The Tag Genome: Encoding Community Knowledge to Support Novel Interaction

JESSE VIG, University of Minnesota
SHILAD SEN, Macalester College
JOHN RIEDL, University of Minnesota

This article introduces the tag genome, a data structure that extends the traditional tagging model to provide enhanced forms of user interaction. Just as a biological genome encodes an organism based on a sequence of genes, the tag genome encodes an item in an information space based on its relationship to a common set of tags. We present a machine learning approach for computing the tag genome, and we evaluate several learning models on a ground truth dataset provided by users. We describe an application of the tag genome called Movie Tuner which enables users to navigate from one item to nearby items along dimensions represented by tags. We present the results of a 7-week field trial of 2,531 users of Movie Tuner and a survey evaluating users' subjective experience. Finally, we outline the broader space of applications of the tag genome.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces—*Collaborative computing*; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces

General Terms: Algorithms, Design, Experimentation, Human Factors

Additional Key Words and Phrases: Tagging, recommender systems, conversational recommenders, machine learning, data mining, information retrieval

ACM Reference Format:
Vig, J., Sen, S., and Riedl, J. 2012. The tag genome: Encoding community knowledge to support novel interaction. ACM Trans. Interact. Intell. Syst. 2, 3, Article 13 (September 2012), 44 pages.
DOI = 10.1145/2362394.2362395 http://doi.acm.org/10.1145/2362394.2362395

1. INTRODUCTION

Tagging systems have flourished across the Web. Users of tagging systems create free-form text descriptors of music, pictures, or encyclopedia articles and use these descriptors to navigate complex information spaces. Users may search items by the tags that they are most interested in, and they can make sense of an item by the tags that other users have applied. Tags can also be used to organize a collection of items, or as a way for users to express their opinions on items to the user community.

While tagging systems provide great value to users, we believe there is an opportunity to develop many new and exciting tagging applications that go beyond traditional

---

Since one of the authors of this article is an editor-in-chief of ACM TiiS, the reviewing of the manuscript was managed in accordance with ACM's policy for such situations, by a TiiS associate editor whose identify was not revealed to the authors.
This material is based upon work supported by the National Science Foundation under grants IIS 03-24851, IIS 05-34420, IIS 09-64695, IIS 09-64697, IIS 08-08692, and IIS 07-29344.
Author's address: J. Vig (corresponding author); email: jesse.vig@gmail.com.

DOI 10.1145/2362394.2362395 http://doi.acm.org/10.1145/2362394.2362395

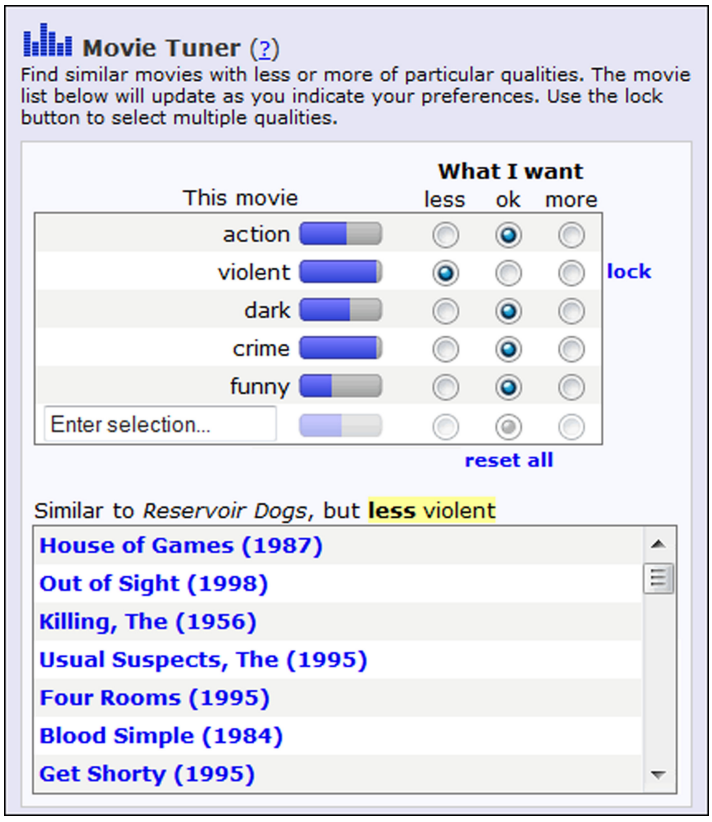

Fig. 1. Movie Tuner interface for *Reservoir Dogs* after a user applies the critique "less violent".

tag browse and search. However, in order to develop many of these applications, a more expressive tagging model is needed. In the following section we introduce a novel tagging application called Movie Tuner that motivates a new type of tagging model that we call the *tag genome*.

1.1. Movie Tuner: A Novel Tagging Interaction

Movie Tuner enables users to navigate an information space using tags, but in a fundamentally different way than traditional tag search or browse. Consider the following 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 is a novel tagging application that enables users to navigate an information space much like Marco did. Figure 1 shows the Movie Tuner interface as

Marco would 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.

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 displays the relevance of tags to items on a explicit and interpretable scale, visualized with the relevance meters shown in Figure 1. This allows users to decide if the level of a particular tag is too high or low for their taste. Behind the scenes, these relevance values allow the system to compare items with respect to tags ("*The Usual Suspects* is *less violent* than Reservoir Dogs").

1.2. Characteristics of the Traditional Tagging Model

In order to support this type of interaction, a new tagging model is needed. The traditional tagging model has two basic properties that make it difficult to apply to the use case described in the preceding text.

Tags are binary. In most traditional tagging systems, users can only express binary relationships between tags and items[1]. 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 like Movie Tuner to compare the relative levels of a tag across items. This is a particular challenge for tagging systems based on the *set* model of tagging [Marlow et al. 2006], which states that the user community may collectively apply a tag just once to an item. The *bag* model of tagging [Marlow et al. 2006], in contrast, allows multiple users to apply the same tag to the same item. Tagging systems based on this model can use tag frequency as an additional signal of tag relevance; if many users apply a particular tag to an item, 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 [Golder and Huberman 2006; Sen et al. 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. For Movie Tuner, it is just as important to identify tags that are not

[1]There has been some work in non-binary tagging systems such as Eck et al. [2007], discussed in Section 2.1.

relevant to each item as it is to identify tags that are highly relevant, since users may request either less or more of a particular tag.

1.3. Extending the Traditional Tagging Model: The Tag Genome

To address these limitations, we introduce an extension of the traditional tagging model 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 3.

The tag genome provides all of the information that Movie Tuner needs to support the interaction we just described. For the scenario depicted in Figure 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 5.2.2), enabling Movie Tuner to find movies that are similar overall to *Reservoir Dogs*.

Movie Tuner 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 Section 6.

1.4. Computing the Tag Genome

In order to implement Movie Tuner or the other applications listed, we first need to construct the underlying tag genome. We describe 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 dataset used to train the model.

