T. Grust et al. (Hrsg.): Datenbanksysteme für Business, Technologie und Web (BTW 2019),
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2019 169

Perceptual Relational Attributes: Navigating and Discovering Shared Perspectives from User-Generated Reviews

Manuel Valle Torre[1] Mengmeng Ye Christoph Lofi[2]

Abstract: Effectively modelling and querying experience items like movies, books, or games in databases is challenging because these items are better described by their resulting user experience or perceived properties than by factual attributes. However, such information is often subjective, disputed, or unclear. Thus, social judgments like comments, reviews, discussions, or ratings have become a ubiquitous component of most Web applications dealing with such items, especially in the e-commerce domain. However, they usually do not play major role in the query process, and are typically just shown to the user. In this paper, we will discuss how to use unstructured user reviews to build a structured semantic representation of database items such that these perceptual attributes are (at least implicitly) represented and usable for navigational queries. Especially, we argue that a central challenge when extracting perceptual attributes from social judgments is respecting the subjectivity of expressed opinions. We claim that no representation consisting of only a single tuple will be sufficient. Instead, such systems should aim at discovering *shared perspectives*, representing dominant perceptions and opinions, and exploiting those perspectives for query processing.

Keywords: Perceptual Attributes; Modelling; User-Generated Attribute Values; Query-By-Example Navigation

1 Introduction

Social judgments like comments, reviews, discussions, or ratings have become an ubiquitous component of most Web applications, especially in the e-commerce domain. Now, a central challenge is using these judgments to improve the user experience by offering new query paradigms. Recommender systems have already demonstrated how ratings can be effectively used towards that end, providing users with proactive guidance within large item databases.

In this paper, we will discuss how to use unstructured reviews to build a structured semantic representation of such items, enabling the implementation of user-driven queries. Thus, we address one of the central challenges of Big Data systems: making sense of huge collections of unstructured user feedback. More specifically, we discuss the challenge of building structured, but latent representations of "experience items" stored in a relational database (like movies, books, music, games, but also restaurants or hotels) from unstructured user

[1] TU Delft, Web Information Systems, Van Mourik Broekmanweg 6, 2628 XE Delft, Netherlands; m.valletorre@tudelft.nl
[2] TU Delft, Web Information Systems, Van Mourik Broekmanweg 6, 2628 XE Delft, Netherlands; c.lofi@tudelft.nl

doi:10.18420/btw2019-11

feedback. Such representations should encode the consensual perception of an item from the point of view of a large general user base. Consequently, this information can then be exploited for allowing semantically richer queries. In the following, we will use movies as a use case. However, the described techniques can easily be transferred to any other domain which has user ratings or reviews available.

While there have been previous works also aiming at representing items in a database based on social judgments (e.g., [LW15], [LN14]), we address one major yet unresolved problem: user judgments are inherently subjective as they represent a user's perception. While rating-based systems are widely used, semantic quality quickly deteriorates when richer information sources like reviews are considered; they are less in quantity but richer in content and thus can express a wider variety of opinion. Furthermore, mining reviews is harder due to the complexity of natural language. Here, aspect-oriented sentiment analysis [YLT17] or document-embeddings [LW15] have been used to create structured representations which then later can be used for database query processing. However, the commonly chosen approach of combining the resulting representations into a single tuple (e.g., by averaging the document embeddings) is often not meaningful. Considering an example case of the movie "Twilight" (2008), typical reviews might express that the movie is a "beautiful romance full of alluring characters" or "an overall stupid movie and a disgrace to the vampire genre". Other reviews might even be unrelated to the item itself, e.g. "my DVD was damaged on arrival, bad seller". Clearly, a meaningful representation of the movie should incorporate all its relevant points of views, thus the aggregation of several user judgments has to be performed carefully. This problem is further aggravated by the current trend towards opinion polarization [MTT17]: publicly stated user opinions for example in reviews or comments are increasingly extreme, making it even more relevant than ever to respect different points of view when representing items.

Therefore, in this paper, we propose to represent each experience item in a database using multiple shared perceptual perspectives, with each perspective representing one major consensual opinion aggregated from multiple user judgments. Our contributions are:

- We present a large-scale *modelling experiment* outlining some of the challenges when modelling perceptual attributes
- We present the *foundations* of shared perspectives for experience items
- We provide an overview of the *design space* of different techniques and methods available to obtain and process such perspectives
- We introduce an adapted variant of the *query-by-example paradigm* intended to interact and query multiple shared perspectives
- We present our *prototype implementation* of a system using shared perspectives, and give insights into its query processing performance using simulations

- We conduct and present a *user study*, giving insights into the usefulness and semantic representativeness of the perspectives in our prototype system
- Based on the results of this study, we *identify shortcomings and challenges* with shared perspectives, and propose additional techniques to address some of them

2 Towards Shared Perspectives and Modelling Perceptual Properties

In the precursor work of this paper [LW15], we already explored preliminary concepts and implementations for encoding social judgments for capturing experience items like movies, books, etc. in relational databases. We used an e-commerce scenario where users can browse for experience items. This type of scenario is a prime application for user-generated judgments: user-friendly interaction with experience items is notoriously difficult, as there is an overwhelming number of those items easily available. Some of them are vastly popular and easily accessible mainstream items, but most of them are relatively unknown long tail products which are hard to discover without suitable support. Even more, the subjective user experience those products will entail (which, for most people, is the deciding factor for buying or consuming the product) is difficult to describe by typically available meta-data like production year, actor names, or even rough genre labels. Due to this problem, web services dealing with experience products enthusiastically embraced techniques for motivating the creation of user-generated judgments in the form of ratings, comments or reviews. In its most naïve (but very common) implementation, rating and review data are simply displayed to users without any additional processing (e.g., as seen in most current video streaming or shopping platforms). Querying and discovering items still relies on traditional SQL-style queries and categorizing based on non-perceptual meta-data (e.g., year, actor list, genre label, etc.). Manually reading these user judgments may help potential new customers to decide if they will like or dislike a certain item, but it does not really help them to discover new items beyond their expertise (i.e., manually reading user judgments works fine if a user knows what she is looking for, but has not yet come to a final buying decision). This led to the development of recommender systems [LSY03, BKV10], which proactively predict which items a user would enjoy. Often, this relies on collaborative filtering techniques [LSY03] which exploit a large number of user-item ratings for predicting a user's likely ratings for each yet-unrated item. While collaborative filtering recommender systems have been proven to be effective [KB11], they have only very limited query capabilities (basically, most recommender system offer just a single static query for each user).

