Intelligent Tagging Systems: Machine Learning for Novel Interaction

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Jesse Vig

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

April, 2012

Acknowledgements

I would like to thank:

- My advisor, John Riedl, for his contagious enthusiasm for research and his sage advice.
- My colleague, Shilad Sen, for welcoming me into the GroupLens lab and setting an impossible standard to follow.
- GroupLens, for their friendship, creativity, and intellectual energy.
- My professors, for showing me the many and diverse sides of computer science that I have tried to incorporate into this thesis.
- MovieLens developers, past and present, for an unparalleled research platform without which none of this work would have been possible.
- My family, for their consistent support and encouragement.

Abstract

The Web puts a vast repository of information at users' fingertips, but the size and complexity of this information space can easily overwhelm users. Recommender systems and tagging systems represent two very different approaches to addressing this information overload. Recommender systems use machine learning and statistical models to automatically retrieve the items of most interest to a particular user. Tagging systems leverage the community's collective knowledge to help users explore the information space themselves.

While both approaches can be very effective, they each have limitations. Recommender systems require little effort from users, but they leave users with little control over the recommendation process. Tagging systems put control in the hands of the user, but—because tags are applied by humans—tagging systems often suffer from issues of tag sparsity.

This thesis explores intelligent tagging systems that combine the machine intelligence of recommender systems with the user control and comprehensibility of tagging systems. We first present *Tagsplanations*, tag-based explanations that help users understand why an item was recommended to them. We then introduce the *Tag Genome*, a novel data structure that uses machine learning to augment human judgments of the relationships between tags and items. Next we discuss *Movie Tuner*, a conversational recommender system based on the Tag Genome that enables users to provide multifaceted feedback using tags.

For each system, we outline the design space of the problem and discuss our design decisions. We evaluate each system using both offline analyses as well as field studies involving thousands of users from MovieLens, a movie recommender system that also supports tagging of movies. Finally, we draw conclusions for the broader space of related applications.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The Web provides the greatest collection of information that the human race has ever created. But the Web's greatest asset is also its biggest liability: the size and complexity of this information can make it difficult for users to navigate this space.

Various tools have been developed to help users overcome this information overload. One of the earliest approaches to help users navigate the Web was to rely on system designers or domain experts to organize the Web for users. For example, Yahoo's[1] early home pages provided a taxonomy of the Web, allowing users to drill down into topics of interest such as *Art*, *Business*, *Computers*, etc. This type of expert-maintained taxonomy provides a clean structure for navigation, but compensating these experts can be expensive; further, these taxonomies may not accurately represent the diverse and evolving interests of the user community [Shirky, 2005].

More recently, two broad themes have emerged in the tools that people use to navigate the Web. First, users have become more involved in the task of organizing the information on the Web. *Tagging systems* enable users to apply free-form labels to items in an information space and share those tags with other users. For example, users of Flickr[2] can tag images with descriptors such as *nature*, *building*, or *waterfall*. The result is a user-defined taxonomy, or *folksonomy*, which users can explore to locate exactly the

[1] www.yahoo.com
[2] www.flickr.com

information they are looking for.

A second theme is that machine learning approaches for navigating the Web have become increasingly popular. Search engines use algorithms such as PageRank [Page et al., 1999] to automatically locate the page that best match a user's query. Recommender systems take the level of automation one step further, suggesting items based on a user's past behavior without even requiring the user to specify a search query. The user simply consumes a list of recommendations; no navigation is required.

Both approaches for navigating information spaces have strengths and weaknesses. Tagging systems give users control to specify the categories that best represent their interests, but these systems often suffer from issues of tag sparsity because there are simply not enough users available or willing to apply all of the relevant tags. Automated systems such as recommender systems require very little effort from users, but they also leave users with very little control over the navigation process; if users are unhappy with their recommendation lists, their only recourse is to rate more items in the hopes that the system will eventually figure out what they really want.

The goal of this thesis is to explore intelligent tagging systems that combine the best elements of both approaches. We develop systems that combine the user control and comprehensibility of tagging systems with the automation of recommender systems. Rather than using machine intelligence to automate navigation – as in the case of recommender systems – we use machine learning to help users understand and explore an information space.

We first introduce a system called *Tagsplanations* that uses tags as a way for recommender systems to *explain* recommendations to end users. We then present the *Tag Genome*, an extension of the traditional tagging model that enables a wide variety of interactions between users and intelligent systems. We explore one particular application of the Tag Genome called *Movie Tuner*, which allows users to guide a movie navigation system by providing rich feedback using tags. Finally, we discuss the broader space of applications of the Tag Genome.

1.2 Contributions

1.2.1 Tagsplanations

Recommender systems and their users engage in a form of dialog: users communicate their preferences to the system and the system provides recommendations in return. The channel of communication between user and system is generally narrow; in a typical recommender system, users articulate their preferences through numeric ratings of items, and the system communicates its recommendations as lists of items and predicted ratings. While this limited form of communication offers benefits such as efficiency and low cognitive load, it omits information that can yield a more informed recommendation process. For example, what are the qualities of a particular item that led the system to recommend it to a particular user? And why did a user accept or reject a particular recommendation?

One way to enhance the dialogue between system and user is to have the system *explain* recommendations to users; for example, Netflix[3] explains movie recommendations by showing users similar movies they have rated highly in the past. We explore *Tagsplanations*, tag-based explanations of recommendations. Tagsplanations explain a recommendation by decomposing a user's preferences for the recommended item into their preferences for its tags, for example, *"We recommend the movie Fargo because it is tagged with* ***quirky*** *and you have enjoyed other movies tagged with* ***quirky****."*. We describe the design space of Tagsplanations, detail our design decisions, and discuss the results of a user study.

1.2.2 The Tag Genome

Two properties of the traditional tagging model have shaped – and to some degree, limited – the design space of tagging systems:

1. *Tags are binary.* Users may apply tags to an item, but usually do not indicate how strongly the tags apply to the items.

2. *Tags are sparse.* Tags only have meaning with respect to an item if someone has actually applied the tag to an item. Since users apply tags voluntarily, there is no

[3] http://www.netflix.com

guarantee that all of the tags that are relevant to an item will actually be applied to that item.

We propose that many novel and exciting tagging applications are possible if we reconsider these two assumptions. We introduce an extension of the traditional tagging model, called the *Tag Genome*, which provides a richer and more complete mapping between tags and items. In contrast to the binary relationship of traditional tagging systems, the Tag Genome measures the relationship between tags and items on a continuous scale. Whereas traditional tagging systems depend on humans to apply tags, the Tag Genome uses machine learning to compute the relationship between tags and items, thereby avoiding the sparsity issues that plague traditional tagging systems.

By providing a continuous and dense mapping between tags and items, the Tag Genome opens the door for novel forms of user interaction. In the next section, we introduce one such application called Movie Tuner that enables users to tune their movie selection along continuous dimensions represented by tags. Later we discuss the broader space of applications of the Tag Genome.

1.2.3 Movie Tuner

Researchers have explored recommender systems that enable users to provide richer feedback than numeric ratings. *Conversational recommenders* allow users to provide multifaceted feedback on recommendations [Linden et al., 1997, Burke et al., 1997, Faltings et al., 2004, Smyth et al., 2004]; a classic example is the Entree restaurant recommender, in which users critique restaurant recommendations by asking for restaurants that are, for example, "less expensive", "more quiet", or "less traditional" [Burke et al., 1997]. In response, the system provides new recommendations that satisfy the users' critiques.

We present Movie Tuner, a conversational recommender system in which users critique items using tags. Movie Tuner enables users to ask for movies that are, for example, "less violent" or "more quirky", since *violent* and *quirky* are tags in the system. Tags enable users to express a much broader range of feedback than traditional conversational recommenders, which typically offer a narrow set of dimensions for users to critique. Further, because tags are applied by the users themselves, they are able

to capture the diverse and changing interests of the user community. While traditional conversational recommenders depend on an expert-maintained knowledge base, Movie Tuner is driven by the Tag Genome.

Chapter 2

Research Context

2.1 Tagging Systems

Tags — free-form labels that users apply to items such as pictures, movies, or books — have become increasingly popular on websites such as Delicious[1] , Flickr[2] , and Amazon[3]. Tagging systems serve many user tasks including navigation, self-expression, organization, learning, and decision support [Sen et al., 2006, Ames and Naaman, 2007, Golder and Huberman, 2006]. Most tagging systems are collaborative in nature: tags applied by one user are visible to everyone, the vocabulary of tags evolves as a "folksonomy" [Golder and Huberman, 2006], and tag cloud visualizations show which tags are most popular for each item [Bateman et al., 2008].

While tags provide many benefits, they also present several challenges. Tags may be of poor quality [Sen et al., 2007], redundant with one another [Golder and Huberman, 2006], or meaningful only to the user who applied them [Sen et al., 2007]. Also, tagging systems often suffer from issues of tag sparsity; there are simply not enough users willing to apply all of the relevant tags for each item. In later chapters we will discuss various approaches for addressing these issues.

Researchers have studied tagging systems from various perspectives. Much of

[1] http://del.icio.us
[2] http://www.flickr.com
[3] http://www.amazon.com

the work in this area seeks to quantitatively model the large-scale dynamics of tagging systems. For example, [Cattuto et al., 2007] apply a stochastic model to explain how tags tend to follow a power-law distribution. More qualitative studies have explored user behavior on tagging systems at the level of an individual. For example, [Ames and Naaman, 2007] presents a taxonomy of users' motivations for tagging based on a qualitative study of users' tagging of photos. Other research has studied the effect of various user interfaces (tag cloud, list, etc.) on tagging systems [Rivadeneira et al., 2007, Schrammel et al., 2009, Bateman et al., 2008]. Finally, other research has explored novel types of tagging systems [Bernstein et al., 2009, Vig et al., 2010].

This thesis falls in the last category of tagging research listed above. Through Tagsplanations and Movie Tuner, we show how to use tags to provide novel forms of interactions to end users. Beyond developing particular tagging systems, we also develop a general framework in the form of the Tag Genome, which supports a wide variety of end-user applications.

2.2 MovieLens Platform

The movie recommender system MovieLens[4] (see Figure 2.1) serves as the research platform for the work in this thesis. The primary purpose of MovieLens is recommendation: users rate movies on a scale from 1/2 to 5 stars and receive recommendations in return. MovieLens has been in continuous use since 1997, with 198,000 users providing a total of 18 million movie ratings.

In 2006, a tagging feature was added to MovieLens, enabling users to apply free-form descriptors such as *violent*, *Meg Ryan*, or *quirky* to movies. MovieLens uses the *bag* model of tagging [Marlow et al., 2006], where multiple users may apply the same tag to a given item. Since the tagging feature was launched, 6,166 users have applied 29,581 distinct tags, resulting in 273,000 total tag applications (a *tag* is a particular word or phrase, a *tag application* is a user's association of a tag with a particular item).

Users may apply tags on two pages in MovieLens: the *movie details* page (see Figure 2.2) and the *movie list* page (see Figure 2.3). The movie details page displays detailed information about a particular movie including cast, director, and a Netflix synopsis.

[4] http://www.movielens.org

Figure 2.1: MovieLens homepage

The movie list page is used to display any list of movies, including search results and personalized recommendations.

2.3 Generalizations to Other Contexts

We focus on the movie domain in this thesis because we have proprietary access to the movie recommender site MovieLens. However, the work in this thesis can apply to any number of other domains, such as music services, news portals, or dating sites. In this section we identify the key dimensions required for the various systems presented in this thesis.

- **Tagsplanations**

  Tagsplanations require two types of user input: tags and ratings. Ratings are used to infer each user's preference profile, and tags are used to describe those preferences in terms that users can easily comprehend. The ratings data must be dense enough to compute meaningful correlations in ratings between items, and the tags must be sufficiently dense to capture a range of reasons a user might like or dislike a particular item. Although we rely on explicit star ratings for the

Figure 2.2: Tags on movie details page

Figure 2.3: Tags on movie list page.

Tagsplanations on MovieLens, implicit ratings such as click-throughs may also be used.

- **Tag Genome**

  The requirements for computing the Tag Genome are somewhat more broad. Tags are necessary to computing the Tag Genome, but – in contrast to Tagsplanations – no ratings data is required. Although we rely on ratings data in computing the Tag Genome on MovieLens, systems designers can use any data that provides a signal of tag relevance. For example, we also mine the text in users' movie reviews in computing the Tag Genome on MovieLens. In Chapter 4, we discuss various types of data that may be used for computing the Tag Genome in other domains. The other requirement for computing the Tag Genome is a user community willing to share the relationships between tags and items on a continuous scale. However, this is only required for a subset of tag-item pairs; a machine learning algorithm trains on this data and then infers the relationship for the vast majority of tag-item pairs.

- **Movie Tuner**

  The requirements for building a Tuner application such as Movie Tuner are the same as those for computing the Tag Genome, since the Tag Genome provides all the information that Tuner applications need. Beyond those basic requirements, the tuner application may also benefit from displaying various information about each item to help users formulate their critiques; in Movie Tuner, for example, we also display each movie's genre, a synopsis of the movie, and the movie's cover art. In domains that provide instant consumption of content, users can formulate their critiques in the process of consuming the content; for example, a user listening to *Moonlight Sonata* on an online music station might tell the system that they would prefer a song that is more *fast* or less *classical*.

Chapter 3

Tagsplanations

3.1 Introduction

While much of the research in recommender systems has focused on improving the accuracy of recommendations, recent work suggests a broader set of goals including trust, user satisfaction, and transparency [Tintarev and Masthoff, 2007b, Bilgic and Mooney, 2005, Sinha and Swearingen, 2002, Herlocker et al., 2000]. A key to achieving this broader set of goals is to *explain* recommendations to users. While recommendations tell users what items they might like, explanations reveal *why* they might like them. An example is the "Why this was recommended" feature on Netflix[1]; Netflix explains movie recommendations by showing users similar movies they have rated highly in the past.

Research shows that explanations help users make more accurate decisions [Bilgic and Mooney, 2005], improve user acceptance of recommendations [Herlocker et al., 2000], and increase trust in the recommender system [Sinha and Swearingen, 2002]. Moreover, studies indicate that users *want* explanations of their recommendations – a survey of users of one movie recommender site showed that 86% of those surveyed wanted an explanation feature added to the site [Herlocker et al., 2000].

While many different types of explanation facilities exist, they all seek to show how a recommended item relates to a user's preferences. As Figure 3.1 illustrates, a common technique for establishing the relationship between user and recommended item is to use

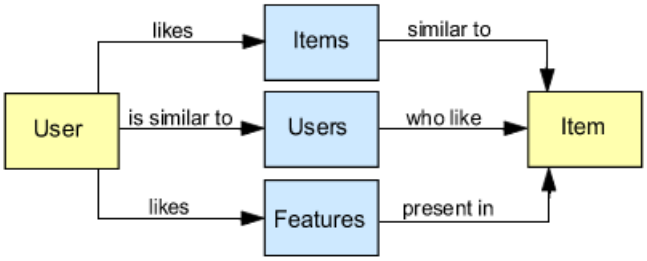

Figure 3.1: Intermediary entities (center) relate user to recommended item.

an *intermediary entity*. An intermediary entity is needed because the direct relationship between user and item is unknown, assuming that the user has not yet tried the item. In the Netflix example above, the intermediary entity is the list of previously rated movies shown in the explanation. The relationship between the user and this list of movies is that the user has rated these movies positively. The relationship between the list of movies and the recommended movie is that other users who liked the movies in the list also liked the recommended movie.

Explanations of recommendations fall into one of three categories: *item-based*, *user-based*, and *feature-based*, depending on the type of intermediary entity used to relate the user to the recommended item. In item-based explanations like the Netflix example, a set of items serves as the intermediary entity. User-based explanations utilize other users as intermediary entities. For example, Herlocker et al. designed a explanation that shows a user how other users with similar taste rated the recommended item [Herlocker et al., 2000]. Feature-based approaches use features or characteristics of the recommended item as intermediary entities. For example, one movie recommender prototype uses movie features including genre, director, and cast to justify recommendations [Tintarev, 2007].

We present a new type of explanation that uses tags as features, which we call a *Tagsplanation*. The intermediary entity for Tagsplanations is a tag or a set of tags. For example:

"We recommend the movie Fargo because it is tagged with ***quirky*** *and you have enjoyed other movies tagged with* ***quirky****."*

Tags have have many qualities that make them useful for explanations. As described in [Golder and Huberman, 2006], tags may describe what an item is, what it is about,

or what its characteristics are – all of which may be useful for explaining a recommendation. Another advantage is that no experts are needed to create and maintain tags, since tags are applied by the users themselves. Furthermore, tags provide both factual and subjective descriptions [Sen et al., 2006]. However, tags present unique challenges, including issues of tag quality and tag redundancy, as discussed in Section 2.1.

We study two aspects of tag-based explanations: the relationship of the tag to the recommended item, which we call *tag relevance*, and the relationship of the user to the tag, which we call *tag preference*. Tag relevance represents the degree to which a tag describes a given item. For example, consider a Tagsplanation for the movie *Pulp Fiction* that uses the tag *dark comedy*. In this example, tag relevance would measure how well *dark comedy* describes *Pulp Fiction*. Tag preference, on the other hand, measures the user's sentiment for the given tag, for example how much the user likes or dislikes dark comedies. Tag relevance and tag preference are independent of one another: the former is item-specific and the latter is user-specific.

Our design of Tagsplanations is motivated by three goals: justification, effectiveness, and mood compatibility. *Justification* is the ability of the system to help the user understand why an item was recommended [Herlocker et al., 2000]. Justification differs from transparency [Tintarev and Masthoff, 2007b] because justifications may not reveal the actual mechanisms of the recommender algorithm. Tintarev et al. define *effectiveness* as the ability of the explanation to help users make good decisions [Tintarev and Masthoff, 2007b]. *Mood compatibility* is the ability of the explanation to convey whether or not an item matches a user's mood. A recent study showed that users are interested in explanations with mood-related features [Tintarev and Masthoff, 2007a].

In this chapter, we investigate the roles of tag relevance and tag preference in Tagsplanations. Specifically, we consider the following research questions:

RQ-Justification: What is the role of *tag preference* and *tag relevance* in helping users understand their recommendation?

Explanations help users understand why a given item was recommended. Tagsplanations promote understanding by demonstrating how a tag relates to the item and how the user relates to the tag. We investigate the role of these two components in helping users understand the recommendation overall.

RQ-Effectiveness: What is the role of *tag preference* and *tag relevance* in helping users determine if they will like the item?

Tag preference measures how much a user likes a particular tag, independent of any particular item, while tag relevance measures how strongly a tag applies to a particular item. We investigate how users process these two metrics in determining whether they will like a particular item.

RQ-Mood: What is the role of *tag preference* and *tag relevance* in helping users decide if an item fits their current mood?

Recommender systems typically do not consider the user's mood when generating recommendations. Explanations provide users with additional information that can help them decide if an item fits their current mood. Tags in particular reveal specific characteristics of an item that may be more or less appealing to a user depending on the user's mood. We investigate the relative importance of tag preference and tag relevance for revealing mood compatibility.

RQ-Tag-Type: What types of tags are most useful in Tagsplanations?

A wide variety of tags may be applied to an item, some of which may be more suitable for explanations than others. As discussed in [Sen et al., 2007], *factual* tags identify facts about an item such as people, places, or concepts, while *subjective* tags express users' opinions of an item. We investigate the relative usefulness of factual tags versus subjective tags, and analyze which specific tags perform best.

To answer these research questions, we developed a Tagsplanation feature for the MovieLens movie recommender site[2] . We investigated various definitions of tag preference and tag relevance and implemented algorithm for computing both metrics. We designed 4 different versions of the Tagsplanation interface, each of which handles tag preference and tag relevance in a different way.

We then conducted an online user study to evaluate how well different versions of the Tagsplanation interface promote the goals of justification, effectiveness, and mood compatibility. We analyzed the results to determine the roles of tag relevance and tag preference in achieving these 3 goals. We also studied the role of different types of tags

[2] http://www.movielens.org

in Tagsplanations.

3.2 Related Work

Item-based explanations. Item-based approaches use items as intermediary entities to relate the user to the recommended item. An example is the "Why this was recommended" feature on Netflix, which shows users their past ratings for a set of related movies. Similarly, Amazon shows users their past purchases that motivated a recommendation of a new item. Studies show that item-based explanations improve users' acceptance of recommendations [Herlocker et al., 2000] and help users make accurate decisions [Bilgic and Mooney, 2005]. As discussed in [Tintarev and Masthoff, 2007a], a shortcoming of item-based explanations is that users may not understand the connection between the explaining items and the recommended item.

Item-based approaches have several properties that we also use in our tag-based approach. First, item-based approaches present the relationship between the user and the set of related items in a way that users can easily interpret. In the Netflix example, the relationship is defined by the rating the user has given to each movie shown in the explanation. We use a similar approach, by expressing the relationship between user and tag as a 1-to-5 star inferred rating. Second, item-based explanations often use ratings correlation as a criteria for choosing the related items. We use a similar approach for Tagsplanations by selecting tags with preference values that strongly correlate with ratings for the item.

User-based explanations. User-based explanations utilize other users as intermediary entities. The relationship between the main user and the explaining users is typically that they share similar tastes, and the relationship between the explaining users and the recommended item is that they have rated the item positively. For example, Herlocker et al. developed an explanation facility that displays a histogram of ratings of the recommended item by other users with similar taste [Herlocker et al., 2000]. This approach was successful at persuading users to try an item, but less effective at helping users make accurate decisions [Bilgic and Mooney, 2005]. Motivated by these results, we study how well Tagsplanations help users make accurate decisions.

Feature-based explanations. Feature-based approaches use qualities or characteristics of the recommended item as intermediary entities. Two types of features have been used in recommendation explanations:

- **Predefined categories.** Tintarev et al. developed a movie recommender prototype that explains recommendations by using categories such as genre, director, and cast [Tintarev, 2007]. While these features have tremendous potential for use in recommendation explanations, using predefined categories like these also presents several challenges. Shirky describes several such issues: (1) experts or system designers are needed to create the categorization scheme, (2) the categorization scheme might not be sufficiently descriptive, and (3) the categories may be too static to accommodate changing needs [Shirky, 2005]. A tag-based approach addresses these issues by putting control in the hands of the users and allowing free-form metadata.

- **Keywords.** Many types of items, such as books, articles, or websites, contain textual content that may be mined for keywords [Mooney and Roy, 2000, Billsus and Pazzani, 1999]. Keyword-based approaches use these words as features in the explanation. The Libra book recommender, for example, extracts keywords from the book text based on their predictive power in a naive Bayesian classifier and uses them to explain the recommendation [Mooney and Roy, 2000]. Bilgic et al. showed that these explanations helps users make more accurate decisions [Bilgic and Mooney, 2005].

  Explanations using item content face several challenges, many of which may be addressed by using tags. One limitation is that items such as music or pictures may not have readily available textual content. In contrast, tags may be applied to virtually any type of item. A second issue is that keywords extracted from content represent data rather than metadata, and therefore may be too low-level. In one example described in [Bilgic and Mooney, 2005], the keywords used to explain a book recommendation are *Heart*, *Beautiful*, *Mother*, *Read*, *Story*. While these words do suggest qualities of the book, meta-level tags such as *fiction*, *mother-daughter relationship*, or *touching* might be more suitable.

We considered adapting keyword-style approaches to generate Tagsplanations. Existing keyword-style approaches [Mooney and Roy, 2000, Bilgic and Mooney, 2005] only require that an item have a set of words and corresponding frequencies. If tags were used rather than words, the same algorithm could be utilized. We chose to develop a new approach for several reasons. First, existing keyword-style approaches require that users rate items using binary categories, such as {*like*, *dislike*}. In the MovieLens recommender system used in our study, users rate items on a 5-star scale. Second, existing approaches do not account for two issues that tagging systems face: low quality tags [Sen et al., 2007] and redundant tags [Golder and Huberman, 2006]. Third, we wished to represent the relationships between tag and item and between user and tag in a more intuitive way than in existing keyword-style approaches. For example, the Libra book recommender display the *strength* of each keyword, which equals the count of the word multiplied by the weight assigned by a naive Bayesian classifier. While these strength values may reflect a user's preferences towards the keywords, users may have difficulty interpreting the values.

