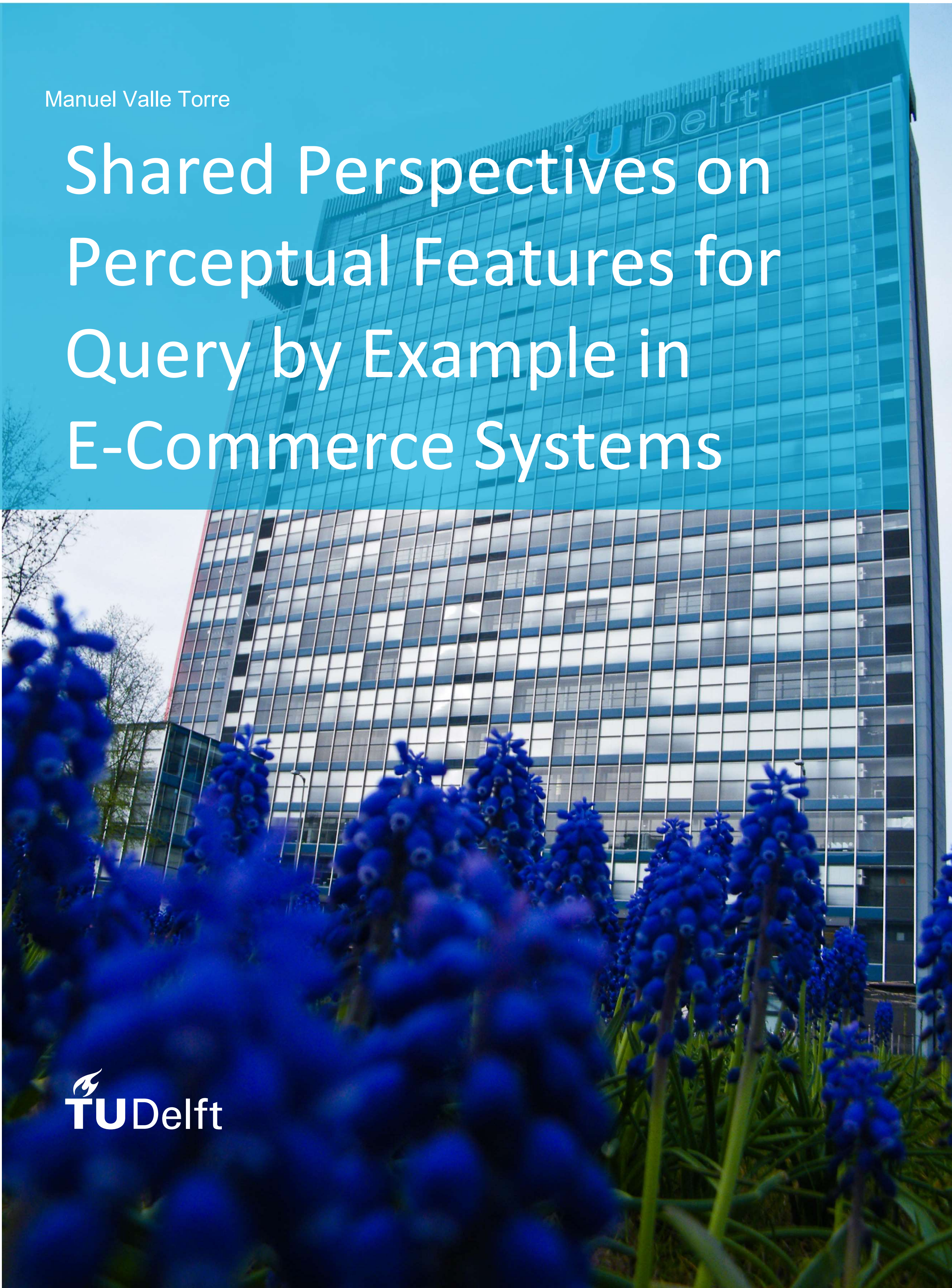


Manuel Valle Torre

Shared Perspectives on Perceptual Features for Query by Example in E-Commerce Systems

TU Delft

Shared Perspectives on Perceptual Features for Query by Example in E-Commerce Systems

By

Manuel Valle Torre
4521870

in partial fulfilment of the requirements for the degree of

Master of Science
in Computer Science / Data Science and Technology

at the Delft University of Technology,
to be defended publicly on January 31, 2018

Supervisor:	Dr. C. Lofi	WIS, EEMCS, TU Delft
Thesis committee:	Prof. Dr. ir. G.J. Houben	WIS, EEMCS, TU Delft
	Dr. C. Lofi	WIS, EEMCS, TU Delft
	Dr. R.J. Krebbers	PL, EEMCS, TU Delft

PREFACE

This report concludes my work for the Masters Degree in Computer Science: Data Science and Technology. In this thesis, the focus is to find a new way for people to query *experience items,* like movies or books, based on the users' perception of them. The basic example query is "*I want a movie like the Green Mile, because it's touching and emotional*". In this query, the user is not concerned if the similar item is a novel adaptation or it is with a great cast, just that it is touching and emotional. With the concepts presented in this thesis, a system can provide results to that query, such as: "*a similar movie is On the Waterfront, because other users also perceived it as touching and emotional*". The main challenge with this type of information is that it is subjective. The research, development, experiments and writing took place in the WIS & ST Masters lab in the EEMCS faculty.

I would like to thank the people that made this possible, starting with my family, my friends, mainly the CS Crew and my girlfriend, that motivated me throughout the process. I'm especially grateful for the learning experience and guidance from my direct supervisor, Christoph, in particular for the writing. I would also like to thank the committee members, Geert-Jan and Robbert for their time, interest in this process, and their valuable feedback. Finally, this thesis, as well as my whole study program in TU Delft would have never happened without the National Center of Science and Technology of Mexico and San Luis Potosi (CONACYT & COPOCYT) that provided me with the scholarship and financial support.

Manuel Valle Torre
Delft, January 2018

ABSTRACT

In this thesis, we focus on database query processing for so-called *experience items*, i.e., items commonly encountered in E-Commerce systems such as books, games or movies which are better described by their perceived subjective consumption experience, or Perceptual Features, than by factual meta-data normally used in SQL-style queries. To realize this, the perceived consumption experiences are extracted from social media feedback, like reviews or ratings, using methods such as Aspect-based Review Extraction or Document Embedding. These are then encoded either explicitly or implicitly as database tuples. We group similar tuples together for every item and determine the representative tuple for each of the groups, which are used in the Query-by-Example paradigm to allow users to explore and query the item space in an interactive and intuitive fashion.

In contrast to previous approaches with similar goals, like the article on 'Exploiting Perceptual Similarity: Privacy-Preserving Cooperative Query Personalization by Lofi and Nieke', we now introduce the notion of Shared Perspectives: paying respect to the subjectivity of user experiences. We do not try to encode only a single summarized experience for each item, but find and store dominant opinions shared by large parts of the user base instead. This allows us to represent controversial or split opinions much more accurately than previous systems, so the user can select which opinion is more relevant for them to find a similar item. We introduce the relevant conceptual foundations for Shared Perspectives, and give an overview of the design space for implementation. Furthermore, we showcase a prototype system, and evaluate it with respect to query performance as compared to previous approaches not featuring Shared Perspectives. We also investigate their semantic quality in a limited user study. As a result of these evaluations, we identify and resolve the new challenge of relevance of Perspectives, since not each commonly shared opinion is equally important or beneficial for query processing.

CONTENTS

1 Introduction **1**
1.1 Contributions . . . 2
1.2 Outline . . . 3

2 Related Work **3**

3 Concepts and Design Space **4**
3.1 Shared Perspectives . . . 4
3.2 Query by Example for Shared Perspectives . . . 5

4 Implementation and Prototype **6**
4.1 Source Data . . . 6
4.1.1 Related Work . . . 6
4.1.2 Implementation . . . 6
4.2 Perceptual Feature Extraction . . . 7
4.2.1 Related Work . . . 7
4.2.2 Implementation . . . 8
4.3 Storage: Relational Databases . . . 9
4.3.1 Related Work . . . 9
4.3.2 Implementation . . . 9
4.4 Clustering: Constructing Shared Perspectives . . . 10
4.4.1 Related Work . . . 10
4.4.2 Implementation . . . 10
4.4.3 Storage of Shared Perspectives . . . 10
4.4.4 Cluster Analysis: Why we Prefer Doc2vec . . . 11
4.4.5 Recalculating Shared Perspectives . . . 11
4.4.6 Alternatives to Shared Perspectives . . . 11
4.5 Query by Example . . . 11
4.5.1 Related Work . . . 12
4.5.2 Implementation . . . 12
4.6 Integrating Example . . . 13

5 Evaluation: User Simulation **13**
5.1 Simulation Set-up . . . 14
5.2 Results . . . 15

6 Evaluation: Semantic Quality of Shared Perspectives **16**
6.1 User Study . . . 16
6.2 Predicting Usefulness Scores . . . 17
6.3 Results . . . 17
6.4 Exploration and Analysis . . . 18

7 Conclusion **18**
7.1 Future Work and Discussion . . . 19

8 References **19**

A Example of User Study **20**

Shared Perspectives on Perceptual Features for Query by Example in E-Commerce Systems

Manuel Valle Torre
4521870
m.valletorre@student.tudelft.nl

1. INTRODUCTION

In current web information systems, users face the challenge of finding information in large collections. Some of these systems are specialized in e-commerce, and they host many types of products or services. One particular case is that of *experience items*. These items have the key characteristic that some, or most, of their value is experienced by users, such as movies, videogames, hotels or restaurants. For example, the experience of a movie with an engaging storyline or a moving performance may be more important for an interested user than its director or distribution company. In consequence, experiences should play an important part in the process of exploring and searching for new items. However, their available information consists mostly of objective and hard-set *factual features*. These factual features are intrinsic to the item and their schema can be easily modelled for a system: movies have a title, release year, director, cast, etc. With a simple query, a user can find the item they are looking for, using any of these features or their combination. If a user wants to find The Green Mile they can just search for the title and most systems will return the appropriate information.

In contrast, the features that are perceived by the users in their experience with the item are rarely available for query. For example, a given user likes The Green Mile because it is "emotional and touching" (and not necessarily because it is an "iconic prison movie" or a "good book adaption"). As a query, the user wishes to find other similar movies with respect to their perspective of interest: being emotional and touching. Such a query is not supported by most systems today, because these *perceptual features* are not commonly available. Furthermore, even if they were available, current systems only discover movies which are similar in their *overall* perception, and not personalized to the interested user. In the following paragraphs we describe how we propose to address this unavailability of perceptual features, and the lack of personalization for the users' interest.

The availability of perceptual features is limited because they are subjective: they depend on every user's experience and not on the item itself. A movie may be perceived as entertaining and witty for some people, while others perceive it as forced and pretentious. Given this subjectivity, modelling the schema of perceptual features for every item and assigning values manually is not feasible. The producing company or the administrator of the e-commerce system cannot decide this up-front. Fortunately, perceptual features and their values can be obtained indirectly from user feedback, where users share the experience they had with an item. Some of these feedback sources include ratings [1], reviews [2] [3] [4], and crowdsourcing [1]. In addition, there are methods that can determine and extract these perceptual features and their values from text documents, such as reviews [5] [6]. These methods take advantage of different Natural Language Processing (NLP) tools: from simple tokenization and stemming [2], to complex onthologies and lexical databases such as WordNet and SentiWordNet [7] [8] [9], and even machine learning approaches like Latent Dirichlet Allocation [10] [3] or document embeddings [4]. The extracted features may allow human interpretation, like quality of acting or scenery, or they may be latent or implicit features with no possible interpretation. In this research we use reviews, since they are detailed descriptions of the user's experience and they are commonly available in most e-commerce systems. We assume that the performance of the aforementioned methods is acceptable to obtain perceptual features and their values from reviews, and use them in a system.

This leads to the next challenge: how to handle all the reviews since there can be hundreds or thousands of them per experience item. While every review conveys the experience of a single user, there is usually a few major *perspectives* that users *share* for a given item. This does not mean that all reviews can be perfectly classified into clearly defined categories, but that some groups have common ground on their perception. These groups, that we call *Shared Perspectives* (SPs), can be used to represent the experience of most users with the item. For example, from 400 reviews for The Green Mile, there may be 180 that say something along the lines of "beautiful and touching movie, full of emotion". Some details can differ, but the general *perspective* is the same. Another 110 agree that it has "great acting and a good story that follows the book quite well", and 80 more say that it is a "good movie with famous actors, it is long but worth it". The rest may be isolated or unique opinions saying that it is an "unrealistic fairy tale in prison", or "such a bad movie, Tom Hanks is so lame", that do not belong to major groups. Therefore, by grouping reviews with similar perceptual feature values, a system can find the Shared Perspectives of the item. For this grouping to be possible, a given system requires that feature values are extracted from the reviews, because processing natural language is not a trivial task.

Once obtained, Shared Perspectives pose the challenge of querying them, since they are conformed by the perceptual features, which are widely varied, subjective, and can even be implicit. Users would need to know the schema of every system and the values of every item to generate useful or even valid queries. One system may have a feature in their schema called 'funniness' while the other one calls it 'comedy level'; for one system the 'action pace' goes from 1 to 10 and in others from 0 to 100. Since some methods even extract implicit features [11] [12], those features and their values simply do not have human interpretation. In addition, for queries on experience items, people usually do not know exactly what they are looking for, and therefore are not clear on the values required for its query. Such querying problems can be addressed with the use of *Query by Example* (QBE), where a user provides an example item and the system returns several similar items. In brief, QBE finds items with similar representations to the example item, the *representation* being a set of values such as a vector or a tuple with the features of the item. The notion of using QBE to avoid the need for structured queries or SQL has been applied before, for example to find similar movies by obtaining the values of perceptual features from ratings [1] or reviews [4]. In these studies, authors obtain a set of implicit perceptual feature values for every item, and use this as their representation in the QBE process. By using QBE, the user supplies an item to find others with roughly the same perceptual features: "something like this, but not this". This starts an iterative process where users can provide one of the resulting items as the new example, until a satisfactory item is found. Such systems do not require that the user has any specific knowledge about the schema, the values, or even the precise target.