1.5. Paper Overview

This article builds on our earlier published work, in which we introduce Movie Tuner [Vig et al. 2011]. In this article, we discuss the tag genome in much more depth: we describe how to compute the tag genome and present a case study of computing the tag genome in the movie domain. We also expand the related work and propose other applications of the tag genome in addition to Movie Tuner.

The structure of this article is as follows. We first present related work. We compare the tag genome to existing data models, and we discuss how Movie Tuner builds on a class of applications called *example-critiquing* systems. We then formally define the tag genome and present guidelines for computing the tag genome using a machine

learning approach. Next, we apply these guidelines to compute the tag genome on MovieLens, a movie recommender system that also supports tagging of movies. We then discuss the design of Movie Tuner and present results from a 7-week field trial. Finally, we propose several other applications of the tag genome.

2. RELATED WORK

Sections 2.1 and 2.2 discuss literature related to the tag genome, while Sections 2.3 and 2.4 focus on systems related to Movie Tuner.

2.1. Tag Prediction

To compute the tag genome, we develop algorithms that predict the relevance of tags to items on a continuous scale. Similarly, traditional tagging systems may utilize *tag prediction* algorithms that automatically predict the relevant tags for an item [Jäschke et al. 2007; Sigurbjörnsson and van Zwol 2008; Wu et al. 2009]. In the following text, we present several examples of tag prediction algorithms that span a variety of domains and techniques.

Heyman et al. [2008] explore techniques for predicting tags for URLs on the Web site del.icio.us[2]. 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. They also use association rule mining to predict tags for a URL based on the other tags that have been applied to the URL.

Ulges et al. [2008] developed a system to predict tags for videos from YouTube[3] 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.

Eck et al. [2007] explore the problem of predicting tags for musical artists. They build a multiclass 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.

Symeonidis et al. [2008] develop a personalized approach to tag prediction, recommending the tags that are relevant to an item for a particular user. Their approach is to apply tensor factorization to the tag-item-user space, transforming tags, items, users, and their interactions into a lower dimensional space. The output of the algorithm is a continuously valued estimate of each user's affinity for each tag with respect to each item, which is used to generate a personalized list of tag recommendations for each item. They evaluate their approach using tagging data from Bibsonomy[4] and Last.fm[5].

[2]http://www.delicious.com/
[3]http://www.youtube.com
[4]http://www.bibsonomy.org
[5]http://last.fm

Like many of the approaches just described, we use machine learning techniques 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, in contrast to Heymann et al. [2008] and Ulges et al. [2008], we predict relevance values on a continuous 0–1 scale rather than predict a binary (*applies*, *does not apply*) output. Eck et al. [2007] also use a non-binary relevance scale. However, they base their ternary relevance scale on tag frequency rather than explicit relevance ratings from individual users.

Like Symeonidis et al. [2008], we apply dimensionality reduction techniques to overcome the sparsity in tagging systems. As described in Section 4.2, we perform singular value decomposition on the tag-item space in computing input features to the tag genome. We do not decompose the full tag-item-user space like Symeonidis et al. [2008], since we do not consider the individual user when predicting tag relevance. The personalized extension of the tag genome proposed in Section 3.2 may benefit from a tensor-based approach, however.

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. VSM represents each document 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 of 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 6) is analogous to the term-document matrix for the VSM. 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 vector space model are based on term frequency.

Many applications of the tag genome are related to traditional VSM applications. 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 5.2.2 we describe how to compute the cosine similarity between two tag genomes using a term-weighting function based on tf-idf. Item retrieval in Movie Tuner (see Section 5.2) is analogous to document retrieval based on the VSM. The

Fig. 2. Entree example-critiquing system for restaurants [Burke 2002].

primary difference is that Movie Tuner retrieves items based on the difference in relevance values instead of always retrieving the items with the highest relevance.

2.3. Example-Critiquing Systems

Researchers have explored conversational recommenders that allow users to give immediate feedback on recommendations and then adjust recommendations accordingly [Burke et al. 1997; Faltings et al. 2004; Linden et al. 1997; 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 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 3 and 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 4, critique dimensions include 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.

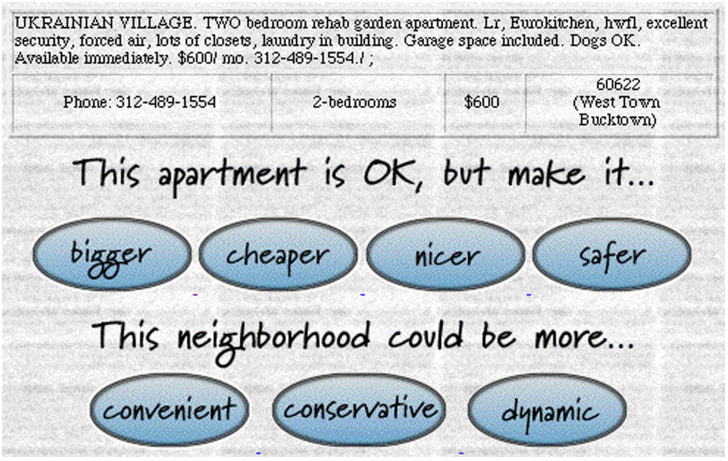

Fig. 3. RentMe example-critiquing system for apartment rentals [Burke et al. 1996].

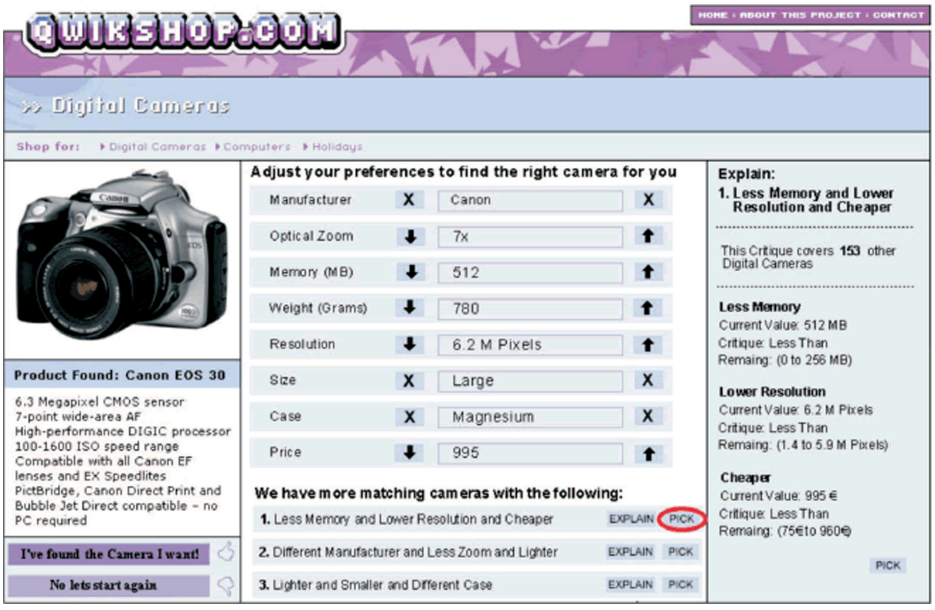

Fig. 4. Qwikshop example-critiquing prototype for digital cameras [McCarthy et al. 2005].