3.3 Design space

The goal of explanations of recommendations is to relate the recommended item to the user's tastes and interests. As discussed in the introduction, a common approach for establishing this relationship is to use an intermediary entity that relates to both user and recommended item. For Tagsplanations, the intermediary entity is a tag. Tagsplanations must clearly identify the relationship between user and tag and between tag and recommended item.

A recommendation explanation may be one of two types: description or justification. *Descriptions* reveal the actual mechanism that generates the recommendation. For example, k-nearest-neighbor item-item algorithms generate recommendations based on a user's ratings for the k items most similar to the recommended item [Sarwar et al., 2001]. In this case, an item-based explanation could be used to accurately depict the algorithm. *Justifications*, on the other hand, convey a conceptual model that may differ from that

of the underlying algorithm. For example, a book recommender might use a k-nearest-neighbor item-item algorithm to recommend books, but may justify a recommendation based on the fact that the book was written by a user's favorite author.

While descriptions provide more transparency than justifications, there are several reasons why justifications might be preferred. First, the algorithm may be too complex or unintuitive to be described in terms that a user can understand. For example, dimensionality reduction algorithms generate recommendations based on a set of latent factors in the data that may not have a clear interpretation [Ellenberg, 2008]. User-based nearest neighbor algorithms recommend items based on a user's similarity to other users [Resnick et al., 1994], but these other users are generally not familiar to the given user. Second, the system designers may want to keep aspects of the algorithm hidden, for example to protect trade secrets. Third, using justification allows a greater freedom in designing explanations since they are not constrained by the recommender algorithm. However, there are ethical considerations in presenting justifications over explanations; the system could potentially be lying to the user to induce desired behavior rather than help them understand the recommendation. We don't address the ethical concern in this thesis, but suggest it as an area for future research.

3.3.1 Relationship between item and tag: tag relevance

We use the term *tag relevance* to describe the relationship between a tag and an item. One possible measure of relevance is tag popularity. That is, have many users applied a given tag to an item or only a few users? A tag applied by many users is probably more relevant to the given item. Another potential measure of relevance is the correlation between item preference and tag preference. That is, do users who like a given tag also tend to like the associated item? A strong correlation may suggest that the tag is highly relevant. While both tag popularity and preference correlation may indicate tag relevance, the two measures may not agree with one another. On MovieLens, the tag *Bruce Willis* is unpopular for the film *Predator* (no one has applied the tag to this movie); we suspect this is because Bruce Willis is not in *Predator*. However, a strong correlation may exist between users' preference for the tag *Bruce Willis* and their preference for *Predator* because *Predator* is the type of movie Bruce Willis is often in.

Tag relevance may be represented as either a binary relationship (*relevant*, *not relevant*) or as a value on a continuous scale. While a binary relationship might convey a simpler conceptual model, it lacks the precision of a continuous scale. Users may wish to know *how* relevant a given tag is, rather than just whether it is relevant or not.

3.3.2 Relationship between user and tag: tag preference

We define tag preference to be the user's sentiment towards a tag. The key design choices concern how to represent this relationship and how to compute the value for a given user and tag. Tag preference may be modeled as a binary relationship (*like*, *dislike*) or as a continuous variable representing the degree to which the user likes or dislikes a tag. A binary approach provides a simpler conceptual model to users, but is less precise than a continuous approach. A user's preference towards a tag may be computed in one of two ways. Preference may be assessed directly, by asking a user his or her opinion of a tag, or it may be inferred based on a user's actions. For example, if a user tends to give high ratings to items that have a particular tag, the system could infer that he or she has a positive preference toward that tag. An advantage of inferring tag preference over asking users directly is that no additional work is needed from the users.

3.4 Design decisions

3.4.1 Platform

We used the MovieLens movie recommender website (see Section 2.2) as a platform for implementing Tagsplanations. MovieLens members rate movies on 5-star scale and receive recommendations based on a k-nearest-neighbor item-item recommender algorithm. MovieLens does not take tags into account when generating movie recommendations; therefore, tag-based explanations serve as a justification rather than a description of the underlying recommender algorithm.

3.4.2 Tag Preference

In this section we first give an overview of our approach, followed by formal definitions. As previously discussed, tag preference may be computed in one of two ways. Tag preference may be measured directly, by asking a user his or her opinion of a tag, or it may be inferred based on user behavior. We use the latter approach, in order to spare users from having to explicitly evaluate many different tags. Specifically, we infer users' tag preferences based on their movie ratings. We use movie ratings because they are the central mechanism on MovieLens by which users express preferences and because they are in abundant supply (the average MovieLens user has rated 75 movies).

To estimate a user's preference for a tag, we compute a weighted average of the user's ratings of movies with that tag. For example, suppose we are estimating Alice's preference for the tag *violence* and Alice has rated the following movies tagged with *violence*: *Planet of the Apes*, *Reservoir Dogs*, *Sin City*, and *Spider-Man 2*. Her inferred preference toward *violence* is a weighted average of those four movie ratings. We chose to use a weighted average for two reasons. First, it provides a simple and computationally efficient algorithm for computing tag preference. Second, it guarantees that the computed tag preference values will lie in the same 0.5 - 5.0 range as the movie ratings, since the output value of a weighted average is bounded by the minimum and maximum input values. This allows us to represent tag preference values using the familiar 5-star paradigm used for movie ratings. Although we considered a binary representation (*like*, *dislike*), we chose the 5-star scale because it provides a fine-grained level of information yet is easy to interpret since it follows the standard of the movie rating scale. Previous studies have shown that users prefer fine-grained rating scales [Cosley et al., 2003].

We weight the average because some movie ratings may suggest tag preference more strongly than others. For example, *Reservoir Dogs* is much more violent than *Planet of the Apes*, so a user's rating for *Reservoir Dogs* is probably a stronger indicator of their preference toward *violence* than their rating for *Planet of the Apes*, even though both movies have been tagged with *violence*. We use tag frequency to measure the relative importance of a movie in determining tag preference. For example, MovieLens users have tagged *Reservoir Dogs* with *violence* 7 times, while *Planet of the Apes* has only been tagged once. Therefore we assign a higher weight to *Reservoir Dogs* when computing a user's preference for *violence*.

We now provide formal definitions for the concepts described above. First, we define *tagshare*[3] , a measure of tag frequency that is used to assign weights to movies. The *tagshare* of a tag t applied to an item i is the number of times t has been applied to i, divided by the number of times *any* tag has been applied to i. We denote this value as tag_share(t, i). For example, on MovieLens the tag *Bruce Willis* has been applied to the movie *Die Hard* 8 times and the number of applications of all tags to *Die Hard* is 56. Therefore tag_share(*Bruce Willis*, *Die Hard*) equals $8/56 = 0.14$. We add a constant smoothing factor of 15 to the denominator when computing tagshare, in order to reduce the value when an item has a small number of tags[4] . This smoothing reflects the possibility that applications of a tag may be due to chance.

We now formally define the measure we use to estimate tag preference, which we call *tag_pref*. Let I_u to be the set of items that user u has rated, let $r_{u,i}$ to be the rating user u has given to item i, and let $\bar{r}_u$ be user u's average rating across all items. User u's tag preference for tag t is computed as follows:

$$\text{tag\_pref}(u,t) = \frac{\left(\sum_{i \in I_u} r_{u,i} \cdot \text{tag\_share}(t,i)\right) + \bar{r}_u \cdot k}{\left(\sum_{i \in I_u} \text{tag\_share}(t,i)\right) + k}$$

tag_pref(u, t) is undefined if user u has not rated any items with tag t. k is a smoothing constant that we assign a value of 0.05[4]. The smoothing constant k accounts for users who have rated few items with a given tag. This smoothing serves to bring the computed tag preference closer to the user's average rating, because ratings of a small number of items may not properly reflect a user's tag preference.

3.4.3 Tag Relevance

As discussed earlier, two possible measures of tag relevance are tag popularity and preference correlation. Tag popularity reflects the number of users who have applied the tag to the movie, while preference correlation is the correlation between users' preference for the tag and their preference for the movie. We chose to use preference correlation to represent tag relevance, because it directly relates tag preference (the relationship between user and tag) with item preference (the relationship between user and item). To

[3] The term *tagshare* was first used by Tim Spalding from LibraryThing in a blog post on February 20, 2007: http://www.librarything.com/thingology/2007/02/when-tags-works-and-when-they-dont.php

[4] We chose smoothing constants based on qualitative analysis over a series of test cases.

determine the preference correlation between item i and tag t, we compute the Pearson correlation between the sequence of ratings for i across all users and the sequence of inferred tag preference values for t across all users. Tag popularity is used implicitly as a filtering criteria; we assign a relevance of zero to tags that have not been applied at least once to a given item. We represent tag relevance on a continuous scale rather a binary one (*relevant*, *not relevant*), because this allows users to discern the relative importance of one tag versus another when both tags are relevant to the item.

We now formally define our measure of tag relevance, which we call *tag_rel*. For a given tag t and item i, we define U_{ti} to be the subset of users who have rated i and have a well-defined tag preference for t. (Users must have rated at least one item with tag t in order to have a well-defined tag preference for t.) We define X to be the set of ratings for item i across all users in U_{ti}, adjusted by each user's average rating. That is, $X = \{r_{u,i} - \bar{r}_u : u \in U_{ti}\}$. We subtract each user's average rating to account for individual differences in rating behavior. We define Y to be the set of inferred tag preference values[5] toward tag t for all users in U_{ti}, adjusted by each user's average rating. That is, $Y = \{\text{tag\_pref}(u,t) - \bar{r}_u : u \in U_{ti}\}$. Tag_rel is defined using Pearson correlation, as follows:

$$\text{tag\_rel}(t,i) = \begin{cases} \text{pearson}(X,Y), & \text{if } t \text{ has been applied to } i; \\ 0, & \text{otherwise.} \end{cases}$$

3.4.4 Tag Filtering

We filter tags based on three criteria:

- **Tag quality.** One study showed that only 21% of tags on MovieLens were of high enough quality to display to the community [Sen et al., 2007]. We filtered tags based on implicit and explicit measures of tag quality. First, we require that a tag has been applied by at least 5 different users and to at least 2 different items. Second, we require that the average thumb rating of a tag across all items satisfy a minimum threshold. As discussed earlier, MovieLens members use thumb ratings to give explicit feedback about tag quality. We used a smoothed average

[5] When computing tag preference values for t, we excluded item i in order to avoid spurious correlations with ratings for i.

of thumb ratings as described in [Sen et al., 2007] and retained the top 30% of tags. However, we chose not to filter tags representing movie genres or names of directors and actors. MovieLens members tended to rate these tags poorly, but we suspect this is due to the fact that genre, cast, and director are displayed next to each film, and tags containing the same information might appear redundant. For Tagsplanations, we do not display this information and therefore such tags would not be redundant.

- **Tag redundancy.** Different tags may have very similar meanings [Golder and Huberman, 2006]. These similarities arise when two tags are synonyms (*film*, *movie*), different forms of the same word (*violence*, *violent*), or at different levels of specificity (*comedy*, *dark comedy*). Our process for filtering redundant tags consists of three steps: preprocessing, redundancy detection, and winner selection. In the preprocessing step, we stem[6] the words in each tag and remove stopwords[7] . In the redundancy detection step, we use a simple heuristic: if two tags in the same explanation contain any of the same words or differ from one another by only 1 character, they are classified as redundant. The winning tag is the one with higher relevance to the given item.

- **Usefulness for explanation.** We removed all tags with an undefined value for tag preference (which occurs when a user has not rated any items with that tag) or with a tag relevance score less than 0.05. We also limited the number of tags we show in each Tagsplanation to 15, in order to conserve screen space and to avoid overloading users with too much information.

3.4.5 Interface

Figure 3.2 shows an example of the Tagsplanation interface. Tag relevance is represented as a bar of varying length, while tag preference is depicted as a number of stars rounded to the nearest half. An arrow indicates sort order of the tags. We designed four variations of the interface, which differ in the data displayed (tag relevance, tag preference, or both) and the sorting order (tag relevance or tag preference):

[6] We used Porter's stemming algorithm as implemented at http://nltk.sourceforge.net

[7] We used a standard list of stopwords from http://nltk.sourceforge.net plus one domain-specific stopword: "movie"

3.5 Experiment

We conducted a within-subjects study of each of the four interfaces: RelSort, PrefSort, RelOnly, and PrefOnly. Subjects completed an online survey in which they evaluated each interface on how well it helped them (1) understand why an item was recommended (*justification*), (2) decide if they would like the recommended item (*effectiveness*), and (3) determine if the recommended item matched their mood (*mood compatibility*). Based on survey responses, we draw conclusions about the role of tag preference and tag relevance in promoting justification, effectiveness, and mood compatibility.

- **Interface 1: RelSort.** Shows relevance and preference, sorts tags by relevance (Fig. 3.2)
- **Interface 2: PrefSort.** Shows relevance and preference, sorts tags by preference (Fig. 3.3)
- **Interface 3: RelOnly.** Shows relevance only, sorts tags by relevance (Fig. 3.4)
- **Interface 4: PrefOnly.** Shows preference only, sorts tags by preference (Fig. 3.5)

3.5.1 Methodology

We invited 2,392 users of MovieLens to participate in the study, which was launched on 08/19/2008. We only included members who had logged in within the past 6 months and who had rated at least 50 movies. 556 users accepted the invitation. In the study, we included movies with at least 10 tags (after tag filtering) and between 1000 and 5000 movie ratings. 93 movies satisfied these conditions. We placed bounds on the number of ratings for each movie in order to find movies that were neither very popular nor obscure, because we wished to provide a mix of movies that some subjects had seen and others had not seen. Clearly a live system would not use such filters. However, we found there was no statistically significant difference in survey responses for movies with a higher number of ratings (4000 to 5000 ratings) versus movies with a lower number of ratings (1000 to 2000). We leave it to future work to determine the minimum amount of rating and tagging data needed to produce good-quality Tagsplanations.

Figure 3.2: RelSort interface, for movie *Rushmore*. (The list of tags is truncated.)

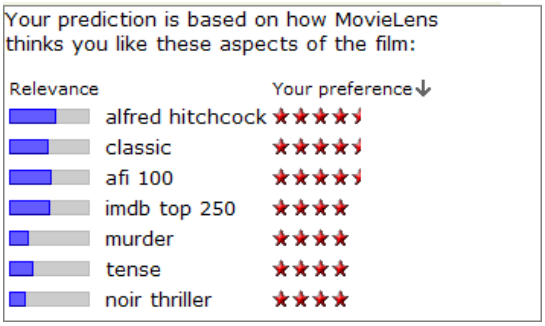

Figure 3.3: PrefSort interface, for movie *Rear Window*.

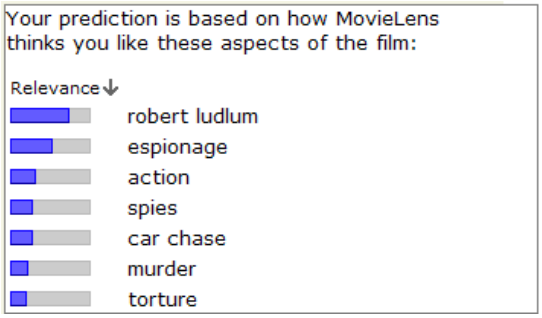

Figure 3.4: RelOnly interface, for movie *The Bourne Ultimatum*.

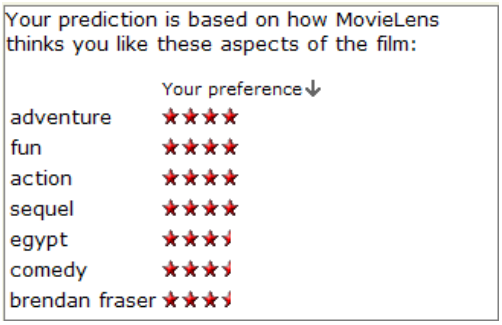

Figure 3.5: PrefOnly interface, for movie *The Mummy Returns*.

Each participant took a personalized online survey, consisting of three parts:

- **Part 1: Evaluation of Tagsplanations**

  We showed each subject Tagsplanations for 4 different movies, drawn randomly from the pool of movies that satisfied the filtering conditions described above. Further, we only showed users a Tagsplanation if they had a well-defined tag preference for at least 7 of the tags (reducing the number of eligible movies per user from 93 to 83, on average). Subjects only saw Tagsplanations for movies they *had not* seen, and they verified whether or not they had seen each movie. Each of the 4 Tagsplanations utilized a different interface (RelSort, PrefSort, RelOnly, or PrefOnly) so that each subject would see all 4 interfaces. To account for order effects, we presented the 4 interfaces in a random order.

  For each Tagsplanation, participants responded to the following statements using a 5-point Likert scale[8] :

  1. *This explanation helps me understand my predicted rating.* (Justification)
  2. *This explanation helps me determine how well I will like this movie.* (Effectiveness)
  3. *This explanation helps me decide if this movie is right for my current mood.* (Mood compatibility)

- **Part 2: Evaluation of particular tags**

  We showed users three randomly chosen tags from the last Tagsplanation they saw in Part 1 of the study. For each tag, they responded to the following statements using a 5-point Likert scale:

  1. *This **element** helps me understand my predicted rating.*
  2. *This **element** helps me determine how well I will like this movie.*
  3. *This **element** helps me decide if this movie is right for my current mood.*

[8] The possible responses were: 1 = strongly disagree, 2 = disagree, 3 = neutral, 4 = agree, 5 = strongly agree.

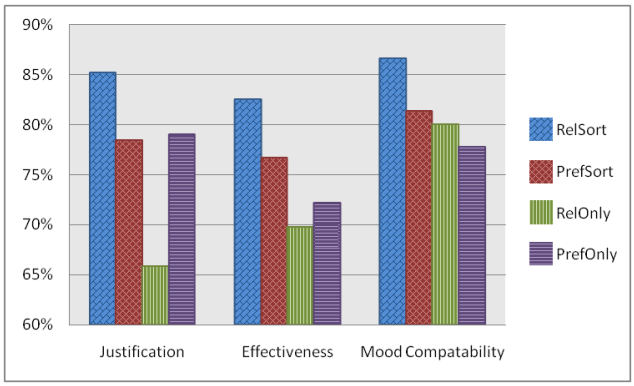

Figure 3.6: Percentage of responses that were *agree* or *strongly agree*, broken down type of interface (RelSort, PrefSort, RelOnly, PrefOnly). Neutral responses were excluded from the calculations.

- **Part 3: Verification of Tagsplanation accuracy**

  Subjects evaluated Tagsplanations for two movies they had seen in the past, drawn from the same pool of movies described above. The purpose was to verify the accuracy of Tagsplanations, since subjects had otherwise only evaluated Tagsplanations for movies they *had not* seen. For each Tagsplanation, subjects responded to the following statement using a 5-point Likert scale:

  Overall this is a good explanation given my knowledge of the movie.

3.5.2 Results

In Part 1 of the study, users responded to a series of statements about 4 different Tagsplanation interfaces. Figure 3.6 shows the percentage of responses that were either *agree* or *strongly agree* for each type of statement (justification, effectiveness, mood compatibility), broken down by interface (RelSort, PrefSort, RelOnly, and PrefOnly). Neutral responses, which accounted for less than 30% of total responses in each category, were excluded from the calculations.

RelSort performed best in all three categories (justification, effectiveness, and mood-compatibility) by a statistically significant margin[9] ($p \leq 0.005$, $p \leq 0.05$, and $p \leq 0.05$ respectively). RelOnly scored lowest in justification by a statistically significant margin

[9] To determine statistical significance we used the Z-test of two proportions.

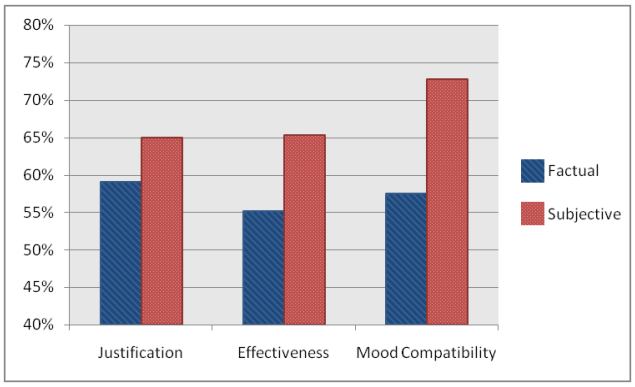

Figure 3.7: Percentage of responses that were *agree* or *strongly agree* for each type of statement, broken down by tag type (factual, subjective). Neutral responses were excluded from the calculations.

($p \leq 0.001$). PrefSort scored higher than RelOnly in effectiveness by a statistically significant margin ($p \leq 0.01$). None of the other differences were statistically significant.

In Part 2 of the study, users responded to a series of statements about individual tags. Figure 3.7 shows the percentage of responses that were *agree* or *strongly agree* for each type of statement (justification, effectiveness, and mood compatibility), summarized by type of tag (subjective or factual).[10] In all three categories, subjective tags outperformed factual tags by a statistically significant margin (justification: $p \leq 0.05$, effectiveness: $p \leq 0.001$, mood compatibility: $p \leq 0.001$). Tables 3.1 and 3.2 show the 10 best and worst performing factual and subjective tags, based on the percentage of responses that were *agree* or *strongly agree* across all types of statements. Neutral responses were excluded from the calculations.

Results from Part 3 of the study reveal that 81.7% of respondents agreed with the statement "*Overall this is a good explanation given my knowledge of the movie*". Neutral responses, which accounted for less than 20% of all responses, were excluded from this calculation. Broken down by interface, the percentage of agreement was 85.4% for RelSort, 80.3% for PrefSort, 75.7% for RelOnly, and 85.7% for PrefOnly.

[10] In previous work [Sen et al., 2007], tags were manually coded as subjective or factual by two independent coders.

Top 10	**score**
afi 100 (laughs)	100.0%
fantasy world	100.0%
world war ii	100.0%
sci-fi	95.2%
action	94.4%
psychology	93.8%
disney	91.7%
satirical	88.5%
drama	87.5%
satire	86.4%

Bottom 10	**score**
male nudity	0.0%
narrated	7.7%
los angeles	12.5%
directorial debut	16.7%
childhood	17.6%
matt damon	20.0%
movie business	25.0%
afi 100	25.5%
new york city	26.7%
amnesia	28.6%

Table 3.1: 10 best and worst factual tags, based on percentage of responses that were *agree* or *strongly agree*. Includes tags with at least 10 responses.

Top 10	**score**
great soundtrack	90.9%
fanciful	90.9%
funny	90.0%
poignant	88.9%
witty	88.0%
dreamlike	87.5%
whimsical	87.5%
dark	87.3%
surreal	86.7%
deadpan	84.2%

Bottom 10	**score**
sexy	15.4%
intimate	25.0%
passionate	25.0%
lyrical	30.8%
meditative	33.3%
brilliant	41.2%
gritty	44.4%
understated	45.5%
fun	50.0%
reflective	50.0%

Table 3.2: 10 best and worst subjective tags, based on percentage of users who agreed with statements about tag. Includes tags with at least 10 responses.

3.5.3 Discussion

RQ-Justification: What is the role of *tag preference* and *tag relevance* in helping users understand their recommendation?

The results in Figure 3.6 suggest that tag preference is more important than tag relevance for justifying recommendations. RelOnly, which only shows tag relevance, performed worst by a significant margin. PrefSort and PrefOnly received virtually the same score, even though PrefSort includes tag relevance and PrefOnly does not. Users may prefer seeing tag preference because they are skeptical that a recommender system can accurately infer their preferences. According to one user, "*The weights (relevance) don't do much good without the values (preferences) they're applied to when it comes to understanding the predicted rating. This is because what movielens thinks my preferences are might differ greatly from my actual preferences*".