These approaches using QBE commonly represent each item of the collection in a high-dimensional perceptual feature space, for example for features build on reviews [1], or for features build on ratings [4]. However, these approaches have been shown to have a major flaw [4]: by incorporating multiple subjective, potentially conflicting, or even irrelevant user feedback documents into a single representation, a significant part of the semantics can be lost. For example, the Green Mile might either be seen as an emotional and touching movie, or as a prison movie, or as a good book adaption, the aforementioned perspectives. Depending on the current user's own viewpoint, a different perspective might be more relevant. Furthermore, some perspectives commonly present in reviews might be irrelevant for the item, for example describing packaging and shipping. Previous approaches would consider a movie similar to another because its representation was constructed by reviews that say it is also shipped late or arrived damaged.

To address this weakness, this work proposes to use Shared Perspectives for Query by Example to help users in the process of querying experience items. With the Shared Perspectives, the user provides an example item and the system returns several items that were perceived in a similar way. The system can indicate the Shared Perspectives of the example item, so the user can identify why the resulting items are similar. This way, the user is involved in the process of similarity, they are informed of *why* the resulting items are similar to the example. Instead of "these are nine similar items to your example because I say so" the system returns "these are three similar items to your example because people also think it is emotional and touching, these three are similar because of good acting in a great book adaptation, and these three because it has famous actors and it is long but worth it". This allows transparency in the process of selecting the new example item to continue the query.

The evaluation of the proposed ideas is not simple, mainly because of the subjectivity surrounding experience items: the users' perception, how it translates in their feedback, and how others interpret it. Therefore, we simulate user behavior in QBE to compare the performance of Shared Perspectives against the aforementioned single representation. We also conduct a user study to analyze and model the semantic quality of Shared Perspectives when used to say that movie A is similar to movie B.

1.1 Contributions

The contributions of this work are presented as answers to the following questions:

RQ - How can we improve the process of searching experience items, based on their perceptual features? We propose to use Shared Perspectives in Query by Example to find items that were perceived in a similar way by other users. The Shared Perspectives provide the perceptual information to understand most users' experience with a given item. The use of Query by Example simplifies the search process for the user. This is the main contribution of this work, however, other questions have to be answered before this can be feasible.

RQ1 - How can we obtain the perceptual features of experience items? The perception of an experience item by a user can be extracted from a review. The extraction is required to obtain values that a system can process, since human language is unstructured and messy. From the broad field of Natural Language Processing, there are many technologies that can be combined in different ways to achieve the extraction results closest to human understanding. For this thesis we study the current situation in this field and adapt existing methods for the prototype implementation.

RQ2 - How can we represent the experience of many users effectively, given that they are subjective? Assuming that perceptual features can be extracted from feedback documents with acceptable precision, there are still a lot of values to analyze. However, not all documents available convey a unique perception of the experience with the item: there is some agreement between them. In consequence, we propose to group together similar perceptions into Shared Perspectives. As a result, an item can be represented by how major groups of users experienced them.

RQ3 - How can we store the extracted feature values for future use? We can use a Relational Database (RDB) with some conceptual delimitations to store perceptual feature values. The discussion of storage of the obtained values for future use is rarely addressed in feature extraction research. However, for Shared Perspectives it is necessary to store them. One reason is that in many systems there will be new feedback documents added continuously, and the Shared Perspectives have to be recalculated. Another reason is simply for implementation, some clustering algorithms require a specific format to calculate distance between tuples, and it may differ from the output format of the extraction methods. By storing the tuples, the format can be adapted when reading from the database. In consequence, we describe and implement the concepts of Perceptual Features, Perceptual Feature Families, and Perceptual Tuples as part of a RDB.

1.2 Outline

In Section 2 we describe the related efforts on exploiting user feedback for perceptual similarity of items, followed by the formalization of concepts behind Shared Perspectives and Query by Example in Section 3. The implementation is subsequently addressed in Section 4, where we describe the steps required for SPs for QBE: data collection, feature extraction, storage, grouping and querying. The relevant related work is presented for every step of the process, along with the concepts defined and the implementation details of this particular prototype. Afterwards, we use the prototype to evaluate these concepts using real-life web data, with both simulated user behavior for performance and an exploratory user study for semantic quality of the Shared Perspectives. During the synthetic evaluation, where we compare performance of SPs to single representation in Section 5, we show that not all Shared Perspectives are useful to relate one item to others. Therefore, a metric of usefulness is created with the assistance of real users in the exploratory study in Section 6, and then its performance is evaluated using the synthetic experiment again. In summary, it is shown that using a few Shared Perspectives to search for items has better performance than one representation per item. Finally we conclude the text in Section 7, along with a brief discussion on future work.

2. RELATED WORK

In this section we present studies that focus on improving the way users find experience items in information systems, by their perceptual features. There are several approaches for this, each one of them with more variety of methods and tools applied. The approaches can be divided in two groups, from the side of the user's attitude: actively searching, as a query; or passively waiting for items, as recommendation.

In this research, the attention is on the active search of experience items using perceptual features. An interesting approach for this search focuses on working with the experiences in *Perceptual Spaces*, a vector space where users and items are positioned together [13]. This research takes advantage of ratings, for their high availability and low handling complexity: if a user gives a high rating to an item, it is assumed that they liked it. The process starts by forming triples (movie, user, score), which are processed in a modified version of Euclidean Embedding factor model to minimize the cost function. Afterwards, a d-dimensional space is mapped with users and items, where items are located close to the users that rated them highly. The users are then removed from the space, leaving only similar items close to each other, with no telling what the similarities are. This is because the unknown d dimensions in space are latent features, they are not directly observed but rather inferred through the model. To better understand the resulting dimensions, in following research, authors make use of crowdsourcing to classify the genre of the movies. If a group of similar movies is classified as funny, and they share a high value in one dimension, that dimension could be labeled as 'funniness'. To search for items in Perceptual Spaces, authors propose to use Query by Example, allowing easy-to-use and personalized queries [1].

Along the same line, another study uses reviews to obtain a single vector representation for a movie, then the representations are used to find similar movies [4]. Focused on the challenge of putting user-generated feedback to practical use, the study works with experience items and reviews, the goal is to obtain the latent features of every movie from the reviews. An important advantage over the work with Perceptual Spaces, is that it does not matter who wrote the reviews or how many, so there is no need to keep extensive user profiles. The authors compare three different extraction methods: TF-IDF, Latent Semantic Aanalysis with TF-IDF, and a document embedding model, to create latent vector representations of 3,284 movies, each one with an average of 8.58 reviews. For all methods, one vector is obtained using all the reviews of a movie as a single document, regardless of their perception: a crucial difference with Shared Perspectives. The reviews could state opposing experiences, or information about the movie format or vendor, and they were all forced into a single representation for the item. Since there is no conventional baseline to evaluate this type of work, the three methods are compared to the outcomes of the study on Perceptual Spaces [1]. The document embedding model was tested with 600, 300 and 100 dimensions. The embedding model with 100 dimension had highest correlation with Perceptual Spaces, out of all methods evaluated, and was therefore considered the best approach. An interesting result is the similarity of unrelated movies, caused by the presence of bad reviews in the corpus of each item. These 'bad' reviews include comments about delivery or provider service, or the format of the movie (VHS, DVD, Blu-ray), and they do not contribute to a meaningful representation of the vector. This is an important problem considering that there is only one vector per movie, so these reviews will affect the performance of the system. In summary, the study is a first approach to the actual application of neural document embeddings to be used as a similarity measure between experience items.

It is important to also review the situation in which the users are not actively searching, but instead items are recommended to them. This is because, in essence, Recommender Systems (RS) also address the challenge of finding experience items by their perception, especially when they are unknown to the user. The development of RS comes from the observation that individuals often rely on suggestions in their daily decisions (e.g. a friend recommends a book, or a critic praises a movie). With this notion in mind, the basic RS take advantage of collaborative-filtering to suggest items that other people with similar behavior have liked. There are many improvements to this basic idea, for example: boosting new or unpopular items, or taking into account user feedback when an item is suggested correctly [14]. There are more user-centric RS approaches, such as demographic [15], where the user gets recommendations based on their profile, nationality, age, location, etc. Another one is community based, where recommendations depend on the network of the user, or their friends' interests [16]. Finally, the closest to the approach of this thesis is content-based, where the recommended items have similar characteristics to the ones that the user liked [17]. The downside of RS is that they need to build a profile for the interested user: what they like or not, and even their demographics or social network, when relevant. The strength of RS is that they are in line with the common behavior of the user, however, it is difficult to adapt them to varying situations. For instance, if a user likes to watch action and horror movies but right now he is taking care of a younger sibling, recommendation results may not be very useful.

There is interest and efforts to obtain *Perceptual Features* from user feedback and use them in the search process for *Experience Items*. There are Recommender Systems that suggest experience items to users based on their characteristics, what they like, what their friends like, their demographics, etc. Some use ratings, others reviews, and some even behavior, but they all generate one *Perceptual Tuple* per item from the user feedback available, and use it to find similar items.

3. CONCEPTS AND DESIGN SPACE

In this section we explain the motivation behind Shared Perspectives (SPs) and their key concepts, which are used in the rest of the text. The main inspiration for SPs is that when people talk about experience items, their perceptual features are usually the center of the conversation. Comments like "this movie is so scary", or "that book is super engaging", are the usual way to communicate such experiences. Yet, this is not common in e-commerce platforms, where only factual features are used. Using factual features in SQL or 'normal' queries works perfectly when the user knows exactly the item they are looking for.

On the other hand, Recommender Systems are there for users that prefer the system to suggest something they may be interested in. The goal of SPs is to cover the middle ground and enable users to search for experience items, supported by the Query by Example process, in an exploratory fashion. This allows for custom search in all kinds of situations, with no system knowledge and no need for user profile or information.

3.1 Shared Perspectives

In this section, we introduce the idea of Shared Perspectives and then describe each concept individually. In Web Information Systems, particularly in e-commerce platforms, users interact with information about experience items. These items have factual features, as well as **Perceptual Features**, which are rarely available in current information systems. Perceptual Features are subjective to each user's experience, and therefore not commonly modelled for every item. Fortunately, when a person uses an item, they can provide feedback, such as a rating or review, of their experience. There are methods available today that are able to encode the experience in that feedback into a set of perceptual feature values: a **Perceptual Tuple**. If every feedback document, like a review, can be encoded into a Perceptual Tuple, there can be hundreds or even thousands of tuples for every item. For most items, we suggest that not all tuples are individually unique, but there are a few major perspectives that most of them share: **Shared Perspectives**. This is, the distance between the tuples can be measured, and therefore they can be clustered into several groups. In addition, every group contains a tuple that has the least distance to the rest of the tuples in that group so it represents it, the **Shared Perspective Tuple**. Every item has a few Shared Perspectives and their respective Shared Perspective Tuples, which represent the experiences of most users with the item. This idea also helps to ignore isolated experiences, resulting in the more supported views only. These concepts can be formalized in the following way:

- *Factual Feature (F):* Single objective attribute of an item, with only one possible value. They are mostly do not depend on each other or other items. This is the traditional, or regular attribute widely used in current systems, for example title, release year, director, etc.
- *Perceptual Feature (PF):* Single perceptual attribute of an item. It shares most characteristics of a factual feature F, with the exception that it belongs to a Perceptual Feature Family. This means that a PF represents only part of its Family, therefore it should be paired with the other members to form a perception, and not used for QBE by itself. The domain of each PF is determined by the extraction method, such as 'acting' for movies, or 'implicit feature_1'.
- *Perceptual Feature Family (PFF):* Ordered set of PFs that are semantically related, $PFF = \{PF_1 ... PF_k\}$. A PFF is the possible output of an extraction method, if a given method generates a vector with k features, there will be k PFs in that Family. There can be several PFFs for a given system, and they are independent of each other.
- *Perceptual Tuple: (PT)* A set of values that belong to an item, for a PFF. This represents the individual perception from an individual feedback document. A PFT is the actual set of feature values extracted from a method. For instance, for a method with a PFF of acting, storyline, and directing, the PFT of a review could be (2, 3, 0). Since Families are independent, an item may or may not have PFTs for different Families from the same document.
- *Perceptual Family Relation (PFR):* Normalized table that stores all the PT for a single Family. There will be as many PFRs as extraction methods used for a given system.
- *Shared Perspective (SP):* Each one of the groups of similar PTs for an item. Depending on the format of the extraction method, a clustering algorithm that can work with the PTs obtained is applied. Parameters and implementation details will vary for every system, as long as similar PTs can be divided into groups.
- *Shared Perspective Tuple (SPT):* The representative tuple of a SP. The representative is a tuple with the most similar values to the rest of the PTs in the same SP, such as a centroid, medoid, or average. The SPTs of an item will its representation in the Query by Example process.
- *Shared Perspective Relation (SPR):* Constructed from the SPTs of one PFF. For example, it would contain all the SPTs obtained by using one extraction method with its respective clustering algorithm on the reviews of movies.

In most cases, users' experiences with an item are not completely unique, but there are major perspectives that groups of users share. By extracting *Perceptual Features* from user feedback, and encoding each user's experience into a *Perceptual Tuple*, we can group similar tuples together into *Shared Perspectives*. From every SP, there is a central *Shared Perspective Tuple* to represent the item, effectively reducing hundreds of tuples for an item into a few.