2.1 Modelling Perceptual Properties

For enabling semantic queries like similarity exploration or query-by-example queries [LW15, LN14], the first step is to find semantically meaningful representations of database items going beyond simple available structured meta-data. As motivation for this work, we

argue that experience items are generally better characterized by their *perceptual properties*, e.g. their mood, their style, or if there are certain plot elements present – information which is rarely explicitly available and expensive to obtain. Furthermore, from a modelling point of view, it is often unclear which perceived properties describe items well.

Thus, in the following we investigate the question of *"How well can perceptual attributes be modelled manually by (semi-skilled) database schema designers"*. This modelling challenge can be approached in multiple different ways, as in [TNK10] where simply the Oscar Award categories are used (like "cinematography" "music" "costume design") - a design decision which might represent some movies better than others, and might not necessarily represent how the general public would describe (and query) for movies.

For this paper, we conducted a *modelling experiment with 180 second year university BSc students*, and asked them to create a ranked list of perceptual attributes (i.e. attributes describing the consumption experience of a movie). The core task was *"Which attributes should be used to describe movies beyond typical structured meta-data as e.g., available in IMDB[3]."* Also, a brief motivation why they think that these attributes would be relevant should be given. Practical limitations, like the challenge of how to obtain the values for such attributes, or the problem that certain attributes might be subjective, were to be ignored.

Tab. 1: Modelling Experiment Example Response

Attribute	Importance	Explanation
Quality of Acting	5	The quality of the acting in a movie can make a great difference in the overall quality. Believable acting can be a great help to a movies perceived quality.
Originality of Storyline	3	Most big movies seem to have the same storyline with some characters changing names, they are predictable and bore a lot of people. A lot of people are therefore interested in movies with original storylines
Amount of Explosions	2	A large amount of explosions will put off some viewers while others will really appreciate them.
Quality of Scenes	3	Simply put fight scenes can differ greatly in quality from the lameness that is Darth Vader vs Palpatine to the awesomeness that is gypsy danger rocket punching a giant monster in the face. People that watch action movies want to see people get punched in an awesome way, not thrown over a railing for example. This can really make or break these kinds of movies for certain viewers.

The students all possessed basic knowledge of database modelling techniques, and conducted

[3] `http://www.imdb.com`

this task as part of their database education curriculum. Students were teamed into pairs, and performed this modelling experiment collaboratively to foster discussions and enforce at least a minimal degree of consensus on the relevance of attributes between the two team members. An example result created by one of the teams is shown in Table 1. That group claims that "acting" (as in "the quality of acting") is the most important attribute while "storyline" or "scenes" are considered slightly less important.

The modelling teams provided between 4 and 10 attributes each, with an average of 6.5 attributes per team. An overview of all modelled attributes is shown in table 2. We manually grouped the participants' responses (e.g., "quality of story", "plot", or "story" are all grouped into "storyline"). 90 teams participated in the experiment, and two attributes received a high degree of consensus: 74 teams mentioned "storyline", and 72 mentioned "acting". Several other attributes like "scenery", "character", or "directing" are only mentioned by roughly half of the teams, and several attributes were only mentioned once (like "Disney'ishness" describing how a much a movie feels like a Disney movie; or how friendly the movie seems to portray animals.)

As part of the experiment's post evaluation, several students complained that it was not intuitive to model perceptual attributes. This also shows in the modelled attributes themselves: beyond attributes like acting and storyline, there is little consensus between the 90 data models created in the experiment, and for many modelled attributes it is debatable what they mean and how important they really are. Furthermore, importance of many attributes seems to be quite subjective. Thus, we conclude that explicitly modelling perceptual attributes is often unfeasible. Furthermore, even if such attributes are modelled, obtaining the actual attribute values for all modelled attributes and items in a database is far from trivial (e.g., [YLT17] uses aspect-oriented review mining for this with somewhat limited success). Thus, for the remainder of this work, we opt for fully automatically generated *latent* perceptual attributes as discussed in the next subsection.

- Perceptual attributes should be considered when describing experience items as they better capture how users see or query for items than traditional structured attributes.
- Perceptual attributes are hard to model explicitly as they are often fuzzy and subjective

2.2 Latent Representation of Perceptual Properties from Reviews

Due to the challenges regarding explicit modelling and mining of perceptual attributes, we investigate latent representations. Latent representations can be mined fully automatically from user judgments like reviews or ratings. However, while these attributes might have a real-world interpretation, that interpretation is typically unknown to us (for example, one attribute might represent how scary a movie is, but this attribute will simply have a generic name and we do not know that it indeed refers to scariness). We consider this an acceptable price to pay for the convenience of obtaining both the data model and item attribute

Tab. 2: Modelling Experiment Most and Least Important Attributes

Aspect	Sum Importance	Avg Importance	mentions
storyline	346	4.67	74
acting	341	4.73	72
scenery	197	4.10	48
character	186	4.53	41
directing	177	4.43	40
sound	162	2.74	59
humor	160	3.01	53
originality	67	2.48	27
pace	49	3.06	16
cinematography	46	2.70	17
child friendliness	3	3	1
babes	2	2	1
morality	2	2	1
animal friendliness	1	1	1
Disney'ishness	1	1	1

values fully automatically. A naive way of creating a latent representation is to embed each item in a high-dimensional vector space (therefore, such techniques are also sometimes called "embeddings") with usually 100-600 automatically created dimensions where each dimension represents an (unlabeled) perceptual attribute (like scariness, funniness, quality of special effects, or even the presence of certain plot elements like "movie has slimy monsters" - but again, while the attributes likely do represent a real world aspect, the actual meaning is unknown to us).