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

Fig. 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.

that they did not have access to the volume of user-generated data that is available today.

2.4. Tag-Based Recommenders

Recent work has explored recommender algorithms that leverage tagging data to recommend items. For example, Tso-Sutter et al. [2008] extend traditional collaborative filtering algorithms by incorporating tagging data into the user-item matrix. Guan et al. [2010] apply a graph-based learning algorithm to recommend items based purely on tagging data. Sen et al. [2009] construct a Bayesian model that estimates a user's preference for items based on their inferred preferences for tags.

Researchers have also developed interfaces that combine tagging and recommendation. 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). 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 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.

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, we apply these guidelines to compute the tag genome in the movie domain.

3.1. Definition

Just as an organism is defined by a sequence of genes, an item in an information space may be defined by its relationship to a set of tags. 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

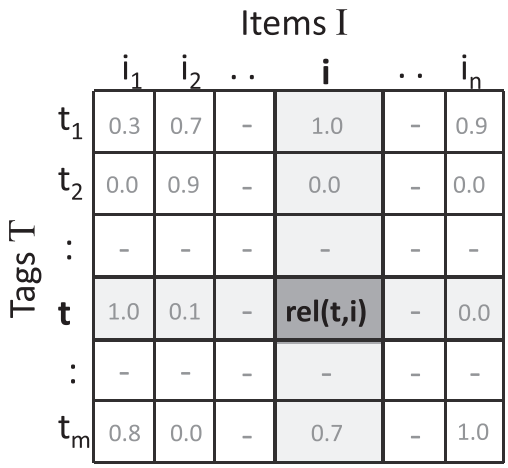

Fig. 6. The tag genome. Each entry in the matrix records the relevance of a tag to an item on a 0–1 scale.

movie domain[6], for example, *rel*(*violent*, *Reservoir Dogs*) = 0.98, *rel*(*violent*, *The Usual Suspects*) = 0.65, and *rel*(*violent*, *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 6. We define G such that

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

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:

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

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 ("the tag genome of *Reservoir Dogs*"), we are referring to G_i; otherwise we are referring to the full matrix G.

3.2. Assumptions

In this section, we discuss the assumptions made in the preceding definition of the tag genome, and we propose some alternative formulations.

Community Consensus Model of Tag Relevance. According to our definition of the tag genome, each tag t has one true relevance value $rel(t, i)$ with respect to each item i. However, tags are often subjective [Sen et al. 2006], so one user's assessment of tag relevance may differ from another's. For example, one user might find *Pulp Fiction* uproariously funny, while another considers it devoid of any humor. An alternative, personalized formulation of the tag genome could take these individual differences into account. Instead of representing tag relevance as $rel(t, i)$, the system could model it as $rel(t, i, u)$, representing the relevance of tag t to item i from the perspective of user u. However, computing $rel(t, i, u)$ requires more sophisticated models than estimating $rel(t, i)$, and it also requires knowledge about the particular user. We leave the personalized extension of the tag genome to future work.

[6]We describe the method for computing these values in Section 4.

Uniform Relevance Scale. For all tags in the genome, the relationship between item and tag is expressed on a continuous scale from 0 to 1. In reality, the nature of this relationship can vary depending on the tag. While some tags represent continuous attributes that may be present to varying degrees, other tags describe binary qualities that either apply to the item or do not. For example, it makes more sense to describe a movie as somewhat *violent* (continuous) as opposed to somewhat *based on a play* (binary). The 0–1 relevance scale is general enough to handle both types of tags; binary attributes may simply be represented as 1 (relevant) or 0 (not relevant). Nonetheless, there may be advantages to explicitly modeling the type of tag. First, the learning model could leverage this information to more accurately predict tag relevance, since only relevance values of 0 or 1 are possible for binary tags. Second, the system could visualize tag relevance in different ways depending on the type of tag. Rather than displaying a bar of varying length (see Figure 1) to convey the relevance of binary tags, the interface could simply indicate "yes" or "no", for example. We leave this extension of the tag genome to future work.

3.3. 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 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. This is similar to the approach used by many example-critiquing systems (see Section 2.3), which use domain experts to populate the underlying knowledge base.

— *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 that you have to compensate the experts, which may be expensive 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 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 7, 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. Other approaches are possible, for example, one could perform matrix factorization on the tag genome. However, these two approaches are not mutually exclusive, since the output from the matrix factorization can be used as an input in the regression. Regression offers a generic framework that can combine features from multiple data sources, as well as the

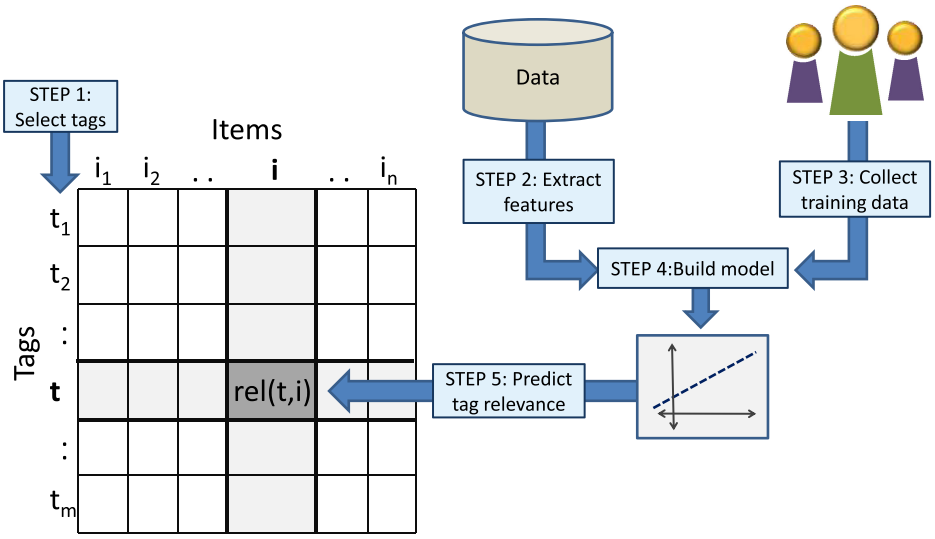

Fig. 7. Computing the tag genome.

output from various models. In Section 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 the following.