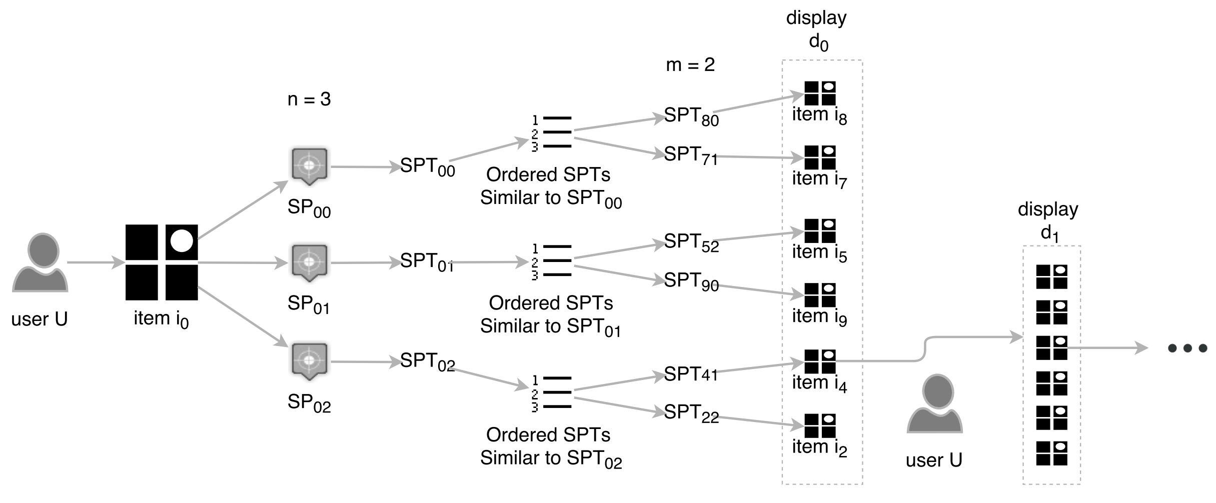


Figure 1: QBE Example Process

3.2 Query by Example for Shared Perspectives

In this section we describe the need for Query by Example, and how the concepts of Shared Perspectives and their Tuples are used in the process. If a user wanted to execute a SQL query on SPs, they would need to know exactly the perceptual features available for query, as well as the values they are looking for. Given that extraction methods can result in any number of perceptual tuples, some of them with implicit features, such SQL queries are not feasible. A similar situation exists in multimedia databases, where items are also represented with tuples of implicit features. Instead of a query on such features, the user can provide an item that resembles what they are looking for, in a QBE: they can give the system an image of a tree on a hill to get a visually similar one back [18]. The system uses the implicit features of the item, regardless of the user's knowledge about them, to find other items with similar features [19].

We propose to use this approach for experience items, where the user can provide a movie with a nice story that follows the book, and get similar movies based on that perception. The use of QBE not only avoids the problem of unfeasible queries on implicit features, but also the challenge of having implicit target, where users don't know what they're looking for. If they knew what they were looking for, a SQL query would suffice. In addition, QBE can be an iterative process, where the user can select an item from the results as the new example, until an item is found with the desired perceptual features. Conceptually, the process of Query by Example with Shared Perspectives is described below and shown in Figure 1:

- User U is thinking of a certain *Experience Item:* i_0 that fits their current situation or mood, and they want something that can provide a similar experience.
- The user provides i_0 as the example of the QBE. This item has n *Shared Perspectives*, therefore SPT_0: $\{SPT_{00}, SPT_{01}, ..., SPT_{0\text{n}}\}$. The system contains the SPTs of each of the k experience items: $i_0, i_1, ..., i_{\text{k}}$ in its collection C.
- For each SPT in SPT_0, the system will calculate the similarity with the rest of the $SPTs : SPT_1, SPT_2, ...,$ SPT_{k} available in the table SPR.
- The system will then select m most similar SPTs for each one of SPT_0.
- It will show the user a *display* d_0 with the m items that have the most similar SPT for each SPT in SPT_0. Therefore d_0 will have a size n x m.
- If any of those items is satisfactory for U, then the process ends.
- If not, the user can select the item they like the most from d_0 and provide this as the new example of QBE.
- This process is iterative, it can be executed until an item is found. The SPTs of items shown in d_0 are discarded when computing d_1, then the ones in d_0 and d_1 for d_2, etc.

For example, Tom wants to watch a movie like The Green Mile. He watched it last week and he experienced it as a movie with nice acting and a story that follows the book quite well. He provides The Green Mile to the system as the example, and gets a display d_0 of six items back, see Figure 1. From those six items, there are two *(m)* for each one of the three *(n)* Shared Perspectives of The Green Mile: *a)* nice emotional movie, *b)* movies with good acting and story close to the book, and *c)* movies that are long but worth it. Tom is not completely convinced by the results, but one of them called Patriot Games, that is similar to The Green Mile because of *b)*, seems interesting. Tom provides Patriot Games as the new example, and receives a new display d_1 with six new movies. One of them is The Hunt for Red October, and Tom decides to watch it. The implementation of a similar system is carried out in the following section, and evaluated afterwards.

Since the *Shared Perspectives* are encoded as *Perceptual Tuples* from user feedback, SQL queries are mostly unfeasible. They can be avoided with the *Query by Example* paradigm, that calculates the similarity between the *Shared Perspective Tuples* of the example item and the items available in the collection, and returns the most similar items. QBE is an iterative process, it discards the need for SQL queries and accommodates for simple use and situations when the user is not clear on their target.

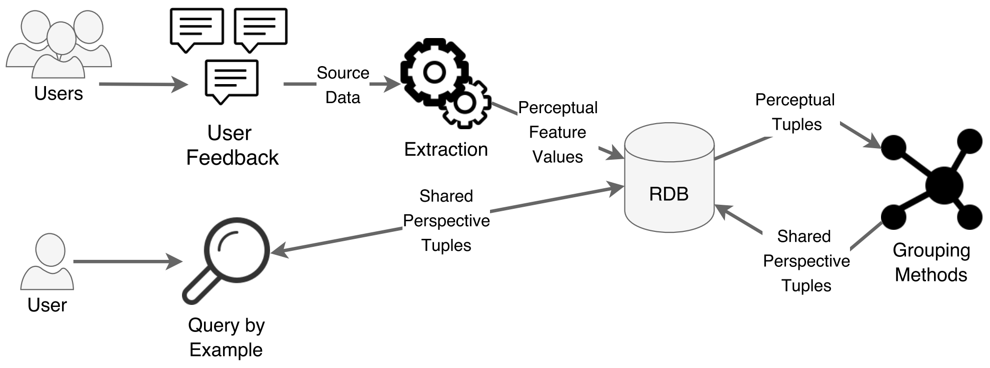


Figure 2: System Design

4. IMPLEMENTATION AND PROTOTYPE

Once the theoretical background is established, we need to carry out the ideas proposed to ground the concepts proposed to attest that they are feasible, which we address in this section. Simplified, if we can encode reviews into tuples and we apply a clustering algorithm on the tuples for every item, we get Shared Perspectives. The Shared Perspective Tuples, center of those SPs, can be the representation of the item in a Query by Example process, so the system is able to find the items with most similar user perception. In Figure 2, we show the general design of this simplified process. Every step of the process is described in this section, together with the related work, and the implementation details.

4.1 Source Data

As it has been established, perceptual features have to be obtained from feedback of actual users. In this section we revise user feedback as information source. The source documents for perceptual features have to contain an explanation or evaluation of the experience with a given item. Ideally, this evaluation is direct, explicit and for one item only. Such feedback documents can be found in social network posts, in dedicated blogs, or as reviews in item collections like online stores or specialized websites. From these different sources, reviews in online stores tend to be direct, explicit, and there are many available for a single item. Online stores try to get reviews mostly from users that actually acquired or experienced the item. By using the reviews, we add a source of information to the data available for items, perceptual information.

In this work, we assume that reviews are:

- Accepted source of information for research and business applications [5]
- Inherently written to contain an opinion about the item [5]
- Important for interested users, used to construct an opinion about the item [20]

In most of today's online platforms, reviews are shown to the interested user, with some ordering and filtering capabilities, but it is ultimately the user's task to read and interpret enough of them to have an idea about the item before having to experience it themselves.

4.1.1 Related Work

Research towards the use of reviews for all types of experience items has been around for years, and is still evolving. Reviews are key in online shopping decisions, since they have significant influence on people when choosing experience items. The extraction, summarization, selection and display of knowledge extracted from reviews is an important challenge in today's state of information systems [5]. Opinion mining from reviews is considered a significant step in research and industry. The current paradigms of simply showing them to users is not enough, and many studies address this need [9] [21]. Most studies claim that their process and results can be reproduced across domains, and some even implement it and compare performance in more than one domain [22]. The use of reviews is not only for the extraction and summarizing, but also for querying process [4].

4.1.2 Implementation

For this thesis we use an existing dataset of Film reviews from Amazon, we adhere to movies because they share common features between them. In addition, the interaction with online movie catalogs is easy to relate to, resulting in simple and engaging examples. The dataset was originally created and organized for research on Recommender Systems [23]. Unfortunately, Amazon Film includes series, movies, documentaries and other types of video material, which are not classified. In an attempt to single out movies, we discard items with titles that include words like 'season', 'collection', 'series', and 'pack'. We only select movies that have at least 100 reviews, and if they have more than 300, the rest are left out to keep a similar number of reviews across movies. Reviews with less than 25 words usually do not convey feature information and are therefore discarded. This is mainly because Amazon requires a minimum of 20 words per review, so users tend to fill this out with repetitive statements. This results in a dataset consisting of ∼375K Amazon movie reviews, for 2041 movies.

A good way to obtain users' experience with an item is from their feedback, such as reviews. Reviews are commonly used in research as a source of information of user's perception, or *Perceptual Tuple.* We use Amazon movie reviews for this implementation.

4.2 Perceptual Feature Extraction

The increase in data generation that came with Web 2.0 caused a rise in efforts to understand it and use it in for research and business [24], which is the topic of this section. Efforts to work with this user-generated content have appeared in all academic areas, from Psychology to Computer Science. For Computer Science, one of the main areas is NLP, together with many relevant branches such as Information Extraction, Web Systems, and Databases [6]. The relevant content source for this work is in e-commerce systems, where a person shares their experience with a product or service using Ratings and Reviews. We assume that reviews convey enough information to extract the values of Perceptual Features. The intention is to translate the perception in the unstructured text review into a value that the system can work with. We roughly separate the extraction methods by explicit and implicit features, the first are easily identified and intuitive for humans, while the second are underlying and very difficult or impossible to label. Furthermore, the explicit features are defined before extracting their values, while the implicit features are labeled after the process, when possible.

4.2.1 Related Work

We first address extraction methods with explicit features, where there are many algorithms and tools used, but their general structure is similar. Almost all explicit feature extraction processes need a set of features and related words to look for, this can be created manually or automatically [25]. For manually created sets, developers of the extraction method decide which features are important for the item, possibly using the help of experts or crowdsourcing to support their decisions [9]. For example, in the movie domain the feature set can include Dialogues, Editing, Cinematography, Music, and Acting. For videogames it can be Storyline, Game Mechanics, Physics, Video Quality or Music. The other way is to automatically generate a set of relevant features by their relative presence in the data corpus in question, and the relatedness with the topic [26]. This is very useful in domains where every type of product has specific features, for example in electronics: screen quality is relevant for a laptop but not so much for a router. After generating the set of relevant features, most methods follow roughly the same steps: extraction continues by isolating sentences that contain these features (or any related keyword) in the document. The next step is to identify any modifiers that affect the feature, for example: "The *acting (feature)* of this movie is *amazing (modifier)*". The system then rates the sentiment and degree that the modifier inflicts on the feature. This process of explicit feature extraction usually generates a summary of features for every item, with their positive or negative values, for example: acting 9/10 and storyline 3/10.

The main technologies used in the process are the following: Part of Speech Tagging to find the nouns and modifiers in a sentence, and Association Mining to find itemsets [7] [2] [6]. The nouns are usually the relevant features, while the itemsets are groups of keywords related to them. In addition, Adjective Identification is used to identify the modifiers or words that affect the features. This can be challenging, for example: "with this acting, why would anyone watch this movie?" has no direct modifiers, however it is highly negative. To assing actual values to the feature, researchers use Orientation Identification [2]. An alternative approach to determine a sentiment score is done with the assistance of lexical databases, like SentiWordNet [8] [7]. Once again, this task is far from trivial, especially for the complexity of natural language, for example: "The cast is amazing, the story sounds beautiful and the photography promises to be engaging, however it fails to deliver" [27]. Every sentence has a direct modifier positively affecting the feature, but in the end they are all cancelled, which may be easy to notice for the reader, but not for an algorithm [5].

In this work we use one explicit feature extraction method called: Aspect-based Review Extraction (AbRE) [9]. This method has the goal of obtaining relevant features for a type of item of any domain, as well as the user's feeling towards each one of them. To determine these features, the first step is to use TF-IDF to find keywords that are frequent in the selected item reviews (movies, in this case), but not so frequent in other 'general' corpus such as news. This is an example of automatic generation of a feature set. The most relevant features for the item are compared to the actual item, using WordNet, to obtain their similarity score. For example, the word 'life' and cting' are relevant in the movie corpus, but according to WordNet, 'acting' is significantly closer to 'movie' than 'life' By using both TF-IDF and the similarity scores, the top 5 words are selected as the most important features for a given domain.

With the feature set defined, the sets of related words to each feature are created by their distance in WordNet, these are all words with a similarity score over 0.75 for each feature. For example the set of acting includes actor, performance, acting, cast, and role. Every word in the review is scanned and if there is a match with a word from an item set, the system looks for adjectives and adverbs in the same sentence. SentiWordNet is used on the adjectives and adverbs affecting the identified word, to assign a value from 0 (negative) to 1 (positive) to the modifier word. Finally, the scores of all modifiers for a word are averaged and normalized from 1 to 5. Implemented in movie reviews, AbRE determines the sentiment score of five defined film features: acting, directing, scenery, character and storyline [9].

The basic process of AbRE is the following:

1. Find top 5 features for the current dataset
2. Identify words related to the features using WordNet
3. Isolate sentences with features or related keywords
4. Identify modifiers (adjectives or adverbs) affecting the keywords
5. Assign sentiment value from 0 to 1 using SentiWordNet
6. Average and normalize into 1-5 scale

Now we turn to implicit feature extraction methods. They have more varied structure, but are generally based on mathematical and statistical modelling, often aided by machine learning. The main difference with explicit feature extraction methods is that both the selection of the features and their values are performed at the same time, resulting in a set of unnamed features. Most methods process all documents at the same time, instead of one by one like explicit methods. Sometimes the resulting features can be labeled, typically by experts or using crowdsourcing. However, in other cases the individual features simply have no human interpretation, and the values have to be processed together. Two known methods are Latent Semantic Analysis (LSA) and Latent Dirichlet Allocation (LDA). LSA assumes that documents that contain similar words have similar topics. For LSA, a matrix is constructed containing a word count per paragraph, and Singular Value Decomposition (SVD) is applied to reduce the number of rows while preserving the structure of the columns. Words are compared by the cosine distance of the angle of their resulting vectors [12]. For documents, the vectors of all the words are aggregated resulting in a single representation, which is used to compare with other documents. LDA is a generative probabilistic model of a corpus where documents are represented as a mixture of topics. Every word in a document attributes to the probability that the document belongs to one of the topics. This allows the selection of the most dominant topic of each document by the words it contains [10]. LDA was proven useful to obtain subtopics of restaurant reviews on Yelp!, with 50 topics modelled. The four most relevant topics are: service, take out, decor and value. The authors use sentiment analysis to rate each subtopic in a review, and use them to predict the actual rating [3]. Both for LDA and LSA, the resulting topics can usually be labeled. In some cases, resulting topics contain words that are hard to classify in a single container, they have to be broad. In a comparative study with LDA and LSA, authors used movie plots to create a recommendation system. By using LSA, they represented each movie plot with 500 topics instead of 220000 keywords, while for LDA, they used 50 topics. In short, LSA required double the computational cost, but had significantly better performance than LDA [28].

