is a process of segmenting strings into smaller parts called tokens (say sub-strings). These tokens are usually words, punctuation marks, sequence of digits, and like. An example is shown in the following table:

Table 7: Example of tokenization process.

Input to Tokenizer	**John is 26 years old.**					
Output of Tokenizer	John	Is	26	Years	Old	.

Lemmatization: Lemmatization is a process of removing any changes in form of the word like tense, gender, mood, etc. and return dictionary or base form of the word. Lemmatizer is a Natural Language Processing tool that aims to remove any changes in form of the word like tense, gender, mood, etc. and return dictionary or base form of word. In Apache OpenNLP, Lemmatizer returns base or dictionary form of the word (usually called lemma) when it is provided with word and its Parts-Of-Speech tag. For a given word, there could exist many lemmas, but given the Parts-Of-Speech tag also, the number could be narrowed down to almost one, and the one is the more accurate as the context to the word is provided in the form of postag.

Language Detection: Language Detection is a task of finding the natural language to which the sample text provided belongs to.

Small overview of OpenNLP and the APIs it provides with NLP which is used to analyze text, allowing machines to understand how human's speak. This human-computer interaction enables real-world applications [29].

6.4 Key features of algorithms

Recommender Systems often need to calculate large number of data and products useful information in high quality time efficient way. With the use of various algorithms the Recommender Systems has the ability to predict whether a particular user would prefer an item or not, based on the user's profile and their activities. The use of Recommender Systems are beneficial to both service providers and users. Recommended System technology is focused on algorithm's that generate recommendations for particular user.

To build recommendation system we can use apache Lucene to search and index documents, and apache mahout algorithm's to process and generate recommendations or apache openNLP algorithms to process interaction based on human computer interactions.

Below are described the key features of this algorithm's:

Figure 19: Key features of this algorithm's: Apache Lucene, Apache Mahout and Apache OpenNlp.

In Figure 19 is a presentation of key functionality that these algorithm provide.

7. Evaluation of TROI search engine

Evaluation as a final phase for software development, has been applied all the time, especially when you have to do with complex software systems. In this respect, the software can be evaluated from different aspects, for example, functionality, reliability, usability, efficiency, maintainability, portability, to name a few [30].

The characteristics of software quality can be also classified on the basis of the end users. Software quality can be classified as below [31]:

- User Perspectives
    1. Reliability
    2. Usability
    3. Functionality
- Developer Perceptive
    1. Maintainability
    2. Efficiency
    3. Portability
- Project Manager Perceptive
    1. Cycle Time
    2. Cost Schedule Pressure

In our case, the TROI is evaluated based on user experience (UX). The author used a survey to handle user's reliability, usability, and reflection of functionality while experiencing the use of TROI search engine. For more details questionnaire is on appendix chapter.

7.1 User and usability evaluation of TROI Search engine

To handle user's reliability, usability, and reflection of functionality while experiencing the use of TROI search engine below are set of questions that has been provided to the users.

PART A

	Question	**Level of response**
Question 1	Did you enjoy your experience with the system?	1 2 3 4 5
Question 2	Is the system easy to use?	1 2 3 4 5
Question 3	Were you able to control the system?	1 2 3 4 5
Question 4	Is the information provided by the system clear enough?	1 2 3 4 5
Question 5	Do you think that this system will be helpful for finding your appropriate job?	1 2 3 4 5
Question 6	Do recommendations were in line with what you were searching for?	1 2 3 4 5

The level of response is divided in to five steps, to handle different user satisfaction. The sample for evaluating our system encounters in total is 30 (30 participants evaluated our system). The age of the participants is within the range of 23-45. Furthermore, the participants were 50% female and the rest males.

A summarizing information regarding the participants:

Total number of participants	**30**
Age (years)	From 23 to 45
Sex	
○ **Male**	○ 15

○ **Female**	○ 15
Level of studies:	
○ **No studies** ○ **Primary studies** ○ **Secondary studies** ○ **Higher studies**	○ 3 ○ 3 ○ 14 ○ 10

8. Result discussion

So, the results of the users that evaluated the system are as follows. 33% of participants enjoyed the experience with the TROI search engine, 33% were somewhat satisfied. 17% were extremely satisfied and only 10% said that this concept of seeking jobs only from one platform may not offer a good experience to the users.