However, users preferred *sorting* by tag relevance. (RelSort significantly outperformed PrefSort.) Thus tag relevance appears to serve best as an organizing principle for helping users understand recommendations. One subject commented: "*sorting by relevance ... feels more useful in terms of seeing how it actually comes up with the final score.*"

RQ-Effectiveness: What is the role of *tag preference* and *tag relevance* in helping users determine if they will like the item?

For effectiveness, tag preference and tag relevance appear to be of roughly equal importance, based on the fact that PrefOnly and RelOnly received similar scores. It is surprising that tag preference would be as important as tag relevance, since users should know their own preferences. One possible reason is that showing tag preference promotes *efficiency* [Tintarev and Masthoff, 2007b], since users can more quickly spot the tags they might like or dislike. Another possibility is that users did not understand that tag preference is user-specific rather than item-specific, and they might have incorrectly thought that tag preference revealed information about the movie. One user commented: "*It is not clear if the preference ratings given are for the aspect in general, or for the value of the aspect exemplified by this movie.*" Again, subjects preferred tag relevance as a sort order for effectiveness. One subject commented: "*I like the relevance, it gives the*

breakdown of what elements the film has and you can gauge what you'll be experiencing when you watch it."

RQ-Mood: What is the role of *tag preference* and *tag relevance* in helping users decide if an item fits their current mood?

As Figure 3.6 shows, RelOnly performed significantly better for mood compatibility than it did for effectiveness or justification. This suggests that relevance plays its most important role in helping reveal mood compatibility. One user commented: "*Exposing the supposedly relevant facets ... allows me to double check relevance against current interests and mood.*" Based on the superior performance of RelSort over PrefSort, tag relevance is again the preferred sort order.

RQ-Tag-Type: What types of tags are most useful in Tagsplanations?

Figure 3.7 shows that subjective tags performed better than factual tags in all three categories. However, these results contradict prior work that showed users prefer factual tags over subjective tags [Sen et al., 2007]. One possible reason is that we filter out tags of low quality and low relevance, while the prior study did not. Subjective tags that survived the filtering step may compare more favorably to factual tags. Another possible reason for the discrepancy is context; subjects in our study evaluate tags in the context of specific tasks (understanding the recommendation, deciding if they will like the item, assessing mood compatibility), while subjects in the earlier study rated tags based on whether they should be shown in a general setting. For these three tasks, in particular assessing mood compatibility, users may prefer subjective tags, while factual tags may be preferred in a more general setting.

While subjective tags generally outperformed factual ones, anecdotal evidence suggests that users prefer a factual tag over a subjective tag if both capture the same idea. For example, users preferred the factual tag *sexuality* (64.9% agreement over all categories) to the subjective tag *sexy* (15.4% agreement), and users preferred the factual tag *violence* (73.8% agreement) to the subjective tag *violent* (57.9% agreement).

Based on the top-rated factual tags in Table 3.1, users prefer factual tags that describe genres (*sci-fi*, *action*, *drama*) or general themes (*world war ii*, *psychology*). Based on the lowest-rated tags, subjects disliked factual tags that were highly specific (*los angeles*, *new york city*, *amnesia*, *movie business*) or tags that describe meta-level

information (*directorial debut*, *narrated*, *afi 100*)

Based on the top-rated subjective tags in Table 3.2, subjects preferred subjective tags that are descriptive (*surreal*, *dreamlike*, *deadpan*) or suggest mood (*dark*, *whimsical*, *poignant*). Based on the lowest rated tags, users dislike subjective tags with sexual themes (*sexy*, *intimate*, *passionate*) or tags that express opinion without providing a description (*brilliant*, *fun*).

Beyond answering these specific research questions, the study also demonstrated the value of Tagsplanations. Among survey respondents who expressed an opinion, over 80% agreed that the RelSort interface achieved the goals of justification, effectiveness, and mood compatibility. Over 85% agreed that the RelSort interface provided a good explanation given their prior knowledge of a movie. One user commented: "*This is as good a description of Waking Life as I could imagine being put together in a dozen words.*"

3.6 Summary

Our study showed that tag relevance and tag preference both play a key role in Tagsplanations. Tag relevance serves best in an organizing role, and both tag relevance and tag preference help promote the goals of justification, effectiveness, and mood compatibility. The study also demonstrated the viability of Tagsplanations. Over 80% of respondents who expressed an opinion agreed that the RelSort interface helped them (1) understand why an item was recommended, (2) decide if they would like the item, and (3) determine if an item fit their current mood.

One limitation of this study is that all data is self-reported. Rather than *asking* subjects how well Tagsplanations promote effectiveness and mood compatibility, one could measure these objectives empirically. Bilgic et al. developed a method for quantifying how well explanations help users make accurate decisions [Bilgic and Mooney, 2005]. One could use this same approach to test how well Tagsplanations promote effectiveness.

Chapter 4

The Tag Genome

4.1 Introduction

In the previous chapters we showed how to use tags to explain recommendations to users. One of the challenges we faced is that tags are not designed for the purpose of explaining recommendations; therefore we had to develop algorithms to adapt tags for this specific purpose. For example, we computed *tag relevance* values that measure the relationship between tags and items on a continuous scale.

There are many other novel and exciting tagging applications that go beyond tag browse and search. However – like Tagsplanations – the traditional tagging model does not fully support these applications. Whereas in the previous chapter we showed how to adapt tags for a specific application, in this chapter we introduce a more general construct called the *Tag Genome* that enables a wide variety of end-user applications. In Chapters 5 and 6 we present a set of concrete applications of the Tag Genome.

We begin by discussing two basic limitations of the traditional tagging model:

Tags are binary. In traditional tagging systems, users may only express binary relationships between tags and items. For example, a user may apply the tag *violent* to *Reservoir Dogs*, indicating that it is a *violent* movie, but she may not indicate *how* violent the movie is. This makes it difficult for a system to determine the relative importance of various tags. For example, if both *Reservoir Dogs* and *The Usual Suspects* have been tagged with violent, how does the system know which is more violent than

the other? One way to extract additional meaning from tags is to consider the frequency with which users apply tags; if many users apply a particular tag, the tag is probably more relevant than if only one or two users have applied it. However, tag frequency is a noisy signal of tag relevance; many factors can influence whether a user decides to apply a particular tag, including the existing tags in the system, the popularity of the item, and the user's personal tendencies [Sen et al., 2006, Golder and Huberman, 2006]. Also, the meaning of tag frequency may not be obvious to the end user; if the tag *violent* has been applied 5 times to *Reservoir Dogs* out of 25 total tag applications, does that mean that *Reservoir Dogs* is extremely violent, or only moderately violent?

Tags are sparse. Because users apply tags voluntarily, there is no guarantee that all of the relevant tags for an item will actually be applied to that item. Further, tagging systems only capture positive relationships between tags and items; users may express that a tag is relevant by applying the tag to the item, but they have no way of indicating that a tag is *not* relevant. These two factors make it difficult to draw any conclusions about tags that have *not* been applied to an item; it may mean that the tag is not relevant to the item, or it may simply mean that no one has gotten around to applying it yet.

4.1.1 Extending the traditional tagging model: the Tag Genome

To address these limitations, we introduce a new data structure called the *Tag Genome*. The Tag Genome records how strongly each tag applies to each item on a continuous scale from 0 to 1 (0 = does not apply at all, 1=applies very strongly), which we call the *relevance* of the tag to the item. Just as a biological genome encodes an organism based on a sequence of genes, the Tag Genome encodes each item based on its relationship to a common set of tags. In contrast to the traditional tagging model, the Tag Genome provides a continuous and dense mapping between tags and items. We describe the Tag Genome in greater detail in Section 4.3.

Movie Tuner, described in the next chapter, is just one example from the broad space of potential applications of the Tag Genome. Another potential application is a refined version of tag search: rather than just specify a tag to search for, users could also indicate the desired level of that tag. For example, someone might search for a

horror movie with "a medium level of *gore* or less". Systems could also use the Tag Genome to help users compare items; for example, if a user is deciding between the movies *WALL-E* and *9*, the system could tell the user that both movies are *animated*, *futuristic*, and *dystopian*, but *WALL-E* is more *humorous* and *cute*, while *9* is more *dark* and *violent*. We describe these and several other potential applications of the Tag Genome in Chapter 6.

4.1.2 Computing the Tag Genome

We present a supervised learning approach for computing the Tag Genome, and we apply this approach to computing the Tag Genome in the movie domain. Using a variety of user-contributed data (tag applications, text-based reviews, and ratings) as inputs, we train a learning model based on users' judgments of tag relevance for a subset of tag-item pairs. We refer to this approach as *community-supervised learning*, since the user community provides both the input data to the learning model as well as the ground truth data set used to train the model.

4.2 Related Work

4.2.1 Tag prediction

Tag prediction algorithms address the problem of tag sparsity by automatically predicting the relevant tags for an item rather than relying on users to come up with the tags on their own [Jschke et al., 2007, Sigurbjörnsson and van Zwol, 2008, Wu et al., 2009]. Below we present several examples from this line of research that span a variety of domains.

Heyman et al. explore techniques for predicting tags for URLs on the website del.icio.us[1] [Heymann et al., 2008]. They treat tag prediction as a binary classification task; for each tag, the positive training examples comprise URLs to which the tag has been applied, and the negative training examples comprise the remaining URLs. They build a separate classification model for each tag using support vector machines with input features extracted from page text, anchor text, and the surrounding host.

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

They also use association rule mining to predict tags for an URL based on the other tags that have been applied to the URL.

Eck et al. explore the problem of predicting tags for musical artists [Eck et al., 2007]. They build a multi-class classifier that predicts whether an artist has "none", "some", or "a lot" of a particular tag. To construct the training set, they label each artist as having "none", "some", or "a lot" of each tag based on how frequently that tag has been applied to the artist's songs. They assign these labels in such a way that there are an equal number of artists assigned to each of the three labels for a particular tag. They then train a classifier using Adaboost with single-level decision trees as weak learners; acoustic features extracted directly from MP3 files serve as the inputs to the classifier.

Ulges et al. developed a system to predict tags for videos from YouTube[2] [Ulges et al., 2008] and other sources. They first segment videos into key frames and extract visual features including color, texture, and motion. They then build a nearest-neighbor classifier that uses these features to measure similarity between videos. They also extract "visual words" that represent characteristics of local patches of images, and compute the probability of each tag given the visual words in an image. They then combine the output of both types of classifiers using a weighted sum.

For computing the Tag Genome, we also use machine learning to infer the relationships between tags and items. Like [Heymann et al., 2008], we extract features from the text associated with an item, as well as the other tags that have been applied to each item. Just as [Eck et al., 2007] and [Ulges et al., 2008] leverage domain-specific features from audio or video files, we leverage the data source most abundant in recommender systems: ratings. However, instead of predicting a binary (*applies*, *does not apply*) or a ternary (*none*, *some*, *a lot*) relevance value, the learning model for the Tag Genome predicts relevance values on a continuous $0 - 1$ scale. Further, the model for the Tag Genome is trained using explicit relevance ratings from individual users rather than relevance values computed based on tag frequency.

4.2.2 Vector Space Model

First introduced in [Salton et al., 1975], the vector space model (VSM) has become a standard model in the field of information retrieval. The VSM represents each document

[2] http://www.youtube.com

from a corpus as a vector of terms and associated term weights; the terms are typically individual words in the corpus, and the term weights reflect the importance of the term to the document. For example, the tf-idf weighting scheme weights terms proportionally to their frequency in the document and inversely to the number of documents they appear in overall [Salton and McGill, 1983]. The collection of term weights for all the documents in a corpus is typically represented as a term-document matrix.

The primary applications of the VSM are document retrieval and document clustering. Because the VSM represents documents as vectors, vector similarity measures such as cosine similarity may be used to measure similarity between two documents or between a document and a search query (which may also be expressed as a term vector). Matrix factorization approaches such as Latent Semantic Indexing (LSI) [Deerwester et al., 1990] can help overcome the sparsity issues of the VSM. LSI uses singular value decomposition to transform term vectors into a lower dimensional concept space; one can then measure the conceptual similarity between two documents or between a document and a query based on the cosine similarity in this lower dimensional space.

In some sense, the Tag Genome is a type of vector space model; the tags in the genome serve as terms, and their respective relevance values provide term weights. The tag-item matrix of the Tag Genome (see Figure 4.1) is analogous to the term-document matrix for the VSM. Just as term vectors in the VSM can be used to measure similarity between documents, the Tag Genome can be used to measure similarity between arbitrary items; in Section 5.4.2 we describe how to compute the cosine similarity between two Tag Genomes using a term-weighting function based on tf-idf. The primary difference between the Tag Genome and the VSM is that the weights in the genome are computed from a machine learning algorithm trained on human judgments of tag relevance, whereas the weights in the VSM are based on term frequency.

4.3 The Tag Genome

In this section we formally define the Tag Genome and present general guidelines for computing the Tag Genome. In Section 4.4 we apply these guidelines to compute the Tag Genome in the movie domain.

4.3.1 Definition

We now formally define the Tag Genome. If T is a set of tags and I is a set of items, we quantify the relationship between each tag $t \in T$ and item $i \in I$ by the *relevance* of t to i, denoted as $rel(t, i)$. $rel(t, i)$ measures how strongly tag t applies to item i on a continuous scale from 0 (does not apply at all) to 1 (applies very strongly). In the movie domain[3] , for example, $rel(\textit{violent}, \textit{Reservoir Dogs}) = 0.98$, $rel(\textit{violent}, \textit{The Usual Suspects}) = 0.65$, and $rel(\textit{violent}, \textit{A Cinderella Story}) = 0.03$.

The Tag Genome G is the collection of relevance values for all tag-item pairs in $T \times I$, represented as a tag-item matrix as shown in Figure 4.1. We define G such that

$$G_{t,i} = rel(t, i) \;\; \forall t \in T, i \in I$$

We also use the term Tag Genome to describe the vector of tag relevance values for a particular item i, which we denote as G_i:

$$G_i = \big\langle rel(t_1, i), \ldots, rel(t_m, i) \big\rangle \;\; \forall t_j \in T$$

Each G_i represents a single column in the matrix G.

When we use the term Tag Genome in the context of a particular item, we are referring to G_i; otherwise we are referring to the full matrix G. This reflects the dual meaning of the term genome in a biological context; genome can refer to the genetic code of a particular organism, or it can refer to common genetic structure shared across a species, e.g. the *human genome*. In the same way that a biological genome may be defined for a species, we define the Tag Genome for a domain, e.g. "the Tag Genome of the movie domain".

4.3.2 Computing the Tag Genome

Now that we have defined the Tag Genome, the next question is how to compute it. That is, how do we come up with $rel(t, i)$ for all $t \in T$ and $i \in I$? We considered three approaches:

User-contributed. In this approach, users of the system rate the relevance of tags to items. This is the similar to the approach taken by traditional tagging systems, which

[3] We describe the method for computing these values in Section 4.4.

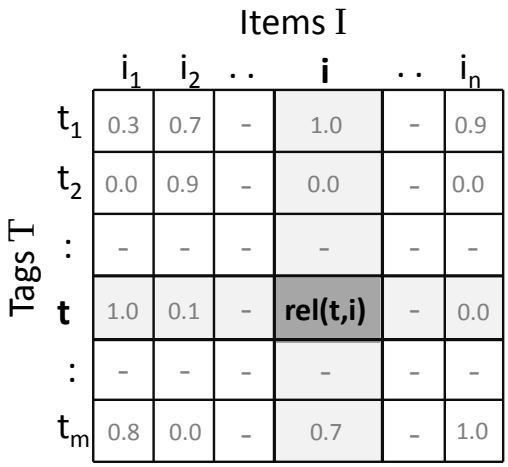

Figure 4.1: The Tag Genome. Each entry in the matrix records the relevance of a tag to an item on a $0 - 1$ scale.

rely on users to apply tags to items; the difference is that users would need to specify a continuous value rather than apply a binary label.

Expert-maintained. Rather than rely on users to contribute relevance ratings, another option is to employ domain experts to provide these ratings.

Machine-learned. Machine learning models can automate the process of computing the Tag Genome. Given a data source and a set of training examples, machine learning models learn a mapping between data attributes and the output variable, in this case tag relevance. Once trained, the model can predict the relevance for all tag-item pairs in the genome.

The challenge of the user-contributed approach is finding enough users to rate the relevance for all tag-item pairs in G. The disadvantage of the expert-maintained approach is the cost of compensating the experts, which may be high if the space of tag-item pairs is large. In both cases, the volume of data required is problematic. In contrast, the machine learning approach learns from a relatively small set of training

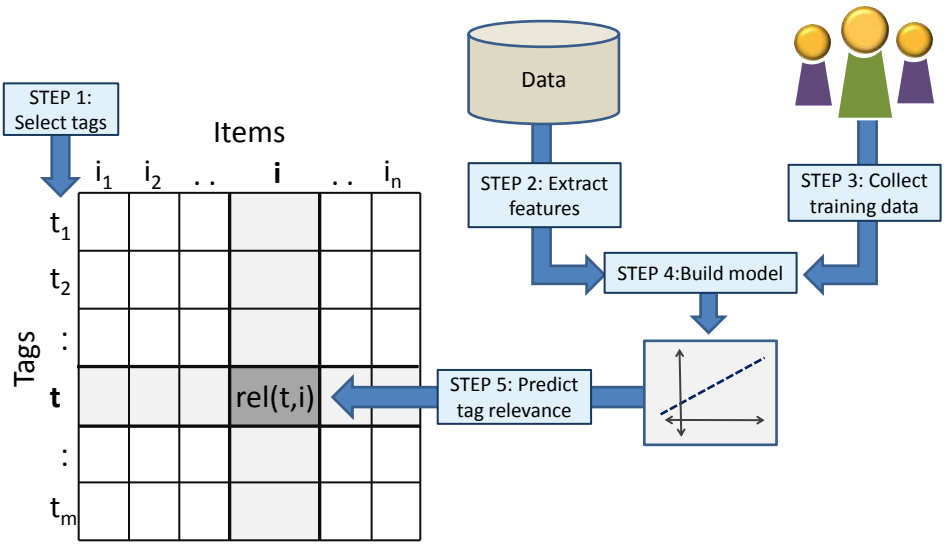

Figure 4.2: Computing the Tag Genome

examples and then computes the relevance for the majority of tag-item pairs. It complements the other two approaches, since relevance ratings from either experts or users can serve as training examples for the learning model.

We now present step-by-step guidelines for computing the Tag Genome using machine learning. As summarized in Figure 4.2, we propose a supervised approach: one first identifies a set of input features, and then trains a regression model on these features using a set of labeled training examples. Regression offers a generic framework that can combine features from multiple data sources, as well as the output from various models. In Section 4.4.2, we describe several features computed from various models such as singular value decomposition.

STEP 1: Select tags. The first step is to choose the set of tags T that constitute the "genes" of the Tag Genome. The goal is to find a set of tags large enough to capture the diverse interests of the user community, but not so large as to (1) make the learning problem intractable, and (2) spread the training data (see Step 3) too thinly across

tags. Various metrics may be used to gauge the community's interest in a particular tag; for example, one might consider how many users have applied the tag or how many have searched by the tag. The appropriate number of tags to choose will depend on the particular domain.

STEP 2: Extract features. The next step is to identify a data source and extract a set of input features for the machine learning model. Data sources might include:

User-generated content. Users contribute a variety of content, such as text (reviews, comments, blogs), ratings, and tags. Each of these data sources may provide features for a machine-learning algorithm. For example, one potential feature from textual data is the frequency with which a tag appears in the text for an item; the more frequently the tag appears, the more likely the tag is relevant to that item. Other features for predicting tag relevance may be less direct; for example, in Section 4.4.2, we describe features computed from users' ratings of items.

System-maintained metadata. Items often have associated metadata; for instance, music albums typically have a genre, title, artist, release date, and description. These metadata may help predict the relevance of particular tags. Knowing that a music album belongs to the "new age" genre, for example, might help predict that the tag *relaxing* has high relevance.

Item content. Many items, such as books, music, and movies, exist in an electronic form that may itself provide a rich set of features. For example, one might construct a feature from a movie's audio file that detects the number of explosions in the movie. This feature might help predict the relevance of tags such as *violent* or *action*. In Section 4.2.1, we discuss algorithms for tag suggestion that extract features from video, audio, and image data; these features may also potentially be used to compute the Tag Genome.

STEP 3: Collect training data. The learning model needs training examples in order to learn the relationship between the input features and tag relevance. Each

training example is a tag-item pair with an associated relevance rating. These relevance ratings may be collected using either the expert-maintained or user-contributed approaches described above, but only for a subset of tag-item pairs in the genome.

We prefer the user-based approach, since it allows the user community to guide the machine learning process; we refer to this approach as *community-supervised learning*. Users could provide this training data at various stages of computing the Tag Genome. Prior to the system computing the Tag Genome, users could provide initial training data through data collection tools like the survey we describe in Section 4.4.3. Once the Tag Genome has been computed, users could provide feedback on these computed values directly in the applications. The system could gradually refine the tag relevance predictions as it gathers more feedback from users.

STEP 4: Build model. The learning model maps input features to tag relevance predictions. Since tag relevance is continuously valued, a natural choice is to use a regression model, where tag relevance is expressed as a weighted combination of input features. There are many different types of regression models, ranging from simple linear regression to more complex models such as generalized linear models or hierarchical models. In Section 4.4.4 we discuss a variety of regression models for predicting tag relevance.

STEP 5: Predict tag relevance. Once the model has been trained, it can be used to predict the relevance for all tag-item pairs in the genome.

4.4 Case Study: Computing the Tag Genome in the Movie Domain

In this section we describe how we compute the Tag Genome for MovieLens, described in Section 2.2.

4.4.1 STEP 1: Select tags

MovieLens users have applied nearly 30,000 distinct tags, ranging from very popular tags such as *classic* (applied by 416 users), *funny* (279 users), and *animation* (236 users) to tags only applied by a single user such as *oh yah*, *sidecar*, or *acorn*. We

Type of data	Source	Description	Mean Volume
Tag applications	MovieLens	Users' applications of tags to movies.	28 tag applications per movie
User reviews	IMDB	Text-based reviews of movies contributed by users.	35,446 total words per movie
Ratings	MovieLens	User ratings of movies, on a scale from 1/2 to 5 stars	1,965 ratings per movie

Table 4.1: Data sources used to predict tag relevance

wished to select the tags most valued by the user community; therefore we filter tags based on popularity, including only those tags applied by at least 10 users. Tags below this threshold tended to be either personal (*jb's dvds*), extremely specific (*archery*), or misspellings of more popular tags (*Quinten Tarantino*). We filter the remaining tags based on a tag quality metric developed in [Sen et al., 2007], excluding tags that scored in the bottom 5 percentile.

After filtering, 1,570 tags remained. 442 of these tags were names of actors or directors. Since MovieLens already stores the actors and director for each film, we simply set $rel(t, i) = 1$ for an actor/director tag t if the actor/director worked on film i, and set $rel(t, i) = 0$ otherwise. We compute the relevance of the remaining 1,128 tags using machine learning as described below. We exclude actor/director tags from the machine learning process so that we could collect more training data for the other tags in the genome for which we had no system-maintained metadata.

4.4.2 STEP 2: Extract features

We extracted features from three types of user-contributed content: tag applications, user reviews, and ratings, as summarized in Table 4.1. The tag applications and ratings came from MovieLens itself, while the user reviews were crawled from the Internet Movie Database website[4] . From each of these data sources we extracted several features, summarized in Table 4.2. Each of these features is defined for a particular tag and item, since what we are trying to predict – tag relevance – is specific to a tag-item pair.