The method that we apply in this work uses neural networks for document embeddings, mapping them to a vector of real numbers [4]. Document embeddings are based on machine learning and statistical analysis, like LDA and LSA. The most known implementation is Doc2vec, developed by Gensim, which populates a dense vector of fixed size as a representation of a variable length document. Doc2vec is used for this work since it can be applied for reviews of any length, returning a vector of the same size. This consistency is vital for the Pearceptual Feature Family structure in a RDB. It is based word2vec, an unsupervised framework trained to predict the next word in an n-gram, but with the addition of a paragraph token that contains the topic of the text. There are two types of document embedding: Distributed Memory Paragraph Vector (dmpv) and Distributed Bag-of-Words (dbow), both show considerable performance for sentiment analysis in movie reviews as well as topic detection from Google Snippets [11]. For this thesis we only focus on dbow since both the authors of the implementation [1] and an empirical study show that it has appropriate performance [29]. In addition, its implementation is more straightforward[2]. Doc2vec is still under research, but it has been used to represent documents with better results than other approaches like Bag of Words or the aggregated vectors of all the words in the document. One key advantage is negative sampling of high frequency words, and the comparison between document and corpus item frequencies [29]. The features and values extracted by this method are implicit, it is simply not possible to identify any meaning from the values themselves.

[1]Authors of Gensim on dbow vs dmpv: https://github.com/piskvorky/gensim/blob/develop/docs/notebooks/Doc2vec-IMDB.ipynb

[2]we also tested dmpv,the resulting vectors were very to dbow but training was significantly slower

4.2.2 Implementation

The methods selected for implementation are Aspect-based Review Extraction and Document Embeddings, representatives of explicit and implicit feature extraction, respectively. We will describe implementation details and an example for each one of them below.

Aspect-based Review Extraction: This method was already applied for the Amazon movie review dataset [9], so it was only necessary to process the values.

Example: *The Green Mile* review

Ok, so it did not deserve best picture. It was still excellent. It has great performances in it. Particularly the guy who never was very famous Michael Jeter or whatever his name is. I love the visuals. I cried at the end. Michael Clarke Duncan is great.

- Perceptual Feature Family: acting, directing, scenery, character and storyline.

- Perceptual Feature Tuple: 3.43, None, 3.12, 3.43, None.

Document Embeddings: A key aspect when using this approach is that there are many implementation choices to be made, from the corpus selected to the tuning of hyperparameters. For topic similarity purposes, in some implementations it is recommended to add other available corpus (such as news or Wikipedia), or to use pre-trained models created by the authors [11]. However, in this work only the Amazon movie reviews were used to train the model, so the 'topic' of the document is the perception of the user. For parameters selection, the decision was to follow the recommended values by the implementation authors, as described before:

- Vectors are kept at 100 dimensions
- The training window is kept at 10 since it showed good performance with documents of similar size (reviews)
- Frequent word subsampling seems to decrease sentiment-prediction accuracy, so it is left out
- dm=0 and dbow_words=1 selects the 'dbow' mode
- A min_count=2 saves model memory significantly, discarding words that only appear in a single document, and are no more expressive than the unique-to-each vectors themselves
- The learning rate is alpha = 0.025 and is kept fixed

Example: *The Green Mile* review

Ok, so it did not deserve best picture. It was still excellent. It has great performances in it. Particularly the guy who never was very famous Michael Jeter or whatever his name is. I love the visuals. I cried at the end. Michael Clarke Duncan is great.

- Perceptual Feature Family: d0, d1, ... d98, d99

- Perceptual Feature Tuple: (-0.640138, 0.422624, ..., -0.0350407, 0.192102)

It is necessary to encode the reviews into *Perceptual Tuples*, so a machine can process them to find *Shared Perspectives*. There are many methods to encode unstructured text for processing, we implement two: *AbRE* with 5 explicit features, using POS tagging, WordNet and SentiWordNet. The other one is *Doc2vec*, it uses neural networks and results in tuples with 100 implicit features.

4.3 Storage: Relational Databases

In this section we describe the concepts related to storage of extracted values as tuples, their structure and basic constraints. This part of the process is rarely discussed in extraction studies like those addressed in Section 4.2. Such studies do not have the need to store the obtained values for future use. For Shared Perspectives and Query by Example, obtaining the tuple of a document is only the first step. By storing these tuples, the SPs can be recalculated when new reviews are introduced to a system, so it is key to keep them in an organized database. In addition, it helps with the general calculations, the application of clustering algorithms, and would be practical for an actual commercial system.

4.3.1 Related Work

In this section we describe the general theory of Relational Databases, with a movie database example. In short, RDBs store information in *relations* (tables) where every *tuple* (record or row) contains *values* (within a domain) for every *attribute* (column). There are special cases where an attribute of a tuple can have multiple values. Usually, multivalued attributes are managed in one of two ways: In the first, the values are simply stored in a single field, so they must be processed for querying, this is rarely done and it is usually advised against. In the second option, the multiple values are normalized, captured in a separate table and linked to the entity. This is done via surrogate keys that refer many values to their respective single item, or cross-reference tables for many-to-many relationships [30]. Multiple values are independent in the same attribute as well as across different attributes [31]. The basic requirement is that the original relation can always be recovered by performing a join [31].

Before proposing how to store perceptions in a RDB, we review the basic concepts of the Relational Database Model. To describe the theory of these relations, we will refer to the Relation Scheme and Instance model. Starting from the Primitive Relation Scheme, the triple $PRS = (\Omega, \Delta, dom)$, where Ω is a finite set of attributes $\{A_1, ..., A_k\}$, Δ is a finite set of domains $\{\delta_1, ..., \delta_k\}$ where each domain is the set of possible values for the respective attribute, $dom : \Omega \to \Delta$ is a function that associates each domain to each attribute. These three concepts together are part of the Relation Scheme or $RS = (PRS, M, SC)$, where M represents the meaning and SC is the set of constraints or conditions of the relation. Together, SC and M help bind the relation to the real world it represents. A tuple is a function $t : \Omega \to \bigcup_{\delta \in \Delta} \delta$ where the value for each attribute belongs to the established domain $t(A) \in dom(A)$ [32].

The RS is only an empty frame. To create a Relation Instance, we need to assign values for each tuple, from the established domain for every attribute, in line with the meaning and constraints. For example, in a movie database, this is an instance of a Relation Scheme for movies, shown in Table 1: $movies = (PRS, M, SC) = (\Omega, \Delta, dom, M, SC)$ [3].

- The relation contains only items of the type movie, from the catalog of Amazon
- The set of attributes $\Omega = \{id, title, year\}$ exist for every movie in this relation, and only have one value for every tuple
- The domain set $\Delta = \{$set of 10-digit Amazon Standard Identification Numbers (ASIN), set of actual titles, set of 4-digit positive numbers$\}$. These have to be defined, it can be through enumeration or syntactical rules
- The function *dom* associates each set of names with their respective attribute, restricted to allowed values for the attribute
- M is the meaning of the relation, these are real movies that were released with an official title, on a certain year
- The set of conditions SC includes that the year has to be the official worldwide premier, not prize or film festival showings
- A tuple has to satisfy all requirements to belong to a relation, its values belong to the domain of each attribute and the tuple fulfills the condition set. For example, a tuple is a function that associates 0780628799 with *id*, The Green Mile with *title*, 1999 with *year*

Table 1: Movies Relation

movies		
id	title	year
0780628799	The Green Mile	1999
0780625129	American History X	1998
076780192X	Close Encounters of the Third Kind	1977

Furthermore, a Database is usually composed by several Relations. Ffor example a movie database may include actor, director, and publisher tables. A Primitive Database Scheme is a finite set of Relation Schemes, each one of them with their corresponding 5-tuple: $PDS = \{RS_i = (\Omega_i, \Delta_i, dom_i, M_i, SC_i) | i \in I\}$. Different relations have different meanings and constraint sets, in the same way, a Database Scheme $DS = (PDS, DM, SDC)$ is formed by the Primitive Database Scheme, plus the overall Database Meaning and the Set of Database Constraints. The Database Meaning and Constraints may not be entirely covered by all the Meanings and Constraints of the Relations it contains, and it has to be defined. For this work, the proposed theory extension is only on relation level, therefore it is not necessary to go into further detail on database level.

4.3.2 Implementation

In this section we propose the storage concepts for the feature values extracted from the source documents or reviews. We apply the concepts of Shared Perspectives described in Section 3.1 to RDB theory, which helps to consolidate them in the implementation.

- *Perceptual Feature:* A single feature extracted from a single method. In the case of AbRE, it is each of the 5 features. The domain includes the range of real numbers from 1 to 5 and 'None', if the feature is not mentioned in the review. For the document embeddings, it is each of the 100 dimensions that represent an implicit feature for the system. The domain is real numbers, usually between -2 and 2.

[3]the RS can also be represented as a 5-tuple

- *Perceptual Feature Family:* Groups of features extracted by the same algorithm, framework or method, in this case there are two: the 5 features extracted with AbRE, and the 100 dimensions of Doc2vec.

- *Perceptual Tuple:* The actual set of values extracted from a review of an item for one family. For instance, for a given review of the Green Mile, with a certain id, the respective values for acting, directing, scenery, character and storyline are: 3.43, None, 3.12, 3.43, None.

- *Perceptual Family Relation:* It is each of the relations that store the perceptions from all the available reviews for all items, joined via surrogate keys.

For the actual database, we used Microsoft SQL Server 2017, because it can easily work with Python, and it allows for Custom Table Types, which we considered useful at the beginning of the research. The idea was to actually restrict the structure of the Perceptual Family Tables from database side, however, this is not possible since every extraction method results in a different scheme. MS SQL was very useful for the prototype implementation, but any Relational Database Management System would equally appropriate.

Storage of *Perceptual Tuples* is required in the case of new reviews, meaning new tuples, and therefore re-clustering. It helps to ground the concepts of *Perceptual Features*, *Families*, and *Tuples*. In this prototype we use MS SQL. By using RDB there is a clear structure that the system can access in the Query by Example process.

4.4 Clustering: Constructing Shared Perspectives

The concepts of Shared Perspectives can be implemented once the Perceptual Tuples are extracted, and this is reviewed in this section. The general notion of SPs consists of grouping together similar perceptions, therefore similar PTs, for every item. As described before, this step of the process requires that distance can be measured between Perceptual Tuples. With this distance, a clustering algorithm is applied to the PTs of an item to find the groups and their representatives. In this section we only discuss the algorithms applied, but there are many available for different implementation requirements.

4.4.1 Related Work

The clustering of documents by topic is an active line of research. This clustering has many applications, for example search optimization by document clusters, or better ways to present query results [33]. In many algorithms, documents are treated as unordered sets of words, in others the order of the words matter. There are challenges of data sparsity and high dimensionality [34], however, the clustering of text documents is an accepted practice and useful for Shared Perspectives.

To group the tuples obtained by Aspect-based Review Extraction, the approach is to use Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) with a custom distance metric [9]. HDBSCAN is an integrated framework for density-based cluster analysis, outlier detection, and data visualization. In short, the algorithm calculates the core distance of each point, and the mutual reachability between points. Points are mutually reachable if they are in each others' core distance. The potential clusters can form a dendogram, where the clusters are decided via a 'minimum cluster size' parameter [35].

For document embeddings, we use spherical k-medoids clustering, which uses cosine similarity as the distance metric. K-medoids is a robust partitioning method to divide a dataset into groups, where the center or medoid of each group is an actual data point. The medoid is the most centrally located object of the cluster, with minimum sum of distances to other points [36]. By using cosine instead of Euclidean distance, the algorithm measures the angle between two vectors in space, in this case the tuples [33]. Tuples with a similarity of 1 have the same orientation, perpendicular vectors have a similarity of 0. This metric is appropriate for document tuples with this level of dimensionality, especially centered around 0, with equally scaled values. This is also recommended as an appropriate similarity metric by the authors of the embedding implementation used [34] [11].

4.4.2 Implementation

The implementation choices for this section depend on the values of the tuples extracted, and they are described below:

HDBSCAN for Aspect-based Review Extraction: The custom distance metric used is the average difference between existing values of two tuples. This distance is calculated as follows: $Dis(review_a, review_b) = avg(|review_a(i) - review_b(i)|)$ for values $(review_a(i), review_b(i) \neq None, i \in (1, 5))$. The custom metric is used primarily to overcome missing values, since most reviews do not refer to all features. These decisions are based on the approach applied by the author of this extraction method [9].

Spherical k-medoids for Doc2vec: The main reason to select k-medoids instead of k-means is that the center of the cluster actually belongs to a tuple, therefore to a review, which should be representative of the rest of reviews in the group. This results in a useful interpretation of the implicit features of Doc2vec, we can actually read the center of every Shared Perspective. For this implementation the same number of clusters is set for all movies, since the final application is a QBE system, and different amounts of representations could affect functionality. This is to avoid that movies with many clusters, and therefore many representations, appear too often in the process. Aside of the final application, the 'elbow method' was applied to select the best number of clusters. Movies show either tendency to an optimal k of 2 or 3, or there is no clear elbow. To adapt to movies where the elbow is difficult to assign, we fix k to 3 clusters for all.

4.4.3 Storage of Shared Perspectives

After defining the concepts and implementation of Shared Perspectives, the next step is to address their storage. Similar to the Perceptual Features proposed, this is simply part of the concepts used in this work. For the theoretical aspect:

- *Shared Perspective Tuple:* The representative of a Shared Perspective. For example the medoids of the Shared Perspectives for The Green Mile, extracted with Doc2vec and clustered via k-medoids.
- *Shared Perspective Relation:* Constructed from the Shared Perspective Tuples of one Feature Family. For example, the relation that contains all the SPTs obtained by using Doc2vec and k-medoids on the Amazon movie reviews.