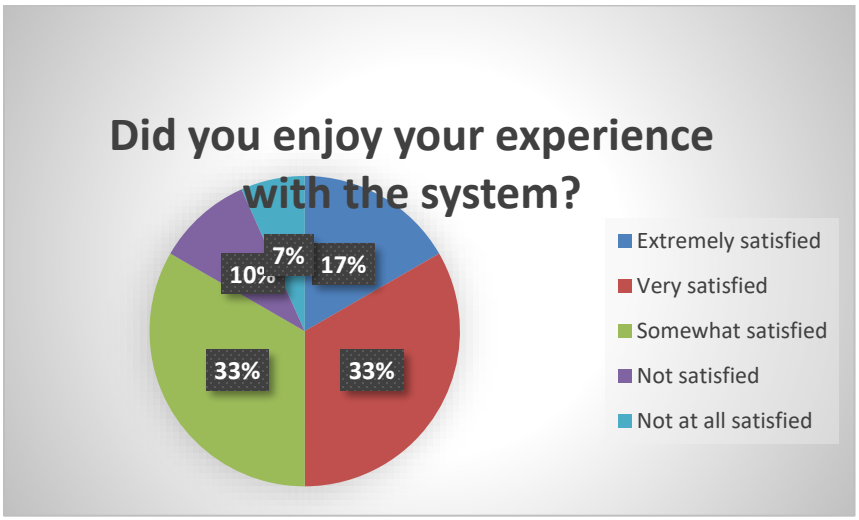

Figure 20 : the user satisfaction based on their experience with the TROI

When the users where asked whether the TROI search engine is easy to use or not, they respond are categorized as follows: 83% of participants where very satisfied, 10 of them where somewhat satisfied, 7% where not at all satisfied.

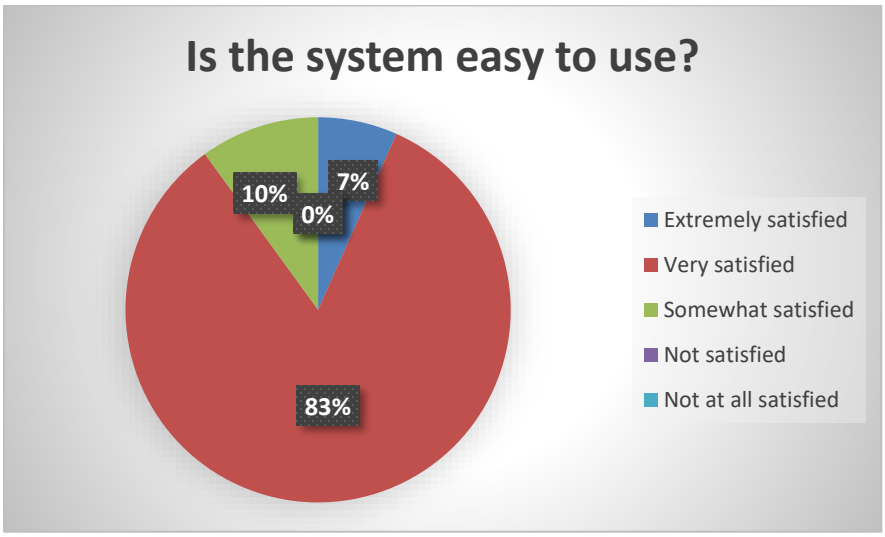

Figure 21: Question 2, whether the system was easy to use

Further, in question 3, when the participants were asked if they are able to control the system, based on what they are searching? 67% of participants were somewhat satisfied, 17% of them where very satisfied, 13% where not satisfied and 3% of them where not at all satisfied.

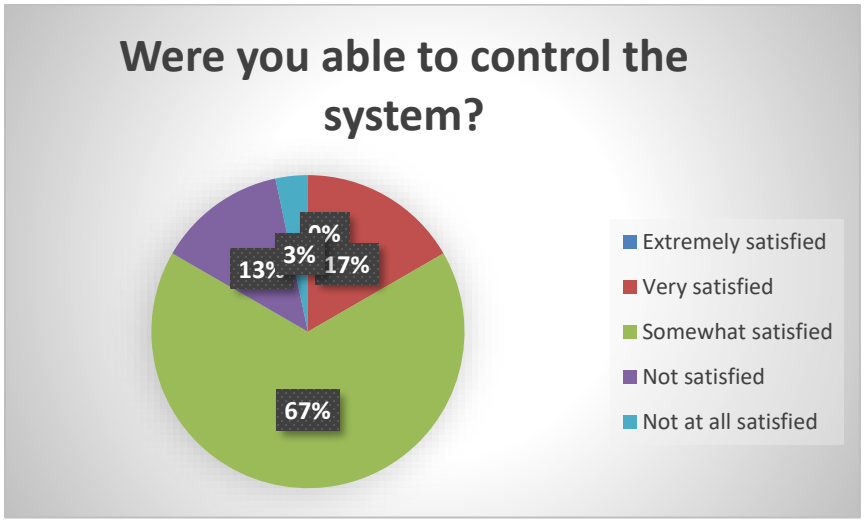

Figure 22 : Question 3, whether the system is easy to be control in the respect what they are searching for

In question 4, when the participants were asked whether they find clear the information within the TROI search engine, 67% of participants were very satisfied, 33% of them where somewhat satisfied.