— *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 4.2.3, 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 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 before, 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 4.3. Once the tag genome has been computed, users could provide additional training data directly in the application, for example, in Movie Tuner, one could allow users to adjust relevance values by clicking on the relevance meters (see Figure 1) and dragging them to a different position. 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, 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. CASE STUDY: COMPUTING THE TAG GENOME IN THE MOVIE DOMAIN

In this section, we describe how we compute the tag genome for MovieLens[7], a movie recommender Web site that also supports tagging of movies. 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).

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 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

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

Table I. Data Sources Used to Predict Tag Relevance

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 ½ to 5 stars	1,965 ratings per movie

Table II. Features Used to Predict Tag Relevance

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.

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 by 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 in the following text. 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.

4.2. STEP 2: Extract Features

We extracted features from three types of user-contributed content: tag applications, user reviews, and ratings, as summarized in Table I. The tag applications and ratings came from MovieLens itself, while the user reviews were crawled from the Internet Movie Database Web site[8]. From each of these data sources we extracted several features, summarized in Table II. 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.

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

4.2.1. 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 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:

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

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[9].

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 $\mathbf{d}'_t$:

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

4.2.2. 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[10] Web site.

[9]We evaluated features using relevance ratings collected in a pilot survey (see Section 4.3.3). These ratings were not used in the final model evaluation.
[10]http://www.imdb.com

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].
— *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 in the following text.
— *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 online Appendix A (available in the ACM Digital Library).
— *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[11].

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). 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:

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

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 $\mathbf{d}'_i$ and $\mathbf{d}'_t$, respectively.

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

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

4.2.3. Features Based on Ratings. Because MovieLens users have contributed over 18-million movie ratings, we wanted to see if we could use this data to help predict tag relevance. But it is not immediately 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? We now describe two features that extract signals of tag relevance by analyzing ratings in aggregate.

[11]We also normalize this feature across all tags as described in Section 4.2.5.

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[12]. 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[13]. We define user u's "rating" of tag t as the smoothed average of u's ratings of items tagged with t:

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

where I_u = items rated by u, and I_t = items tagged with t.

We smooth the computed value 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 these users:

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

4.2.4. 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 preexisting 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 4.3).

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

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

[13]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.

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 4.4.2 because it resulted in the feature that peformed best in predicting the relevance ratings from the pilot survey (see Section 4.3.3). We define regress-tag(t, i) as the relevance prediction for tag t and item i.

4.2.5. 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, tag-count, and tag-share to bring them closer to a normal distribution. This transformation also resulted in stronger correlations with the relevance ratings in the pilot survey (see Section 4.3.3).

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 8, 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[14] and a much larger set of users.

4.3.1. Survey Design. On each page of the survey (see Figure 9), subjects rated the relevance of a particular tag to 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[15]. 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.

4.3.2. 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 online Appendix B (available in the ACM Digital Library). The main features of this algorithm are as follows.

— *Importance-based sampling*. We wanted 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[16]. We measure the importance of tag t by popularity(t), which we define as the number of distinct MovieLens users who have

[14] excluding actor/director tags as discussed in Section 4.1.
[15] http://developer.netflix.com.
[16] For the tag-specific and multilevel models described in Section 4.4.

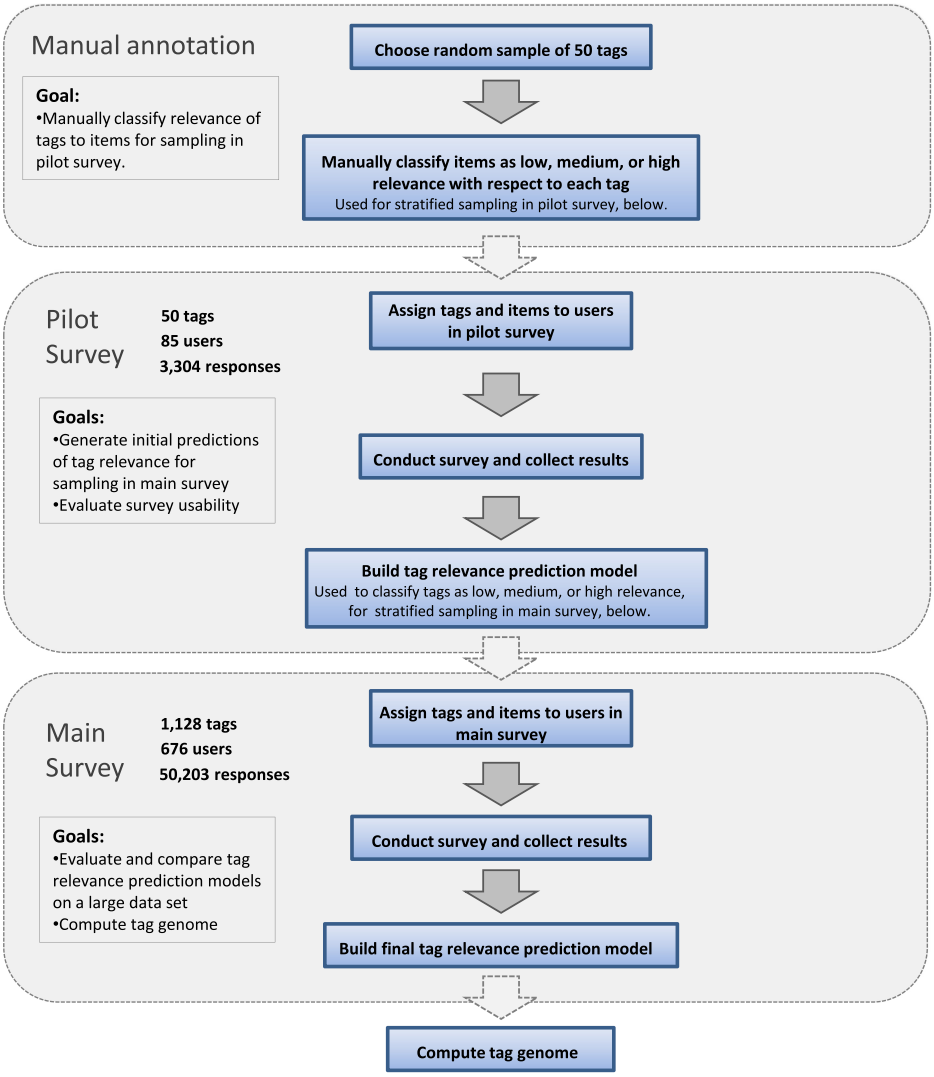

Fig. 8. Overview of process for collecting training data.