[4] http://www.imdb.com

Type of data	**Feature**	**Description**
Tag applications	tag-applied(t, i)	1 if tag t has been applied to item i, 0 otherwise.
	tag-count(t, i)	Number of times tag t has been applied to item i
	tag-share(t, i)	tag-count(t, i) divided by total number of tag applications for item i
	tag-lsi-sim(t, i)	Similarity between tag t and item i using latent semantic indexing, where each document $\mathbf{d}_i$ is the set of tags applied to item i
User reviews	text-freq(t, i)	Frequency with which tag t appears in text reviews of item i
	text-lsi-sim(t, i)	Similarity between tag t and item i using latent semantic indexing, where each document $\mathbf{d}_i$ is the set of words in user reviews of item i
Ratings	avg-rating(t, i)	Average rating of item i
	rating-sim(t, i)	Cosine similarity between ratings of item i and aggregated ratings of items tagged with t
All	regress-tag(t, i)	Predicted value for $rel(t, i)$, based on a regression model using tag-applied as the output variable and the other features as the input variables.

Table 4.2: Features used to predict tag relevance

Features based on tag applications

When a user applies tag t to item i, she is expressing that t is relevant to i to some degree. The following features use this binary signal of tag relevance to help predict $rel(t, i)$.

Tag-applied: tag-applied(t, i) equals 1 if tag t has been applied to item i, 0 otherwise.

Tag-count: tag-count(t, i) equals the number of times tag t has been applied to item i. This feature reflects the fact that tags applied many times to an item may be more relevant than those applied only a few times or not at all.

Tag-share: tag-share(t, i) equals the number of times tag t has been applied to item i, divided by the number of times any tag has been applied to item i. This feature controls for the fact that the tag-count feature is biased towards popular items that attract a large number of tag applications.

Tag-lsi-sim: The above three features provide a signal of tag relevance only if tag t has actually been applied to item i. The other tags applied to an item can also provide clues that a tag is relevant; for example, if the tags *love story* and *hilarious* have been applied to an item, it is likely that the tag *romantic comedy* is highly relevant.

A common technique for finding relationships between terms – in this case tags – is *latent semantic indexing* (LSI) [Deerwester et al., 1990]. LSI applies rank-reduced singular value decomposition to a term-document matrix in order to express the documents and terms in a lower dimensional concept space. One can then measure the conceptual similarity of a term and a document based on their cosine similarity in this concept space.

To use LSI for tags, we construct a term-document matrix (see Section 4.2.2) where each document comprises the tags that have been applied to a particular item. Formally, we associate each item i with a document vector $\mathbf{d}_i$, which represents one column in the term-document matrix:

$$\mathbf{d}_i = \big\langle \text{tag-applied}(t_1, i), \ldots, \text{tag-applied}(t_m, i) \big\rangle \; \forall t_j \in T$$

We used tag-applied as the term weight function rather than tag-count or tag-share, because tag-lsi-sim performed best in predicting tag relevance with tag-applied as the

term weight function[5] .

For each tag t, we define a pseudo-document $\mathbf{d}_t$, which has a term weight of 1 for tag t and 0 for other terms.

We then apply rank-reduced singular value decomposition to the term-document matrix. Based on this matrix factorization, we transform vectors $\mathbf{d}_i$ and $\mathbf{d}_t$ into a lower-dimensional concept space; $\mathbf{d}'_i$ and $\mathbf{d}'_t$ denote the transformed versions of $\mathbf{d}_i$ and $\mathbf{d}_t$ respectively. We define tag-lsi-sim(t, i) as the cosine similarity of $\mathbf{d}'_i$ and $\mathbf{d}'_t$:

$$\text{tag-lsi-sim}(t, i) = \cos(\mathbf{d}'_t, \mathbf{d}'_i)$$

Features based on user reviews

When users review items, they naturally use words or phrases that are relevant to that item. The following features extract signals of tag relevance based on how frequently tags and related terms appear in user reviews from the Internet Movie Database[6] website.

Text-freq: One simple predictor of $rel(t, i)$ is the frequency with which tag t appears in the combined reviews of i, which we denote as text-freq(t, i). We performed the following steps to computing this feature:

Stopword removal. Stopwords represent common words in a corpus that provide little informational value, such as *the*, *is*, or *and*. We removed all stopwords from the reviews based on a standard list of stopwords provided with the Natural Language Toolkit [Bird et al., 2009]. We also removed sequences of words in each review that closely matched the title of the reviewed movie. We did this because users often referred to the title in their reviews, artificially boosting the frequency of title words.

Stemming. Word stemming is the practice of reducing a word to its stem or root form, so that different forms of the same word may be equated. We stemmed words using the Porter Stemmer [Porter, 1980] as implemented by the Natural Language Toolkit [Bird et al., 2009].

[5] We evaluated features using relevance ratings collected in a pilot survey (see Section 4.4.3). These ratings were not used in the final model evaluation.

[6] http://www.imdb.com

Tokenization. Tokenization is the process of dividing a block of text into meaningful units, or tokens. We extracted tokens for each tag (which may span one or more words) in the genome as well as the 1000 most frequent words in the corpus that are not tags. We use the non-tag tokens for computing the text-lsi-sim feature described below.

Smoothing. Smoothing can help improve the accuracy of frequency estimates, especially when little data is available. We initially tried Laplacian smoothing [McCallum and Nigam, 1998] when calculating text-freq, but achieved better results using a Bayesian approach, which we described in detail in Appendix A.

Normalization. Word frequency may vary considerably across tags; popular terms such as *funny* or *action* will naturally appear more frequently in user reviews than more obscure terms such as *nonlinear* or *magical realism*. To standardize text-freq across tags, we compute z-scores by subtracting the mean and dividing by the standard deviation specific to each tag[7] .

Text-lsi-sim: The text-lsi-sim feature is similar to tag-lsi-sim, described in detail above. Whereas tag-lsi-sim constructs each document vector $\mathbf{d}_i$ from the tags that users have applied to item i, text-lsi-sim constructs each $\mathbf{d}_i$ from the tag and word tokens in user reviews of i (see *Tokenization*, above). We include non-tag tokens from the reviews because they can help reveal semantic relationships between tags and items. If *terms* denotes the set of distinct tag and word tokens, we define each document $\mathbf{d}_i$ as:

$$\mathbf{d}_i = \big\langle \text{text-freq}(i, term_1), \ldots, \text{text-freq}(i, term_n) \big\rangle \;\forall\; term_k \in Terms$$

We then perform rank-reduced singular value decomposition on the term-document matrix, and we express item i and tag t as vectors in the lower-dimensional concept space, denoted as $\mathbf{d}'_t$ and $\mathbf{d}'_t$ respectively.

We define text-lsi-sim(t, i) as the cosine similarity of these concept vectors:

$$\text{text-lsi-sim}(t, i) = \cos(\mathbf{d}'_t, \mathbf{d}'_i)$$

Features based on ratings

Because MovieLens users have contributed over 18 million movie ratings, we wished to see if we could use this data to help predict tag relevance. But it is not immediately

[7] We also normalize this feature across all tags as described in Section 4.4.2.

clear how to do this; for example, when a user rates *Die Hard* as 4 stars, how does that tell us that a particular tag is more or less relevant to this movie? Below we describe two features that extract signals of tag relevance by analyzing ratings in aggregate.

Avg-rating: avg-rating(t, i) is the average rating for item i. This feature is item-specific and provides a weak signal of tag relevance for certain tags. For example, the average rating of movies tagged with *stupid* is 3.2 stars, while movies tagged with *intelligent* average 3.9 stars (compared to an average rating of all movies of 3.5 stars). Therefore if a movie's average rating is low, it is more likely that *stupid* is relevant and less likely that *intelligent* is relevant.

Rating-sim: For recommender systems like MovieLens, a common way to measure the similarity between two items is to compute the cosine similarity of their respective ratings vectors[8]. The rating-sim feature uses a similar approach to compute the affinity between a tag and an item. Although tags have no ratings associated with them, one can compute ratings for tags by aggregating the ratings of items to which the tag has been applied [Vig et al., 2009]. We define rating-sim(t, i) as the cosine similarity between the vector of ratings for item i across users and the vector of computed ratings for tag t across users. This is similar to the approach used to estimate tag relevance in [Vig et al., 2009].

Let $r_{u,i}$ denote user u's rating of item i, adjusted by user-item mean[9] . We define user u's "rating" of tag t as the smoothed average of u's ratings of items tagged with t:

$$r_{u,t} = \frac{\sum_{i' \in I_u \cap I_t} r_{u,i'} + \bar{r}_u}{|I_u \cap I_t| + 1},$$

$$\text{where } I_u = \text{items rated by } u, \text{ and } I_t = \text{ items tagged with } t$$

We use Laplacian smoothing in the above expression by adding the user's average rating $\bar{r}_u$ to the numerator and 1 to the denominator. $r_{u,t}$ is undefined if $|I_u \cap I_t| = 0$. We exclude item i (for which we are calculating $rel(t, i)$) from I_u in order to avoid artificially boosting the cosine similarity between $r_{u,t}$ and $r_{u,i}$ as computed below.

Let U' denote the set of users for whom both item rating $r_{u,i}$ and tag rating $r_{u,t}$ are defined. We define rating-sim(t, i) as the cosine similarity between $r_{u,i}$ and $r_{u,t}$ across

[8] The ratings vector for an item is the collection of ratings for that item, indexed by user.

[9] We first subtract the user mean from each rating, then compute each item's mean rating based on these user-mean adjusted ratings, and finally subtract the adjusted item mean from the user-mean adjusted rating.

these users:

$$\text{rating-sim}(t,i) = \frac{\sum_{u \in U'} r_{u,t} \cdot r_{u,i}}{\sqrt{\sum_{u \in U'} r_{u,t}^2} \cdot \sqrt{\sum_{u \in U'} r_{u,i}^2}}$$

Meta-feature

The regress-tag feature blends the output of several other features.

Regress-tag: Tag applications represent a crude form of relevance assessment: when a user applies tag t to item i they are expressing that t is relevant to item i to some degree. The regress-tag feature uses these approximate relevance ratings to train a regression model that uses the other features as inputs. This approach parallels how we train the final regression model for computing the Tag Genome; the difference is that we use continuous relevance ratings collected from a survey when training the final model. While binary tag applications are not as precise as the continuous relevance ratings, they represent pre-existing data that can complement the continuous ratings; this is especially helpful for tags that have only a small number of relevance ratings from the survey (see Section 4.4.3).

We constructed positive training examples from tag-item pairs (t, i) where t has been applied to i; we assign these pairs the maximum relevance rating of 1, based on the assumption that if tag t has been applied to i it is probably highly relevant. We generated negative examples by pairing each tag t with a random set of items not tagged with t, and assigning these pairs a relevance rating of 0. We weighted the examples in the regression model such that the weighted count of negative examples would each equal the number of positive examples for each tag. We did this to avoid an imbalance between the positive and negative examples and to maintain consistency across tags, since some tags had many more positive examples than others.

Once we generated our training set, we fit a regression model using the features text-freq, text-lsi-sim, avg-rating, and rating-sim as inputs. We did not use the tag-applied feature because it is the same as the output variable in the regression, nor did we use tag-count, tag-share, or tag-lsi-sim since they are closely related to tag-applied. We used the nonlinear multilevel regression model described in Section 4.4.4, because it resulted in the feature that peformed best in predicting the relevance ratings from the pilot survey (see Section 4.4.3). We define regress-tag(t, i) as the relevance prediction

for tag t and item i.

Preprocessing

We normalized all of the features by subtracting by the mean and dividing by the standard deviation across all tag-item pairs in the genome. We log transformed text-freq to bring it closer to a normal distribution; this transformation also resulted in a stronger correlation with the relevance ratings in the pilot survey (see Section 4.4.3).

4.4.3 STEP 3: Collect training data

We collected training examples by conducting a survey in which we asked MovieLens users to rate the relevance of tags to movies. As summarized in Figure 4.3, we conducted two phases of this survey: a pilot phase, which was restricted to a smaller subset of tags and users, and the main phase, which included all tags in the genome[10] and a much larger set of users.

Survey design

On each page of the survey (see Figure 4.4), subjects rated the relevance of a particular tag to a each of 6 movies on a scale from 1 (does not apply at all) to 5 (applies very strongly). Although we model tag relevance as a continuous variable, we chose a discrete rating scale because past work suggests that users have difficulty with continuous scales [Sparling and Sen, 2011]. We personalize each survey, only including movies that the subject had rated in the past. To help subjects recall the details of each movie, we included a link ("Summary") to each movie's cover art as well as a short synopsis, both made available through the Netflix API[11] . We also provided space for free-form comments. Subjects could repeat the survey up to 3 times, each time with a new set of tags and movies.

[10] excluding actor/director tags, as discussed in Section 4.4.1
[11] http://developer.netflix.com

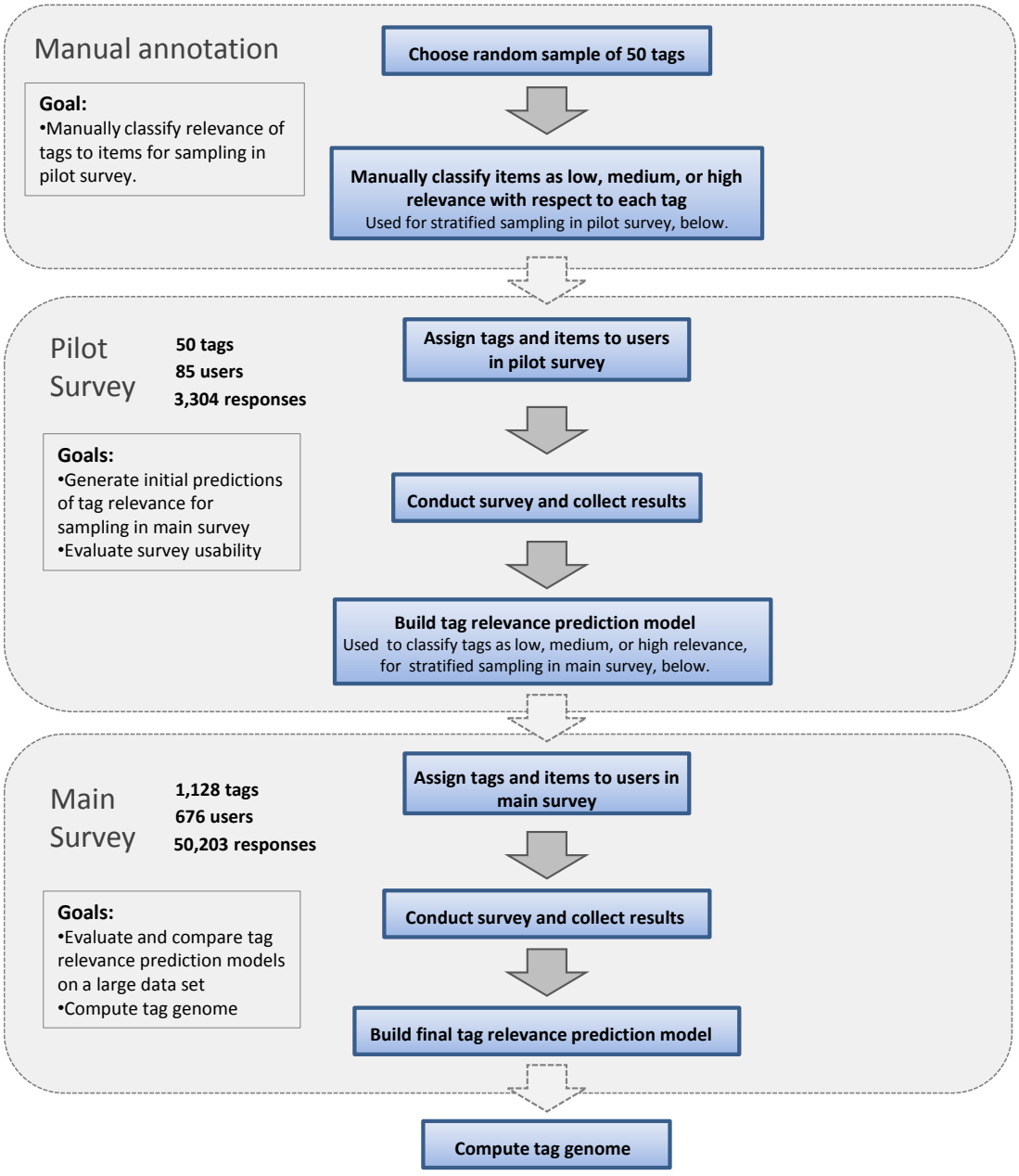

Figure 4.3: Overview of process for collecting training data.

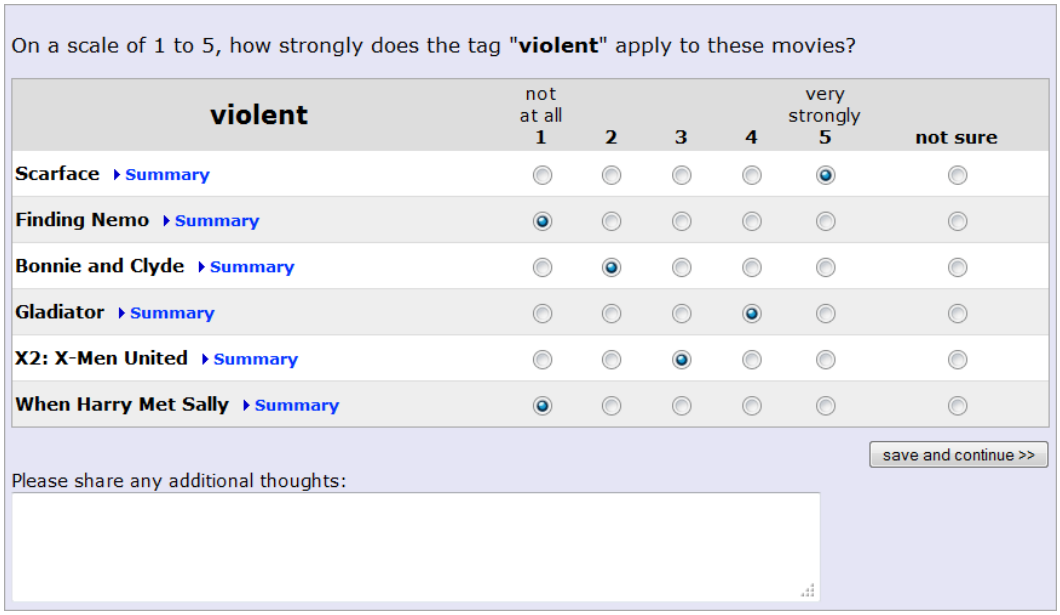

Figure 4.4: Page from survey.

Sampling methodology

In this section, we discuss how we select the tag and associated movies to show on each page of the survey. We present the complete algorithm in Appendix B. The main features of this algorithm are:

Importance-based sampling. We wished to collect more relevance ratings for more important tags, since more training data for a particular tag enables one to more accurately predict relevance for that tag[12]. We measure the importance of tag t by popularity(t), which we define as the number of distinct MovieLens users who have applied tag t. We use tag popularity because it reflects how important the tag is to the overall community. In order to collect more data for popular tags, we assign each tag t to a number of subjects proportional to log(popularity(t)). Since tag popularity follows a power law distribution, we apply a log transform so that popular tags don't completely dominate the training set.

Stratified sampling. We sample movies for the survey using a stratified approach based

[12] For the tag-specific and multilevel models described in Section 4.4.4

on tag relevance: for each tag we show to a user, we select two movies that we predict will have low relevance with respect to the tag, two movies we predict will have medium relevance, and two that we predict will have high relevance. We sample movies in this way in order to achieve a balanced training set with a roughly equal number of low, medium, and high relevance ratings; without stratified sampling, it is likely that most – if not all – of the movies would have low relevance with respect to a given tag. In order to predict in advance which movies have low, medium, or high relevance with respect to each tag, we first train a relevance prediction function on data collected from the pilot survey. We describe this in more detail below.

Pilot survey

Prior to the main phase of the survey, we conducted a pilot phase in which we invited a smaller number of users to take the survey for a subset of 50 tags. Besides testing the usability of the survey, the purpose of the pilot survey was to collect data to train an initial relevance prediction model that we use to sample movies for the main survey (see *Stratified sampling*, above).

We used the same sampling algorithm for the pilot survey that we later use for the main survey (see Appendix B). In order to perform stratified sampling in the pilot survey, we needed to first classify items as low, medium, or high relevance with respect to each tag. To do this, we handcrafted a rule-based classifier based on the strongest individual predictor of tag relevance, which appeared to be text-freq. For each of the 50 tags in the pilot survey, we manually choose two cutoff values for text-freq: one value separated low relevance items from medium relevance items, and the other separated medium relevance items from high relevance items. We classified each item i as low, medium, or high relevance with respect to tag t based on the value of text-freq(t, i) relative to these cutoffs.

85 users took the pilot survey, providing a total of 3,304 relevance ratings. As mentioned above, the reason we conducted the pilot survey was to train a relevance prediction function for sampling of movies in the main survey. We trained a nonlinear, multilevel regression model as described in Section 4.4.4, because it performed best based on cross-validation of the ratings from the pilot survey. Although the pilot survey data used to train this model only included 50 tags, this model can predict relevance

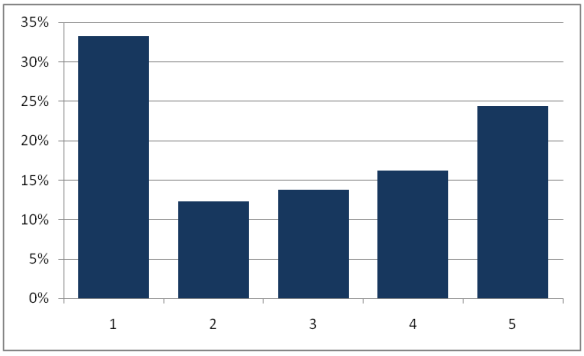

Figure 4.5: Distribution of relevance ratings in main phase of survey (1=Does not apply at all, 5= Applies very strongly)

.

for any tag in the genome (see Section 4.4.4).

Main survey

We invited 5,320 MovieLens users to participate in the main phase of the survey. We included the 1,128 tags from the genome that were not names of actors or directors (see Section 4.4.1), and we drew from the 7,716 movies with at least 100 ratings. We assigned tags and movies to each user based on the sampling algorithm described in Section 4.4.3. We used the relevance prediction model trained on the data collected in the pilot survey for the stratified sampling component: we classify items with a relevance prediction of 1/3 or less as low relevance, those with a predicted relevance of 1/2 to 2/3 as medium relevance, and those with a predicted relevance of 2/3 or higher as high relevance.

676 users participated in the survey (13% response rate), providing relevance ratings for a total of 50,203 tag-movie pairs. 53% of those taking the survey elected to take it multiple times (with different tags and movies each time). Figure 4.5 shows the overall distribution of relevance ratings. On average, we collected 45 ratings per tag.

4.4.4 STEP 4: Build model

In this section we describe the learning models we use to predict tag relevance. Many different approaches are possible; we implemented six regression models that span a range of techniques. The models vary along two dimensions:

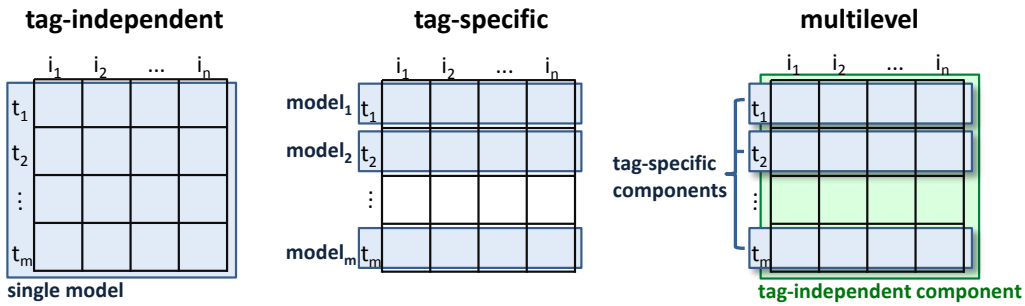

Figure 4.6: Three different approaches with respect to model granularity. The tag-independent approach assumes a unified model across all tags, while the tag-specific approach learns a separate model for each tag. The multilevel approach combines tag-independent and tag-specific approaches.