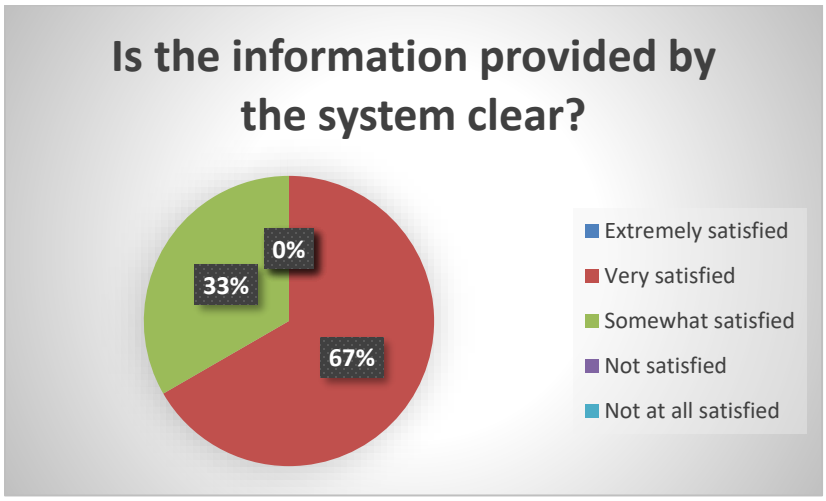

Figure 23 : Question 4, whether the information provided by TROI are clear

In question 5, when the participants were asked whether the TROI search engine was helpful for finding jobs? As shown in Figure 8.5, 48% of participants were very satisfied, 39% of them where somewhat satisfied, 10% of them where not satisfied, 3% extremely satisfied. So, in this case the users concluded that they need to use TROI more often in order to be able to evaluate whether it was helpful or not.

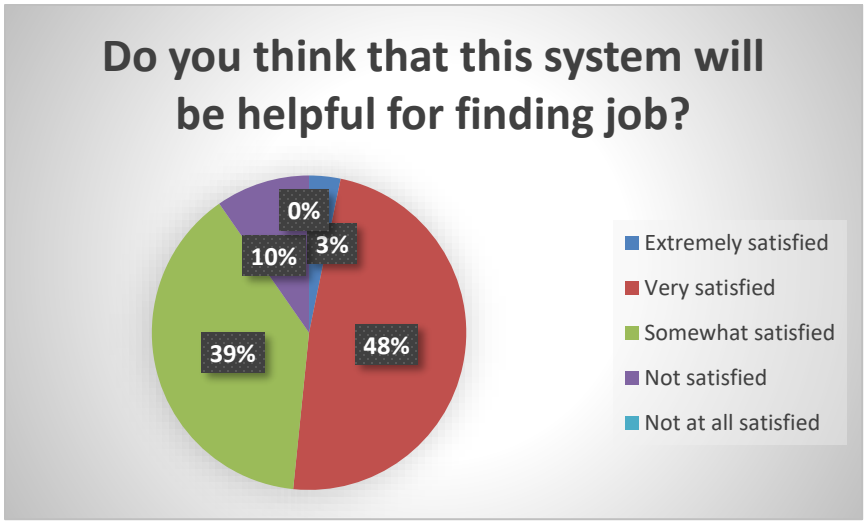

Figure 24: Question 5, whether TROI will be helpful for facilitating the job seeking process

In question 6, when the participants were asked whether the recommendations were in line with what you were searching for? As shown n Figure 8.6, 54 % of participants were for very satisfied, 43% were somewhat satisfied, 3% of participants where not satisfied.

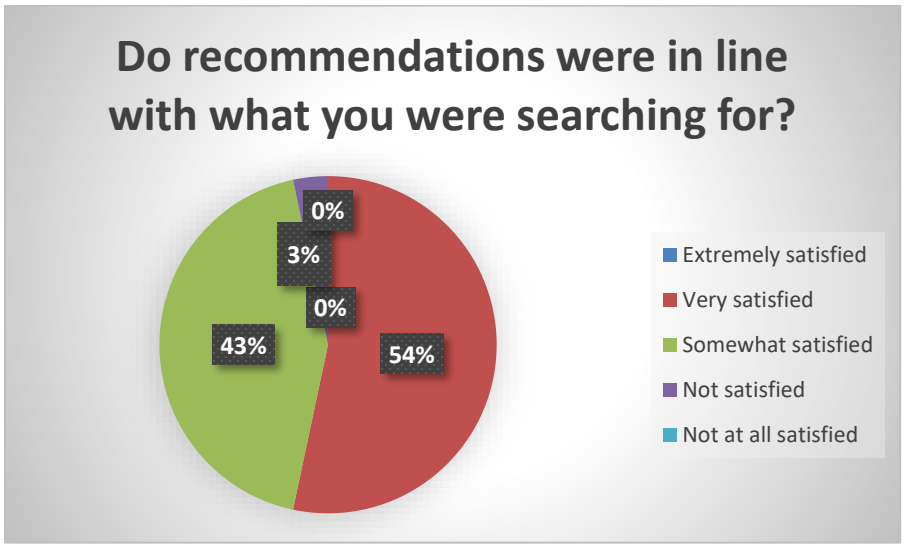

Figure 25: Question 6, whether TROI recommended jobs appropriately

Conclusion

Nowadays recommended systems are playing a big role for both service providers and users. From the service provider's perspective: it increases the user satisfaction, business revenues and number of users that hit their websites. From the user perspective, the system will increase the motivation while searching for jobs, because TROI suggestions are appropriate with their profile using these factors: previous click, preferences, likes, time spent in a specific job In order to predict the appropriate data for the users, a number of steps needed to be followed, starting from data retrieval until to data representation. In order to have a better system respond time, the generalization of query have been realized through ontology approaches. The reason ontologies are becoming so popular is in large part due to what they promise: a shared and common understanding of some domain that can be communicated between people and application systems. By knowing the role of recommended systems and ontology we end up to build TROI search engine.