Even without explicitly labeling the attributes resulting from embeddings, latent representations can already provide tremendous benefits with respect to the user experience: They can directly be used by most data analytic algorithms like clustering, supervised labeling, or regressions. Also, from a user's perspective, such representations can be used with great effect to allow for semantic example-based navigation queries as we have shown in [LN14] for movies, or be used to increase the ease-of-consumption of review texts [YLT17].

To obtain such latent representations for e.g., movies, early approaches like [SLB12, LN14] relied on decomposing user-item-rating matrices using techniques not too unlike those also used in recommender systems [Sa01]. As the needed large user-rating corpora are hard to obtain nowadays due to privacy concerns, our later work [LW15] relied on the openly available reviews. Here, a very simple heuristic was used: *Similar movies should feature similar reviews, thus by embedding all user reviews into a latent space (by using e.g., techniques like Latent Semantic Analysis [Li12] or Document Embeddings [LB16]), and aggregating them into a single tuple with latent attributes, an effective representation*

of experience items can be created which can be used for example-based and similarity queries. However, this heuristic showed several shortcomings as discussed next.

- Perceptual attributes can be automatically extracted in an implicit latent form, e.g., using review or rating mining techniques
- Latent perceptual attributes have no clear explicit semantics, but can still be effectively used for navigational query-by-example queries

2.3 Towards Shared Perspectives

In [LW15], we relied on Amazon movie reviews to build representations of latent perceptual properties of movies. While we could show that the resulting attributes showed mostly comparable performance when used for a query-by-example system compared to a rating-based latent representation, we encountered several cases of unexpected behaviors. Thus, we manually inspected a selection of movies and their supposedly most similar titles. Here, it turned out that there are indeed many good matches in the similarity list. However, there are also some titles which are highly similar with respect to the latent perceptual attributes even though the movies themselves are very different (e.g., for the movie "Terminator 2", both "Robocop" (a good match) and "Dream Girls Private Screenings" (a surprisingly bad match) are both considered similar). The reason for this irritating behavior seems to be that there are many "bad" reviews. "Bad" reviews are not discussing the movie itself, but other issues and do therefore not contribute to a meaningful representation. Typical examples are "I had to wait 5 weeks for delivery of the item! Stupid Amazon!", "Srsly?! Package damaged on delivery?", "I ordered the DVD version, got the Blue Ray!". For "Dream Girls", reviewers seem to be mostly concerned with the bad quality of the DVD version in comparison to the older VHS release. A similar issue is described in several reviews of the original DVD release of Terminator 2. Such reviews should therefore not be considered when building latent perceptual representations, however it is not a trivial process, since cleaning noisy web-data is a live line of research.

A second issue became apparent much later: for many movies, reviews often show very polarizing viewpoints, and aggregating them into a single tuple represents neither point of view well. Consider for example the teen vampire romance movie "Twilight" with the opinions "This is the worst movie of all times, with cheesy characters, dump story line, and really bad sparkling vampires" and "This movie is the best movie ever made, so romantic and beautiful, the love between the lead characters is so enjoyable." (Note that most review for this movie follow either the first or second opinions - only few people believe that it is just an average movie...) By combining the (high-dimensional numeric vector) embeddings of such two highly polarizing opinions, the resulting combined embedding would likely not be useful. Therefore, it is essential to store strong opinions independently to each other and be considerate when aggregating judgments of users on perceptual properties.

However, we argue that while perceptual properties expressed by users are highly subjective, they are not necessarily unique and there is often a smaller set of shared point of views (or perspectives) when perceiving items: For example, out of 400 reviews for the drama "The Green Mile" in our Amazon dataset, essentially 180 express some variation of " this is a beautiful and touching movie, full of emotion" - while each review might be using slightly different words, the *perspective* is the same. Another 110 agree that the movie has "great acting and a good story that follows the book quite well", and 80 reviews claim variants of "it is a good movie with famous actors, it is long but worth it". The remaining reviews usually express some isolated fringe opinions, like "it is an unrealistic fairy tale in prison", or "such a bad movie, Tom Hanks is so lame", or indeed "I ordered this movie and the package was damaged" —opinions which are typically not shared by many others.

Therefore, we propose to group similar user judgments expressing perceptual properties into *shared perspectives*, and aggregate each shared perspective into one single latent representation. Thus, shared perspectives retain major potentially conflicting and/or polarizing opinions while still aggregating the underlying social judgments to minimize storage space and query complexity. Then we retain only major shared perspectives and discard lesser ones. Considering the "The Green Mile" example, this movie would then be represented by it's traditional meta-data like title, actors, release year, etc., and in addition with three latent perceptual tuples representing the three major shared perspectives. This allows to perform similarity navigation from "The Green Mile" to other movies which are perceived as equally emotionally touching, or have equally good acting/plot following a novel. This is illustrated in figure 1: instead of traditional QBE which only relies on a single, hidden-to-the-user, item similarity, the users can choose to navigate along a shared perspective to discover new items *which are similar with respect to that perspective*. Also, several perspectives can be combined during query processing: "Twilight", a teen vampire romance, is characterized by having two very distinct and conflicting shared perspectives (see above), one deeply embracing the movie for its romantic plot and great characters, the other perspective hating the movie for its plot and characters. A similar situation can be found for "Warm Bodies", a teen zombie romance, which features nearly the same perceptions and perspectives.

- User perception are often highly opinionated, polarizing, or even contradicting. This needs to be respected when representing items, and a single perceptual representation of each item is usually insufficient.
- Still, there is often a smaller number of dominant perception which is consensual shared by larger user groups. Focusing on these *shared perspectives* is a good compromise for efficiently representing experience items while still respecting polarized opinions.

3 Shared Perspectives

This is the core section of this work, where the foundation and implementation of Shared Perspectives is described.

Figure 1: Query-by-Example Step using Shared Perspectives

3.1 Foundation and Terminology

In the following, we introduce new concepts relevant for representing shared perspectives. A database system wishing to use shared perspectives to represent a type of real-world items $I = \{I_1, I_2, ..., I_{n_I}\}$ has a set of *factual attributes* $A_F = \{A_1, ..., A_{n_{AF}}\}$. These attributes A_F represent traditional relational attributes, and we use the term *factual* (as opposed to *perceptual*) to describe that there is at most one value for each item and attribute, and that the attributes have been agreed upon during the schema design phase. Each item $x \in I$ can be represented by a factual tuple $x_F \in D_F$, stored in the factual relation $R_F \subseteq D_F = A_1 \times ... \times A_{n_{AF}}$ using those factual attributes.