(1) *Linear versus nonlinear.* Linear regression models express tag relevance as a linear combination of features, while nonlinear models support nonlinear combinations of features. Linear models have the advantage of simplicity, while nonlinear models can more precisely capture the relationship between input features and tag relevance.

(2) *Model granularity.* The learning model must be able to predict the relevance of any of 1,128 distinct tags, each of which may exhibit very different behavior. Therefore a natural question is to what degree we should tailor the model to each tag. On one extreme, we could build a model that treats all tags the same and learns a single set of regression coefficients for all tag-item pairs in the genome; we refer to this type of model as a *tag-independent* model. On the other extreme, we could build a completely different learning model for each tag t that only applies to tag-item pairs with tag t; we call this the *tag-specific* model. Both approaches are illustrated in Figure 4.6.

The advantage of a tag-independent model is that there are fewer parameters to learn; the model only needs to learn a single set of regression coefficients across all tags, and it can use the entire training set to do so. In contrast, tag-specific models must learn separate regression coefficients for each of the 1,128 tags, which can lead to overfitting since each tag has an average of only 45 training examples. However, tag-specific models can exploit relationships between the features and tag relevance that depend on the tag. For example, the relationship between avg-rating and tag relevance is likely tag-specific: for tags with a positive connotation like *good acting*, one would expect a

positive correlation between an item's average rating and tag relevance, whereas the opposite is likely true for tags with a negative connotation like *predictable*. However, most features have been designed to be meaningful across tags; text-freq, text-lsi-sim, tag-applied, tag-lsi-sim, rating-sim and regress-tag should correlate positively with relevance regardless of the tag. Nonetheless, the optimal weights for these features may vary according to tag.

Multilevel models offer a compromise between the tag-independent and tag-specific models. As we discuss in more detail below, multilevel models learn regression coefficients that reflect characteristics of each tag as well as qualities shared across all tags. Multilevel models can capture tag-specific relationships between feature values and tag relevance, while avoiding the overfitting problems of the tag-specific model.

We now present 6 specific regression models that each use a unique combination of the approaches outlined above (2×3). In the following definitions, $\mathbf{x}_{t,i}$ denotes the feature vector for the tag-item pair (t, i), including a constant term. $\boldsymbol{\beta}$ denotes a coefficient vector of the same length as $\mathbf{x}_{t,i}$.

Linear models

The linear models express $rel(t, i)$ as a linear function of the feature vector $\mathbf{x}_{t,i}$.

Tag-independent: For the tag-independent model, we use a single regression equation:

$$rel(t, i) = \mathbf{x}_{t,i}^{\mathsf{T}} \boldsymbol{\beta}$$

Here the vector of coefficients $\boldsymbol{\beta}$ is a constant and does not vary by tag.

Tag-specific: In this model, the vector of coefficients $\boldsymbol{\beta}_t$ is specific to each tag:

$$rel(t, i) = \mathbf{x}_{t,i}^{\mathsf{T}} \boldsymbol{\beta}_t$$

Here we solve a separate regression equation for each tag t, using only the training data associated with t to train each model. If no training data exists for t, we predict a value of 0.5.

Multilevel: In the multilevel, or hierarchical, model [Gelman and Hill, 2007, Raudenbush and Bryk, 2002], the vector of coefficient $\boldsymbol{\beta}_t$ is also specific to each tag. Unlike the tag-specific model, however, each $\boldsymbol{\beta}_t$ is modeled as a random variable sampled from a multivariate normal prior distribution $\mathrm{N}(\mu, \Sigma)$:

$$rel(t,i) = \mathbf{x}_{t,i}^{\mathsf{T}} \boldsymbol{\beta}_t$$
$$\boldsymbol{\beta}_t \sim \mathrm{N}(\mu, \Sigma)$$

The hyperparameters μ and Σ are estimated from the data along with each $\boldsymbol{\beta}_t$. μ is the "average" $\boldsymbol{\beta}_t$ across all tags, and represents the tag-independent component of this model. The estimate for each $\boldsymbol{\beta}_t$ depends on both this prior distribution $\mathrm{N}(\mu, \Sigma)$ as well as the training data specific to tag t. When little training data is available for t, $\boldsymbol{\beta}_t$ will mostly be determined from the prior distribution and its estimated value will be very close to μ. As more training data is available for t, the estimate of $\boldsymbol{\beta}_t$ may deviate further from μ. Thus the multilevel model is able to capture tag-specific behavior, while avoiding overfitting when little data is available for a particular tag.

Nonlinear models

There are many different types of nonlinear regression models, including nonlinear least squares [Kelley, 1999], SVM regression [Drucker et al., 1997], and generalized linear models [McCullagh and Nelder, 1989, Gelman and Hill, 2007]. We chose to use a generalized linear model, for two reasons. First, we wished to implement both a single level and multilevel version of the nonlinear model, and generalized linear models support both. Second, generalized linear models are widely used and are supported in standard statistical packages such as R.

Generalized linear models extend linear regression by introducing a nonlinear link function, which transforms a linear combination of input features into a probability distribution for the output variable. The choice of link function depends on the nature of the problem. Since we are predicting an output variable that lies in a fixed range of $[0, 1]$, we chose a sigmoidal transform using the logit link function. For notational convenience, we refer to the inverse logit:

$$\mathrm{logit}^{-1}(x) = \frac{e^x}{1 + e^x}$$

This is the same approach used in logistic regression. Logistic regression, however, is traditionally used in classification, where the output is the probability of the positive

class:

$$\Pr(y = 1) = \text{logit}^{-1}\left(\mathbf{x}^{\mathsf{T}}\boldsymbol{\beta}\right)$$

In our case, we are predicting a continuous output variable, $rel(t, i)$. Therefore we need to adapt the traditional logistic regression model. We do so by treating $rel(t, i)$, which lies on the range $[0, 1]$, as a probability. From a probabilistic viewpoint, one can think of $rel(t, i)$ as the certainty that tag t is relevant to item i: a value of 1 indicates that t is relevant to i with complete certainty, while a value of 0 indicates that t is certainly irrelevant to i. Values in between represent varying degrees of certainty that t is relevant to i.

Tag-independent: The tag-independent nonlinear model assumes a single vector of coefficients $\boldsymbol{\beta}$, as in the linear case:

$$rel(t, i) = \text{logit}^{-1}\left(\mathbf{x}_{t,i}^{\mathsf{T}}\boldsymbol{\beta}\right)$$

Tag-specific: The tag-specific nonlinear model uses a separate coefficient vector $\boldsymbol{\beta}_t$ for each tag:

$$rel(t, i) = \text{logit}^{-1}\left(\mathbf{x}_{t,i}^{\mathsf{T}}\boldsymbol{\beta}_t\right)$$

As in the linear case, we solve a separate regression equation for each $\boldsymbol{\beta}_t$ using only the training data for t.

Multilevel: The multilevel nonlinear model follows a form similar to the linear version:

$$rel(t, i) = \text{logit}^{-1}\left(\mathbf{x}_{t,i}^{\mathsf{T}}\boldsymbol{\beta}_t\right),$$
$$\boldsymbol{\beta}_t \sim \text{N}(\mu, \Sigma)$$

Model Fitting

We fit the regression models using the R functions [R Development Core Team, 2010, Bates and Sarkar, 2007] shown in Table 4.3. We clamped the predictions from the linear models to lie in the $[0, 1]$ interval; this step was not necessary for the generalized linear models since their output is already restricted to the $[0, 1]$ range.

Prior to fitting the models, we converted the survey ratings from a 1-5 scale to the $[0, 1]$ interval using a linear transformation. However, we found that all models

	Tag-independent	**Tag-specific**	**Multilevel**
Linear	lm	lm	lmer (tag as grouping factor)
Nonlinear	glm	glm	glmer (tag as grouping factor)

Table 4.3: R functions used for each model

performed better when we binarized the ratings used to train each model; we did this by mapping survey ratings of 1 or 2 to a relevance value of 0, mapping ratings of 4 and 5 to a relevance value of 1, and discarding ratings of 3. We only applied this binarization when *fitting* the models, never when validating the models.

Feature selection

We performed initial feature selection to eliminate low-value or redundant features. We eliminated tag-count and tag-share, because neither feature improved the performance of regression models over just using the tag-applied feature alone[13] .

We performed an additional round of feature selection to prevent model overfitting. As discussed earlier, our regression models vary in how susceptible they are to overfitting. In order to select the features most appropriate for each model, we used the *wrapper* approach to feature selection [Guyon and Elisseeff, 2003, Kohavi and John, 1997] where the model itself is used to evaluate a candidate set of features. We used a forward-selection algorithm for choosing features: beginning with an empty set of features, we incrementally added the feature that resulted in the lowest mean absolute error for the model. We used cross-validation to compute the MAE in order to properly assess the generalization error; we stopped adding features once the MAE began to increase due to model overfitting.

We repeated the feature selection process for each round of the cross-validation described below. We only used the training partition from each round for feature selection, reserving the test partition for model evaluation.

[13] We evaluated the models using ratings collected from the pilot survey.

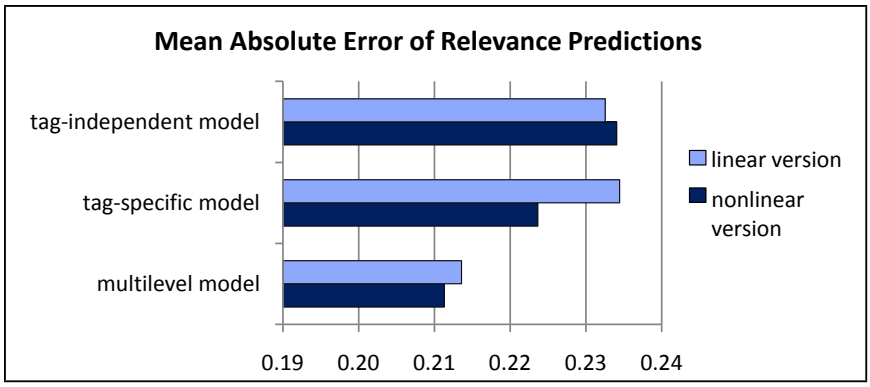

Figure 4.7: Performance of each model using 10-fold cross validation on the relevance ratings from the main survey. All models performed significantly better than a baseline model that used the mean rating (MAE = 0.36). Differences in MAE are statistically significant ($p < 0.05$), with the exception of the difference between the linear, tag-specific model and the nonlinear, tag-independent model.

Model Evaluation

We evaluated the 6 regression models using 10-fold cross-validation on the ratings collected in the main phase of the survey. As shown in Figure 4.7, the multilevel models performed significantly better than the tag-specific and tag-independent models ($p < 0.001$). This follows our assertion that the multilevel models combine the best aspects of the tag-specific and tag-independent models.

The relative advantages of nonlinear versus linear vary according to the specific model. For the multilevel models, the nonlinear model performed better than the linear model by a statistically significant ($p < 0.001$) margin. For the tag-specific models, the nonlinear model also performed significantly better than the linear model ($p < 0.001$). However, the nonlinear version of the tag-independent model performed worse than the linear version by a small but statistically significant ($p < 0.001$) margin.

One possible reason why the nonlinear models performed better in the tag-specific and multilevel case is that these models are able to capture tag-specific differences in the nature of the nonlinear relationship between feature values and tag relevance. For tags that exhibit a continuous range of relevance values – for example, *funny*, *exciting*, or *violent* – the relationship between feature values and tag relevance is likely to follow a relatively smooth curve. For tags that describe binary attributes – for example, *based on*

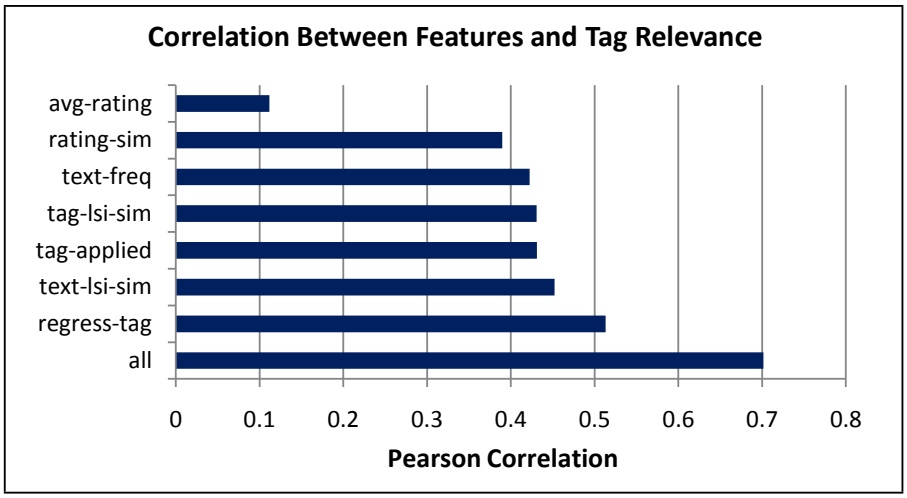

Figure 4.8: Performance of each individual feature based on correlation with tag relevance ratings from the main survey. The *all* feature represents the prediction from the best performing regression model, which uses all the other features as inputs. All differences are statistically significant ($p < 0.001$) except for the differences between text-freq, tag-lsi-sim, and tag-applied.

a play, *oscar*, or *foreign* – the relationship between feature values and tag relevance may be closer to a step function. Depending on the model coefficients, the logit link function can approximate both types of relationships. Since the tag-independent model uses the same coefficients for all tags, it is not able to capture these tag-specific differences in the shape of the curve.

Figure 4.8 shows the strength of individual features based on their correlation with the survey ratings[14]. The meta-feature regress-tag had the highest correlation with the ratings, which is to be expected since this feature combines several of the other features. text-lsi-sim was the next strongest feature, which is not surprising given the high volume of textual data (see Table 4.1). The features tag-applied and tag-lsi-sim were nearly as strong as the text-lsi-sim, despite the fact that the tagging data was relatively sparse (see Table 4.1). This suggests that individual tag applications can provide a relatively strong signal of relevance.

The rating-sim feature was weaker than the other tag-based and text-based features,

[14] We excluded the features tag-count and tag-share from the final model as discussed in Section 4.4.4.

	tag-applied	**tag-lsi-sim**	**text-freq**	**text-lsi-sim**	**avg-rating**	**rating-sim**	**regress-tag**
tag-applied	-	0.689	0.390	0.400	0.156	0.356	0.396
tag-lsi-sim	0.689	-	0.493	0.485	0.070	0.438	0.446
text-freq	0.390	0.493	-	0.664	-0.007	0.318	0.378
text-lsi-sim	0.400	0.485	0.664	-	0.025	0.314	0.399
avg-rating	0.156	0.070	-0.007	0.025	-	0.214	0.387
rating-sim	0.356	0.438	0.318	0.314	0.214	-	0.678
regress-tag	0.396	0.446	0.378	0.399	0.387	0.678	-

Table 4.4: Pairwise correlations between features

but still showed a fairly strong correlation. The avg-rating feature was the weakest based on its low overall correlation with the survey ratings. However, as suggested earlier, the predictive power of avg-rating seems to be largely tag-specific. For example, the correlation between avg-rating and relevance ratings for *good acting* was 0.625, while the correlation between avg-rating and ratings for *predictable* was -0.418. Thus the avg-rating may provide a stronger signal of tag relevance for the tag-specific and multilevel models than the tag-independent models.

Table 4.4 shows the pairwise correlations between each of the features. tag-lsi-sim and tag-applied correlate most strongly; this follows from the fact that they both depend on tag applications. Although tag-lsi-sim(t, i) is calculated based on *all* tags applied to i, this set of tags may include t, which is also used to compute tag-applied(t, i). regress-tag correlated strongly with rating-sim, which is one of the features used to compute regress-tag. The avg-rating feature showed the weakest overall correlation with other features, consistent with the fact that it was the weakest predictor overall.

4.4.5 STEP 5: Predict tag relevance

We use the nonlinear, multilevel model to predict tag relevance, since this model achieved the lowest mean absolute error in our evaluation (see Section 4.4.4). We train the final model using the full set of relevance ratings from the survey.

4.5 Summary

This chapter introduces the Tag Genome, a novel data structure that extends the traditional tagging model in two ways. First, the Tag Genome encodes a continuous notion of tag relevance, which measures how strongly tags apply to items on a $0-1$ scale. Second, the Tag Genome provides a dense structure that computes the relevance of every tag in the genome, regardless of whether that tag has been applied to an item.

We presented a general machine learning approach to computing the Tag Genome, which we demonstrated on the movie tagging system MovieLens. To compute the Tag Genome in MovieLens, we first extracted features from user-generated content including tag applications, text reviews, and ratings, and then we trained our learning model using human judgments of tag relevance collected from a survey of MovieLens users. Results showed that tag applications and the appearance of tags in text provided the strongest signals of tag relevance. We found that hierarchical regression models most accurately predicted tag relevance, and that linear models performed nearly as well as more complex nonlinear models.

Movies represent just one of many possible domains that might benefit from the Tag Genome. The Tag Genome can be computed for any information space with sufficient training data, along with a community that is willing to share their views of the relationships between tags and items. System designers will want to choose a set of training data suitable for their domain. On MovieLens, we found that text reviews and tag applications provided the richest data for learning the Tag Genome, but other types of media may be used in other domains. In the music domain, for example, one might extract features such as tempo, volume, and pitch from audio tracks to help learn the relevance of tags such as *relaxing*, *upbeat*, or *jarring*. As discussed in Section 4.2.1, researchers have developed tag recommendation algorithms for a variety of data types including images, songs, and videos; the same features used in those algorithms might also be used to compute the Tag Genome.

We encourage designers to extend and refine the definition of the Tag Genome that we present in Section 4.3.1. An alternate definition of the Tag Genome could take into account whether tags represent binary or continuous attributes; for example, it makes more sense to describe a movie as somewhat *violent* (continuous) as opposed

to somewhat *based on a book* (binary). Another model for the Tag Genome might distinguish between factual tags (*action*, *black and white*, *japanese*) versus subjective tags (*funny*, *inspiring*, *depressing*) [Sen et al., 2006]. Relevance values for subjective tags could be personalized; instead of computing $rel(t, i)$, the system could compute $rel(t, i, u)$, representing the relevance of tag t to item i from the perspective of user u. Various user-specific features could be used to predict $rel(t, u, i)$. For example, user u's predicted rating for item i could help predict evaluative tags such as *funny* or *exciting*; this is analogous to the avg-rating feature used to predict $rel(t, i)$.

The remainder of this thesis covers the rich and diverse set of applications of the Tag Genome. In the next chapter, we present a detailed case study of one such application called *Movie Tuner*. In Chapter 6, we discuss the broader space of applications of the Tag Genome.

Chapter 5

Movie Tuner

5.1 Introduction

In the previous chapter we introduced the Tag Genome, an extension of the traditional tagging model that captures a richer and more complete mapping between tags and items. In this chapter we explore a concrete application of the Tag Genome called *Movie Tuner*. Movie Tuner provides a novel form of interaction that is based on tags, but that offers a fundamentally different form of navigation than traditional tagging systems. We motivate this system with a hypothetical dialogue between a movie navigation system and a user Marco:

Marco: *I'd like to watch a movie, but I'm not exactly sure what I want.*

System: *How about When Harry Met Sally, Up, or Reservoir Dogs?*

Marco: *Reservoir Dogs looks like a possibility, please tell me more.*

System: *It's a dark, extremely violent, crime film that has some action and is slightly funny.*

Marco: *I'm not in the mood for something quite that violent.*

System: *Then how about The Usual Suspects? It's like Reservoir Dogs, but somewhat less violent.*

Marco: *I'll take it!*

Movie Tuner enables users to navigate an information space much like Marco did.

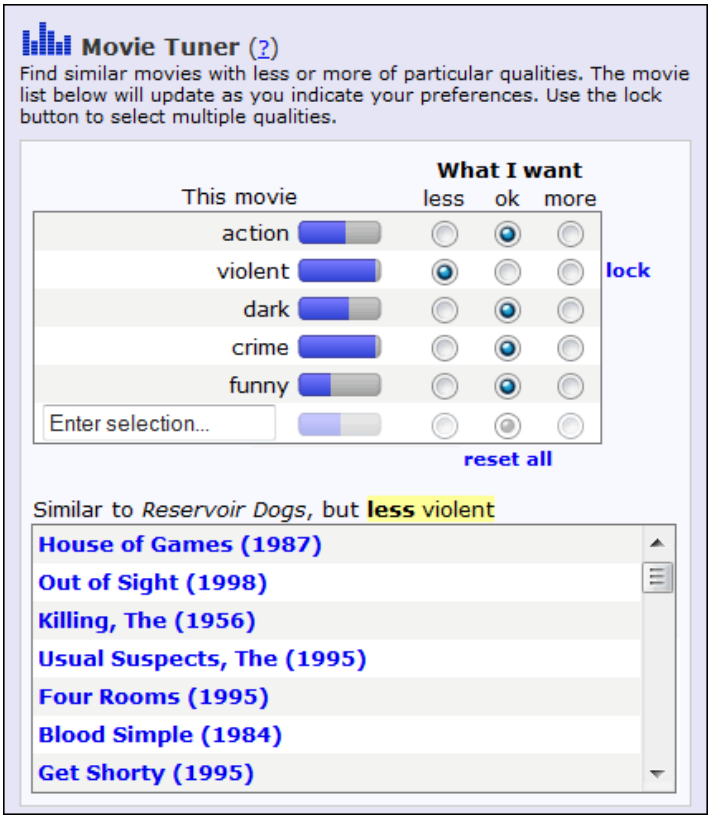

Figure 5.1: Movie Tuner interface for *Reservoir Dogs*, after a user applies the critique "less violent".

Figure 5.1 shows the Movie Tuner interface as Marco might have seen it after selecting *Reservoir Dogs*. The interface displays a set of tags (*action*, *violent*, *dark*, *crime*, *funny*) each with a relevance meter indicating how strongly *Reservoir Dogs* exhibits that quality. Marco clicked the *less* button alongside the *violent* tag; in response, the system displayed a list of movies that are "Similar to Reservoir Dogs, but less violent", including *The Usual Suspects*, which Marco eventually chose.

Navigating a space by critiquing specific items, known as *example critiquing*, has been studied in the context of knowledge-based systems. For example, in the Entree system users critique restaurant recommendations based on price ("less expensive"), style ("more traditional"), atmosphere ("quieter"), and other criteria [Burke et al., 1997]. In Movie Tuner, users critique items with respect to tags. Three advantages of tag-based critiquing versus traditional example critiquing are: (1) the tags are dimensions chosen by users, and hence are expressed in their own language; (2) tags are resources that are freely generated by users of the system, with only modest support from the system designers [Shirky, 2005]; and (3) tags describe both factual and subjective aspects of items [Sen et al., 2006]. The design of Movie Tuner focuses on two primary interactions from example-critiquing systems: *applying critiques* refers to how users tell the system what they would like to change about an item ("I'd like something less violent than Reservoir Dogs"), and *responding to critiques* describes how the system chooses new items in response to a user's critiques ("Here are the movies similar to Reservoir Dogs, but less violent...").

This form of navigation has several key differences versus traditional tag-based navigation. First, tags are used to react to specific items ("I'd like something less violent than *Reservoir Dogs*"), in contrast to traditional tag search, where users begin by specifying one or more tags ("I'd like something *not violent*"). Presenting tags in the context of specific items of interest to users is consistent with prior studies that suggest that people formulate preferences by interacting with the available choices rather than deciding in advance what they want [Payne et al., 1993]. Second, the system explicitly models the relevance of tags to items on a consistent 0-1 scale. This allows the system to compare items with respect to tags ("The Usual Suspects is *less violent* than Reservoir Dogs").