TROI is offered to the users as a search engine which provide help about seeking jobs in Kosovo region. So, in this way we know which job positions is under which category, and we have structured data, while we need to generalize the query.

There are a lot of software foundation communities that have built a lot of tools, algorithms for search and recommend data. I searched under apache foundation algorithms to find algorithm's which will help in information retrieval step and building recommendation through various step. Since in Kosovo the level of unemployment is very high, we are trying to keep informed people for open job positions. The role that this technology on TROI search engine is offering currently is a solutions for open jobs positions only in Kosovo region, but TROI could accessed also from the broader market.

Bibliography

1. Advances in Web-Age Information Management 6th International Conference, WAIM 2005, Hangzhou, China, October 11 – 13, 2005.
2. Recommendation systems: Principles, methods and evaluation F.O.Isinkayea Y.O.Folajimib B.A.Ojokohc
3. Introduction to Recommender Systems Handbook (Francesco RicciEmail authorLior RokachBracha Shapira)
4. Koren, Y., Bell, R.M., Volinsky, C.: Matrix factorization techniques for recommender systems. IEEE Computer (2009)
5. Montaner, M., Lopez, B., de la Rosa, J.L.: A taxonomy of recommender agents on the internet. Artificial Intelligence Review 19(4), 285–330 (2003)
6. Beginners Guide to learn about Content Based Recommender Engines [Shuvayan Das] [https://www.analyticsvidhya.com/blog/2015/08/beginners-guide-learn-content-based-recommender-systems/]
7. From Vector Space Models to Recommender Systems Jon Krakauer [http://trigonaminima.github.io/recommender-systems/tfidf/nlp/2016/11/02/From-Vector-Space-Models-to-Recommender-Systems/]
8. Cosine similarity https://masongallo.github.io/machine/learning,/python/2016/07/29/cosine-similarity.html
9. Euclidean Distance https://www.edureka.co/blog/understanding-euclidean-distance-and-cosine-similarities-in-mahout/
10. Euclidean Distance http://www.inf.ed.ac.uk/teaching/courses/inf2b/learnnotes/inf2b-learn-note02-2up.pdf
11. Collaborative Filtering http://recommender-systems.org/collaborative-filtering/
12. Hybrid Recommender Systems: Survey and Experiments Robin Burke California State University, Fullerton Department of Information Systems and Decision Sciences

13. Diploma Thesis at Department for Agent Technologies and Telecommunications Prof. Dr.-Ing. habil. Sahin Albayrak Faculty IV - Electrical Engineering and Computer Science Technical University Berlin - A Hybrid Approach to Recommender Systems based on Matrix Factorization presented by Stephan Spiegel
14. Methods and applications for ontology-based recommender systems [Doctoral Dissertation][Tuukka Ruotsalo]
15. A Short History of Ontology: It's not just a Matter of Philosophy Anymore [Charles Roe] [http://www.dataversity.net/a-short-history-of-ontology-its-not-just-a-matter-of-philosophy-anymore/]Ide, Nancy, and David Woolner. "Historical ontologies." *Words and Intelligence II* (2007): 137-152.
16. Maja HadzicPornpit, WongthongthamTharam, DillonElizabeth Chang: Ontology-Based Multi-Agent Systems
17. What Are Ontologies, and Why Do We Need Them? [B. Chandrasekaran and John R. Josephson, Ohio State University V. Richard Benjamins, University of Amsterdam].
18. A Recommender System for Job Seeking and Recruiting Website [Yao Lu React Group, EPFLLausanne, Switzerland] [Sandy El Helou React Group, EPFL Lausanne, Switzerland] [Denis Gillet React Group, EPFL Lausanne, Switzerland]
19. A survey of job recommender systems [Shaha T. Al-Otaibi1] and [Mourad Ykhlef2].
20. An Introduction to Agile Software Development http://www.serena.com/docs/repository/solutions/intro-to-agile-devel.pdf
21. International Journal of Advanced Research in Computer Engineering & Technology Volume 1, Issue 1, March 2012 Software Reliability Evaluation: A Survey Based Feroza Haque , Dr Sanjeev Bansal .
22. Pireva, Krenare, and Petros Kefalas. "A Recommender System Based on Hierarchical Clustering for Cloud e-Learning." International Symposium on Intelligent and Distributed Computing. Springer, Cham, 2017.
23. Introduction to use-case-diagram [http://whatis.techtarget.com/definition/use-case-diagram]