4.4.4 *Cluster Analysis: Why we Prefer Doc2vec*

For this implementation, the most relevant difference of the clusters that result from AbRE and Doc2vec is in their hypothetical application. We consider that AbRE would be better as additional filtering in a traditional search process. For example, writing a query with genre or an actor like in current information systems, and then adding a level to features that the user considers important for the query. An example query could be: 'comedy Adam Sandler' with acting greater than 4 and storyline greater than 3. In general, the resulting clusters of AbRE with HDBSCAN have agreement on some of the 5 features extracted, but anything else like emotional level, action pace, or if it is appropriate for kids, is not taken into account. For example, a review like: "The actors deliver a great performance, it is very realistic but way too violent for kids" would be in the same cluster as "I love the acting in this movie, but it is not very accurate historically". Those two perspectives agree on good acting, but they have a completely different focus, and that is missed by the extraction method. On the other hand, Doc2vec provides more detailed representations, and the clusters have a higher level of semantic similarity, closer to what would be interpreted by users. Although Doc2vec has the drawback of implicit features, with the text of the respective Shared Perspective Tuple (the medoid), we have some information about the clusters. Therefore, we consider the use of Doc2vec better suited for QBE. With this in mind, we continue the implementation only using Doc2vec and k-medoids.

4.4.5 *Recalculating Shared Perspectives*

The Recalculation of Shared Perspectives was not implemented since we worked with a static dataset, however it is important to consider the process of updating and recalculating Shared Perspectives. The idea of SPs is to represent the item in the most complete way, covering all the major perspectives that people can have about them. Maybe a movie gets a prize, the author of a book explains a key feature in their material, or a videogame gets an update fixing some important bugs. Big changes or not, new feedback will continuously roll in, and updating SPs has to be carefully executed.

There can be several ways to update SPs, depending on the implementation needs and the rate that new reviews are added. The naive way is a periodic update: simply wait for a week or month storing all new tuples, apply again the clustering algorithm to all tuples of the item, and update the SPs and SPTs. Another way could be by calculating the distance between every new tuple and the existing SP representatives, if it is below a threshold, the new tuple can be added to the SP of the closest one. Those tuples that are below threshold distance can be stored until they reach a predefined value, then all SPs are recalculated for the item. The silhouette score (consistency) of every cluster can also be restricted by a threshold, and if a cluster of a movie surpasses this value, only the Shared Perspectives of a single item are recalculated.

4.4.6 *Alternatives to Shared Perspectives*

In the beginning of this research, the only idea was to improve, or at least create a new option for the current way of searching for experience items. We considered that the answer was in user feedback, particularly reviews, and that we should be able to use the different perspectives portrayed in them. Before Shared Perspectives, two options were reviewed with this purpose, but were eventually discarded.

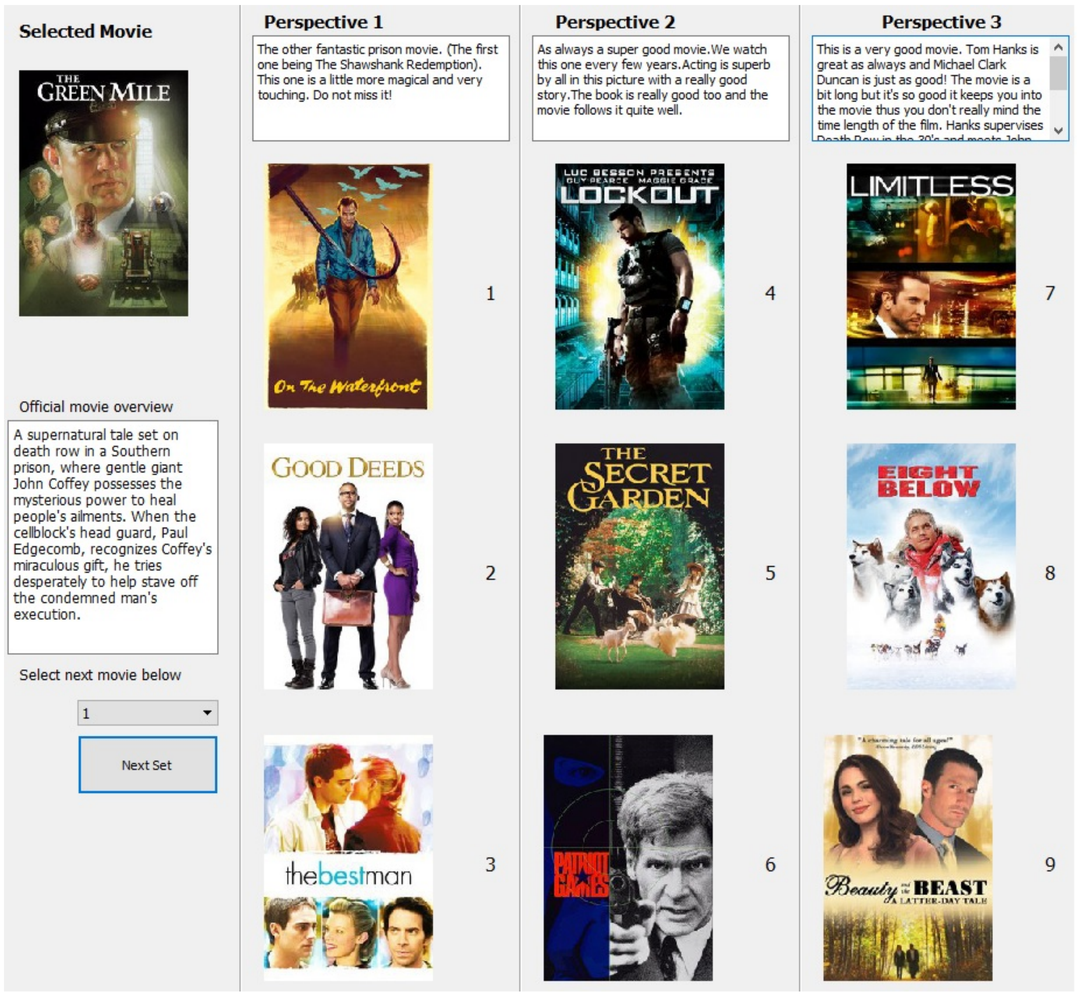
Probabilistic Databases (PDB): PDBs were developed with messy or fuzzy data as the main focus, engaging RDBs to work with several levels of uncertainty [37]. Their main applications were described by the authors in sensor data or automatic data extraction. For example, if a fact is extracted from a website, there in uncertainty that the extraction was correct, plus uncertainty of the website itself being correct [38]. Another example is in scanning handwriting: the first character of a student number had 40% probability of being a 5 or 60% of being 6. Given that most students numbers start with 5, then the probability was increased, etc. This research area was reviewed as an option that would produce the perspective with higher probability of being correct, it still forces all the perceptions into a single representation. Since we try to respect the subjectivity of perception, we stopped the work on PDBs.

Subspace Clustering (SSC): Given the nature of the feature based extraction method used in this work, we reviewed the option to use SSC for the construction of the Shared Perspectives. SSC is a type of projected clustering that takes into account only the dimensions relevant to the cluster, to reduce noise or deal with sparsity [39]. There are several methods and types, such as grid or density focused. Methods are usually classified as one of two types: the first is top-down, which discovers clusters in full-space and prunes the irrelevant dimensions. The other one is bottom-up, where clusters are found in lower dimensions and more dimensions are added if they are deemed relevant enough [40]. Unfortunately, SSC cannot work properly with the missing or *(None)* values like the tuples obtained via AbRE. Most algorithms would either create one cluster, or classify everything as noise. On the other hand, when working with the tuples generated by Doc2vec, it is indeed the combination of the whole tuple that carries meaning. In addition, dimensions cannot be interpreted by humans, so there is really no need to restrict the dimensions. It was tried regardless, and the nature of the tuple representations prohibited the process to finish, or caused the system to run out of memory.

The idea of *Shared Perspectives* is that there are similar perceptions of a given item and they can be grouped. For a prototype implementation, we can group perceptual tuples using clustering algorithms. We use *HDBSCAN* and *spherical k-medoids* for the tuples obtained from AbRE and Doc2vec, respectively. Doc2vec and k-medoids show more interesting clusters, so we continue the prototype with these only. In a real implementation, there would be new reviews added constantly, so the recalculation of SPs would have to be considered.

4.5 Query by Example

In this section we describe the implementation of the concepts of Query by Example shown in Section 3.2. In summary, to avoid the need of difficult, unintuitive, or downright impossible queries required for Perceptual Features, we propose to use the QBE paradigm. Originated from multimedia databases, QBE lets the users discover unknown items based on roughly similar features to the example provided.



The other fantastic prison movie. (The first one being The Shawshank Redemption). This one is a little more magical and very touching. Do not miss it!

As always a super good movie.We watch this one every few years.Acting is superb by all in this picture with a really good story.The book is really good too and the movie follows it quite well.

This is a very good movie. Tom Hanks is great as always and Michael Clark Duncan is just as good! The movie is a bit long but it's so good it keeps you into the movie thus you don't really mind the time length of the film. Hanks supervises

Official movie overview

A supernatural tale set on death row in a Southern prison, where gentle giant John Coffey possesses the mysterious power to heal people's ailments. When the cellblock's head guard, Paul Edgecomb, recognizes Coffey's miraculous gift, he tries desperately to help stave off the condemned man's execution.

Figure 3: Prototype Example

4.5.1 *Related Work*

As mentioned in Section 3.2, QBE is usually applied in databases that store some type of multimedia, mainly because it is difficult for users to construct queries, since implicit features are not understandable and may require schema knowledge for every particular system [18]. From images, to music and video, the use of QBE expands the usual queries, which mostly contain metadata such as titles, years, author, artists, and (sometimes) content descriptions. Implementations of QBE are such as the aforementioned example in image search, where the user supplies an image and the system will return 'visually' similar pictures [19]. Another implementation is in music information retrieval, where users can record or even hum part of a song, to retrieve information of the actual song, or the most similar available [41]. Along the lines of music, but based in Collaborative Filtering, one study tries to find similar artists to the one provided as example [42]. Finally, the previous research on Perceptual Spaces [1], and Latent Vectors [4], also use QBE to find items that were perceived in a similar way by people.

4.5.2 *Implementation*

All the analysis and processing in this work was performed in Python, therefore the implementation of the application was also built with it, and connected to MSSQL, using PyQt tools for the creation of the graphical interface.

A typical process of a user finding a movie using QBE on SP, consists of the following steps:

1. User selects a movie with desired characteristics, in order to find a similar one
2. The user is presented with a *display* of 9 movies, 3 for each of the 3 perspectives of the selected movie, like in Figure 3
3. If the desired movie is in the display, then the process ends. If it is not in the display, the user selects the next best option
4. The user can repeat steps 2-3 until they are satisfied with the result

Query by Example is used in multimedia databases because the implicit features of their items result in difficult queries for users: instead of complicated and unintuitive queries, users can give an item and get some similar items back. The *Perceptual Features* extracted from text often result in implicit features, similar to multimedia, therefore *QBE* can be used in an equivalent way. The *QBE* process can be iterative: the user can provide an example item, then select one from the result as a new example until the target is reached.

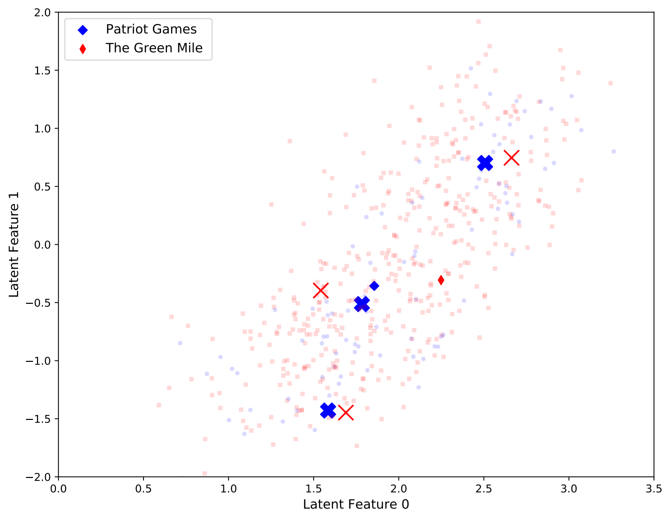


Figure 4: Cluster Example of Review Embedding

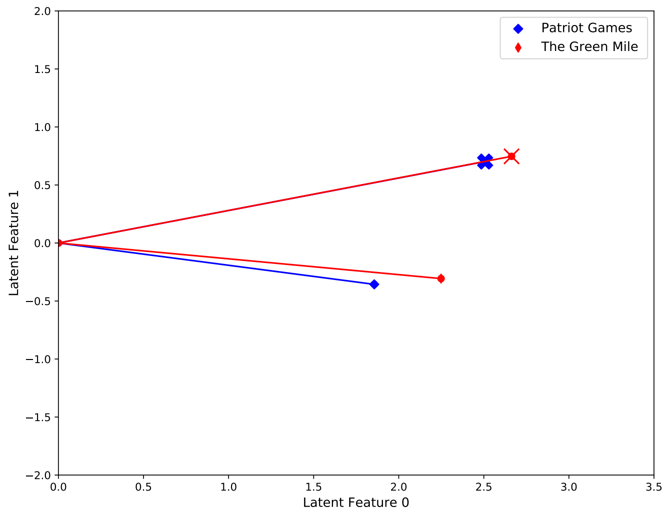


Figure 5: Cosine Distances for 1 and 3 Perspectives

4.6 Integrating Example

Putting together the extraction and Shared Perspectives implementation for this example, the result would look like Figure 4. In this example, the Perceptual Tuples of the reviews of The Green Mile (red slim markers) and Patriot Games (blue thick markers) are shown. The diamonds are the respective medoids of the cluster if we set $k = 1$ for k-medoids, most information systems use a single representation for every item. The crosses are the medoids if we set $k = 3$, representing the different perspectives of the item. Items are traditionally represented with one tuple (diamonds), while multiple Shared Perspectives is the proposed way (crosses). As we can see, the crosses on the top right are respectively closer to each other than the pair of diamonds. By drawing a line from the origin point in (0, 0) to each respective marker, we can observe the angles that are used to compute cosine similarity, like shown in Figure 5. The cosine similarity between the diamonds is 0.9986, and 0.99999988 between the crosses. For the Green Mile and Patriot Games the tuples of those crosses were extracted from *"As always a super good movie. We watch this one every few years. Acting is superb by all in this picture with a really good story. The book is really good too and the movie follows it quite well."* and *"Very good movie version of Tom Clancy's book. I still think Harrison Ford is the best Jack Ryan. Really good plot and good acting. I highly recommend this one."*, respectively. This shows that for two movies, that may not be recognized as related by traditional system, there is a relationship of "recommendable movies from a book, with a good story and good acting". *Note:* Singular Value Decomposition was applied to reduce from 100 dimensions to 2, this is only for illustration purposes, and explains the high cosine similarities.