Movie Tuner is driven by the Tag Genome, described in the previous chapter, which

is built automatically by applying machine-learning to user-contributed content. The Tag Genome provides all of the information that Movie Tuner needs to support the interaction we describe above. For the scenario depicted in Figure 5.1, the Tag Genome enables Movie Tuner to display the relevance meters showing how strongly the tags *action*, *violent*, *dark*, *crime*, and *funny* apply to *Reservoir Dogs* (the rightmost position of the relevance meters indicates a relevance value of 1, the leftmost a value of 0). Movie Tuner can find items that are "less violent" than *Reservoir Dogs* by selecting items that have lower relevance values for the tag *violent*. The Tag Genome can also be used to measure similarity between two items (see Section 5.4.2), enabling Movie Tuner to find movies that are similar overall to *Reservoir Dogs*.

In this chapter, we first summarize related work and discuss how Movie Tuner builds on the existing critiquing and tagging literatures. We then detail the design of applying critiques and responding to critiques in Movie Tuner. Finally, we present the results of a 7-week field study of Movie Tuner.

5.2 Related Work

5.2.1 Example-critiquing systems

Researchers have explored conversational recommenders that allow users to give immediate feedback on recommendations and then adjust recommendations accordingly [Linden et al., 1997, Burke et al., 1997, Faltings et al., 2004, Smyth et al., 2004]. One type of feedback supported by these systems is a *critique*, which describes what the user thinks is wrong with a particular example. For example, in the Entree system shown in Figure 5.2, users critique restaurant recommendations based on price ("less $$"), style ("more traditional"), atmosphere ("quieter"), and other criteria. The system then responds by selecting a new set of results that satisfies the user's critique. This type of conversational recommender is often referred to as an *example-critiquing* system. Figures 5.3 and 5.4 show other example-critiquing systems for apartments and digital cameras, respectively.

Example-critiquing systems generally offer the user a narrow set of dimensions for critiquing items, and these dimensions are typically chosen by designers of the system. For example, in the QwikShop system in Figure 5.4, critique dimensions include

Figure 5.2: Entree example-critiquing system for restaurants [Burke, 2002]

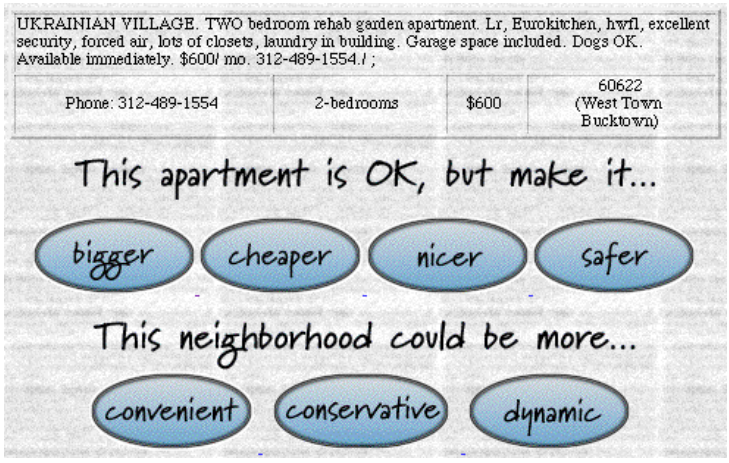

Figure 5.3: RentMe example-critiquing system for apartment rentals [Burke et al., 1996]

Figure 5.4: Qwikshop example-critiquing prototype for digital cameras [McCarthy et al., 2005]

manufacturer, zoom level, memory, weight, resolution, size, case, and price. Moreover, example-critiquing systems are traditionally knowledge-based; for example, the Entree recommender system has an underlying database with the cuisine, price, style, and atmosphere of every restaurant in the system.

The example-critiquing paradigm motivates our design of Movie Tuner. In Movie Tuner, tags serve as the dimensions along which users critique items. For example, users may ask for a movie that is "more funny" or "less violent" because *funny* and *violent* are tags in the system. However, in contrast to the compact set of system-engineered dimensions typically provided by example-critiquing systems, tags provide a broad range of feedback in the language of the users themselves. Further, Movie Tuner requires no underlying knowledge base that knows how violent *Die Hard* is, or how much action is in *Forrest Gump*. Rather, this information is generated automatically by machine learning models based on user-contributed content. One compelling reason for earlier critiquing systems relying on expert-based systems is that they did not have access to the volume of user-generated data that is available today.

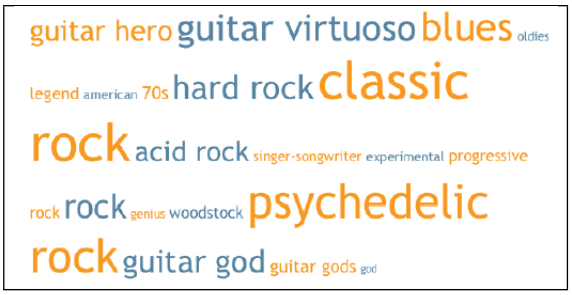

Figure 5.5: The "textual aura" for the musical artist Jimi Hendrix in the Music Explaura system [Green et al., 2009]. Users can steer the system to related artists by dragging tags to make them larger or smaller.

5.2.2 Tag-based recommenders

Recent work has explored systems that combine tagging and recommendation [Sen et al., 2009, Niwa et al., 2006, Green et al., 2009]. The most similar work to Movie Tuner is the Music Explaura system [Green et al., 2009], in which users "steer" music recommendations using tags (see Figure 5.5). MrTaggy [Kammerer et al., 2009] is a tagging system that supports exploration by enabling users to provide positive or negative feedback to tags associated with particular items.

Movie Tuner differs from these systems in several ways. First, Movie Tuner provides an explicit measure of tag relevance on a $0-1$ scale that is based on a gold-standard set of tag relevance values provided by users. Second, Movie Tuner provides a novel interface for visualizing tag relevance and applying critiques. Third, we evaluate Movie Tuner in a live user study involving thousands of users, comparing multiple algorithms for suggesting tags as well as multiple algorithms for retrieving items in response to users' critiques.

5.3 Applying Critiques

Users apply critiques to tell the system what they wish to change about a particular item, for example "less violent" or "more action". Below we outline the design space for how users may apply critiques, and we discuss our design decisions.

5.3.1 Critique dimensions

Critique dimensions represent the dimensions along which users may critique an item. In Movie Tuner, tags serve as critique dimensions. For example, some of the tags on MovieLens are *action, violent*, and *quirky*; with these tags as critique dimensions, a user might request a movie that has "more action", is "less violent", or is "more quirky". We include all 1,570 tags in the genome as critique dimensions.

As shown in Figure 5.1, Movie Tuner displays tags in a list, with a *relevance meter* next to each tag indicating its relevance to the current item. (We discuss later how the system chooses the tags to display.) Other visualizations should work as well, such as a tag cloud with varying font size [Green et al., 2009]. We used the relevance meter to more precisely represent the 0 to 1 relevance scale.

5.3.2 Critique direction

In most example-critiquing systems, users critique an item by specifying a *direction* along a critique dimension, for example "*less* expensive". However, some example-critiquing systems also enable the user to provide a *magnitude*, for example "*at least $100* cheaper" [Chen and Pu, 2006]. Although specifying the magnitude gives users more control over their critiques, it requires more fine-grained input from the user.

In Movie Tuner, we chose to use a direction-only approach. Enabling users to specify the magnitude of the critique, perhaps with a slider, would give users additional control, but we chose the direction-only approach because it requires lower cognitive load. We denote an individual critique as a tuple (t, d), for tag $t \in T$ and direction $d \in \{-1, +1\}$, where -1 indicates *less* and $+1$ indicates *more*.

As shown in Figure 5.1, users choose a critique direction by clicking a "less" or "more" radio button next to a particular tag. The default "ok" selection indicates that the user does not wish to apply a critique with respect to the tag. As an alternative to the three radio buttons, we had also considered having two checkboxes, one for "less" and one for "more", but found that in initial trials users felt compelled to check one box for every tag shown. With a default selection of "ok", users understood that they could simply ignore a particular tag.

5.3.3 Unit versus compound critiques

A *unit critique* is constrained to a single critique dimension ("less violent"), while a *compound critique* [Smyth et al., 2004, Zhang and Pu, 2006] spans multiple dimensions, ("less violent and more action"). Although compound critiques enable faster navigation, they also require more work from the user at each step.

Movie Tuner supports both unit and compound critiques. To apply a compound critique, users must explicitly *lock* the original critique in order to combine it with other critiques (see Figure 5.1); otherwise, the original critique will be reset to the "ok" position when they select additional critiques. We require explicit locking because in initial trials users often forgot to undo their original critique before selecting other unit critiques. As a result the critiques became increasingly complex, and did not match the users' intentions.

5.3.4 System-suggested versus user-initiated critiques

In systems that provide a small number of critique dimensions, a common design choice is to display all critique dimensions and let users choose from them when applying critiques. In Movie Tuner, however, the number of critique dimensions (i.e. tags), far exceeds the available screen space. We considered two alternatives: in a *system-suggested* model, the system displays a small set of possible tags and users choose among them, while in a *user-initiated* model, users must enter the tags they wish to use in critiques.

We chose a mixed-initiative model where users may either choose from a set of system-suggested tags, or enter additional tags of their own. We chose to suggest tags because studies have shown some users have difficulty thinking of tags [Sen et al., 2006]. As shown in Figure 5.1, Movie Tuner displays 5 system-selected tags for each item. We display 5 tags per item in order to provide users with a variety of choices while conserving screen space.

Users may also enter tags not suggested by the system in an auto-complete text box ("Enter selection") below the tags currently displayed; however, users may only enter tags that are among the 1,570 tags included as critique dimensions. Once entered, the tag is displayed above along with its relevance meter as well as radio buttons for setting

the critique direction. Users may also use the text box simply to inquire about the relevance of a tag to an item ("How *realistic* is The Bourne Identity?").

5.3.5 Tag selection algorithm

We now describe how we choose the tags to display for a particular item. We select tags based on three objectives: we choose tags that are *valuable for critiquing* an item, because the primary purpose of displaying the tags is to help users apply critiques; we choose *popular* tags, because the tags should be ones that users care about; and we choose *diverse* tags, because an orthogonal set of tags enables more efficient navigation. Below we define metrics for each of these objectives, and we describe a multi-objective optimization algorithm for selecting the tags to display.

Critique value. We define two metrics for evaluating how useful a tag is for critiquing a particular item. One metric favors *descriptive* tags; for example, *violent* is highly descriptive for *Reservoir Dogs* because it is an extremely violent movie. The other metric favors tags that *discriminate* among the space of similar items; for example, *action* is a discriminating tag for *Reservoir Dogs*, because many similar movies have either more action (e.g. *Kill Bill Vol. 1*) or less action (e.g. *Sexy Beast*).

To measure how *descriptive* a tag t is with respect to an item i, we simply use $rel(t, i)$, the relevance of t to i (see Section 4.3.1). To measure how *discriminating* a tag t is with respect to an item i, we define a metric called *critique entropy*, which measures how evenly t separates the items neighboring i.

To compute critique entropy for tag t relative to item i, we partition the set N of neighbors of i (defined in Section 5.4.3) into 3 subsets N_{+1}, N_{-1}, and N_0. N_{+1} comprises neighbors of i that satisfy the critique "more t", N_{-1} comprises neighbors of i that satisfy the critique "less t", and N_0 comprises the remaining neighbors of i. Formally,

$$N_{+1} = \{j | j \in N, rel(t, j) > rel(t, i) + 0.25\}$$
$$N_{-1} = \{j | j \in N, rel(t, j) < rel(t, i) - 0.25\}$$
$$N_0 = N - N_{+1} \cup N_{-1}$$

We chose the value of 0.25 based on our qualitative analysis over a series of test cases.

This is a simplified version of the critique satisfaction model presented in Section 5.4, and is only used for the purpose of computing critique entropy.

We define critique entropy to be the Shannon entropy of the distribution of items over N_{+1}, N_{-1}, N_0. Formally,

$$\text{critique-entropy}(t, i) = \sum_{d \in \{+1,-1,0\}} -\frac{|N_d|}{|N|} \cdot \log\Big(\frac{|N_d|}{|N|}\Big)$$

Just as Shannon entropy measures the evenness of a distribution, critique entropy measures how evenly the critiques associated with a tag divide the space of neighboring items.

Popularity. We measure tag popularity by the number of distinct users who have applied a tag t, denoted as popularity(t). We apply a log transform to popularity to make the distribution more normal.

Diversity. We measure the diversity of a set of tags based on how dissimilar the tags are to one another. To measure similarity between two tags t and u, we take the cosine similarity of their relevance values across all items in I, which we denote as tag-sim(t, u). Later we will show how we use this tag similarity metric to choose a diverse set of tags.

Multi-objective optimization. Since we wish to satisfy three different objectives (critique value, popularity, diversity) simultaneously, we express the problem of choosing tags as a multi-objective optimization problem [Fletcher, 1981]. One approach for solving multi-objective optimization problems is to define an *aggregate objective function* that takes all objectives into account and computes a single utility value for each candidate solution. One may also frame the problem as a *constrained optimization problem*, where some of the objectives are expressed as constraints while others are included in the objective function.

We chose to express the tag selection problem as a constrained multi-objective optimization problem over the space of all tag sets of size 5. We define an aggregate objective function that evaluates each candidate tag set based on the objectives described above, and we also set constraints to ensure that the chosen tag set satisfies each objective to a minimal degree. We constructed two versions of the optimization problem, one that measures critique value based on tag relevance (favors descriptive tags) and one that measures critique value based on critique entropy (favors discriminating tags).

We chose the specific problem formulation below based on a series of trials with various objective functions and constraint combinations. We did not include diversity in the objective function, because we found that simply setting a constraint based on maximum pairwise similarity between tags produced sufficiently diverse tag sets. We combine popularity and critique value in the objective function by taking their product. We preferred this approach to a weighted sum because, since it is scale invariant, it requires no parameter estimation. We then add these values for all tags in the set in order to produce a single value for the entire set.

Problem formulation. Given an item i, find the set of tags $S \subset T$ that maximizes the following objective function:

$$\begin{aligned} \underset{S}{\text{maximize}} \quad & \left\{ \sum_{t \in S} \text{critique-value}(t, i) \cdot \log(\text{popularity}(t)) \right\} \\ \text{subject to} \quad & |S| = 5 \\ & \text{popularity}(t) \geq 50 \quad \forall t \in S \\ & \text{tag-sim}(t, u) < 0.5 \quad \forall t, u \in S,\ t \neq u \end{aligned}$$

We designed two versions of the objective function, one where $\text{critique-value}(t, i) = rel(t, i)$ (favors descriptive tags), and one where $\text{critique-value}(t, i) = \text{critique-entropy}(t, i)$ (favors discriminating tags). In the latter case, we added the following constraint[1]:

$$\text{critique-entropy}(t) \geq 0.325 \quad \forall t \in S$$

Table 5.1 shows an example of the tags each version generates.

Because finding exact solutions to combinatorial optimization problems is computationally expensive, we designed a greedy algorithm to find an approximate solution. The algorithm begins with an empty set of tags, then iteratively adds the tag that maximizes the objective function subject to its constraints, stopping when the size of the tag set equals 5.

[1] 0.325 is the Shannon entropy of the distribution {0.9, 0.1, 0.0}

Descriptive	Discriminating
sci-fi (0.99)	fantasy (0.50)
comedy (0.98)	space (0.67)
action (0.95)	superhero (0.37)
adventure (0.85)	future (0.35)
comic book (0.75)	tense (0.38)

Table 5.1: Tags chosen for *Men in Black* by each tag-selection algorithm. Relevance values are shown in parentheses.

5.4 Responding to Critiques

After a user critiques an item, the system must respond by retrieving new items that satisfy the critique. In this section we describe the algorithm for responding to critiques on Movie Tuner. The algorithm chooses items based on two objectives: 1) the items should be sufficiently different along the critique dimension, and 2) the items should be similar overall to the original item. We first define an objective measure of *critique distance*, the difference between items along the critique dimension. We then define a measure of the similarity between items. Finally, we present an algorithm that chooses items based on satisfying these two metrics simultaneously.

5.4.1 Critique distance

Users specify the direction of their critiques, but the system must determine how far to move in that direction. For example, if a user asks for a movie with less action than *Independence Day*, the system must decide whether to choose a movie like *Star Trek: Generations*, which still has a reasonable amount of action, or a movie like *Contact*, which has very little action.

To formalize these concepts, we introduce a metric called *critique distance* that measures the difference in tag relevance between two items with respect to a particular critique. For example, if a user applies the critique "less action" to *Independence Day*, then the critique distance to *Star Trek: Generations* is $rel(action, \textit{Independence Day}) - rel(action, \textit{Star Trek: Generations}) = 0.97 - 0.46 = 0.51$. Formally, if i_c is the critiqued item, i_r is the retrieved item, and (t, d) is the critique with tag $t \in T$ and direction

$d \in \{-1, +1\}$, then

$$\text{critique-dist}(i_c, i_r, t, d) = \max(0, (rel(t, i_r) - rel(t, i_c)) \cdot d)$$

To determine the appropriate critique distance when choosing new items, we define a *critique satisfaction* metric that determines how strongly an item satisfies a critique based on critique distance. Below we define two alternative critique satisfaction metrics: linear-sat and diminish-sat. In Section 5.4.3 we describe how we use these critique satisfaction metrics in conjunction with item similarity to sort critique results.

linear-sat: The *linear* critique satisfaction model, linear-sat, assumes that critique satisfaction is proportional to critique distance. This model assumes that greater critique distance is always better, and that the rate of improvement stays constant as the critique distance increases. This model suggests that users want to move as far as possible along the critique dimension.

Formally, if i_c is the critiqued item, i_r is the retrieved item, and (t, d) is the critique, $t \in T, d \in \{-1, +1\}$, then

$$\text{linear-sat}(i_c, i_r, t, d) = \text{critique-dist}(i_c, i_r, t, d)$$

diminish-sat: The *diminishing returns* model, diminish-sat, also assumes that greater critique distance is better, but that the rate of improvement decreases as the critique distance increases. This model suggests that users want a certain amount of change along the critique dimension, but that differences beyond that threshold have little value. Formally,

$$\text{diminish-sat}(i_c, i_r, t, d) = 1 - e^{-5 \cdot \text{critique-dist}(i_c, i_r, t, d)}$$

This formula is based on the negative exponential utility function. We chose the value of -5 based on qualitative analysis over a series of 30 test cases; this value tended to produce critique results that were noticeably different along the critique dimension, but not as different as those generated from the linear model.

5.4.2 Item similarity

When responding to a critique, the system should choose items that satisfy the critique, but are otherwise similar to the original item. One approach for measuring similarity

between items is to use a domain-specific similarity metric; for example, in movie recommenders like MovieLens, a common similarity metric is the ratings correlations between movies. Alternatively, one could measure similarity of items based on the similarity of their Tag Genomes. We prefer the latter approach because it is domain-independent, and because the dimensions (i.e. tags) used to critique items are the same ones used to assess similarity. This means that items will tend to be similar along the dimensions visible to users.

We define the similarity between items i and j as the weighted cosine similarity of their Tag Genomes G_i and G_j (see Section 4.3.1). The cosine similarity reflects the overall similarity of the relevance values between the items across the entire set of tags T. We used a weighted version of cosine similarity to account for the fact that some tags may be more important than others in determining similarity between items.

We denote the weighted cosine similarity between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ based on weight vector $\boldsymbol{w}$ as

$$\cos(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}) = \frac{\sum_{k=1,\ldots,n} w_k \cdot x_k \cdot y_k}{\sqrt{(\sum_{k=1,\ldots,n} w_k \cdot x_k^2)} \cdot \sqrt{(\sum_{k=1,\ldots,n} w_k \cdot y_k^2)}}$$

We assign weights to tags based on two criteria: *tag popularity* and *tag specificity*. We assign more weight to popular tags because they reflect dimensions that more users care about. We define tag popularity of tag t as the number of users who have applied t, denoted as popularity(t). We apply a log transform to make the distribution more normal.

We also assign higher weight to tags that are more specific, because specific tags can more uniquely identify similarities between items. For example, if two movies have the tag *dark comedy* in common, they are more likely to be similar than if they simply had the tag *comedy* in common. We measure tag specificity based on a modified version of *inverse document frequency*, a metric used in the tf-idf weighting scheme to assess term specificity [Salton and McGill, 1983]. In our case, we define doc-freq(t) as the number of items where the relevance of t is greater than 1/2. We apply a log transform to the document frequency to bring it closer to a normal distribution.

Putting all of this together, we define the similarity between items i and j as

$$\text{sim}(i, j) = \cos(G_i, G_j, \boldsymbol{w}),$$
$$\text{where } w_k = \frac{\log(\text{popularity}(t_k))}{\log(\text{doc-freq}(t_k))}$$

For all of the computations below, we normalize similarity values by subtracting the average similarity of all item pairs (0.61). Normalizing in this way yields similarity values with greater proportional variation, which helps balance the effects of similarity versus critique distance in the algorithm discussed in the next section.

5.4.3 Algorithm for responding to critiques

We now describe an algorithm that uses the above metrics to choose items in response to user critiques. Our general approach is to display a small set of highly relevant results, but let users explore a larger result set if they wish. The interface design reflects this approach: critique results are displayed in a scrollable window sorted in descending order of goodness-of-fit to the critique, as shown in Figure 5.1.

The algorithm has two steps: a *filtering* step that establishes the basic requirements for an item to be included in the critique results, and a *sorting* step that orders the remaining items in descending order of goodness-of-fit to the critique.

Filtering. As discussed above, the system must choose items that are sufficiently different along the critique dimension, but are similar overall to the original item. Accordingly, the algorithm filters items based on both objectives. Filtering by similarity has the added benefit that it reduces the number of items to evaluate when responding to critiques of a particular item, enabling the data to be stored client-side.

- **Filtering based on critique distance.** Given a critiqued item i_c, tag t, direction $d \in \{-1, +1\}$, any result i_r must satisfy critique-dist$(i_c, i_r, t, d) > 0$.

- **Filtering based on overall similarity.** Given a critiqued item i_c, any result i_r must be among the k-nearest neighbors of i_c, based on the similarity metric defined in 5.4.2. We considered setting a minimum similarity value instead of using similarity rank, but found that the range of similarity scores varied between items. We chose a value of $k = 250$, because items outside that range tended to be

Linear	**Diminishing Returns**
Aladdin (1992)	Shrek (2001)
Sword in the Stone (1963)	Shrek 2 (2004)
Toy Story (1995)	Toy Story 2 (1999)
Robin Hood (1973)	Aladdin (1992)
Looney, Looney, Looney Bugs Bunny Movie (1981)	Shrek the Halls (2007)

Table 5.2: Top-5 results for the critique "more classic than *Shrek the Third*" for both versions of the algorithm. The linear model favors more classic movies while the diminishing returns model favors similar movies.

considerably different from the critiqued item, and this value produced sufficiently long results lists to satisfy most users.

Sorting. The goal of the sorting step is to identify the items that most strongly satisfy the critique based on critique distance and are most similar to the original item. We sort results using a metric called *critique fit* that combines both objectives:

Given a critiqued item i_c, retrieved item i_r, tag t, direction $d \in \{-1, +1\}$,

$$\text{critique-fit}(i_c, i_r, t, d) = \text{critique-sat}(i_c, i_r, t, d) \cdot \text{sim}\big(i_c, i_r\big)$$

We implemented two versions of the sorting algorithm, one where critique-sat = linear-sat, and one where critique-sat = diminish-sat. The choice of function determines the tradeoff between critique distance and overall similarity. When critique-sat = linear-sat, the tradeoff between critique distance and overall similarity is the same at any critique distance. When critique-sat = diminish-sat, the tradeoff favors increased similarity over increased critique distance as critique distance increases. Table 5.2 shows sample results for both versions of the algorithm.