24. Bootstrap Introduction https://www.tutorialrepublic.com/twitter-bootstrap-tutorial/bootstrap-introduction.php
25. Lucene - Indexing Process https://www.tutorialspoint.com/lucene/lucene_indexing_process.htm
26. Apache Lucene - Index File Formats [https://lucene.apache.org/core/3_0_3/fileformats.html]
27. Introduction to Apache Lucene was published on December 08, 2013 (revised: 12/08/2013) By Karthik Kumar.
28. Introducing Apache Mahout - Scalable, commercial-friendly machine learning for building intelligent applications [https://www.ibm.com/developerworks/library/j-mahout/].
29. Apache OpenNLP Tutorial https://www.tutorialkart.com/opennlp/apache-opennlp-tutorial/
30. Evaluation of Software Systems, University of Ulster Newtownabbey, BT 37 0QB, N.Ireland
. [Günther Gediga][Kai-Christoph Hamborg]
31. International Journal of Advanced Research in Computer Engineering & Technology Volume 1, Issue 1, March 2012 Software Reliability Evaluation: A Survey Based Feroza Haque , Dr Sanjeev Bansal .
32. Eclipse Tutorial https://www.tutorialspoint.com/eclipse/eclipse_tutorial.pdf
33. Introduction to pom.xml https://maven.apache.org/guides/introduction/introduction-to-the-pom.html
34. Introduction to apache maven https://maven.apache.org/what-is-maven.html
35. Apache Maven https://www.ibm.com/developerworks/library/j-5things13/index.html
36. Spring Web MVC framework https://docs.spring.io/spring/docs/3.0.x/spring-framework-reference/html/mvc.html
37. Java Server-Side Programming https://www.ntu.edu.sg/home/ehchua/programming/java/JavaServerPages.html

38. Introduction to JavaServer Pages https://www.ibm.com/developerworks/java/tutorials/j-introjsp/j-introjsp.html
39. Introduction to the Oracle Database https://docs.oracle.com/cd/B19306_01/server.102/b14220/intro.htm
40. Hibernate java persistence framework tutorialspoint.com
41. An Introduction to Oracle SQL Developer Data Modeler http://www.oracle.com/technetwork/developer-tools/datamodeler/sqldeveloperdatamodeleroverview-167687.html

Appendix

TROI Search Engine – Eclipse IDE

TROI is developed in Eclipse IDE. Eclipse is an integrated development environment (IDE) for developing applications using the Java programming language and other programming languages such as C/C++, Python, PERL, Ruby to name a few. The Eclipse platform which provides the foundation for the Eclipse IDE is composed of plug-ins and is designed to be extensible using additional plug-ins. Developed using Java, the Eclipse platform can be used to develop rich client applications, integrated development environments, and other tools. Eclipse can be used as an IDE for any programming language for which a plug-in is available. The Java Development Tools (JDT) project provides a plug-in that allows Eclipse to be used as a Java IDE, PyDev is a plugin that allows Eclipse to be used as a Python IDE, C/C++ Development Tools (CDT) is a plug-in that allows Eclipse to be used for developing application using C/C++, the Eclipse Scala plug-in allows Eclipse to be used an IDE to develop Scala applications and PHP Eclipse is a plug-in to eclipse that provides complete development tool for PHP.

You can download eclipse from http://www.eclipse.org/downloads/. The download page lists a number of flavors of eclipse.

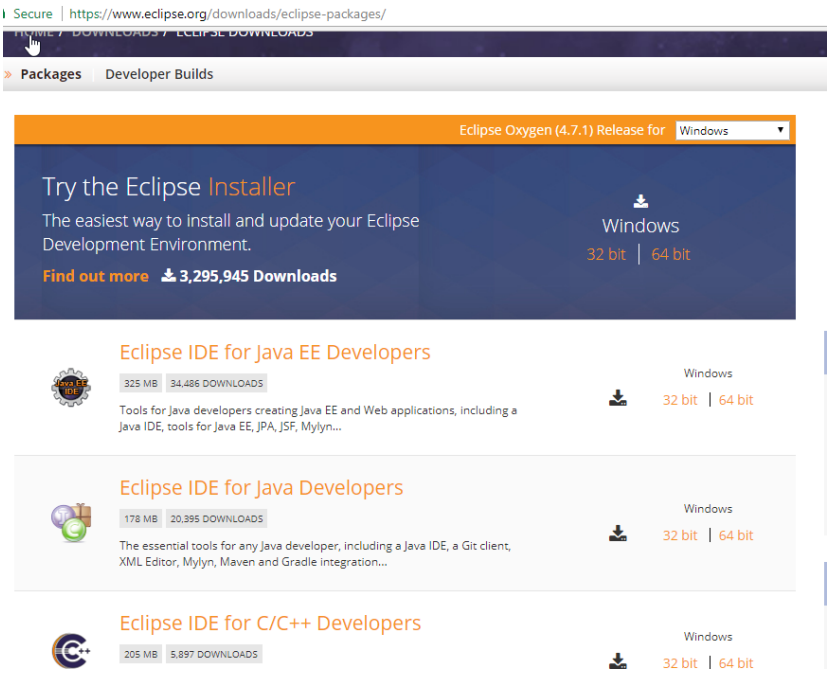

Figure 26: Eclipse download page

The capabilities of each packaging of eclipse are different. Java developers typically use Eclipse Classic or Eclipse IDE for developing Java applications. The drop down box in the right corner of the download page allows you to set the operating system on which eclipse is to be installed. You can choose between Windows, Linux and Mac. Eclipse is packaged as a zip file [32].

TROI Search Engine - Apache Maven Project

Maven, a Yiddish word meaning accumulator of knowledge, was originally started as an attempt to simplify the build processes in the Jakarta Turbine project. There were several projects each with their own Ant build files that were all slightly different and JARs were checked into CVS. We wanted a standard way to build the projects, a clear definition of what the project consisted of, an easy way to publish project information and a way to share JARs across several projects.