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)). 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 on tag relevance. For each we 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

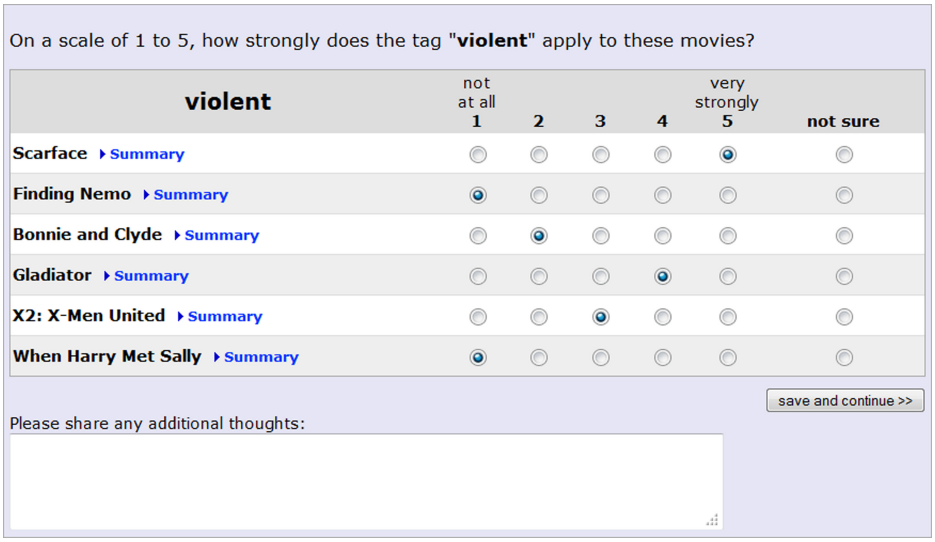

Fig. 9. Page from survey.

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 in the following text.

4.3.3. 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).

We used the same sampling algorithm for the pilot survey that we later use for the main survey (see Appendix B in the online appendix available in the ACM Digital Library). 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 earlier, 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 4.4.2 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 for any tag.

4.3.4. 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 4.1), and we drew from the 7,716 movies with at

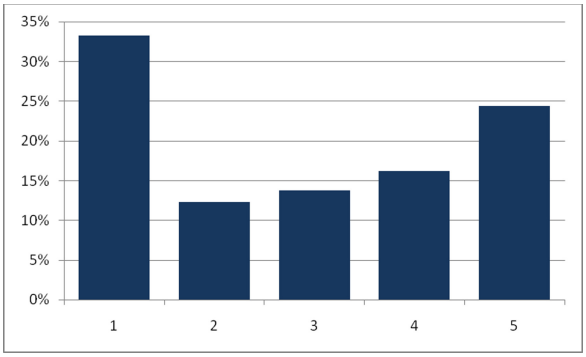

Fig. 10. Distribution of relevance ratings in main phase of survey (1=Does not apply at all, 5= Applies very strongly).

least 100 ratings. We assigned tags and movies to each user based on the sampling algorithm described in Section 4.3.2. 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 ⅓ or less as low relevance, those with a predicted relevance of ½ to ⅔ as medium relevance, and those with a predicted relevance of ⅔ 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 10 shows the overall distribution of relevance ratings. On average, we collected 45 ratings per tag.

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.

(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 11.

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,

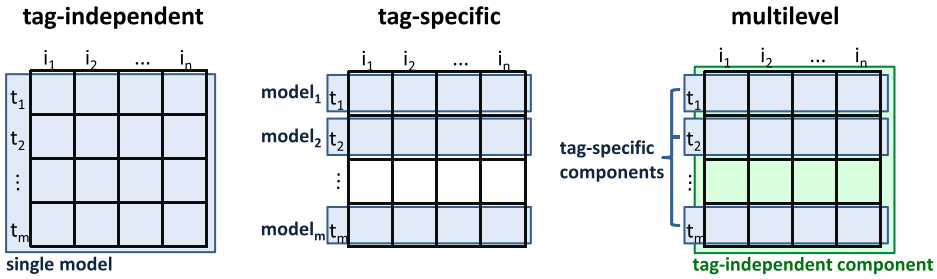

Fig. 11. Three different approaches with respect to model granularity. The tag-independent approach assumes a unified model across all tags, while the tag-specific approach learns a separate model for each tag. The multilevel approach combines tag-independent and tag-specific approaches.

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 in the following text, 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 just outlined (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 $\mathbf{x}_{t,i}$.

4.4.1. 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 β_t is modeled as a random variable sampled from a multivariate normal prior distribution $N(\mu, \Sigma)$:

$$rel(t, i) = \mathbf{x}_{t,i}^{\mathsf{T}} \beta_t$$
$$\beta_t \sim N(\mu, \Sigma).$$

The hyperparameters μ and Σ are estimated from the data along with each β_t. μ is the "average" β_t across all tags and represents the tag-independent component of this model. The estimate for each β_t depends on both this prior distribution $N(\mu, \Sigma)$ as well as the training data specific to tag t. When little training data is available for t, β_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 β_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.

4.4.2. 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 [Gelman and Hill 2007; McCullagh and Nelder 1989]. 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:

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

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:

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

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-idependent nonlinear model assumes a single vector of coefficients β, as in the linear case:

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

Tag-specific. The tag-specific nonlinear model uses a separate coefficient vector β_t for each tag:

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

As in the linear case, we solve a separate regression equation for each β_t using only the training data for t.

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

$$rel(t, i) = \text{logit}^{-1}(\mathbf{x}_{t,i}^{\mathsf{T}}\beta_t),$$
$$\beta_t \sim \text{N}(\mu, \Sigma).$$

4.4.3. Model Fitting. We fit the regression models using the following functions from the R programming language: lm (linear tag-independent and linear tag-specific), lmer (linear multilevel), glm (nonlinear tag-independent and nonlinear tag-specific), and glmer (nonlinear multilevel) [R Development Core Team 2010; Bates and Sarkar 2007]. For the lmer and glmer functions, we used the tag as the grouping factor. 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 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.

4.4.4. 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[17].

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 next. We only used the training partition from each round for feature selection, reserving the test partition for model evaluation.

4.4.5. Model Evaluation. We evaluated the 6 regression models using tenfold cross-validation on the ratings collected in the main phase of the survey. As shown in Figure 12, 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

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

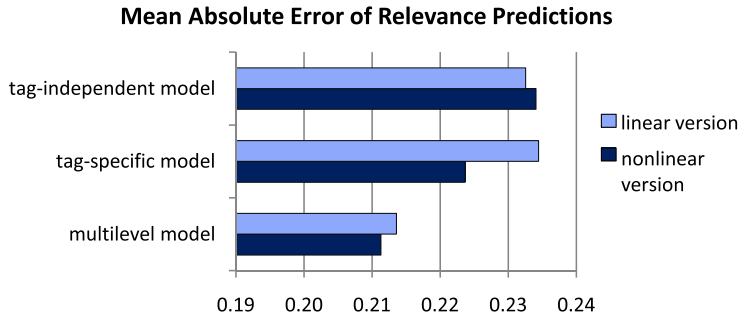

Fig. 12. Performance of each model using tenfold cross-validation on the relevance ratings from the main survey. All models performed significantly better than a baseline model that used the mean rating (MAE = 0.36). Differences in MAE are statistically significant ($p < 0.05$), with the exception of the difference between the linear tag-specific model and the nonlinear tag-independent model.