In addition, there is also the set of perceptual attributes $A_P = \{P_1, ..., P_{n_P}\}$ with $D_P = P_1 \times ... \times P_{n_P}$. In this paper, we showcase a system using latent perceptual features, thus the set A_P is not chosen by the schema designer, but is typically the result of some sort of algorithm *miner* automatically mining user judgments like reviews or ratings. For example, a document embedding algorithm will typically produce 100 to 300 such attributes. For each item x, there is in addition to the factual tuple x_F also a *set of shared perspectives* $x_{SP} = \{x_{SP_1}, ..., x_{SP_{nxsp}}\} \subseteq D_P$. Finally, the item x is represented by (x_F, x_{SP}). Such a tuple is not in the first normal form (as x_{SP} is a set). Therefore, a real-life application using traditional relational databases without special extensions would typically realize this by normalizing and having several relations using joins to build (x_F, x_{SP}) on the application side.

In order to obtain the shared perspectives for an item $x \in I$, each item x also features several user judgments $UJ_x \subseteq UJ$ like reviews, ratings, discussions, or just explicitly provided

user feedback. The aforementioned mining algorithm *miner* transforms such a judgment into perceptual tuples, i.e. $miner : UJ \rightarrow D_P$. By applying *miner* to each judgment in UJ_x, the set of all perceptual tuples $x_P \subseteq D_P$ is created, i.e. when a movie has 1000 reviews from different users, x_P will contain a perceptual tuple for each review. In our implementation, we do not store x_P, but only use it to discover the shared perspectives x_{SP}. Discovering shared perspectives is realized by the function $g\&a$ (group and aggregate), $g\&a : P(D_P) \rightarrow P(D_P)$. This function will group all perceptual tuples, and aggregate the bigger groups into a single tuple in order to produce the shared perspectives, i.e. $x_{SP} = g\&a(x_P)$.

We discuss the design space for implementing the functions *miner* and $g\&a$ in the following sections.

- Experience Items I can have factual A_F and perceptual attributes A_P. Perceptual attributes of each item x can be obtained from user judgments $UJ_x \subseteq UJ$ using a *miner* grouped into shared perspectives x_{SP} with an appropriate algorithm $x_{SP} = g\&a(x_P)$.
- The shared perspectives of an item $x_{SP} = \{x_{SP_1}, ..., x_{SP_{nxsp}}\} \subseteq D_P$ can be compared to other shared perspectives of an item to find similar perspectives, and therefore similar items.

3.2 Query-By-Example using Shared Perspectives

Shared perspectives, especially the implementation chosen in this work with latent attributes, are hard to use with traditional SQL-style querying. Thus, we propose to use a variant of iterative query-by-example for allowing user to interact with the item space. This type of querying sits between SQL-style querying (where users have to specify exactly what they are looking for), and recommendations (where users specify little to nothing, and the system actively recommends). In [LN14], it has been shown that this type of querying works well with users who only have a vague idea of the items they want, and QBE querying was deemed an enjoyable and playful experience by the users in that study.

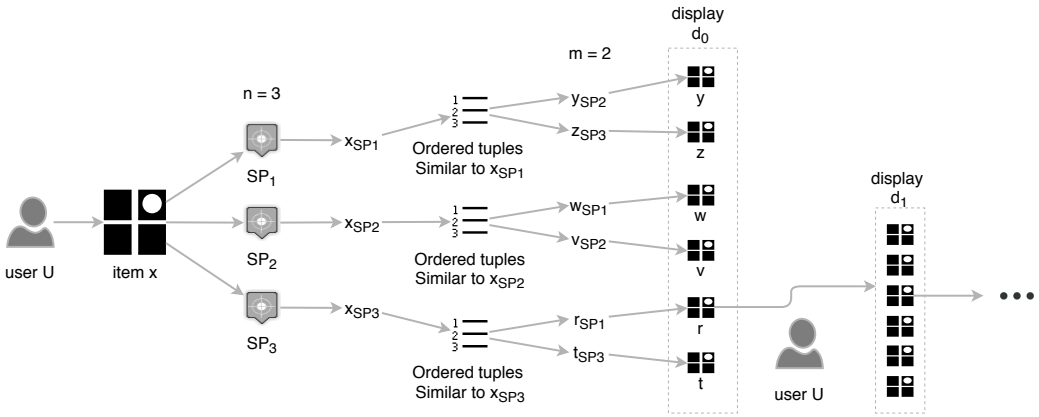
A core concept in querying is semantic similarity *sim* between two perceptual tuples, i.e. $sim : D_p \times D_p \rightarrow [0..1]$. While there are several approaches towards implementing this, we chose with a simple cosine-distance-based similarity.

The shared-perspective-enabled QBE process can be summarized as follows (also see figure 2): The user starts with an example item she likes. Then, for each shared perspective that item has, the system will discover up to n other items which have a shared perspective which is most similar to the current one (n is 3 in our system). Each of these items (n items per shared perspective) are then shown to the user, and the user can select from that display the item she likes most. Then, the process can be repeated until the user is satisfied (see algorithm 1).

Algorithm 1 Query on SP

procedure X ITEM QUERY BY EXAMPLE
  $x \leftarrow UserProvidedExampleItem \in I$
  while user wants to continue **do**
    $display \leftarrow \emptyset$
    for each $sp \in x_{\mathrm{SP}}$ **do**
      $maxItemSim \leftarrow newRelation(item, similarity)$
      for each $(y \in I) \wedge (y \notin display)$ **do**
        $maxSim \leftarrow \max\ (sim(sp, y_{\mathrm{SP}}))$
        $maxItemSim.add(y, maxSim)$
      $display.add(top_{\mathrm{n}}(maxItemSim))$
    $show(display)$
    $x \leftarrow UserSelectedItem \in display$
  end while

Figure 2: Query-by-Example: Shared Perspectives and Displays



- Each item has multiple shared perspectives using perceptual attributes. Shared perspectives are created by representing user judgments as perceptual tuples which are then grouped and aggregated by group.
- For querying, a variant of Query-By-Example can be used. After selecting a start example, we show the user several items which are similar with respect to the different shared perspective of the start item. Then, the user can select a new item she likes and the process starts anew until the user is satisfied.