Maven is an excellent build tool for Java™ developers, and you can use it to manage the life cycle of your projects as well. As a life-cycle management tool, Maven operates against phases rather than Ant-style build "tasks." Maven handles all phases of the project life cycle, including validation, code generation, compilation, testing, packaging, integration testing, verification, installation, deployment, and project site creation and deployment.

Eclipse provides an excellent plugin m2eclipse which seamlessly integrates Maven and Eclipse together. Some of features of m2eclipse are listed below −

- You can run Maven goals from Eclipse.
- You can view the output of Maven commands inside the Eclipse, using its own console.
- You can update maven dependencies with IDE.
- You can Launch Maven builds from within Eclipse.
- It does the dependency management for Eclipse build path based on Maven's pom.xml.
- It resolves Maven dependencies from the Eclipse workspace without installing to local Maven repository (requires dependency project be in same workspace).
- It automatic downloads the required dependencies and sources from the remote Maven repositories.
- It provides wizards for creating new Maven projects, pom.xml and to enable Maven support on existing projects
- It provides quick search for dependencies in remote Maven repositories. [33, 34, 35]

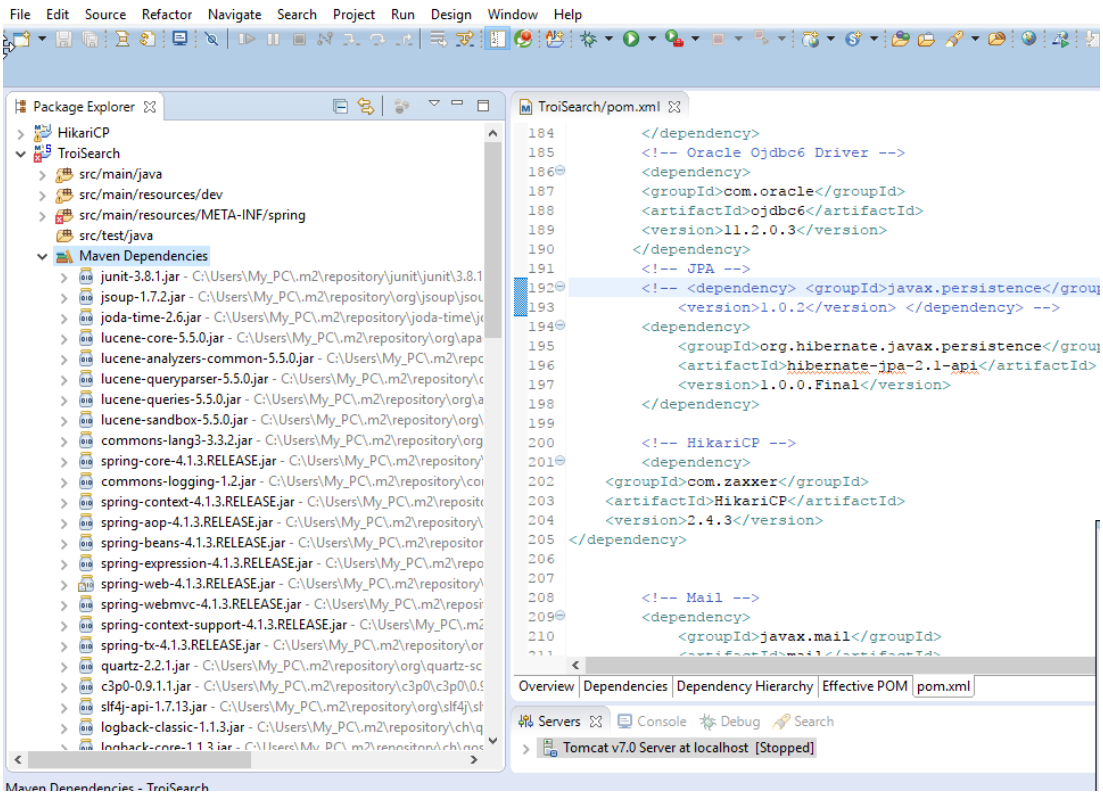

Figure 27: Eclipse TROI search engine in Maven Project.

TROI Search Engine Framework's

Spring MVC - The Spring Web model-view-controller (MVC) framework is designed around a DispatcherServlet that dispatches requests to handlers, with configurable handler mappings, view resolution, locale and theme resolution as well as support for uploading files. The default handler is based on the @Controller and @RequestMapping annotations, offering a wide range of flexible handling methods. With the introduction of Spring 3.0, the @Controller mechanism also allows you to create RESTful Web sites and applications, through the @PathVariable annotation and other features.