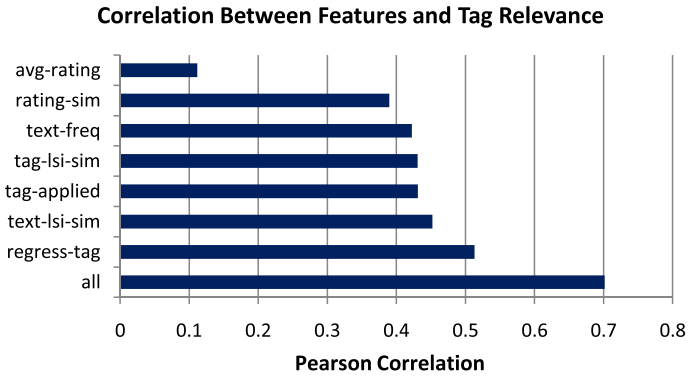

Fig. 13. Performance of each individual feature based on correlation with tag relevance ratings from the main survey. The *all* feature represents the prediction from the best performing regression model, which uses all the other features as inputs. All differences are statistically significant ($p < 0.001$) except for the differences between text-freq, tag-lsi-sim, and tag-applied.

($p < 0.001$). 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 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 13 shows the strength of individual features based on their correlation with the survey ratings[18]. The metafeature regress-tag had the highest correlation with the ratings, to be expected since this feature combines several of the other features.

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

Table III. Pairwise Correlations Between Features

	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	-

text-lsi-sim was the next strongest feature, which is not surprising given the high volume of textual data (see Table I). 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 I). 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, 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 III 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.

4.5. 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 4.4.5). We train the final model using the full set of relevance ratings from the survey.

5. MOVIE TUNER

In this section, we discuss the design of the Movie Tuner application introduced in Section 1.1. Movie Tuner is inspired by a class of applications called *example-critiquing systems*, which enable users to navigate an information space by critiquing specific examples (see Section 2.3). For example, in the Entree system, users critique restaurants 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.

We focus on two primary interactions in Movie Tuner: *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...").

5.1. 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". We now outline the design space for how users may apply critiques, and we discuss our design decisions.

5.1.1. 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 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.

5.1.2. 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 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.

5.1.3. 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 to keep it in effect as they choose additional critiques (see Figure 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.

5.1.4. 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 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?").

5.1.5. 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. We now 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 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 5.2.3) 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 5.2, 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,

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

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, 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 that follows 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 critique-value(t, i) = $rel(t, i)$ (favors descriptive tags), and one where critique-value(t, i) = critique-entropy

Table IV. Comparison of Tag-Selection Algorithms

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)

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

(t, i) (favors discriminating tags). In the latter case, we added the following constraint[19]:

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

Table IV 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.

5.2. 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.

5.2.1. 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 $rel(action, Independence\ Day) - rel(action, Star\ Trek:\ Generations) = 0.97 - 0.46 = 0.51$. 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

[19]0.325 is the Shannon entropy of the distribution {0.9, 0.1, 0.0}.

critique based on critique distance. In the following we define two alternative critique satisfaction metrics: linear-sat and diminish-sat. In Section 5.2.3, 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.

5.2.2. 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 3.1). 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

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

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 ½. 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 that follow, 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.

5.2.3. Algorithm for Responding to Critiques. We now describe an algorithm that uses the preceding 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 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, 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 Section 5.2.2. 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 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 $d \in \{-1, +1\}$,

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

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 trade-off between critique distance and overall similarity. When critique-sat

Table V. Comparison of Critique Satisfaction Algorithms

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)

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.

= linear-sat, the trade-off between critique distance and overall similarity is the same at any critique distance. When critique-sat = diminish-sat, the trade-off favors increased similarity over increased critique distance as critique distance increases. Table V shows sample results for both versions of the algorithm.

Compound and null critiques. The previous 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.

5.3. Design of Field Study

We conducted a field study of Movie Tuner on the MovieLens Web site, 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 can 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[20], 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 (2 x 2). 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 5.1.5. The

[20]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.

Table VI. Survey Results

Statement (abbreviated)	μ	**% agree**	**% disagree**
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

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.)

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 5.2.1.

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 a series of statements, summarized in Table VI, using a 5-point Likert scale[21]. 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 could 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.

5.4. 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,

[21] 1 = strongly disagree, 2 = disagree, 3 = neutral, 4 = agree, 5 = strongly agree

Table VII. Most Popular System-Suggested Tags

Top 10 positive	frac	Top 10 negative	frac
nudity (full frontal)	0.19	coen brothers	0.08
nudity (full frontal)	0.15	religion	0.07
sexuality	0.11	holocaust	0.07
scary	0.09	world war ii	0.06
nudity (topless)	0.09	christmas	0.06
lesbian	0.08	western	0.06
black comedy	0.08	pixar	0.05
psychological	0.07	suicide	0.05
dark comedy	0.07	vampires	0.05
cyberpunk	0.07	police	0.04

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.

and whether they use compound or unit critiques. We then explore how users interact with items displayed in response to their critiques.

5.4.1. 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 5.1.5, 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 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 just described, 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 might 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 VIII. Most Popular User-Entered Tags

Top 10 positive	count	Top 10 negative	count
nudity	36	comedy	21
comedy	32	violence	9
mystery	30	violent	8
nudity (topless)	29	drugs	5
romance	29	horror	5
sex	28	sex	5
action	26	cheesy	4
surreal	21	dark	4
funny	18	nudity	4
erotic	16	predictable	4

User-entered tags most frequently used in each critique direction.)

Table VII shows the system-suggested tags users were most likely to choose in each critique direction[22]. 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*.

Users entered their own tag rather than choose a system-selected tag for 9% of critiques. Table VIII 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?

[22]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.

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'."

5.4.2. 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 ritiqued[23]. 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 5.2.1) 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. This suggests that Movie Tuner can support a range of approaches for responding to users' critiques. System designers might wish to incorporate other objectives when choosing items in response to critiques, 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 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.

6. APPLICATIONS OF THE TAG GENOME

Movie Tuner is just one example of the type of application that the tag genome can support. In this section, we outline the broader space of applications of the tag genome.

6.1. Applications Motivated by Vector Arithmetic

Many applications of the tag genome can be framed in terms of vector arithmetic, as illustrated in Figure 14. Figure 14(a) 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 14(b). 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 trade-offs between items when making decisions [Jedetski et al.

[23]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.