Compound and null critiques. The above definition applies to unit critiques. For compound critiques, we simply take the product of the critique fit values for each of the individual critiques. When no critique has been applied, we order results based on similarity only.

5.5 Design of Field Study

We conducted a field study of Movie Tuner on the MovieLens website, in which we empirically evaluated Movie Tuner based on activity logs and survey data.

We added the Movie Tuner interface to two screens on MovieLens: the *movie details* page, and the *movie list* page. The movie details page displays detailed information about a particular movie including cast, director, a Netflix synopsis, a tag cloud for the movie, and Movie Tuner. The movie list page is used to display any list of movies, including search results and personalized recommendations. We added an icon users may click to see Movie Tuner. We do not show Movie Tuner for the least popular movies (< 50 ratings), because these movies tended to have too little data to accurately compute the Tag Genome. In total, we display Movie Tuner for 8,871 distinct movies.

The primary data source for our analyses comprised activity logs collected during a 7-week period that Movie Tuner was in place, running from July 14, 2010 through September 1, 2010, and the 7-week period just before the launch. These logs track all activity on Movie Tuner, including page views[2], critiques applied, and items selected.

We used a between-subjects design so that subjects could respond to survey questions based on their overall experience in a single experimental condition. Each user was assigned to one of four experimental groups based on two manipulated factors (2x2). One factor determined how Movie Tuner selected tags to display for an item: specifically, whether the algorithm favored *descriptive* tags (*rel* metric) or *discriminating* tags (critique-entropy metric), as described in Section 5.3.5. The other factor determined how Movie Tuner chose items in response to a critique: specifically, whether the algorithm used the linear (linear-sat) or the diminishing returns (diminish-sat) model of critique distance, as discussed in Section 5.4.1.

On 08/26/10, we invited 910 Movie Tuner users to an online survey, of whom 160 (18%) participated. We included users who had viewed Movie Tuner at least once and consented to participate in studies on MovieLens. In the survey, users responded to

[2] We did have one logging problem during the time of the experiment. We did not lose any Movie Tuner data, but due to a data collection bug, movie detail page view data for pages that did not include Movie Tuner was lost between 07/22/10 and 7/29/10, and between 08/17/10 and 08/30/10. We believe this page view data would have been similar to the page view data that was correctly collected, so the lost data should not substantively affect the results.

Statement (abbreviated)	μ	% **agree**	% **dis-agree**
I would like the Movie Tuner feature to remain.	4.2	79	6
Movie Tuner is fun to use.	3.9	74	9
I like having the ability to specify critiques.	4.3	89	4
The tags shown helped me learn about the movie.	3.5	59	12
I liked seeing the tags.	3.9	72	6
The tags made sense to me.	4.0	81	8
The similar movies helped me discover movies I had not seen.	3.4	54	22
The similar movies helped me find movies I was interested in.	3.8	67	10
The similar movies were actually similar to the main movie.	3.6	60	7
Applying critiques helped me to discover movies I had not seen.	3.5	65	19
Applying critiques helped me find movies I was interested in.	3.7	71	11
Movies displayed in response to my critiques made sense.	3.8	68	8

Table 5.3: Survey questions and aggregated responses (5-point Likert scale). Percent (dis)agree equals the number of *(dis)agree* or *strongly (dis)agree* responses divided by total number of responses. For questions below the double line, we only included responses from users who actually applied critiques (71% of respondents.)

a series of statements, summarized in Table 5.3, using a 5-point Likert scale[3] . For each statement, we showed subjects a screenshot of the Movie Tuner interface for the movie *Pulp Fiction*, in order to help them recall their experience with Movie Tuner. We recognized that users may be influenced by the example shown to them when answering questions; therefore we displayed screenshots for each subject that matched how the interface would look given their experimental group. Additionally, we emphasized to subjects that they should respond based on *their* experience with Movie Tuner.

[3] 1 = *strongly disagree*, 2 = *disagree*, 3 = *neutral*, 4 = *agree*, 5 = *strongly agree*

5.6 Results

During the 7-week field trial, 2,531 users viewed the Movie Tuner interface a total of 49,099 times, and 1,037 users applied a total of 12,298 critiques. Overall feedback on MovieLens was positive; 89% of survey respondents liked being able to apply critiques, 74% found Movie Tuner fun to use, and 79% wanted Movie Tuner to remain available on MovieLens. Daily page views of the movie detail page increased by 52% ($p < 0.001$, t-test). One user commented, *"The best thing to come by in MovieLens (besides the product itself). Strongly recommended this to my friends and some picked MovieLens up just because of this addition. Love it!"*

In this section we empirically evaluate users' interactions with Movie Tuner, based on activity logs and user self-report. We first examine how users apply critiques in Movie Tuner, based on the types of tags they choose, how they choose critique direction, and whether they use compound or unit critiques. We then explore how users interact with items displayed in response to their critiques.

5.6.1 Applying critiques

Choosing tags. For 91% of critiques, users chose system-suggested tags rather than entering their own tags. This is consistent with interaction models suggesting people prefer recognition over recall [Smith et al., 1990]. Besides facilitating critique application, the system-selected tags provided other benefits: 72% of respondents like seeing the tags in Movie Tuner and 59% said the tags helped them to learn about the movie (compared to 12% who felt the tags did not help them learn).

As discussed in Section 5.3.5, we implemented two algorithms for choosing tags, one that favored descriptive tags and one that favored discriminating tags. Survey results show that more subjects in the descriptive-tags group (87%) felt the system-suggested tags made sense to them compared to subjects in the discriminating-tags groups (74%). The differences are statistically significant both in percent agreement ($p < 0.05$, Z-test of proportions) and mean response ($p < 0.05$, t-test). We found no other statistically significant differences in survey responses between the two groups.

We compared critiques applied by users in each group, and we found that users in the discriminating-tags group chose a positive ("more") direction for 71% of their

Top 10 positive	frac
nudity (full frontal -	0.19
nudity (full frontal)	0.15
sexuality	0.11
scary	0.09
nudity (topless)	0.09
lesbian	0.08
black comedy	0.08
psychological	0.07
dark comedy	0.07
cyberpunk	0.07

Top 10 negative	frac
coen brothers	0.08
religion	0.07
holocaust	0.07
world war ii	0.06
christmas	0.06
western	0.06
pixar	0.05
suicide	0.05
vampires	0.05
police	0.04

Table 5.4: System-suggested tags most likely to be used in each critique direction, based on the fraction of times the tags were displayed that users chose them for critiques.

critiques, compared to 66% for users in the descriptive-tags group ($p < 0.01$, Z-test of proportions). This may be explained by the fact that the tags displayed by the descriptive-tags algorithm had a mean relevance of 0.81 to the movie displayed, while those displayed by the discriminating-tags algorithm had a mean relevance of only 0.48. As we will discuss later, users were more likely to apply critiques in a positive direction when tag relevance was low. However, we found no significant differences in the number of critiques applied or the proportion of users who applied critiques in the two groups.

With the exception of the differences described above, users in the two tag-selection groups exhibited similar behavior and expressed approximately the same level of satisfaction with Movie Tuner. This shows that Movie Tuner can support a range of tag-selection algorithms, and system designers may wish to explore algorithms that incorporate other objectives. For example, a system might choose a set of tags that capture a range of moods, or it might choose tags that steer users toward items that are otherwise hard to find.

Table 5.4 shows the system-suggested tags users were most likely to choose in each critique direction[4]. For positive critiques, many of the top-10 tags had sexual themes; for negative critiques, many of the tags described sensitive topics such as *religion*, *holocaust*, or *suicide*.

[4] Based on the number of times users applied the tag in that direction divided by the number of times the tag was displayed. We only included tags displayed at least 100 times

Top 10 positive	count
nudity	36
comedy	32
mystery	30
nudity (topless)	29
romance	29
sex	28
action	26
surreal	21
funny	18
erotic	16

Top 10 negative	count
comedy	21
violence	9
violent	8
drugs	5
horror	5
sex	5
cheesy	4
dark	4
nudity	4
predictable	4

Table 5.5: User-entered tags most frequently used in each critique direction.

Users entered their own tag rather than choose a system-selected tag for 9% of critiques. Table 5.5 shows the most popular user-entered tags for each critique direction. For positive critiques, the top-10 tags reflect similar themes to what we saw for the system-selected tags. For negative critiques, several tags mirror the criteria used to determine MPAA ratings (*violence*, *drugs*, *sex*, *nudity*), suggesting that some users are seeking to avoid movies with content they consider objectionable, perhaps because they wish to find movies appropriate for a younger audience. In both directions, the user-entered tags appear to be more general than the system-selected tags, suggesting that users may find it easier to recognize specific tags than to recall them.

Choosing direction. Users applied 68% of their critiques in the positive ("more") direction, compared with 32% in the negative ("less") direction ($p < 0.001$, Z-test of proportions). We expected that users would be more likely to select "more" for low-relevance tags compared to high-relevance tags, since there is greater distance to travel in the positive direction. To test this hypothesis, we divided critiques into three buckets based on the relevance of the critique tag t to the critiqued item i: *low relevance* ($rel(t,i) < \frac{1}{3}$), *medium relevance* ($\frac{1}{3} \leq rel(t,i) < \frac{2}{3}$), and *high relevance* ($rel(t,i) \geq \frac{2}{3}$). The proportion of positive critiques in each bucket were 70.2%, 69.7%, and 66.1% respectively; the differences between the high relevance bucket and the other buckets were statistically significant ($p < 0.01$, Z-test of proportions), but the difference between the low and medium relevance buckets were not. These results show that lower tag relevance does correspond with a greater frequency of positive critiques, but that the

effect is fairly weak. Among critiques with $rel(t, i) > 0.95$, users still chose a positive direction 66% of the time. Future research should explore why users choose positive critiques most of the time: is it because tags tend to reflect attributes that people like, or because users find it more natural to navigate in a positive direction?

Unit versus compound critiques. Compound critiques were popular, comprising 24% of all critiques applied. 37% of users who applied a critique applied at least one compound critique. Several subjects who didn't realize compound critiquing was available asked for the feature in their comments. One subject wrote, *"I would like the Movie Tuner to permit adjusting two or more qualities at the same time. For example, if I am at the tuner for the movie 'The Girl Who Played with Fire', I would like to be able to search for movies that are both 'less violent and less sexually graphic'."*

5.6.2 Critique Results

We also analyzed how users interacted with the results that Movie Tuner displayed in response to their critiques. On average, users clicked on 1.2 results for every movie they critiqued[5] . Survey results indicate that users were satisfied with the critique results: 68% of subjects who applied critiques thought the critique results made sense, 71% felt that applying critiques helped them find movies they were interested in, and 65% thought that applying critiques helped them find movies they had not seen. To make it easier to find movies they had not seen, several users asked for the option to exclude movies they had already rated.

We compared how users in the *linear* and *diminishing-returns* groups (see Section 5.4.1) responded to critique results. We found no statistically significant differences between the groups based on user self-report or observational data such as number of click-throughs. Additional work is needed to determine the optimal algorithm for choosing items in response to user critiques. System designers may wish to incorporate other objectives in these algorithms, such as predicted item rating or diversity of items.

Besides displaying movies in response to users' critiques, Movie Tuner also displays an initial list of "similar movies" when the user first visits a movie page. This feature proved popular: users clicked on the "similar movies" 12,626 times. Survey results

[5] This count only includes results clicked while a critique was in place; users also clicked on the similar movies shown when no critique was in place.

indicate that users liked seeing the similar movies: 67% of subjects thought the similar movies helped them find movies they were interested in, and 54% thought it helped them to find movies they hadn't seen (versus 22% who did not think it helped find movies they hadn't seen). Further, 60% thought that the movies shown were actually similar to the main movie (versus 7% who did not), suggesting that the similarity metric worked properly.

5.7 Summary

In this chapter we introduced Movie Tuner, a system for navigating an information space using natural language critiques based on community tags. In contrast to traditional tag search, Movie Tuner lets users formulate their preferences adaptively by critiquing particular examples. In contrast to traditional example-critiquing systems, Movie Tuner builds its knowledge base automatically by applying machine learning to user-contributed content via the Tag Genome, rather than by relying on paid experts.

We approached this problem from a design perspective, exploring two design dimensions. First, we examined how the system should suggest tags to users. We implemented two algorithms, one that favored *descriptive* tags and one that favored *discriminating* tags. Survey participants felt that descriptive tags made more sense to them than discriminating tags. Users who saw descriptive tags tended to apply fewer positive critiques, most likely because descriptive tags represented attributes that were already fully present in the current item. Second, we explored how to choose items in response to users' critiques. We implemented *linear* and *diminishing returns* models of critique satisfaction based on critique distance. We found that users were equally satisfied and exhibited similar behavior with both approaches.

Initial tests suggest that Movie Tuner is an important and valuable tool. 89% of subjects liked being able to critique movies, and 79% wanted Movie Tuner to remain available on MovieLens. One user wrote, "*Movie Tuner instantly made MovieLens many times more valuable and useful for me! It generally works well and sometimes extremely well. Please keep it available!*" Over 1,000 users applied a total of 12,000 critiques, and views of movie detail pages on MovieLens increased by over 50%.

Since the results show that Movie Tuner may support a range of implementations,

we encourage system designers to explore alternate designs. For example, some users suggested they would like more fine-grain control over their critiques. One user wrote, "*As opposed to a less/more function, the ability to slide the bar and set an amount would be welcomed.*" Alternatively, system designers may explore more organic implementations such as speech-based interfaces; for example, someone listening to a personalized radio station could simply say "less classical" or "more mellow" to select a song that better fits their mood.

Chapter 6

Other Applications of the Tag Genome

Movie Tuner is just one example of the type of application that the Tag Genome can support; in this chapter we outline the broader space of applications of the Tag Genome.

6.1 Applications motivated by vector arithmetic

Many applications of the Tag Genome may be framed in terms of vector arithmetic, as illustrated in Figure 6.1. Figure 6.1a shows how the critiquing interaction in Movie Tuner can be interpreted as adding a vector of tag relevance differences ("more *action* and less *dialogue*") to the Tag Genome of the starting item (*Pulp Fiction*) to find the result (*Kill Bill, Vol. 1*), which has a Tag Genome that is close to the sum of these two vectors.

An inverted form of the same equation can be used to compare items, as shown in Figure 6.1b. In this case, the user specifies two items (*The Bourne Supremacy* and *Mission Impossible*), and the system computes the differences in the Tag Genomes of the items ("more *gritty*, more *realistic*, less *Tom Cruise*"). This type of comparison can help users understand the tradeoffs between items when making decisions [Pu and Chen, 2005, Jedetski et al., 2002]. The Tag Genome can also reveal similarities between items [Green et al., 2009], for example that *The Bourne Supremacy* and

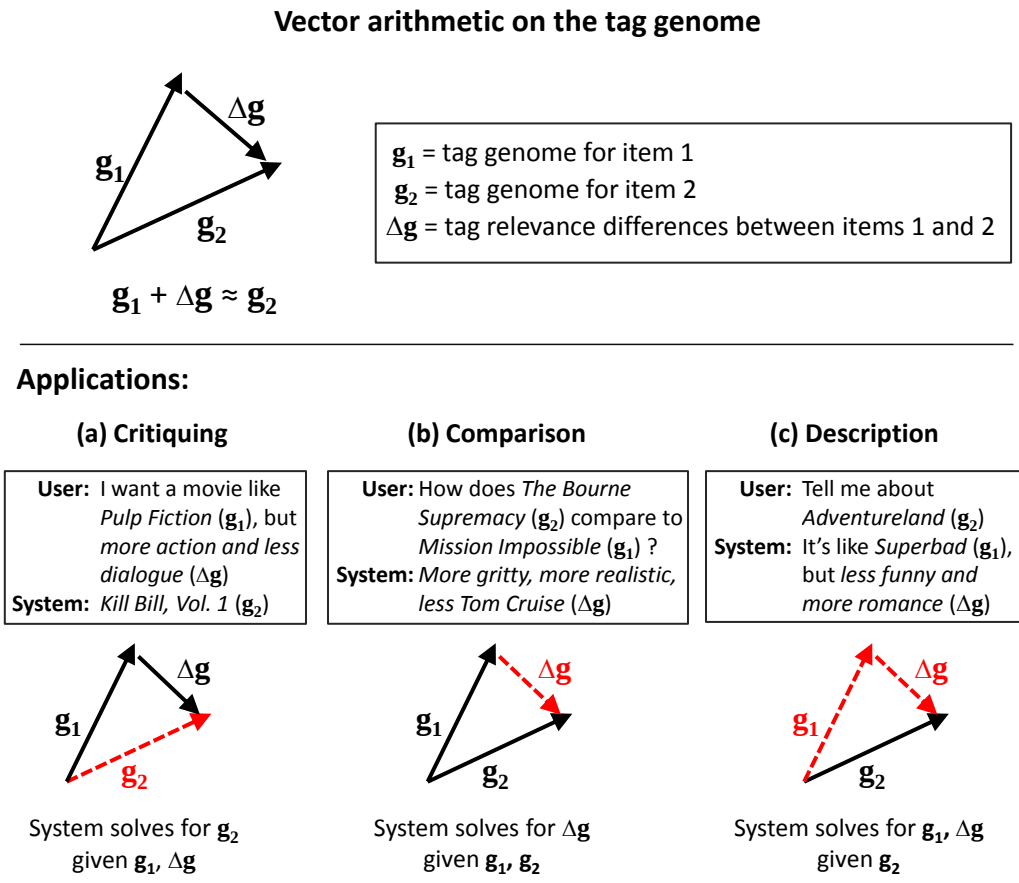

Figure 6.1: Applications of the Tag Genome inspired by vector arithmetic.

Mission Impossible both share the qualities *action*, *thriller*, and *spies*. Various applications – for example, recommender systems – display items that are similar to some reference item; research shows that users also want to understand *how* these items are similar [Hingston and Kay, 2006, Tintarev and Masthoff, 2007a, Vig et al., 2009].

As shown in Figure 6.1c, another application is to describe an item (*Adventureland*) in terms of a familiar item (*Superbad*) plus a vector of tag relevance differences ("less *funny*, more *romance*"). The familiar item could be an item that the user has rated in the past, or simply a popular item that is familiar to most users. The advantage of describing one item in terms of another is that it provides a very compact description compared to standard textual descriptions.

6.2 Applications motivated by genomics

In biological systems, genetic recombination serves to combine the traits of two organisms to produce a new organism; in the same way, recombining the Tag Genomes of two items can yield an item that shares traits from both items. One potential application is finding items that satisfy multiple users; for example, if Elena is in the mood for the movie *Sahara* (*adventure*, *action*), and her friend Jeff is in the mood for *The Bridges of Madison County* (*drama*, *romance*), the system might suggest the movie *Australia*, which possesses both sets of attributes. Another application is to describe a new item as a combination of two familiar items; for example, a system might describe the movie *The Time Traveler's Wife* as "*The Notebook* (*love story, sentimental*) meets *The Butterfly Effect* (*time travel, mind-bending*)".

6.3 Extensions to traditional tagging applications

Search. In traditional tag search, a user keys in a tag and the system retrieves items to which that tag has been applied. The Tag Genome enables users to more precisely specify their search criteria, indicating not only the tag they are interested in, but also the desired level of that tag, e.g. "medium level of action or less". Users could save these searches and use them as context-sensitive filters; for example, a user might have a filter for movies to watch with her young children (e.g. "no violence"), but another

for choosing movies to watch with her teenage brother (e.g. "medium level of violence or lower").

Browse. Traditional tagging systems support various browsing interfaces; users can browse the tags associated with a particular item, and then browse other items by clicking on one of the tags. Faceted navigation [Yee et al., 2003] provides a very different browsing experience that allows users to filter items by choosing options within various facets such as size (small, medium, large), price (less than $10, $10-$20, $20+), or location (Mexico, France, Egypt). Past work has explored faceted browsing systems using tags as facets [Feinstein and Smadja, 2006]; the Tag Genome can extend these systems by representing tags as continuous dimensions rather than binary attributes. Facets based on continuous dimensions can provide unique forms of interaction [Teevan et al., 2008]; instead of displaying check boxes next to each facet (tag), the system could provide slider controls for users to specify a range of relevance values, similar to the approach described in [Shneiderman, 1994]. Alternatively, the system could offer a set of discrete choices within each facet by binning relevance values (e.g. *low*, *medium*, *high*) [Teevan et al., 2008].

Item Summarization. Tagging systems often visualize the set of tags applied to an item as a cloud or list [Halvey and Keane, 2007, Bateman et al., 2008, Rivadeneira et al., 2007], providing users with a compact summarization of the item. Both types of visualizations convey the relative importance of tags, typically based on tag frequency: the most frequently applied tags are shown at the top of a tag list, or displayed with the largest font in a tag cloud. The $0 - 1$ relevance scale in the Tag Genome opens up the possibility of visualizing tag relevance on an absolute scale. For example, Movie Tuner displays "relevance meters" next to each tag (see Figure 5.1). Many other visual metaphors are possible, such as sliders on a sound mixer, or dials on a volume control.

Tag clouds and lists only show tags that users have applied to an item, reflecting the sparse nature of the traditional tagging model as discussed in Section 4. The Tag Genome, in contrast, encodes tag relevance for every tag in the genome, regardless of whether the tag has been applied to the item. This dense representation can support new forms of item summarization. For example, a system might display a scrollable

list with all the tags in the genome along with their relevance values, which users could sort by factors such as relevance to the item, relevance to the user (based on inferred preferences for the tag), or overall tag popularity. Users could also query the relevance of particular tags, for example by entering them in an autocomplete text box as with Movie Tuner. Alternatively, the system could choose a small number of tags to display based on factors such as tag relevance, diversity, or user interest, as discussed in Section 5.3.5.

6.4 The Tag Genome as feature vector

We have discussed how the Tag Genome enables new types of user interactions. But it can also support back-end processes that represent items as feature vectors in order to perform various computations. As a feature vector, the Tag Genome has two useful qualities: (1) it combines data from any number of sources into a compact and diverse set of features and (2) it is human comprehensible. Machine learning algorithms could use the Tag Genome as an input feature vector for classification or regression models; since these feature are human-interpretable, they can also be used to explain certain types of models to users [Poulin et al., 2006, Možina et al., 2004, Mooney and Roy, 2000, Vig et al., 2009] or even allow users to adjust the model [Kulesza et al., 2009]. Clustering algorithms could use the Tag Genome to compute similarity between items (see Section 5.4.2) as well as labeling the resulting clusters [Manning et al., 2008] based on the tags with high relevance across each cluster.

Chapter 7

Conclusion

In this thesis we explored intelligent tagging systems. Rather than using machine learning to automate navigation, these systems use machine learning to provide users with greater understanding and control in navigating an information space themselves. The user response was overwhelmingly positive; users expressed that these systems helped them understand the information space, make better decisions, and find items of interest to them.

We first explored Tagsplanations, explanations of recommendations based on community tags. We discussed two key components of Tagsplanations: tag relevance, the degree to which a tag describes an item, and tag preference, the user's sentiment toward a tag. We then described novel algorithms for computing both components. Survey results suggest that Tagsplanations serve not only to explain recommendations, but also to help users make better decisions about items and to assess how well items fit their current mood.

We then introduced the Tag Genome, a data structure that uses machine learning to augment human judgments of the relationships between tags and items. We showed how the Tag Genome addresses limitations of traditional tagging systems and supports new forms of interaction. We provided general guidelines for computing the Tag Genome, and we demonstrated this approach by computing the Tag Genome in the movie domain.

Next we showed how to use the Tag Genome to enable a new kind of tagging interaction in the form of Movie Tuner. We discussed the design of Movie Tuner, and we presented the results of 7-week field study of Movie Tuner on MovieLens. Results show

that Movie Tuner was very popular among MovieLens users and that users enjoyed the additional control that Movie Tuner provided.