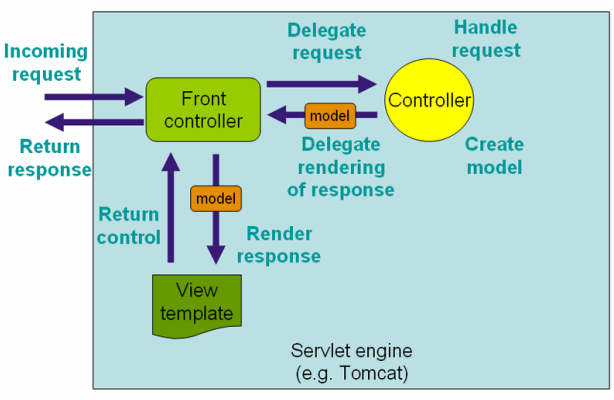

Figure 28: Dispatcher Servlet

Spring's web module includes many unique web support features:

- Clear separation of roles. Each role -- controller, validator, command object, form object, model object, DispatcherServlet, handler mapping, view resolver, and so on -- can be fulfilled by a specialized object.
- Powerful and straightforward configuration of both framework and application classes as JavaBeans. This configuration capability includes easy referencing across contexts, such as from web controllers to business objects and validators.
- Adaptability, non-intrusiveness, and flexibility. Define any controller method signature you need, possibly using one of the parameter annotations (such as @RequestParam, @RequestHeader, @PathVariable, and more) for a given scenario.
- Reusable business code, no need for duplication. Use existing business objects as command or form objects instead of mirroring them to extend a particular framework base class.
- Customizable binding and validation. Type mismatches as application-level validation errors that keep the offending value, localized date and number binding, and so on instead of String-only form objects with manual parsing and conversion to business objects.
- Customizable handler mapping and view resolution. Handler mapping and view resolution strategies range from simple URL-based configuration, to sophisticated,

purpose-built resolution strategies. Spring is more flexible than web MVC frameworks that mandate a particular technique.

- Flexible model transfer. Model transfer with a name/value Map supports easy integration with any view technology.
- Customizable locale and theme resolution, support for JSPs with or without spring tag library, support for JSTL, support for Velocity without the need for extra bridges, and so on.
- A simple yet powerful JSP tag library known as the spring tag library that provides support for features such as data binding and themes. The custom tags allow for maximum flexibility in terms of markup code. For information on the tag library descriptor, see the appendix entitled Appendix F, spring.tld
- A JSP form tag library, introduced in Spring 2.0, that makes writing forms in JSP pages much easier. For information on the tag library descriptor, see the appendix entitled Appendix G, spring-form.tld
- Beans whose lifecycle is scoped to the current HTTP request or HTTP Session. This is not a specific feature of Spring MVC itself, but rather of the WebApplicationContext container(s) that Spring MVC uses. These bean scopes are described in Section 3.5.4, “Request, session, and global session scopes”

So spring is a powerful Java application framework, used in a wide range of Java applications [36].

JAVA Server Pages (JSP) technology

JSP is one of the most powerful, easy-to-use, and fundamental tools in a Web-site developer's toolbox. JSP combines HTML and XML with Java TM servlet (server application extension) and JavaBeans technologies to create a highly productive environment for developing and deploying reliable, interactive, high-performance platform-independent Web sites.

JSP facilitates the creation of dynamic content on the server. It is part of the Java platform's integrated solution for server-side programming, which provides a portable alternative to other server-side technologies, such as CGI. JSP integrates numerous Java application technologies, such as Java servlet, JavaBeans, JDBC, and Enterprise JavaBeans. It also separates information presentation from application logic and fosters a reusable-component model of programming.

What exactly is JSP? Let's consider the answer to that question from two perspectives: that of an HTML designer and that of a Java programmer. If you are a Java programmer, you can look at JSP as a new higher-level way to write servlets. Instead of directly writing servlet classes and emitting HTML from your servlets, you write HTML pages with Java code embedded in them. The JSP environment takes your page and dynamically compiles it. Whenever a user agent requests that page from the Web server, the servlet generated from your JSP code is executed, and the results are returned to the user. A simple dynamic JSP in TROI Search:

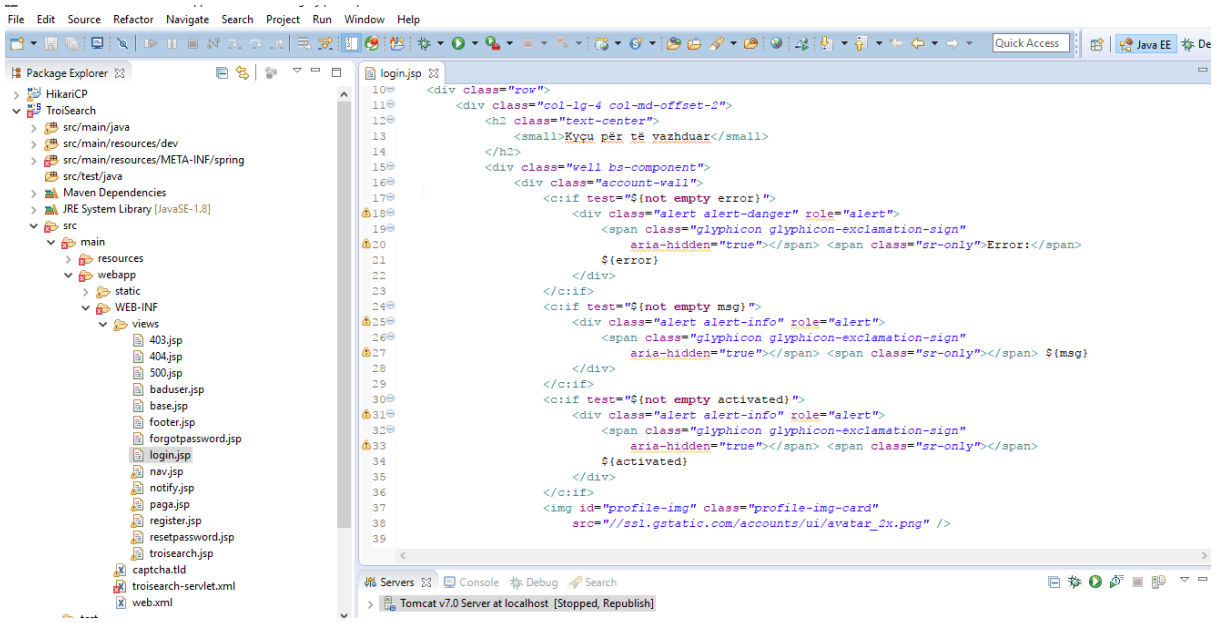

Figure 29: JSP Files in TROI search engine implementation

Advantages of JSP

- Separation of static and dynamic contents: The dynamic contents are generated via programming logic and inserted into the static template. This greatly simplifies the creation and maintenance of web contents.
- Reuse of components and tag libraries: The dynamic contents can be provided by re-usable components such as JavaBean, Enterprise JavaBean (EJB) and tag libraries - you do not have to re-inventing the wheels.
- Java's power and portability.

JSP is based on Servlet. In fact, we shall see later that a JSP page is internally translated into a Java servlet. We shall also explain later that "Servlet is HTML inside Java", while "JSP is Java inside HTML".

Apache Tomcat Server

JSPs, like servlets, are server-side programs run inside a Java-capable HTTP server. Apache Tomcat Server (@ http://tomcat.apache.org) is the official reference implementation (RI) for Java servlet and JSP, provided free by Apache (@ http://www.apache.org) - an open-source software foundation.

You need to install Tomcat to try out JSP [37, 38].

Database – Hibernate Framework

Oracle Database -

An Oracle database is a collection of data treated as a unit. The purpose of a database is to store and retrieve related information. A database server is the key to solving the problems of information management. In general, a server reliably manages a large amount of data in a multiuser environment so that many users can concurrently access the same data. All this is accomplished while delivering high performance. A database server also prevents unauthorized access and provides efficient solutions for failure recovery.

Oracle Database is the first database designed for enterprise grid computing, the most flexible and cost effective way to manage information and applications. Enterprise grid computing creates large pools of industry-standard, modular storage and servers. With this architecture, each new system can be rapidly provisioned from the pool of components. There is no need for peak workloads, because capacity can be easily added or reallocated from the resource pools as needed.

Hibernate -

Object-relational mapping (ORM) is a programming method to map the objects in java with the relational entities in the database. In this, entities/classes refers to table in database, instance of classes refers to rows and attributes of instances of classes refers to column of table in database. This provides solutions to the problems arise while developing persistence applications using traditional JDBC method. This also reduces the code that needs to be written.

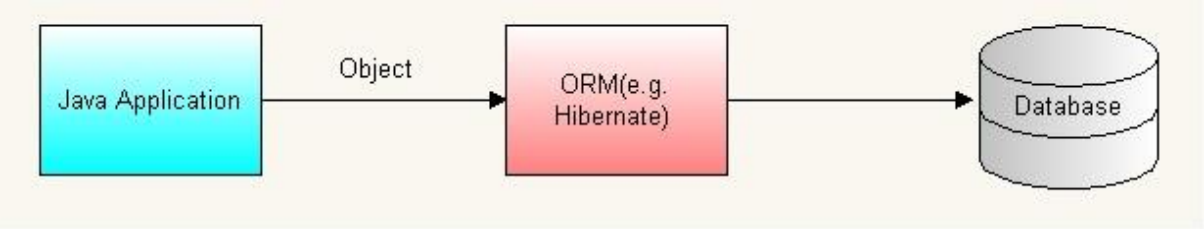

Figure 30: Object-relational mapping (ORM)

Hibernate is a pure Java object-relational mapping (ORM) and persistence framework that allows you to map plain old Java objects to relational database tables. The main goal of hibernate is to relieve the developer from the common data persistence related tasks. It maps the objects in the java with the tables in the database very efficiently and also you can get maximum using its data query and retrieval facilities. Mainly by using Hibernate in your projects you can save incredible time and effort [39].

HQL - Hibernate uses HQL (Hibernate Query Language), which is similar to SQL, but Hibernate's HQL provides full support for polymorphic queries. HQL understands object-oriented concepts like inheritance, polymorphism, and association

Hibernate will definitely increase the performance of your application and help you reduce the development time for your application — and hence the cost. So by considering the above-mentioned advantages.

SQL Developer Tool

Oracle SQL Developer Data Modeler is a new, graphical data modeling tool that facilitates and enhances communication between data architects, database administrators, application

developers and users, and simplifies the data modeling development process itself. Using SQL Developer Data Modeler users can create, browse and edit, logical, relational, physical, multi-dimensional, and data type models. The generation of DDL scripts improves productivity and promotes the use of standards [40, 41].