4 Implementation Design Space

In this section, we outline some areas of the design space for implementing shared perspectives into an information system, and explain the choices of our prototype system.

4.1 Dataset

Underlying our prototype is an existing dataset of movie reviews crawled from Amazon [Mc15]. We only considered movies that have at least 100 reviews, and use maximum 300 (randomly selected) reviews per movie. We also discard reviews with less than 25 words as they are commonly uninformative, This results in a dataset consisting of around 375K movie reviews, for 2,041 movies overall.

4.2 Extracting Perceptual Attributes

In this section, we discuss the design choices for implementing the aforementioned function *miner* which translates reviews into perceptual attributes. In a good review, a user will take the time to briefly summarize the content of an item, and then expresses her feelings and opinions towards it - the task for *miner* is to represent these opinions in such a way that the similarity measures used by the QBE process work well.

In an earlier prototype we focused on explicitly modelling perceptual attributes and using aspect-oriented opinion mining to extract perceptual tuples [YLT17]. However, this had a considerable manual overhead and did result into only few usable attributes which could be extracted reliably. Thus, in this work, we focus on extracting a fixed number of latent perceptual attributes as in [LW15]. In [LW15] we discussed the advantages of different implementation techniques like LSA [DL05] or LDA [BNJ03]. However, in the last few years there has been a surge of approaches proposing to represent the semantics of text documents using neural language models. A prominent example is word2vec, representing the semantics of words as a fixed-length vector [Mi13], and the later version doc2vec which represents documents as vectors [LM14]. Studies in [LB16] suggest that the semantic performance of doc2vec in typical document tasks like clustering or retrieval seems to be quite good, outperforming other approaches based on bag-of-words, word vectors, or LSA/LDA. This has also been shown for the special case of reviews [LM14].

In our prototype, we used the document embedding implementation of doc2vec provided by Gensim [4], using distributed bag-of-words representation (dbow). Typically, such document embeddings need to be trained on a large corpus before they can be used. In scenarios where only a smaller amount of text is available, the solution is often to rely on models pre-trained on Wikipedia or Google News. However, as we had a large review corpus available [Mc15], we trained on all Amazon reviews without excluding anything.

The parameters for training and the embeddings were:

- Vectors are kept at 100 dimensions. Typically, 100 to 300 dimensions is considered good for document similarity tasks [LB16].

[4] https://github.com/piskvorky/gensim/

- The training window is 10 since it showed good performance with documents of similar size (reviews)
- Frequent word subsampling seems to decrease sentiment-prediction accuracy, so it is not applied
- We do not consider any words which are mentioned only once in the whole corpus
- 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 Tuple: (-0.640138, 0.422624, ..., -0.0350407, 0.192102)

- The *miner* selected for this work is doc2vec, applied to amazon movie reviews, with 100 dimensions.
- Hyper-parameter values were chosen in line with the recommendations of the authors of doc2vec and other implementations working with similar documents.

4.3 Aggregating Shared Perspectives

As part of the creation process of shared perspectives, individual perceptual tuples representing single reviews need to be aggregated and summarized to implement the function $g\&a$. We opted for spherical k-medoids clustering to group perceptual tuples for an item [DM01]. Alternatively, we also considered HDBSCAN, but this did not notable improve the results [Ca15]. The 'elbow method' was applied to select the best number of clusters. After clustering, we only retain up to 3 clusters. This is a simplifying design choice to keep the user interface easy and accessible. However, a quick inspection showed that most reviews would only show 2-3 clusters anyway (typically, this is reviews from people who hate the movie, reviews from people who love it, and balanced views). Thus this limitation will only sacrifice semantics in case of very diverse opinions. The shared perspective tuple is representing a cluster is chosen using the medoids. We prefer this solution over k-means, which would create an artificial shared perspective tuple which does not relate to a real user review.

- Spherical k-clustering was implemented as the choice of aggregating or $g\&a$ for this work.
- k=3 was decided since manual inspection of the 'elbow method' for several movies showed that most reviews would only show 2-3 clusters, plus it simplifies design and user interface for the evaluation in Section 5.3.

5 Evaluation

We evaluated our system in two ways: the first one is a query simulation akin to those performed in [LN14]. Here, the core idea is that we assume the existence of a hypothetical "target object" the user is searching for, and simulate the user interactions leading to that item. While this simulation is artificial and does not resemble a real-life user interaction, it still gives insights into the effectiveness of some design decisions. The second evaluation is using real users, and having them interact with the system to evaluate the perceived usefulness and semantic quality of the shown shared perspectives.

5.1 Evaluation: QBE User Simulation

The general effectiveness of QBE using perceptual attributes without shared perspectives has been shown in [LN14, LW15]. Therefore, in this section we focus on the effect the introduction of shared perspectives has on the QBE process. As a baseline, we force our system to only consider a single perspective (i.e., use the medoid of all perceptual tuples resulting from reviews as the only shared perspective). This resembles the setup in [LW15], which uses only a single tuple to represent a movie. We compare this to a version of the system in which we consider up to 3 shared perspectives.

The artificial evaluation scenario is a as follows: choose a random start example movie, and choose a random target movie. Then perform perform query-by-example iterations choosing always the displayed movie which is closest to the target. As an evaluation metric, we count the number of iterations necessary to traverse from the start example to the target. We assume that using shared perspectives, fewer iterations are needed as the display selection has a wider semantic spread then when using only a single perspective, since SPs can help to 'get out' of dense similarity neighborhoods. For instance from "The Green Mile" to "Tinkerbell", there is a perspective that relates them: "The Green Mile" has a "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".

Note that this evaluation is very artificial from a semantic point of view: shared perspectives with QBE are designed to help users who have a vague idea of the style/type/general flavor of item they are looking for. They will not have a particular item in mind (if they had, they could simply retrieve it using SQL). Thus, we assume that users will choose a starting example which is in the proximity of their unclear target, and then clarify their preferences during the query process (e.g., "I know what I am looking for as soon as I see it.")