In summary, the prototype consists of the following:

1. Amazon reviews for 2041 movies
2. Extraction methods. Explicit: Aspect-based Review Extraction (AbRE) Implicit: Document Embeddings (Doc2vec)
3. Store as tuples in RDB in SQL Server
4. Group Shared Perspectives using HDBSCAN for AbRE and Spherical k-medoids for Doc2vec. Store Shared Perspective Tuples in RDB.
5. Iterative QBE implementation in Python using the SPTs from Doc2vec and k-medoids.

5. EVALUATION: USER SIMULATION

Given that we have a functional prototype, in this section we propose and execute a simulation of the process of searching for a movie using Query by Example. This is an evaluation comparing the performance of using SPs against the status-quo of a single representative tuple per item, by measuring the iterations from starting to target item.

The main goal of this work is to improve the process of searching experience items, based on the perception of the item. Since QBE is a user-focused process, we assume that the the process is completed when the user finds an item. As a result, the selected performance metric is the number of iterations or steps required from the first example provided, i_0, until the target is in the resulting display, d. We choose this measure because the selection of a new example is the only action that the interested user will perform during the query process.

Evaluating the performance of QBE when using the proposed Shared Perspectives is a challenge for several reasons:

- The reviews as a source of information are inherently subjective, so the interested user may not agree with the perspectives that represent a selected movie in the application
- The extraction process selected for the prototype is not close enough to a full human understanding This means that sometimes the perceptions grouped together may have similarities not relevant to the process
- User behaviour may differ from a system evaluation, users can decide to keep exploring options or change the initial type of movie they had in mind throughout the process
- The target movie is implicit: "I'll know it when I see it", as discussed in Section 3.2, therefore it is not possible to assert if the user actually reached the goal. There may even be multiple items satisfactory for the user

One option for evaluation, where some of the individual differences could be mitigated, would be to conduct a large user study. However, this would not eliminate the problems of implicit targets or unpredictable user behavior. This would be better for a satisfaction evaluation, comparing the proposed approach to current systems and asking users for their opinion. Since it does not allow to evaluate the number of iterations from start to target, this study is not pursued. Instead, we propose to simulate the query process of a user, with a set of predefined pairs of start and target movies. The simulation selects the new example movie from a display, consequently moving to the next display, until it reached the predefined target movie. The number of steps from start to target is tracked, as it is the evaluation measure. This will allow to compare the performance of a single representative tuple against multiple representative tuples per item. By simulating user behavior, there is no influence from the users' perception on the movies, and there is no subjectivity of interpretation of reviews. In addition, the target movie in a pair is known from the beginning and maintained throughout the process, and the selection of the next movie is based on the actual distance between representations.

In the experiment we compare the use of Three Shared Perspectives per item, as described in the prototype, against a Single Representation (SR). For this single representation, we use the same Perceptual Tuples computed using Doc2vec with the Amazon reviews as for Shared Perspective. In the clustering step, we set $k = 1$ and apply k-medoids to every item, effectively forcing a single cluster, and a single representative tuple.

The experiment simulates the Query by Example process shown in Section 3.2, and the displays of movies are generated the same way as in the prototype system [1] [43]. The graphic interface is not used, since it is irrelevant for the simulation. After the display is generated, the simulation selects the next 'example' movie, i_1, by the similarity of its own SPTs to the SPTs of the target. The goal is to calculate and compare how many steps are needed from a starting movie to a target movie, to prove that by using multiple item representations, the simulation (and therefore a user) would find their goal item in less iterations.

A difference to consider between the predefined pairs and real user behavior is that when a user selects a starting movie, the target should be roughly similar, while for the experiment it is completely random. We expect that SPs should outperform SR, also when the pairs are unrelated, since it can 'get out' of one type of movies to get to the target movie. For seemingly unrelated movie pairs, for instance from The Green Mile to Tinkerbell, there may be a perspective that relates them. For example, The Green Mile has the "good story that follows the book" perspective, which leads to a display that includes Matilda, which has the perspective "beautiful family movie with a message", and the next display contains Tinkerbell. In contrast, for the case of the SR, it takes several iterations to get out of the "must-watch emotional drama" movies to reach kids movies. In conclusion, for random pairs as well as in reality, the average number of steps should be smaller when using multiple representations.

We want to evaluate the process of searching for Experience Items, so we use the number of iterations from start to target items, because selecting a new example item is the only user input. However, this process is full of uncertainties and unpredictable behavior, making its evaluation difficult. As a result, we propose a simulation with predefined start and target items. We compare performance of using *Three Shared Perspectives* against a *Single Representation* per item. We calculate this by using k=3 and k=1 in the k-medoids algorithm, respectively.

5.1 Simulation Set-up

The basic steps for both simulating user behavior with Single Representation and Shared Perspectives are:

1. The starting movie is now called selected movie, the current example
2. System generates display d with 9 most similar movies based on selected movie
3. If target movie is in display, finish. If not, select the best option as new example movie from display
4. Discard movies in display for the rest of the process
5. Repeat 2-4 until target movie is in display

The two simulations have a few differences, as described below:

Single Representation Simulation

- Step 2: The display is created with the 9 movies that have the Representative Tuples (RT) with highest cosine similarity to the RT of the selected movie.
- Step 3: The most similar movie from the display to the target is the one with the RT with the highest cosine similarity to the one of the target movie.

Three Shared Perspectives Simulation

- Step 2: The display is created with three sets of three movies. Each set stands for each one of the three SPs of the selected movie. For example, for Shared Perspective A, the three movies that have a SPT with highest cosine similarity.
- Step 3: The most similar movie from the display to the target is the movie with any one SPT with the highest cosine similarity to any of the three SPTs of the target. This means for a display of nine movies, with three SPTs each, and three SPTs of target, 81 values are calculated and the best one is chosen.

We compare the number of iterations of the proposed *Shared Perspectives* against a *Single Representation* of the item. Single Representation, or one tuple per item, is how current systems manage the perceptual tuples. The simulation basically performs the same process as the prototype, without the graphical interface. It executes the *Query by Example*, then selects the most similar item to the target from the display as the new example.

5.2 Results

The average steps it takes from start to target movie for 25, 50 and 100 different pairs of movies is shown in Table 2. There is an improvement in the performance with Shared Perspectives, as it takes 30% less steps to the target, as with a Single Representation. The frequency distribution of steps for the experiment with 100 pairs is shown in Figure 6. This graph shows that by using SPs, around 80% of pairs reach the target movie in less than 50 steps, compared to 65% with SR. In addition, SR has a high percentage of simulations that take over a hundred steps, this may be for those movies pairs that seem unrelated, like the Green Mile and Tinkerbell. On the other hand, it shows that one pair processed with SPs required 201 steps, while the maximum for SR is 152, affecting the average; this particular case is reviewed later.

Table 2: Average steps with 25, 50 and 100 pairs

	Single Representation	Shared Perspectives
25	37.12	24.28
50	33.28	21.84
100	40.58	32.54

The resulting average number of steps is high compared to other research that uses a similar simulation for evaluation [1] [4], where the results were between 10 and 17 steps. This is mainly because, for this thesis, the display is constructed in a naive way, considering only the selected movie. In the similar research, *informative* strategies in the generation of the display proved to significantly improve results. These strategies aim to maximize the information gain in every step. The display generation in related research uses Bayesian probability, therefore informative strategies mean showing movies that would greatly affect the probability towards a certain type of movie (the target movie). We could also also achieve better results by not just considering the selected movie but also the target movie's representation when creating the display. However, as discussed in the related work, this can produce confusing results, for example showing Finding Nemo in the display of The Terminator because it is more informative for the system [1].

As a qualitative analysis of the results, we can review an example of the iterations followed by the simulation using Shared Perspectives, from start to target. The process from The Back-up Plan, a romantic comedy, to In Time, a sci-fi action thriller, takes only two iterations. The movie that is selected from the first display is Life as We Know It, because it is "definitely a cute movie, somewhat predictable", and the movie selected from the second display is Definitely Maybe, for being a "cute and enjoyable chick-flick, even if you're not a girl". In the third display, created for Definitely Maybe, the target movie In Time appears for being a movie with "nice story and surprisingly good acting, a refreshing change for the genre".

Another example is the procees from The Verdict, a lawyer drama, to Fantasia, an animated Disney movie with classical music, which takes ten iterations. It goes through movies like Zulu, for its "acting and power", to Real Steel, for being "a much deeper film than expected", to The Starfighter, for being "awesome sci-fi for the whole family", finally leading to a display with "movies from my time that my kids simply love". The simulation also showed unexpected bad examples, from the classic horror thriller, House on Haunted Hill, to the musical drama, Rent, the simulation takes 96 steps. After the 4th display, it goes from Jumanji to Toy Story 2 because the users "have seen it so many times", leading the simulation into many similar steps. It eventually reaches other theater plays, that users have also seen many times, eventually leading to the target movie. The special case with 201 steps shown in Figure 6, it the movie pair Transamerica to Intolerable Cruelty. All three perspectives of the target movie somewhat state that it "seemed good, but failed to deliver", and they lack any specific features. The simulation algorithm selects many perspectives that mention either George Clooney or Catherine Zeta Jones, as they are the only outstanding keywords. For this analysis, it seems that the users' perception movie is so bland that the simulation struggles to reach it.

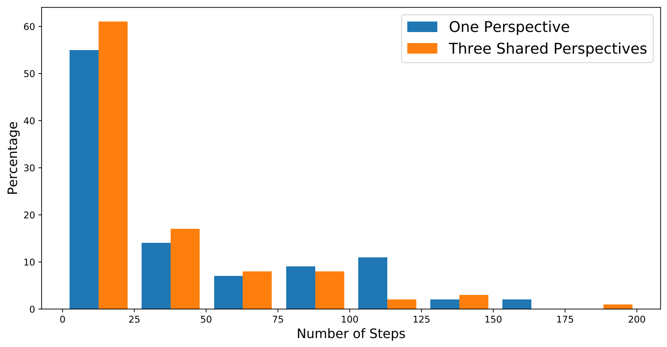


Figure 6: Histogram of Steps with 100 pairs

In previous work using reviews as data source, it was also discovered that some of them are not very useful to relate movies, for example, they found that two movies were related because of grainy video [4]. These reviews seem to appear across all types of movies and do not represent a useful relationship for effectively using QBE. A very common one, includes people writing about video quality: they "bought the movie on Blu-ray", or "the improvement from VHS to DVD is great". Another perspective that appears frequently is the vendor or physical format: "the DVD was scratched", "the delivery was a week late", etc. We conduct a user study to examine the difference between useful and useless perspectives. For example for a movie, the text of one Shared Perspective, and the most similar movies:

- Useful: Hercules → Animated classic from childhood → Aladdin, Lion King, Beauty and the Beast

- Not useful: Hercules → Had it in VHS, bought Blu-ray → Terminator, American Pie, Hook

The simulation could select the next example movie because of a not useful perspective, which may send the process into a repetitive direction. Therefore we think that the performance can be affected for a simulation that follows the most similar element without an understanding of what is a 'good' similarity. We ask participants to rate the Usefulness of a Perspective, in other words its Semantic Quality and use it to improve the process.

Using three *Shared Perspectives* requires, on average, 30% fewer steps than the *Single Representation* to reach the target item. Qualitative analysis shows that some Perspectives are more *useful* than others to relate movie A to B, so we explore this *Semantic Quality* in a user study.

6. EVALUATION: SEMANTIC QUALITY OF SHARED PERSPECTIVES

By observing the process of the simulations using Shared Perspectives, we found that some SPs are not very useful to relate movies to each other, so they have low semantic quality. In this section we analyze the impact of this semantic quality by asking users to rate a few SPs. We model these ratings so we can predict them for the rest of the SPs. This is important since the main use of SPs is to find movies that are perceived similarly.

Authors of previous work discuss that low quality perspectives can be solved with previous data cleaning, or by using topic modelling techniques to separate this type of data [4]. In this case, we inform the user about the Semantic Quality of the SPs, so they can decide if they want to select movies related with a given SP or not. To accomplish this, we first have to discover if there is a difference between good and bad similarity relationships. If there is a noticeable difference between them, the system could be able to model and label this, and show the users a quality rating, for a more useful application. In summary, we need to take the following steps:

1. Ask people to rate usefulness of some perspectives
2. Find a relationship to model usefulness score as a function of available data
3. Predict the scores for the rest of the perspectives

In the user study, a SP is useful if the movies that the prototype selected based on this SP 'make sense' for the participant. In other words, a SP has high Semantic Quality if the participants would use that SP to recommend a movie to a friend. Predicting the Usefulness Score for the rest of the SPs is necessary because there are 2041 movies, so 6123 SPs to rate, which is not feasible for a study of this scale.

Some reviews are not directly related to the experience with the movies, but other elements like the format or the retailer. These form Shared Perspectives that are less *useful* to find movies with similar experience. Therefore we try to quantify and use this as a *'Usefulness Score'*. For this, we need to ask users to rate some SPs, model these ratings using available data, and use the model to predict scores for all unrated perspectives.

6.1 User Study

The purpose of this study is to get some ratings of the quality of Shared Perspectives and use them to predict the Quality of the rest of the SPs. To obtain ratings of usefulness, we asked seven participants to rate if a SP is useful or not to relate a selected movie to the set of three similar movies. The participants include people from 18 to 55 years old, from different countries and education levels. The setup of the experiment is the following:

- The study was a self contained application that stores the ratings provided
- A digital version of the application was sent to the participants
- The interface showed one selected movie, with a display of nine similar movies, one set of three for each of the three SPs. The experiment looks similar to the prototype, and an example of the interface is provided in Appendix A.
- Instructions for the participants were:
  - Rate if you agree that the perspective text that relates the movies in the set to the selected movie is useful (Totally Agree, Somewhat Agree, Don't Agree)
  - This is, if at least two of the three movies in the set are related to the example, taking in account the text of the perspective
  - Example of useful relationship: Disney classic from childhood
  - Example of not useful relationship: It looks better in blu-ray than VHS

The results obtained are between five and seven ratings, for seven starting movies, therefore we examine 21 perspectives. The values can be encoded as a Likert Scale: Totally Agree = 2, Somewhat Agree = 1, Don't Agree = 0. To determine the *usefulness score* for every perspective, we average the numerical values obtained. While this may not be appropriate for hypothesis testing with statistical significance, averaging a Likert Scale is sufficient for exploration. It is helpful in this case, to have a general sense of the 'usefulness score' of every perspective. For example My Dog Skip related to P.S. I Love You, The Good Life of Timothy Green and Bee Movie because "My wife and I loved this movie. A heart warming story" has a score of 1.5. In contrast The Good, The Bad and The Ugly related to The Towering Inferno, Shane and The Uninvited because "The digital restoration looks really great", received an average of 0.5.