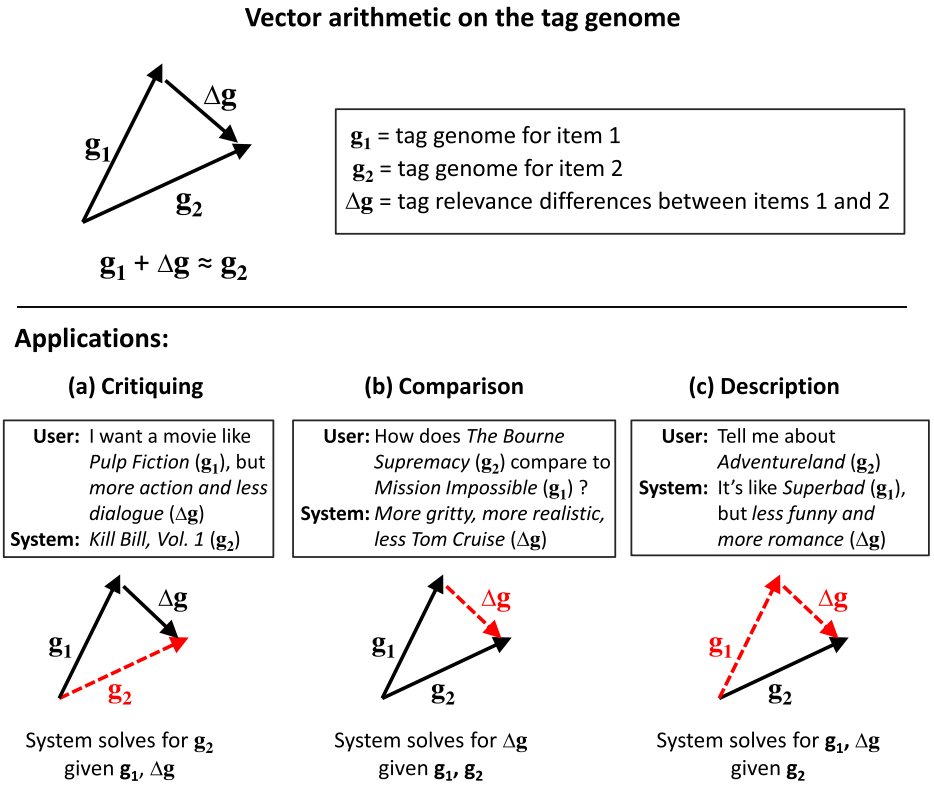

Fig. 14. Applications of the tag genome inspired by vector arithmetic.

2002; Pu and Chen 2005]. The tag genome can also reveal similarities between items [Green et al. 2009], for example, that *The Bourne Supremacy* and *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 2007; Vig et al. 2009].

As shown in Figure 14(c), 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. 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, for instance, "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 (i.e., tag) the system could provide slider controls for users to specify a range of relevance values, similar to the approach described by 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 [Bateman et al. 2008; Halvey and Keane 2007; 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 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 1.2. 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 5.1.5.

6.3. The Tag Genome as Feature Vector

We have discussed how the tag genome enhances user interaction. But it can also support backend 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 [Mooney and Roy 2000; Možina et al. 2004; Poulin et al. 2006; 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 5.2.2) as well as label the resulting clusters [Manning et al. 2008] based on the tags with high relevance across each cluster.

7. CONCLUSION

This article introduces the tag genome, a 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 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.

In addition to defining and computing the tag genome, we showed how the tag genome can enrich user interaction in tagging systems. We developed Movie Tuner, a system for navigating an information space using natural language critiques based on the tag genome. 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, rather than by relying on paid experts.

We approached Movie Tuner 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-grained control over their critiques in Movie Tuner. 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.

We believe that the tag genome can support many exciting applications across multiple domains. Tuners represent just one class of applications. In Section 6, we proposed several other types of applications, including search, faceted browsing, and item comparison. We hope that system designers will explore these and other applications in the coming years.

ACKNOWLEDGMENTS

The authors thank Loren Terveen, Daniel Kluver, Rich Davies, Tony Lam, and the rest of GroupLens for their feedback and assistance with this article. We thank the members of MovieLens for their participation in the survey, as well as their feedback and suggestions for Movie Tuner.

REFERENCES

BATEMAN, S., GUTWIN, C., AND NACENTA, M. 2008. Seeing things in the clouds: The effect of visual features on tag cloud selections. In *Proceedings of the 19th ACM Conference on Hypertext and Hypermedia (Hypertext'08)*. ACM, New York, NY, 193–202.

BATES, D. AND SARKAR, D. 2007. Linear mixed-effects models using S4 classes. http://CRAN.R-project.org.

BIRD, S., LOPER, E., AND KLEIN, E. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.

BURKE, R. 2002. Hybrid recommender systems: Survey and experiments. *User Model. User-Adapted Interac. 12,* 4, 331–370.

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 (AAAI'96)*. AAAI Press, 462–468.

BURKE, R. D., HAMMOND, K. J., AND YOUNG, B. C. 1997. The FindMe approach to assisted browsing. *IEEE Expert 12*, 32–40.

CHEN, L. AND PU, P. 2006. Evaluating critiquing-based recommender agents. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI'06)*. Vol. 1, 157–162.

DEERWESTER, S. C., DUMAIS, S. T., LANDAUER, T. K., FURNAS, G. W., AND HARSHMAN, R. A. 1990. Indexing by latent semantic analysis. *J. Amer. Soc. Info. Sci. 41,* 6, 391–407.

DRUCKER, H., BURGES, C. J. C., KAUFMAN, L., SMOLA, A., AND VAPNIK, V. 1997. Support vector regression machines. In *Advances in Neural Information Processing Systems*, Vol. 9, 155–161.

ECK, D., LAMERE, P., BERTIN-MAHIEUX, T., AND GREEN, S. 2007. Automatic generation of social tags for music recommendation. In *Advances in Neural Information Processing Systems*, Vol. 20.

FALTINGS, B., PU, P., TORRENS, M., AND VIAPPIANI, P. 2004. Designing example-critiquing interaction. In *Proceedings of the 9th International Conference on Intelligent User Interfaces (IUI'04)*. ACM, New York, NY, 22–29.

FEINSTEIN, D. AND SMADJA, F. 2006. Hierarchical tags and faceted search: The RawSugar approach. In *SIGIR Conference on Research and Development in Information Retrieval Workshop on Faceted Search*. 23–25.

FLETCHER, R. 1981. *Practical Methods of Optimization: Vol. 2: Constrained Optimization*. John Wiley and Sons.

GELMAN, A. AND HILL, J. 2007. *Data Analysis Using Regression and Multilevel Hierarchical Models*. Cambridge University Press, Cambridge, UK.

GELMAN, A., CARLIN, J. B., STERN, H. S., AND RUBIN, D. B. 2003. *Bayesian Data Analysis, 2nd Ed.*. Chapman & Hall/CRC.

GOLDER, S. A. AND HUBERMAN, B. A. 2006. Usage patterns of collaborative tagging systems. *J. Info. Sci. 32*, 198–208.

GREEN, S. J., LAMERE, P., ALEXANDER, J., MAILLET, F., KIRK, S., HOLT, J., BOURQUE, J., AND MAK, X. 2009. Generating transparent, steerable recommendations from textual descriptions of items. In *Proceedings of the ACM Conference on Recommender Systems (RecSys'09)*. ACM, New York, NY, 281–284.