The basic steps for this evaluation are:

1. For the currently selected movie x, the system generates a new display with 9 movies as described in section 3.2. When using shared perspectives, it is the 3 most similar

movies with respect to the 3 perspectives of x. This is, the 3 movies with a shared perspective tuple with the least cosine distance to x_{SP_1}, 3 more for x_{SP_2}, and 3 for x_{SP_3}. When not using shared perspectives, it is simply the 9 movies with least cosine distance to the single tuple x.

2. If target movie is in display, finish. If not, select the best option (i.e., movie which is most similar to the target) as new example movie from display

3. Repeat

Results: The average steps it takes from start to target movie for 175 different pairs of movies is 38 when using only a single representation, and 28.03 when using shared perspectives. Thus, shared perspectives reduce the average number of required interaction steps by roughly 30% (again, note, that in a real use case, there will be significantly fewer steps as start movies are chosen closer to the implicit "target"). The frequency distribution of the number of steps for this experiment is shown in Figure 3. 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. While this is an improvement, there is however also number of movie pairs for which the simulation takes more than 175 - something that does not happen when not using shared perspectives. This odd behavior will be investigated later in section 5.3.

Please note that these numbers seem high when compared to the best results reported in [LN14, LW15], where the results were between 10 and 17 steps. In those works, we employed an additional Bayesian probability user model which did not solely rely on similarity, but also provided shortcuts to the movie which is the predicted target of the user based on that model. Similar techniques could also be applied to shared perspectives. However, like mentioned in those works, this is beneficial when the goal is to reach a predefined target, which in reality is not known to user or system. Therefore in a real application, where there is no such target, displaying items that are highly informative for the system, like showing Terminator as similar to Finding Nemo, would be confusing to the user.

- For evaluation, we propose a simulation with a starting and target movie to mimic the behavior of a user.
- We compare the performance of a single representation of a movie against the multiple shared perspectives.

5.2 User Evaluation

The purpose of this user study is to obtain some insights on the quality of shared perspectives for querying from a user's point of view. We do not consider this an exhaustive quantitative analysis, but rather a quicker exploration to obtain an intuition on the quality of the approach. Especially, we are interested in how useful different perspectives are perceived, and what

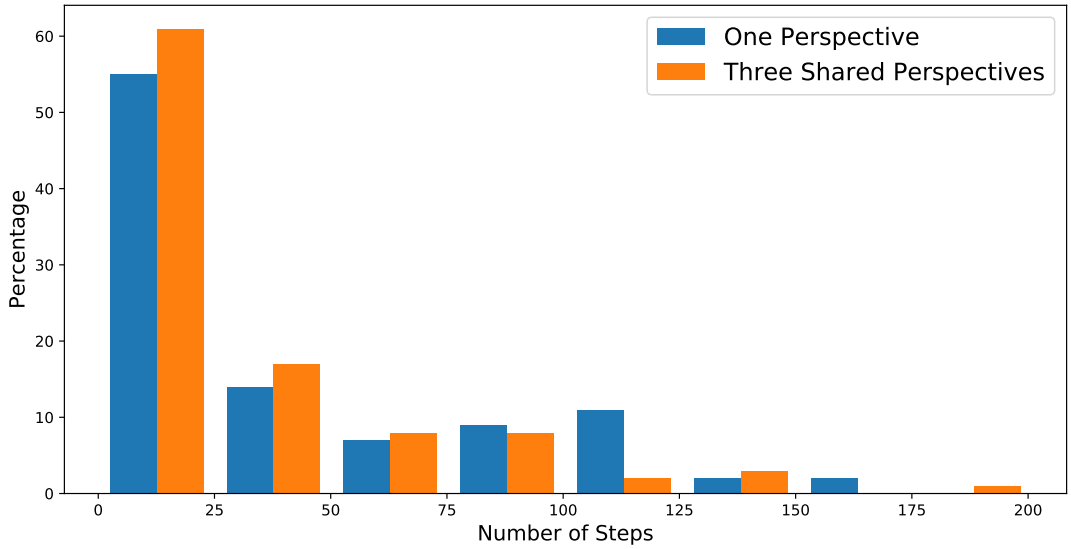


Figure 3: Histogram of Steps with 175 pairs

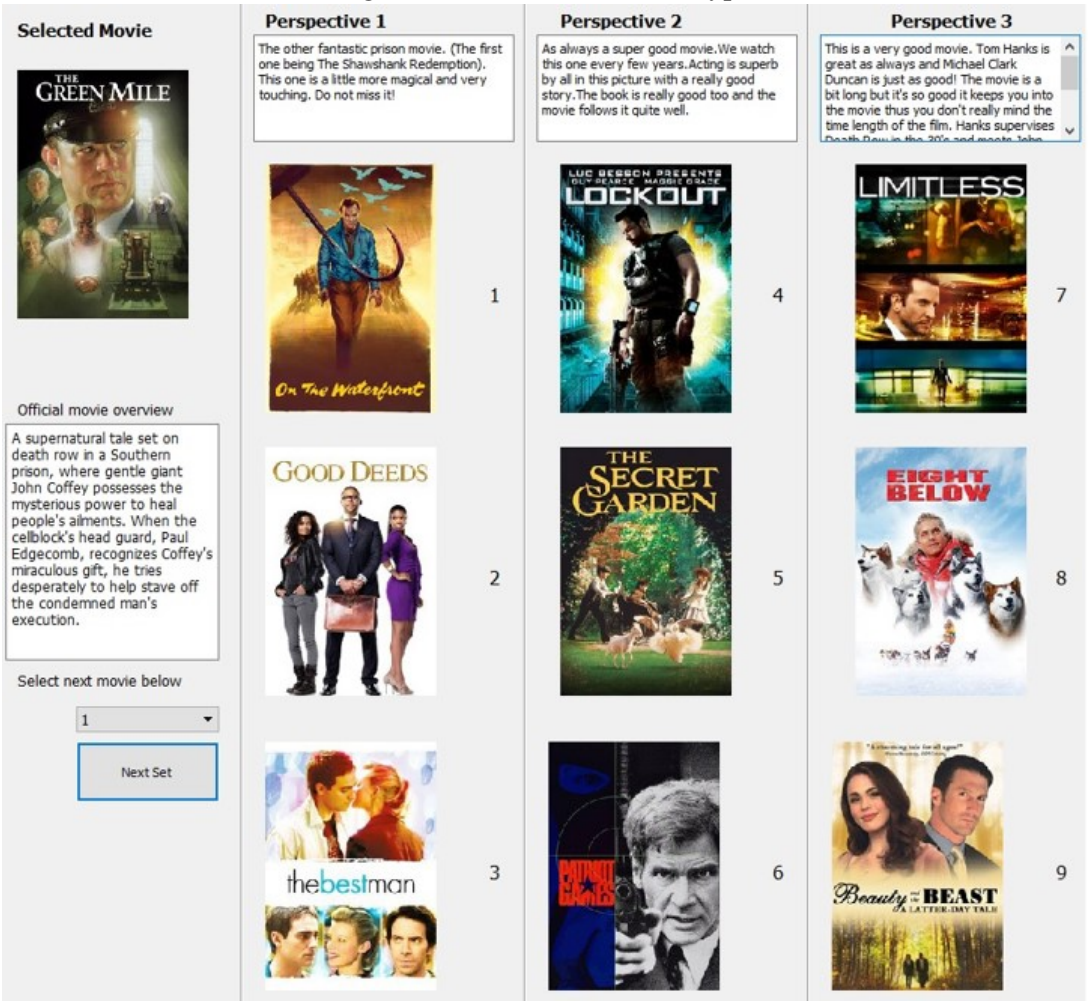
their semantics could be (remember: shared perspectives result from semantically clustering reviews. Thus the question is: What is the commonality of the reviews which contributed to a perspective? Is it a meaningful semantic?).