As a small scale user study, there are many threats to validity, so it is important to underline that this is just an exploratory experiment. The main deficiencies of this study are:

- Not enough response to be an appropriate statistical sample
- Only movies with reviews under 500 characters were selected, because participants reported that long reviews made the process too cumbersome
- The extraction methods are not perfect, and the interpretation of reviews is different across participants
- The experiments were not supervised, participants performed it remotely

This study is enough to provide with information for analysis on the quality of the Perspectives as a relationship between movies. These results can be used for an optimization of the simulation, as well as a useful addition to what would be a user-oriented application.

In the study we ask users to rate on a scale from 0 to 2, if they agree that a perspective is useful. We average the ratings to obtain a single *Usefulness Score* for every SP. It is a small scale study but it is appropriate and useful for exploration purposes.

6.2 Predicting Usefulness Scores

We compare the scores obtained with the available data of the SPs, to model them as a function and use it to predict the scores of the rest of SPs. The modelling depends in the extraction method used, the distance metrics and the clustering algorithm. The values that we have for a given SP consist of its tuple, SPT_{00}, and the rest of available SPTs in the relation, with their respective cosine similarity. For every SPT, 6120 cosine similarities are calculated, and ordered from high to low. For example, for SPT_{00} of The Good, the Bad and the Ugly the most similar SPT has a cosine similarity of 0.624, the next one 0.622, until SPT 6120 with a similarity of -0.01.

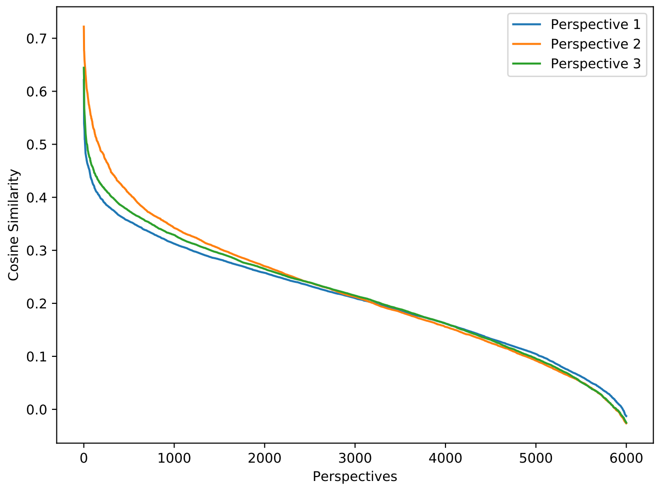


Figure 7: Distribution of Cosine Similarity Values for The Good, the Bad and the Ugly

Using the average cosine similarity, median, or other aggregates may not be adequate because they are dependent on how unique the SPT is. Therefore we decided to focus on distribution metrics. The idea is that for useful SPTs, the behavior of its distribution may be different from the behavior of not useful ones. Figure 7 shows the cosine similarity values, in descending order, for each of the Perspectives of The Good, the Bad and the Ugly. The respective Usefulness Scores for each Shared Perspective is 1.4, 0.6 and 0.8. The figure shows that large difference in distribution behavior is more evident with a smaller number of perspectives, on the left side of the figure. We observed a similar trend in distribution graphs of many other movies. To compare these differences in distribution, we use the Pearson Median Skewness (PMS), also called Second Skewness Coefficient. PMS is unaffected by scale, and we can use it to compare all SPTs. It is defined as $(3 * (mean - median))/standarddeviation$. Preliminary analysis showed that the Perspectives with high Usefulness Score have a small difference between the mean and the median, while the opposite is true for Perspectives with bad scores.

We calculate the PMS and compare it to the Usefulness Score provided by the user study for each Perspective. To find the best result, we calculated the correlation of the Usefulness Score with the PMS for many different sets of Perspectives, ranging from 5 to 5000. By computing the PMS of the cosine similarities of the 105 most similar Perspectives, the correlation with the Usefulness Scores is -0.72, meaning that higher PMS leads to lower Semantic Quality. Figure 8 shows the PMS of the first 105 cosine similarities in x vs the Usefulness Score in y, where every point is one Shared Perspective from the User Study.

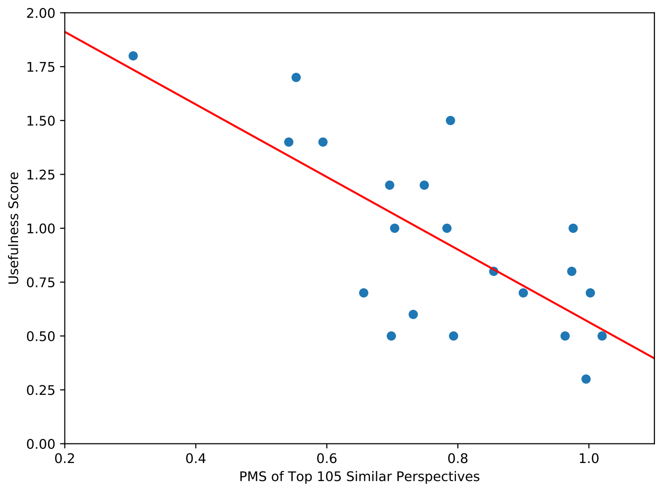


Figure 8: Usefulness Score vs. Pearson Median Skewness of Top 105 Most Similar Perspectives

We used linear regression to fit a line to the data points in Figure 8 with an R-squared of 0.52. This value for R-squared is acceptable for exploration of user behavior, especially with such a simple model. Using second or third degree polynomial regression did not improve the performance considerably, therefore we decide to keep the model as simple as possible with a linear function. By modelling this function, we can use it to predict the scores of all Shared Perspectives and optimize the similarity calculations of the simulation.

To predict the scores of all unrated *Shared Perspectives*, we need to find a relationship between the *Usefulness Scores* obtained from the study and the data available. This data consists mostly of the values of the cosine similarities between the selected perspective and the rest. We found the Pearson Median Skewness of the 105 most similar perspectives has the highest (negative) correlation. We model this as a function to obtain the scores of all perspectives.

6.3 Results

To evaluate the impact of the predicted Usefulness Scores in the QBE process, we run the same simulation as in Section 5, but with the Usefulness Score added to the cosine similarities when selecting the new example movie from the display. This will give an advantage to useful perspectives over their less useful counterparts. For this implementation, the optimization yields the best results when the calculated Usefulness Score is divided by 15 and added to the cosine similarity in the selection step.

As shown in Table 3, an improvement is observed when using the Usefulness Score optimization in the simulation. More importantly, it showed reduction of the number of steps for pairs with extreme values, for example from 82 to 41 steps. It should be noted that in rare cases the optimization caused an increase of steps, however, this was only

Table 3: Average steps with 25, 50 and 100 pairs

	One SP	Three SP	Three SP + Usefulness
25	37.12	24.28	22.96
50	33.28	21.84	18.7
100	40.58	32.54	31.56

in low numbers of steps, like from four to seven. The main intention of evaluating the impact of the Usefulness Score for the simulation was to reduce the numbers of steps for extreme cases. While the average result is not extraordinary, this could really benefit the actual user by showing them the predicted Usefulness Scores, which is the main intention of the usefulness scores overall. As mentioned before, if the intention was only to obtain better results for the simulation, the generation of the display would be the first section to improve. By focusing on the Usefulness Score of Perspectives instead, users have more information for better decisions. The prediction of the scores can be improved, starting with a bigger dataset (large scale user study) and a more complex model.

We use the calculated *Usefulness Scores* to optimize the simulation when selecting the new example item. This is mainly as an improvement for the user, however, we also execute an evaluation with the same simulation described in Section 5. While there is a slight improvement in average steps from start to target item, the important result is in the reduction of number of steps for movie pairs with extreme values. This optimization only works in this particular implementation, but it helps to show that we can improve performance by focusing on the Shared Perspectives instead of the original data, as proposed in previous research [4].

6.4 Exploration and Analysis

The evaluation of the extraction methods used, in particular Doc2vec, was not in the focus of this work, but it can be used for further exploration of the results obtained. For this, we asked participants if each one of the three reviews that were selected as similar by the prototype is similar to the respective Perspective. By checking a box for each review, participants agree that it is similar to their respective Perspective text, by not checking it, they consider that the text is not similar. This can be seen in Appendix A. The boxes were unchecked by default, and the study was not supervised, so there is a risk that participants forgot to answer. Participants rated 26 reviews, on average 0.4 of the reviews were perceived as similar. For 13 reviews the results were consistent across all participants: eight were considered as not related and five as related. Comparing to the Perspectives reviewed for the study, in nine out of 21, at least two of the three reviews in a set have an average rating over 0.5, and in seven of those, the respective 'Usefulness Score' is over 1. With a large scale user study, this could be used in future work to improve the model created to automatically rate the usefulness of perspectives.

As an interesting addition, we asked participants to label these automatically selected perspectives with a few words, as a mini-crowdsourcing experiment. In the user study interface, we provided an empty text box for each of the three SPs of the selected movie. The results emphasize how different participants interpreted the perspectives. For instance, for The Green Mile, the perspective about a "movie with great acting and engaging story, that follows the book quite well", three participants wrote labels mentioning acting, two wrote about the story and only one mentioned the book. On the other hand, there is agreement in some cases, for Jungle Book, all participants agree that the second perspective is "Disney classics", or the first perspective for The Good, the Bad and the Ugly is a "great western with very good actors". With a higher participation, this could definitely be used to discover a few keywords to describe the Perspectives instead of the medoid. It would be an improvement particularly when perspectives describe the story or the cast: a person can easily interpret that the perspective is describing the plot in a positive way and just state 'good plot', while a system would struggle to reach this conclusion.

In user study, we asked participants if the selected reviews were actually similar to the *SP* of the selected movie. This confirmed the uncertainty we tried to avoid with the simulation, where participants not always agree if *SP*s are related or not We also asked if they could label those selected reviews by their shared characteristics. It shoed that they focus on completely different characteristics for the same sets of related movies.

7. CONCLUSION

In this work we introduced a novel way of extracting perceptions from user feedback and grouping them into multiple representations for an experience item, called Shared Perspectives, that can be applied to a QBE process. Using reviews adds perceptual features to the current metadata for those items, which includes technical, bibliographic, organizational, and content-based information. Clustering similar perceptions covers the opinions of majority groups, while respecting some subjectivity. To ground the concepts of SPs, a prototype of QBE with Shared Perspectives was implemented. We used movie reviews, with AbRE and Doc2vec as explicit and implicit feature extraction methods. With a clustering algorithm, such as HDBSCAN or k-medoids, similar perceptions are grouped. We present a basic extension of the Relational Model to store values for Perceptual Features, Tuples and Shared Perspectives. In a synthetic experiment, the use of Shared Perspectives showed improved results in query performance against Single Representation per item. To automatically rate the usefulness of a perspective (how useful it is to relate movie A to B), we modelled a basic heuristic with semantic information obtained from a small user study. The modelled heuristic is for this particular implementation and can be improved, mainly with more user ratings and more values. The user study also provided some interesting insights on the subjectivity of reviews. There are other implementation choices to be made in addition to the concepts presented, for instance: how to manage new Perceptual Tuples, recalculate Shared Perspectives, using improved extraction methods, the use of centroids instead of medoids, the use of both regular metadata with SPTs to calculate displays, whether or not showing Usefulness Score or applying this Score to order the elements in the display. However, the core concepts remain: obtaining a Perceptual Tuple per feedback document, grouping similar ones into Shared Perspectives, and using the Shared Perspective Tuples to represent the items in QBE.

7.1 Future Work and Discussion

Together with the research, the idea was to create a useful application that can used by people. This involves many possibilities for improvement of the idea and implementation. Some of the most pressing points are:

- Optimization of Extraction Methods: Tune parameters, use larger corpus, evaluate encoding and similarity
- Implementation and Scaling: How to add new reviews to clusters, and therefore how to recalculate Shared Perspectives
- Evaluation: User Satisfaction compared to the current information systems, apply reinforcement of useful perspectives, use click data as feedback for good and bad perspectives

The implementation will be continued and improved as a personal project, hopefully resulting in a functional, standalone application that can be used for movie enthusiasts. The intention is to also create a code repository in the form of a tutorial, available for people interested in NLP, Perceptual Feature-based search, recommendation, and databases. This is in part for own interest and in part for the response obtained in the user study. After completing the required instructions, participants continued to interact with the prototype. There was positive feedback and comments about the general idea, the resulting movies, and their perspectives.

The prototype and all examples were with movies, but this general process should perform appropriately in different domains as long as there are reviews, and the items are properly categorized. The process was implemented using reviews for the 'Videogames' category of Amazon, with similar qualitative results to movies. For example perspectives of "bought it for my son and he loves playing it", or "the story is great as long as you have played the previous one". In contrast, when applied to the 'Pets' category, the behavior was more erratic because the category is too general, there is food, toys, medicine, clothing, etc. Without proper sub-categories, it is difficult to find related products.

For other applications, an important consideration is that Shared Perspectives can be used by the system owners instead of the users, for example in Business or Political Intelligence applications [44] [45]. This can lead to better feedback for the creators of content (i.e. movie or video game production companies) so they know how their items are perceived by major groups. In addition, it can be used by the online platform itself (i.e. Amazon), to differentiate their customers for better service. For platform owners and what we considered not useful, such as delivery and packaging, may be of actual interest [5].

8. REFERENCES

[1] Christoph Lofi and Christian Nieke. *Exploiting Perceptual Similarity: Privacy-Preserving Cooperative Query Personalization.* Springer International Publishing, 2014.

[2] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '04*, 04:168, 2004.