One limitation of these systems is that the underlying algorithms are not transparent to the end user. For example, both Movie Tuner and Tagsplanations display tag relevance values, but neither system explains how these values are computed. An interesting research direction would be to take the explanations one level deeper and allow users to explore how these values are computed. For example, Movie Tuner could show excerpts from the user reviews [Yatani et al., 2011] that influenced the computed tag relevance values. Tagsplanations could show users which past ratings led the system to its estimate of tag preference.

Even better, the systems could allow users to scrutinize these algorithms and provide feedback on the computed values. For example, if a user disagrees with their inferred preference for a tag, she could override the value by indicating a different number of stars. Or if a user of Movie Tuner disagreed with a tag relevance value, she could simply drag the slider to the correct position. The underlying algorithms could then incorporate this feedback into its future computations.

We designed Movie Tuner and Tagsplanations as distinct systems, but they also have much in common: both systems describe the relationship between tags and items using continuously-valued tag relevance values, and both provide an interaction around a specific item. Future work may explore unified systems that combine elements from both systems. For example, the Tag Genome that drives Movie Tuner could also provide the tag relevance values for Tagsplanations. The same type of machine learning approach for computing the Tag Genome might also be used to compute users' preferences for tags, providing a tag-user genome to complement the tag-item genome described in this thesis.

System designers might also explore Tagsplanation systems that incorporate the critiquing interaction of Movie Tuner; instead of simply explaining the predicted rating to users, the Tagsplanation could play an active role in guiding users to other items. For example, after a Tagsplanation tells a user that it has given a low prediction to *Reservoir Dogs* because it is *violent*, it could guide the user to other movies that are similar but less violent.

We have presented several systems that combine user-applied tags with automated

machine learning algorithms. Our results suggest that human intelligence and machine intelligence are not opposing design elements but rather complementary. We showed how human judgments of tag relevance can guide the machine learning process that computes the Tag Genome; the Tag Genome, in turn, can help users to understand and explore an information space themselves. We hope that this work will stimulate more research into the rich interplay between human and machine intelligence.

Chapter 8

References

[Ames and Naaman, 2007] Ames, M. and Naaman, M. (2007). Why we tag: Motivations for annotation in mobile and online media. In *CHI '07: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 971–980, New York, NY, USA. ACM.

[Bateman et al., 2008] Bateman, S., Gutwin, C., and Nacenta, M. (2008). Seeing things in the clouds: The effect of visual features on tag cloud selections. In *Hypertext '08: Proceedings of the Nineteenth ACM Conference on Hypertext and Hypermedia*, pages 193–202, New York, NY, USA. ACM.

[Bates and Sarkar, 2007] Bates, D. and Sarkar, D. (2007). Linear mixed-effects models using S4 classes.

[Bernstein et al., 2009] Bernstein, M., Tan, D., Smith, G., Czerwinski, M., and Horvitz, E. (2009). Collabio: a game for annotating people within social networks. In *UIST '09: Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology*, pages 97–100, New York, NY, USA. ACM.

[Bilgic and Mooney, 2005] Bilgic, M. and Mooney, R. J. (2005). Explaining recommendations: Satisfaction vs. promotion. In *Proceedings of Beyond Personalization Workshop, IUI*.

[Billsus and Pazzani, 1999] Billsus, D. and Pazzani, M. J. (1999). A personal news agent that talks, learns and explains. In *AGENTS '99: Proceedings of the Third*

Annual Conference on Autonomous Agents, pages 268–275, New York, NY, USA. ACM.

[Bird et al., 2009] Bird, S., Loper, E., and Klein, E. (2009). *Natural Language Processing with Python*. O'Reilly Media Inc.

[Burke, 2002] Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370.

[Burke et al., 1996] Burke, R. D., Hammond, K. J., and Young, B. C. (1996). Knowledge-based navigation of complex information spaces. In *Proceedings of the 13th National Conference on Artificial Intelligence*, pages 462–468. AAAI Press.

[Burke et al., 1997] Burke, R. D., Hammond, K. J., and Young, B. C. (1997). The FindMe approach to assisted browsing. *IEEE Expert*, 12:32–40.

[Cattuto et al., 2007] Cattuto, C., Loreto, V., and Pietronero, L. (2007). Semiotic dynamics and collaborative tagging. *Proceedings of the National Academy of Sciences*, 104(5):1461–1464.

[Chen and Pu, 2006] Chen, L. and Pu, P. (2006). Evaluating critiquing-based recommender agents. In *AAAI 2006: Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1*, pages 157–162.

[Cosley et al., 2003] Cosley, D., Lam, S. K., Albert, I., Konstan, J. A., and Riedl, J. (2003). Is seeing believing? How recommender system interfaces affect users' opinions. In *CHI '03: Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 585–592, New York, NY, USA. ACM.

[Deerwester et al., 1990] Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., and Harshman, R. A. (1990). Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407.

[Drucker et al., 1997] Drucker, H., Burges, C. J. C., Kaufman, L., Smola, A., and Vapnik, V. (1997). Support vector regression machines. *NIPS '97: Advances in Neural Information Processing Systems*, 9:155–161.

[Eck et al., 2007] Eck, D., Lamere, P., Bertin-Mahieux, T., and Green, S. (2007). Automatic generation of social tags for music recommendation. In *NIPS '07: Advances in Neural Information Processing Systems*, volume 20.

[Ellenberg, 2008] Ellenberg, J. (2008). The psychologist might outsmart the math brains competing for the netflix prize. *Wired Magazine*.

[Faltings et al., 2004] Faltings, B., Pu, P., Torrens, M., and Viappiani, P. (2004). Designing example-critiquing interaction. In *IUI '04: Proceedings of the 9th International Conference on Intelligent User Interfaces*, pages 22–29, New York, NY, USA. ACM.

[Feinstein and Smadja, 2006] Feinstein, D. and Smadja, F. (2006). Hierarchical tags and faceted search: The rawsugar approach. In *SIGIR 2006 Workshop on Faceted Search*, pages 23–25.

[Fletcher, 1981] Fletcher, R. (1981). *Practical Methods of Optimization: Vol. 2: Constrained Optimization*. John Wiley and Sons.

[Gelman et al., 2003] Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (2003). *Bayesian Data Analysis, Second Edition*. Chapman & Hall/CRC.

[Gelman and Hill, 2007] Gelman, A. and Hill, J. (2007). *Data analysis using regression and multilevel hierarchical models*. Cambridge University Press, New York.

[Golder and Huberman, 2006] Golder, S. A. and Huberman, B. A. (2006). Usage patterns of collaborative tagging systems. *Journal of Information Science*, 32:198–208.

[Green et al., 2009] Green, S. J., Lamere, P., Alexander, J., Maillet, F., Kirk, S., Holt, J., Bourque, J., and Mak, X.-W. (2009). Generating transparent, steerable recommendations from textual descriptions of items. In *RecSys '09: Proceedings of the 2009 ACM Conference on Recommender Systems*, pages 281–284, New York, NY, USA. ACM.

[Guyon and Elisseeff, 2003] Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182.

[Halvey and Keane, 2007] Halvey, M. J. and Keane, M. T. (2007). An assessment of tag presentation techniques. In *WWW '07: Proceedings of the 16th International Conference on the World Wide Web*, pages 1313–1314, New York, NY, USA. ACM.

[Herlocker et al., 2000] Herlocker, J. L., Konstan, J. A., and Riedl, J. (2000). Explaining collaborative filtering recommendations. In *CSCW '00: Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, pages 241–250, New York, NY, USA. ACM.

[Heymann et al., 2008] Heymann, P., Ramage, D., and Garcia-Molina, H. (2008). Social tag prediction. In *SIGIR '08: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 531–538, New York, NY, USA. ACM.

[Hingston and Kay, 2006] Hingston, M. and Kay, S. J. (2006). User friendly recommender systems (honors thesis).

[Jedetski et al., 2002] Jedetski, J., Adelman, L., and Yeo, C. (2002). How web site decision technology affects consumers. *IEEE Internet Computing*, 6:72–79.

[Jschke et al., 2007] Jschke, R., Marinho, L. B., Hotho, A., Schmidt-Thieme, L., and Stumme, G. (2007). Tag recommendations in folksonomies. In *PKDD '07: Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases*, volume 4702, pages 506–514. Springer.

[Kammerer et al., 2009] Kammerer, Y., Nairn, R., Pirolli, P., and Chi, H. (2009). Signpost from the masses: Learning effects in an exploratory social tag search browser. In *CHI '09: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 625–634.

[Kelley, 1999] Kelley, C. T. (1999). Iterative methods for optimization. *SIAM Frontiers in Applied Mathematics*, (18).

[Kohavi and John, 1997] Kohavi, R. and John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273 – 324.

[Kulesza et al., 2009] Kulesza, T., Wong, W.-K., Stumpf, S., Perona, S., White, R., Burnett, M. M., Oberst, I., and Ko, A. J. (2009). Fixing the program my computer learned: Barriers for end users, challenges for the machine. In *IUI '09: Proceedings of the 14th International Conference on Intelligent User Interfaces*, pages 187–196, New York, NY, USA. ACM.

[Linden et al., 1997] Linden, G., Hanks, S., and Lesh, N. (1997). Interactive assessment of user preference models: The automated travel assistant. In *User Modeling '97*, pages 67–78. Springer.

[Manning et al., 2008] Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

[Marlow et al., 2006] Marlow, C., Naaman, M., Boyd, D., and Davis, M. (2006). HT06, tagging paper, taxonomy, flickr, academic article, to read. In *Hypertext '06: Proceedings of the Seventeenth Conference on Hypertext and Hypermedia*, pages 31–40, New York, NY, USA. ACM.

[McCallum and Nigam, 1998] McCallum, A. and Nigam, K. (1998). A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, pages 41–48. AAAI Press.

[McCarthy et al., 2005] McCarthy, K., Reilly, J., McGinty, L., and Smyth, B. (2005). Experiments in dynamic critiquing. In *IUI '05: Proceedings of the 10th International Conference on Intelligent User Interfaces*, pages 175–182, New York, NY, USA. ACM.

[McCullagh and Nelder, 1989] McCullagh, P. and Nelder, J. (1989). *Generalized Linear Models, Second Edition*. Chapman & Hall/CRC.

[Mooney and Roy, 2000] Mooney, R. J. and Roy, L. (2000). Content-based book recommending using learning for text categorization. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, pages 195–204, New York, NY, USA. ACM.

[Možina et al., 2004] Možina, M., Demšar, J., Kattan, M., and Zupan, B. (2004). Nomograms for visualization of naive bayesian classifier. In *PKDD '04: Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 337–348, New York, NY, USA. Springer-Verlag New York, Inc.

[Niwa et al., 2006] Niwa, S., Doi, T., and Honiden, S. (2006). Web page recommender system based on folksonomy mining. In *ITNG '06*, pages 388–393, Washington, DC, USA. IEEE Computer Society.

[Page et al., 1999] Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab. Previous number = SIDL-WP-1999-0120.

[Payne et al., 1993] Payne, J. W., Bettman, J., and Johnson, E. J. (1993). *The Adaptive Decision Maker*. Cambridge University Press.

[Porter, 1980] Porter, M. (1980). An algorithm for suffix stripping. *Program: Electronic Library and Information Systems*, 14(3):130–137.

[Poulin et al., 2006] Poulin, B., Eisner, R., Szafron, D., Lu, P., Greiner, R., Wishart, D. S., Fyshe, A., Pearcy, B., MacDonell, C., and Anvik, J. (2006). Visual explanation of evidence in additive classifiers. In *Proceedings of the 18th Conference on Innovative Applications of Artificial Intelligence - Volume 2*, pages 1822–1829. AAAI Press.

[Pu and Chen, 2005] Pu, P. and Chen, L. (2005). Integrating tradeoff support in product search tools for e-commerce sites. In *EC '05: Proceedings of the 6th ACM Conference on Electronic Commerce*, pages 269–278, New York, NY, USA. ACM.

[R Development Core Team, 2010] R Development Core Team (2010). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

[Raudenbush and Bryk, 2002] Raudenbush, S. and Bryk, A. (2002). *Hierarchical Linear Models (Second Edition)*. Sage Publications, Thousand Oaks, CA.

[Resnick et al., 1994] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). GroupLens: An open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, North Carolina, United States. ACM Press.

[Rivadeneira et al., 2007] Rivadeneira, A. W., Gruen, D. M., Muller, M. J., and Millen, D. R. (2007). Getting our head in the clouds: Toward evaluation studies of tagclouds. In *CHI '07: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 995–998, New York, NY, USA. ACM.

[Salton and McGill, 1983] Salton, G. and McGill, M. (1983). *Introduction to Modern Information Retrival*. McGraw-Hill.

[Salton et al., 1975] Salton, G., Wong, A., and Yang, C. S. (1975). A vector space model for automatic indexing. *Commun. ACM*, 18:613–620.

[Sarwar et al., 2001] Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th International Conference on the World Wide Web*, pages 285–295, Hong Kong. ACM Press.

[Schrammel et al., 2009] Schrammel, J., Leitner, M., and Tscheligi, M. (2009). Semantically structured tag clouds: An empirical evaluation of clustered presentation approaches. In *CHI '09: Proceedings of the 27th International Conference on Human Factors in Computing Systems*, pages 2037–2040, New York, NY, USA. ACM.

[Sen et al., 2007] Sen, S., Harper, F. M., LaPitz, A., and Riedl, J. (2007). The quest for quality tags. In *GROUP '07: Proceedings of the 2007 International ACM Conference on Supporting Group Work*, pages 361–370, New York, NY, USA. ACM.

[Sen et al., 2006] Sen, S., Lam, S. K., Rashid, A. M., Cosley, D., Frankowski, D., Osterhouse, J., Harper, F. M., and Riedl, J. (2006). tagging, communities, vocabulary, evolution. In *CSCW '06: Proceedings of the 2006 ACM Conference on Computer Supported Cooperative Work*, pages 181–190, New York, NY, USA. ACM.

[Sen et al., 2009] Sen, S., Vig, J., and Riedl, J. (2009). Tagommenders: Connecting users to items through tags. In *WWW '09: Proceedings of the 18th International Converence on Word Wide Web*, pages 671–680, New York, NY, USA. ACM.

[Shirky, 2005] Shirky, C. (2005). Ontology is overrated. http://www.shirky.com/writings/ontology_overrated.html.

[Shneiderman, 1994] Shneiderman, B. (1994). Dynamic queries for visual information seeking. *IEEE Software*, 11:70–77.

[Sigurbjörnsson and van Zwol, 2008] Sigurbjörnsson, B. and van Zwol, R. (2008). Flickr tag recommendation based on collective knowledge. In *WWW '08: Proceeding of the 17th International Conference on the World Wide Web*, pages 327–336, New York, NY, USA. ACM.

[Sinha and Swearingen, 2002] Sinha, R. and Swearingen, K. (2002). The role of transparency in recommender systems. In *CHI '02 Extended Abstracts on Human Factors in Computing Systems*, pages 830–831, New York, NY, USA. ACM.

[Smith et al., 1990] Smith, D. C., Irby, C., Kimball, R., Berplank, B., and Harslem, E. (1990). Designing the Star user interface. *Human-Computer Interaction*, pages 237–259.

[Smyth et al., 2004] Smyth, B., McGinty, L., Reilly, J., and McCarthy, K. (2004). Compound critiques for conversational recommender systems. In *Web Intelligence '04*, pages 145–151, Washington, DC, USA. IEEE Computer Society.

[Sparling and Sen, 2011] Sparling, E. I. and Sen, S. (2011). Rating: How difficult is it? In *RecSys '11: Proceedings of the 2011 ACM Conference on Recommender Systems*, New York, NY, USA. ACM.

[Teevan et al., 2008] Teevan, J., Dumais, S., and Gutt, Z. (2008). Challenges for supporting faceted search in large, heterogeneous corpora like the web. In *HCIR '08: The Second Workshop on Human-Computer Interaction and Information Retrieval*, Redmond, WA, USA.

[Tintarev, 2007] Tintarev, N. (2007). Explanations of recommendations. In *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*, pages 203–206, New York, NY, USA. ACM.

[Tintarev and Masthoff, 2007a] Tintarev, N. and Masthoff, J. (2007a). Effective explanations of recommendations: User-centered design. In *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*, pages 153–156, New York, NY, USA. ACM.

[Tintarev and Masthoff, 2007b] Tintarev, N. and Masthoff, J. (2007b). A survey of explanations in recommender systems. In *IEEE 23rd International Conference on Data Engineering Workshop*, pages 801–810.

[Ulges et al., 2008] Ulges, A., Schulze, C., Keysers, D., and Breuel, T. M. (2008). A system that learns to tag videos by watching YouTube. In *ICVS'08: Proceedings of the 6th International Conference on Computer Vision Systems*, pages 415–424, Berlin, Heidelberg. Springer-Verlag.

[Vig et al., 2009] Vig, J., Sen, S., and Riedl, J. (2009). Tagsplanations: Explaining recommendations using tags. In *IUI '09: Proceedings of the 13th International Conference on Intelligent User Interfaces*, pages 47–56, New York, NY, USA. ACM.

[Vig et al., 2010] Vig, J., Soukup, M., Sen, S., and Riedl, J. (2010). Tag expression: tagging with feeling. In *UIST '10: Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology*, pages 323–332, New York, NY, USA. ACM.

[Wu et al., 2009] Wu, L., Yang, L., Yu, N., and Hua, X.-S. (2009). Learning to tag. In *WWW '09: Proceedings of the 18th international Conference on the World Wide Web*, pages 361–370, New York, NY, USA. ACM.

[Yatani et al., 2011] Yatani, K., Novati, M., Trusty, A., and Truong, K. N. (2011). Review spotlight: a user interface for summarizing user-generated reviews using adjective-noun word pairs. In *CHI '11: Proceedings of the 2011 Annual Conference on Human Factors in Computing Systems*, pages 1541–1550, New York, NY, USA. ACM.

[Yee et al., 2003] Yee, K.-P., Swearingen, K., Li, K., and Hearst, M. (2003). Faceted metadata for image search and browsing. In *CHI '03: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 401–408, New York, NY, USA. ACM.

[Zhai and Lafferty, 2001] Zhai, C. and Lafferty, J. (2001). A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR '01: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 334–342, New York, NY, USA. ACM.

[Zhang and Pu, 2006] Zhang, J. and Pu, P. (2006). A comparative study of compound critique generation in conversational recommender systems. In *Adaptive Hypermedia '06*, pages 234–243. Springer.

Appendix A

Smoothing algorithm

Here we present the details of the algorithm for smoothing text-freq(t, i). We use a Bayesian approach in which we treat term frequency as the unknown parameter of a Binomial distribution with a Beta prior. Our approach is similar to Bayesian smoothing with a Dirichlet prior [Zhai and Lafferty, 2001], but is simpler to implement as it handles each term independently instead of using a joint multinomial distribution.

Let $N_{t,i}$ denote the number of occurrences of tag t in user reviews of item i. Let N_i denote the total number of terms (including tag and non-tag terms) in user reviews of item i.

We treat each $N_{t,i}$ as an observation from a binomial distribution with a beta prior specific to each tag:

$$N_{t,i} \sim \text{Binomial}(N_i, p_{t,i}) \qquad p_{t,i} \sim \text{Beta}(\alpha_t, \beta_t)$$

$p_{t,i}$ represents the probability that an arbitrary term in the reviews of i equals t. We use the posterior estimate of $p_{t,i}$ as the smoothed value of text-freq(t, i).

To compute the posterior estimate of $p_{t,i}$, we first estimate the hyperparameters α_t and β_t. We follow the Empirical Bayes method [Gelman et al., 2003] and estimate α_t and β_t from the data. To do this, we first compute non-Bayesian estimates of $p_{t,i}$:

$$\hat{p}_{t,i} = \frac{N_{t,i}}{N_i}$$

We then compute the sample mean and variance of these values across all items:

$$\mu_t = \frac{\sum_{i \in I} \hat{p}_{t,i}}{|I|} \qquad \sigma_t^2 = \frac{\sum_{i \in I} (\hat{p}_{t,i} - \mu_t)^2}{|I|}$$

We then compute the parameters α_t and β_t based on the sample mean and variance:

$$\alpha_t = \mu_t\left(\frac{\mu_t(1-\mu_t)}{\sigma_t^2} - 1\right) \qquad \beta_t = (1-\mu_t)\left(\frac{\mu_t(1-\mu_t)}{\sigma_t^2} - 1\right)$$

Finally, we compute the posterior estimate of $p_{t,i}$:

$$p_{t,i} = \frac{N_{t,i} + \alpha_t}{N_i + \alpha_t + \beta_t}$$

which serves as the smoothed value for text-freq(t, i)

Appendix B

Sampling algorithm

Algorithm 1: Assign tags and items to users for survey

Input: tags T, items I, users U, tag relevance classfier `tag-rel-classify`
Output: set of tags T_u for each user $u \in U$ and set of items $I_{u,t}$ for each user u and tag $t \in T_u$
Description: Assign tags and movies to each user taking the survey. The input function `tag-rel-classify` is for the startified sampling (see Section 4.4.3).

foreach $t \in T$ **do**
  `// Partition items into low, medium, and high relevance buckets.`
  $low_t \leftarrow \{i \in I : \texttt{tag-rel-classify}(t, i) = low\}$
  $med_t \leftarrow \{i \in I : \texttt{tag-rel-classify}(t, i) = medium\}$
  $high_t \leftarrow \{i \in I : \texttt{tag-rel-classify}(t, i) = high\}$
  $AssignmentEase_t \leftarrow \min(\sum_{i \in low_t} |R_i|, \sum_{i \in med_t} |R_i|, \sum_{i \in high_t} |R_i|)$
    `/*` R_i `= ratings for item` i`. Since we only assign tag` t `to users who have rated at least 2 items in` low_t`,` med_t`, and` $high_t$`, the ease of assigning the tag is based on how many ratings exist for items in those groupings. */`
end
foreach $u \in U$ **do**
  $T_u \leftarrow \emptyset$ `/* Initialize each user's tags to empty set */`
end
$NumTagsPerUser \leftarrow 8$
$NumUserTagPairs \leftarrow |U| \times NumTagsPerUser$
Sort T by sortkey(t) = $AssignmentEase_t$ `/* Sort tags by how easy they are to assign */`
foreach $t \in T$ **do**
  `// Choose number of users proportional to log(popularity(`t`)), where`
  `// popularity(`t`) = number of distinct users who have applied` t`. This way we`
  `// we get more data for popular tags that more people care about.`
  $NumUsersForTag \leftarrow NumUserTagPairs \times \texttt{log}(\texttt{popularity}(t))/\sum_{t' \in T} \texttt{log}(\texttt{popularity}(t'))$
  $NumUsersAssigned \leftarrow 0$
  Sort U by sortkey(u) = $|T_u|$ `/* Sort users by number of tags assigned so far */`
  foreach $u \in U$ **do**
    if $|T_u| == NumTagsPerUser$ **then**
      break `/* Remaining users in` U `have been assigned all of their tags */`
    end
    `// Construct 3 buckets of items user has rated that have low, medium, and`
    `// high relevance with respect to tag` t`.`
    $bucket_1 \leftarrow low_t \cap \texttt{items-rated-by}(u)$
    $bucket_2 \leftarrow med_t \cap \texttt{items-rated-by}(u)$
    $bucket_3 \leftarrow high_t \cap \texttt{items-rated-by}(u)$
    if $|bucket_1| \geq 2$ *and* $|bucket_2| \geq 2$ *and* $|bucket_3| \geq 2$ **then**
      $T_u \leftarrow T_u \cup \{t\}$ `/* Assign tag to user */`
      `// Randomly sample 2 items from each bucket`
      $I_{u,t} \leftarrow \texttt{sample}(bucket_1, 2) \cup \texttt{sample}(bucket_2, 2) \cup \texttt{sample}(bucket_3, 2)$
      $NumUsersAssigned \leftarrow NumUsersAssigned + 1$
    end
    if $NumUsersAssigned == NumUsersForTag$ **then**
      break`// Done assigning tag to users`
    end
  end
end
return T_u, $I_{u,t}$