To this end, we asked seven participants to interact with the system and rate how useful a perspective is based on the constructed QBE display. The participants include people from 18 to 55 years old, from different countries and education levels. The setup of the experiment is the following:

- We sent a standalone desktop application to the study's participants
- The application implemented the QBE workflow as outlines in section 3.2, with a display of 9 movies (3 for each shared perspective) in each iterations step.
- We showed the users the text of the medoid review which represents a perspective (see figure 4 for a user interface example)
- Instructions for the participants were to rate for each perspective how well the shown review snippet represents the selected movies on a 3-point scale: well represented (score 2), somewhat represented (1), not represented well (0).

All 7 participants evaluated the same set of 7 displays. For each shared perspective thus evaluated, we computed a "usefulness score" based on the user feedback. Here, some perspectives got higher ratings, like for "Real Steel" which had a perspective connecting to "The Last Starfighter" which can be summarized as "awesome sci-fi for the whole family", or "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" with a usefulness score of 1.5. However, even when looking at this small-scale study involving only

Figure 4: User Interface Prototype



few users and example displays, some problems became apparent: “The Good, The Bad and The Ugly” relates to “The Towering Inferno”, “Shane” and “The Uninvited” because “The digital restoration looks really great”. This perspective was not perceived as useful with a score 0.5. This problem was already encountered in [LW15]. However, in that work, such reviews affected the whole item representation. However, when using shared perspectives, it usually results in only one bad perspective, while typically one or two useful ones remain. As such, this problem is less severe for shared perspectives than it was for [LW15], but still this is an unsatisfying result.

As an additional exploration, we provided users with the option to label shared perspectives with a few keywords. Even though we obtained only a smaller number of labels this way, some of them were quite descriptive: all participant for example noted that the second perspective for “The Jungle Book” represents “Disney classics”, or that the first perspective for “The Good, the Bad and the Ugly” should be something along the lines of “great western with very good actors”. In contrast, the first perspective in “The Green Mile” received labels along the lines of “good acting”, or “good story” with no clear con census. Other

perspectives received no labels at all. In our future works, we will extend this preliminary inspection with a larger scale user study to provide reliable insights into the semantics of shared perspectives. For now, we remain confident that while not all perspective have a discernible meaning, many actually have one.

- In addition to the simulation for evaluation, a user-study is conducted to rate the usefulness of shared perspectives to find similar movies.
- The usefulness is for each shared perspective, and can be defined as: how good is $x_S P_1$ to say that movie x is related to movie w because of $w_S P_2$, to movie y because of $y_S P_1$ and z because of $z_S P_1$.

5.3 Predicting Usefulness Scores

Motivated by the results in the last section, we tried to look into the problem of not useful shared perspectives. The goal of this to learn from user feedback, and predict for each perspective a usefulness score.

In a production system, this could be a continuous process where users are given the option to down-vote a perspective as not useful, and the system is considering this feedback when computing usefulness scores. Then, perspectives could be presented ordered by their usefulness, or skipped all together if deemed very unuseful. Also, the calculation of all similarities could be modified by usefulness.

We manually analyzed all shared perspectives involved in the last experiment. For those, non-useful perspectives are closely similar to perspectives of a large number of movies. On the other hand, useful perspectives are only similar to perspectives of a smaller number of movies. This can be justified by the following intuition: useful perspectives should be specific to that movie, and only few other movies should share that perspective, like being a “heart-warming children movie” or being a “Disney classic". If a perspective (i.e., a particular topic featured in reviews) is present in a large number of movies, it is likely not useful. This would for example be the case for “My packaging was damaged” or “Image quality is grainy due to HD upscale” - such sentiments can afflict any movie no matter it’s actual qualities or content.

To capture this intuition, we calculate the Pearson Median Skewness (PMS) score, also called Second Skewness Coefficient of a given shared perspective tuple to all other shared perspective tuples in the database, calculating a usefulness score for that perspective based on the difference between mean and median of PMS.

We extended the simulation experiment from section 5.1 by adjusting the similarity calculation with usefulness scores, thus de-emphasizing unuseful similarities. This improved the number of simulation steps from 38 when using only a single representation, to 28.03 when simply using shared perspectives, to 26.05 when using shared perspectives weighted

by usefulness. More importantly, while this average improvement is quite small, extreme outlier cases saw improvement: e.g., one pair of movies which took 82 steps now only takes 41; and others went from 205 to 170.