[3] James Huang, Stephanie Rogers, and Eunkwang Joo. Improving Restaurants by Extracting Subtopics from Yelp Reviews. *iConference 2014 Berlin*, pages 1–5, 2014.

[4] Christoph Lofi and Philipp Wille. Exploiting social judgements in big data analytics. *CEUR Workshop Proceedings*, 1458:444–455, 2015.

[5] John Prager. Open-Domain QuestionAnswering. *Foundations and Trends® in Information Retrieval*, 1(2):91–231, 2006.

[6] Gobinda G. Chowdhury. Natural language processing. *Annual Review of Information Science and Technology*, 37(1):51–89, 2005.

[7] Tun Thura Thet, Jin Cheon Na, and Christopher S.G. Khoo. Aspect-based sentiment analysis of movie reviews on discussion boards. *Journal of Information Science*, 36(6):823–848, 2010.

[8] Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. SentiWordNet 3.0 : An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining SentiWordNet. *Analysis*, 10:1–12, 2010.

[9] Mengmeng Ye. *Aspect-Based Review Extraction for E-Commerce Products.* Masters, T. U. Delft, Delft, Netherlands, 2017.

[10] Andrea Morandi, Marceau Limousin, Jack Sayers, Sunil R. Golwala, Nicole G. Czakon, Elena Pierpaoli, Eric Jullo, Johan Richard, and Silvia Ameglio. X-ray, lensing and Sunyaev Zel'dovich triaxial analysis of Abell 1835 out to R_{200}. *CrossRef Listing of Deleted DOIs*, 1:993–1022, nov 2011.

[11] Quoc V. Le and Tomas Mikolov. Distributed Representations of Sentences and Documents. In *31st International Conference on Machine Learning*, volume 32, pages 1188–1196, Beijing, China, 2014.

[12] Susan T. Dumais. Latent semantic analysis. *Annual Review of Information Science and Technology*, 38(1):188–230, 2005.

[13] Joachim Selke, Christoph Lofi, and Wolf-Tilo Balke. Pushing the boundaries of crowd-enabled databases with query-driven schema expansion. *Proceedings of the VLDB Endowment*, 5(6):538–549, 2012.

[14] Toine Bogers and Antal Van Den Bosch. Collaborative and content-based filtering for item recommendation on social bookmarking websites. *Submitted to CIKM*, 9:9–16, feb 2009.

[15] Tariq Mahmood and Francesco Ricci. Towards learning user-adaptive state models in a conversational recommender system. *LWA*, 7:373–378, 2007.

[16] Ofer Arazy, Nanda Kumar, and Bracha Shapira. Improving social recommender systems. *IT Professional*, 11(4):38–44, 2009.

[17] Dunja Mladenić. Text learning and related intelligent agents: a survey. *IEEE Intelligent Systems*, 14(4):44–54, 1999.

[18] Myron Flickner, Harpreet Sawhney, Wayne Niblack, Jonathan Ashley, Qian Huang, Byron Dom, Monika Gorkani, Jim Hafner, Denis Lee, Dragutin Petkovic, David Steele, and Peter Yanker. Query by Image and Video Content: The QBIC System. *Computer*, 28(9):23–32, 1995.

[19] Kyoji Hirata and Kato Toshikazu. Query by visual

example. *International Conference on Extending Database Technology*, pages 56–71, 1992.

[20] Beverley A. Sparks, Helen E. Perkins, and Ralf Buckley. Online travel reviews as persuasive communication: The effects of content type, source, and certification logos on consumer behavior. *Tourism Management*, 39:1–9, 2013.

[21] Ana-Maria Popescu and Oren Etzioni. Extracting Product Features and Opinion from Reviews. In *Natural language processing and text mining*, pages 9–28. Springer, London, 2007.

[22] Yohan Jo and Alice H. Oh. Aspect and sentiment unification model for online review analysis. *Proceedings of the fourth ACM international conference on Web search and data mining - WSDM '11*, page 815, 2011.

[23] Ruining He and Julian McAuley. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. In *International Conference on World Wide Web (WWW '16)*, pages 507–517, 2016.

[24] Rafal A. Angryk and Jacek Czerniak. Heuristic algorithm for interpretation of multi-valued attributes in similarity-based fuzzy relational databases. *International Journal of Approximate Reasoning*, 51(8):895–911, 2010.

[25] V. K. Singh, R. Piryani, A. Uddin, and P. Waila. Sentiment analysis of movie reviews: A new feature-based heuristic for aspect-level sentiment classification. In *Proceedings - 2013 IEEE International Multi Conference on Automation, Computing, Control, Communication and Compressed Sensing, iMac4s 2013*, volume 361, pages 712–717. IEEE, mar 2013.

[26] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - EMNLP '02*, 10(July):79–86, 2002.

[27] Peter D. Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, pages 417–424, 2002.

[28] Sonia Bergamaschi and Laura Po. Comparing LDA and LSA topic models for content-based movie recommendation systems. *Lecture Notes in Business Information Processing*, 226:247–263, 2015.

[29] Jey Han Lau and Timothy Baldwin. An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation. *CoRR*, abs/1607.0:78–86, 2016.

[30] Sam Lightstone, Toby Teorey, Tom Nadeau, and H.V. Jagadisk. *Database Modeling and Design: Logical Design*. Morgan Kaufmann Publishers Inc., 5th edition, 2011.

[31] Carlo Batini, Stefano Ceri, and Shamkant B Navathe. *Conceptual database design*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1992.

[32] Jan Paredaens, Paul de Bra, Mark Gyssens, and Dirk van Gucht. *The Structure of the Relational Database Model*. Springer-Verlag New York, Inc., New York, NY, USA, 1989.

[33] Duc Thang Nguyen, Lihui Chen, and Chee Keong Chan. Multi-viewpoint Based Similarity Measure and Optimality Criteria for Document Clustering. *Information Retrieval Technology*, pages 49–60, 2010.

[34] Joydeep Ghosh and Alexander Strehl. Similarity-based text clustering: A comparative study. In *Grouping Multidimensional Data: Recent Advances in Clustering*, pages 73–97. Springer-Verlag, Berlin/Heidelberg, 2006.

[35] Ricardo J. G. B. Campello, Davoud Moulavi, Arthur Zimek, and Jörg Sander. Hierarchical Density Estimates for Data Clustering, Visualization, and Outlier Detection. *ACM Transactions on Knowledge Discovery from Data*, 10(1):1–51, 2015.

[36] Shie Mannor, Xin Jin, Jiawei Han, Xin Jin, Jiawei Han, Xin Jin, Jiawei Han, and Xinhua Zhang. K-Medoids Clustering. *Encyclopedia of Machine Learning*, pages 564–565, 2011.

[37] Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. Probabilistic Databases. In *Synthesis Lectures on Data Management*, volume 3, pages 1–180. Morgan & Claypool Publishers, 2011.

[38] Christoper Koch. MayBMS A System for Managing Large Probabilistic Databases. *Managing and Mining Uncertain Data SE - 6*, 35:1–34, 2009.

[39] Lance Parsons, Ehtesham Haque, and Huan Liu. Subspace clustering for high dimensional data. *ACM SIGKDD Explorations Newsletter*, 6(1):90–105, 2004.

[40] Emmanuel Müller, Stephan Günnemann, Ira Assent, and Thomas Seidl. Evaluating clustering in subspace projections of high dimensional data. *Proceedings of the VLDB Endowment*, 2(1):1270–1281, 2009.

[41] Nastaran Borjian. Query-by-example music information retrieval by score-based genre prediction and similarity measure. *International Journal of Multimedia Information Retrieval*, 6(2):155–166, 2017.

[42] Brian McFee, Luke Barrington, and Gert Lanckriet. Learning Similarity from Collaborative Filters. In *International Society of Music Information Retrieval Conference*, pages 345 – 350, 2010.

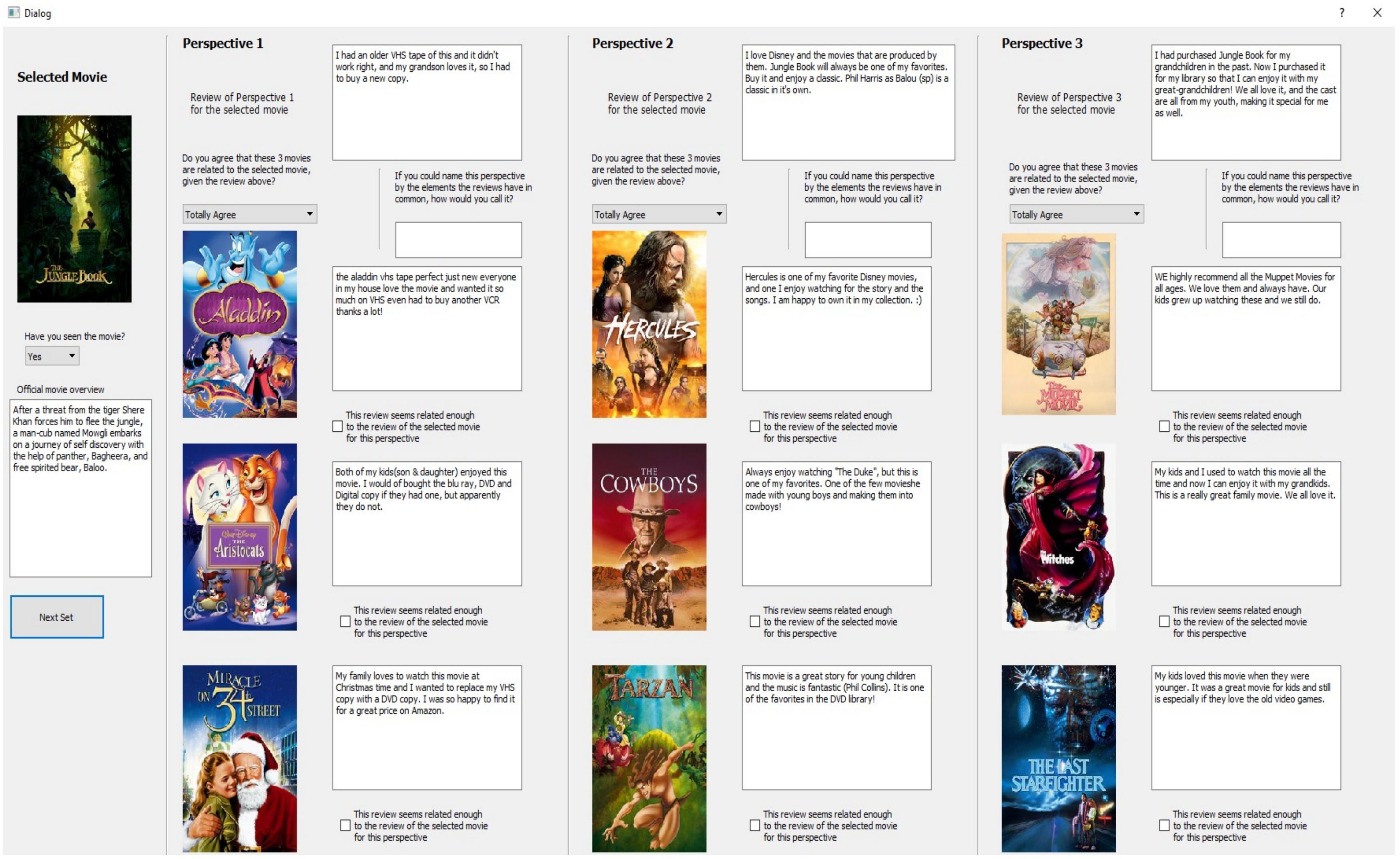
[43] Christoph Lofi, Christian Nieke, and Wolf Tilo Balke. Mobile product browsing using Bayesian retrieval. *Proceedings - 12th IEEE International Conference on Commerce and Enterprise Computing, CEC 2010*, pages 96–103, 2010.

[44] Michael Laver, Kenneth Benoit, and John Garry. Extracting policy positions from political texts using words as data. *American Political Science Review*, 97(2):311–331, 2003.

[45] Robert Malouf and Tony Mullen. Taking sides: user classification for informal online political discourse. *Internet Research*, 18(2):177–190, 2008.

APPENDIX

A. EXAMPLE OF USER STUDY


Dialog
Selected Movie
Have you seen the movie?
Yes
Official movie overview
Next Set
Perspective 1
Review of Perspective 1 for the selected movie
Do you agree that these 3 movies are related to the selected movie, given the review above?
Totally Agree
If you could name this perspective by the elements the reviews have in common, how would you call it?
This review seems related enough to the review of the selected movie for this perspective
Perspective 2
Review of Perspective 2 for the selected movie
Perspective 3
Review of Perspective 3 for the selected movie

After a threat from the tiger Shere Khan forces him to flee the jungle, a man-cub named Mowgli embarks on a journey of self discovery with the help of panther, Bagheera, and free spirited bear, Baloo.

Perspective 1

I had an older VHS tape of this and it didn't work right, and my grandson loves it, so I had to buy a new copy.

the aladdin vhs tape perfect just new everyone in my house love the movie and wanted it so much on VHS even had to buy another VCR thanks a lot!

Both of my kids(son & daughter) enjoyed this movie. I would of bought the blu ray, DVD and Digital copy if they had one, but apparently they do not.

My family loves to watch this movie at Christmas time and I wanted to replace my VHS copy with a DVD copy. I was so happy to find it for a great price on Amazon.

Perspective 2

I love Disney and the movies that are produced by them. Jungle Book will always be one of my favorites. Buy it and enjoy a classic. Phil Harris as Balou (sp) is a classic in it's own.

Hercules is one of my favorite Disney movies, and one I enjoy watching for the story and the songs. I am happy to own it in my collection. :)

Always enjoy watching "The Duke", but this is one of my favorites. One of the few movieshe made with young boys and making them into cowboys!

This movie is a great story for young children and the music is fantastic (Phil Collins). It is one of the favorites in the DVD library!

Perspective 3

I had purchased Jungle Book for my grandchildren in the past. Now I purchased it for my library so that I can enjoy it with my great-grandchildren! We all love it, and the cast are all from my youth, making it special for me as well.

WE highly recommend all the Muppet Movies for all ages. We love them and always have. Our kids grew up watching these and we still do.

My kids and I used to watch this movie all the time and now I can enjoy it with my grandkids. This is a really great family movie. We all love it.

My kids loved this movie when they were younger. It was a great movie for kids and still is especially if they love the old video games.

Figure 9: User Study Example Display