This result motivates us to investigate this approach more thoroughly in our future works, improving the way how we predict usefulness in a larger setup with more users and feedback. Especially, the calculation of usefulness scores should relate to actual user feedback instead of being based on intuitions as discussed in this section. In addition, modelling and calculating the score of perspectives can be used to remove noisy reviews, a current problem with applications of web-data.

- Leveraging the usefulness scores obtained before, a function of usefulness was modelled to then calculate the scores of all shared perspectives in the corpus.
- The scores are then evaluated in a simulation similar to Section 5.1 with promising results.

6 Conclusions

In this paper, we discussed the challenge of representing experience items like movies, books, or music. For such items, perceptual attributes should be modelled, which is an inherently difficult task during the design phase but also later when items need to be described with respect to these properties. To underline this claim, we presented a brief study with 180 students who were asked to model perceptual properties for movies. This task was shown to be hard, and the results between participants were not very consistent.

Therefore, in this paper we proposed the use of latent perceptual attributes which are automatically mined from user judgments like ratings of reviews. While such latent attributes have no explicit human-understandable semantics, they can be effectively used for query-by-example navigational queries. However, previous works revealed a shortcoming with this approach: typically, each item is represented using a single tuple. Especially when highly opinionated and polarized social judgments are used to generate the latent perceptual attributes, performance suffers as the resulting tuple represents neither viewpoint well. To rectify this, we introduced the concept of *shared perspectives*, perceptual tuples representing a dominant consensual point of view for an item. Each database item can have several shared perspectives, thus striking a balance between aggregating social judgments for storage and query efficiency, and still retaining conflicting opinions for better semantic representatives and querying.

We discussed our prototype implementation for a query-by-example system exploiting shared perspectives, and showed evaluations with synthetic queries but also a human user study. Based on the feedback gathered during evaluations, we suggested an improvement of our approach which takes the semantic usefulness of a perspective into account.

For future improvements, we aim at applying our approach on real-live problems in additional domains, as for example on large music repositories. Furthermore, the QBE query processing process can benefit from additional tuning by for example combining also the objective structured meta-data into the similarity measures, and also including additional user modelling to allow users to discover desired items even quicker (as e.g, in [LN14]). Also, a large-scale user study into the semantics of shared perspectives is planned.

References

[BKV10] Bell, Robert M.; Koren, Yehuda; Volinsky, Chris: All together now: A perspective on the Netflix Price. CHANCE, 23(1):24–24, apr 2010.

[BNJ03] Blei, David M; Ng, Andrew Y; Jordan, Michael I: Latent Dirichlet Allocation. Journal of Machine Learning Research, 3:993–1022, 2003.

[Ca15] Campello, Ricardo J. G. B.; Moulavi, Davoud; Zimek, Arthur; Sander, Jörg: Hierarchical Density Estimates for Data Clustering, Visualization, and Outlier Detection. ACM Transactions on Knowledge Discovery from Data, 10(1):1–51, 2015.

[DL05] Dumais, Susan; Landauer, Thomas: Latent semantic analysis. Annual Review of Information Science and Technology, 38(1):188–230, 2005.

[DM01] Dhillon, Inderjit S.; Modha, Dharmendra S.: Concept Decompositions for Large Sparse Text Data Using Clustering. Machine Learning, 42(1):143–175, Jan 2001.

[KB11] Koren, Yehuda; Bell, Robert: Advances in Collaborative Filtering. In: Recommender Systems Handbook, S. 145–186. 2011.

[LB16] Lau, Jey Han; Baldwin, Timothy: An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation. CoRR, abs/1607.0:78–86, 2016.

[Li12] Liu, Chien Liang; Hsaio, Wen Hoar; Lee, Chia Hoang; Lu, Gen Chi; Jou, Emery: Movie rating and review summarization in mobile environment. IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews, 42(3):397–407, 2012.

[LM14] Le, Quoc V.; Mikolov, Tomas: Distributed Representations of Sentences and Documents. In: 31st International Conference on Machine Learning. Jgg. 32, Beijing, China, S. 1188–1196, 2014.

[LN14] Lofi, Christoph; Nieke, Christian: Exploiting Perceptual Similarity: Privacy-Preserving Cooperative Query Personalization. In: Web Information Systems Engineering – WISE 2014. Springer International Publishing, Cham, S. 340–356, 2014.

[LSY03] Linden, G.; Smith, B.; York, J.: Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Computing, 7(1):76–80, jan 2003.

[LW15] Lofi, Christoph; Wille, Philipp: Exploiting social judgements in big data analytics. CEUR Workshop Proceedings, 1458:444–455, 2015.

[Mc15] McAuley, J.; Targett, C.; Shi, J.; van den Hengel, A.: Image-based recommendations on styles and substitutes. In: ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR). Santiago de Chile, Chile, 2015.

[Mi13] Mikolov, Tomas; Sutskever, Ilya; Chen, Kai; Corrado, Greg S.; Dean, Jeff: Distributed Representations of Words and Phrases and their Compositionality. Advances in Neural Information Processing Systems, 21:3111–3119, 2013.

[MTT17] Matakos, Antonis; Terzi, Evimaria; Tsaparas, Panayiotis: Measuring and moderating opinion polarization in social networks. Data Mining and Knowledge Discovery, 31(5):1480–1505, 2017.

[Sa01] Sarwar, Badrul; Karypis, George; Konstan, Joseph; Reidl, John: Item-based collaborative filtering recommendation algorithms. Proceedings of the tenth international conference on World Wide Web - WWW '01, S. 285–295, 2001.

[SLB12] Selke, Joachim; Lofi, Christoph; Balke, Wolf-Tilo: Pushing the boundaries of crowd-enabled databases with query-driven schema expansion. Proceedings of the VLDB Endowment, 5(6):538–549, 2012.

[TNK10] Thet, Tun Thura; Na, Jin-Cheon; Khoo, Christopher SG: Aspect-based sentiment analysis of movie reviews on discussion boards. Journal of information science, 36(6):823–848, 2010.

[YLT17] Ye, Mengmeng; Lofi, Christoph; Tintarev, Nava: Memorability of Semantically Grouped Online Reviews. In: Semantics 2017. Amsterdam, Netherlands, sep 2017.