GUAN, Z., WANG, C., BU, J., CHEN, C., YANG, K., CAI, D., AND HE, X. 2010. Document recommendation in social tagging services. In *Proceedings of the 19th International Conference on World Wide Web (WWW'10)*. ACM, New York, NY, 391–400.

GUYON, I. AND ELISSEEFF, A. 2003. An introduction to variable and feature selection. *J. Machine Learn. Res. 3*, 1157–1182.

HALVEY, M. J. AND KEANE, M. T. 2007. An assessment of tag presentation techniques. In *Proceedings of the 16th International Conference on the World Wide Web (WWW'07)*. ACM, New York, NY, 1313–1314.

HEYMANN, P., RAMAGE, D., AND GARCIA-MOLINA, H. 2008. Social tag prediction. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'08)*. ACM, New York, NY, 531–538.

HINGSTON, M. AND KAY, J. 2006. User friendly recommender systems (honors thesis).

JEDETSKI, J., ADELMAN, L., AND YEO, C. 2002. How web site decision technology affects consumers. *IEEE Internet Comput. 6*, 72–79.

JÄSCHKE, R., MARINHO, L. B., HOTHO, A., SCHMIDT-THIEME, L., AND STUMME, G. 2007. Tag recommendations in folksonomies. In *Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'07)*. Vol. 4702, Springer, 506–514.

KAMMERER, Y., NAIRN, R., PIROLLI, P., AND CHI, H. 2009. Signpost from the masses: Learning effects in an exploratory social tag search browser. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'09)*. 625–634.

KELLEY, C. T. 1999. Iterative methods for optimization. In *SIAM Frontiers in Applied Mathematics,* 18.

KOHAVI, R. AND JOHN, G. H. 1997. Wrappers for feature subset selection. *Art. Intell. 97,* 1–2, 273–324.

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 *Proceedings of the 14th International Conference on Intelligent User Interfaces (IUI'09)*. ACM, New York, NY, 187–196.

LINDEN, G., HANKS, S., AND LESH, N. 1997. Interactive assessment of user preference models: The automated travel assistant. In *Proceedings of User Modeling'97*. Springer, 67–78.

MANNING, C. D., RAGHAVAN, P., AND SCHÜTZE, H. 2008. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK.

MARLOW, C., NAAMAN, M., BOYD, D., AND DAVIS, M. 2006. HT06, tagging paper, taxonomy, flickr, academic article, to read. In *Proceedings of the 17th Conference on Hypertext and Hypermedia (Hypertext'06)*. ACM, New York, NY, 31–40.

MCCALLUM, A. AND NIGAM, K. 1998. A comparison of event models for naive bayes text classification. In *Proceedings of the Conference on Artificial Intelligence Workshop on Learning for Text Categorization*. AAAI Press, 41–48.

MCCARTHY, K., REILLY, J., MCGINTY, L., AND SMYTH, B. 2005. Experiments in dynamic critiquing. In *Proceedings of the 10th International Conference on Intelligent User Interfaces (IUI'05)*. ACM, New York, NY, 175–182.

MCCULLAGH, P. AND NELDER, J. 1989. *Generalized Linear Models* 2nd Ed. Chapman & Hall/CRC.

MOONEY, R. J. AND ROY, L. 2000. Content-based book recommending using learning for text categorization. In *Proceedings of the 5th ACM Conference on Digital Libraries*. ACM, New York, NY, 195–204.

MOŽINA, M., DEMŠAR, J., KATTAN, M., AND ZUPAN, B. 2004. Nomograms for visualization of naive bayesian classifier. In *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'04)*. Springer-Verlag Berlin, 337–348.

PAYNE, J. W., BETTMAN, J., AND JOHNSON, E. J. 1993. *The Adaptive Decision Maker*. Cambridge University Press. Cambridge, UK.

PORTER, M. 1980. An algorithm for suffix stripping. *Program: Elect. Library Info. Syst. 14,* 3, 130–137.

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 (AAAI'06)*. Vol. 2, AAAI Press, 1822–1829.

PU, P. AND CHEN, L. 2005. Integrating tradeoff support in product search tools for e-commerce sites. In *Proceedings of the 6th ACM Conference on Electronic Commerce (EC'05)*. ACM, New York, NY, 269–278.

R DEVELOPMENT CORE TEAM. 2010. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

RAUDENBUSH, S. AND BRYK, A. 2002. *Hierarchical Linear Models* 2nd Ed. Sage Publications, Thousand Oaks, CA.

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 *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'07)*. ACM, New York, NY, 995–998.

SALTON, G. AND MCGILL, M. 1983. *Introduction to Modern Information Retrival*. McGraw-Hill.

SALTON, G., WONG, A., AND YANG, C. S. 1975. A vector space model for automatic indexing. *Comm. ACM 18*, 613–620.

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 *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW'06)*. ACM, New York, NY, 181–190.

SEN, S., HARPER, F. M., LAPITZ, A., AND RIEDL, J. 2007. The quest for quality tags. In *Proceedings of the International ACM Conference on Supporting Group Work (GROUP'07)*. ACM, New York, NY, 361–370.

SEN, S., VIG, J., AND RIEDL, J. 2009. Tagommenders: Connecting users to items through tags. In *Proceedings of the 18th International Converence on Word Wide Web (WWW'09)*. ACM, New York, NY, 671–680.

SHNEIDERMAN, B. 1994. Dynamic queries for visual information seeking. *IEEE Softw. 11*, 70–77.

SIGURBJÖRNSSON, B. AND VAN ZWOL, R. 2008. Flickr tag recommendation based on collective knowledge. In *Proceeding of the 17th International Conference on the World Wide Web (WWW'08)*. ACM, New York, NY, 327–336.

SMITH, D. C., IRBY, C., KIMBALL, R., BERPLANK, B., AND HARSLEM, E. 1990. Designing the Star user interface. In *Human-Computer Interaction*, 237–259.

SMYTH, B., MCGINTY, L., REILLY, J., AND MCCARTHY, K. 2004. Compound critiques for conversational recommender systems. In *Proceedings of (Web Intelligence'04)*. IEEE Computer Society, Los Alamitos, CA, 145–151.

SPARLING, E. I. AND SEN, S. 2011. Rating: How difficult is it? In *Proceedings of the 2011 ACM Conference on Recommender Systems (RecSys'11)*. ACM, New York, NY.

SYMEONIDIS, P., NANOPOULOS, A., AND MANOLOPOULOS, Y. 2008. Tag recommendations based on tensor dimensionality reduction. In *Proceedings of the ACM Conference on Recommender Systems (RecSys'08)*. ACM, New York, NY, 43–50.

TEEVAN, J., DUMAIS, S., AND GUTT, Z. 2008. Challenges for supporting faceted search in large, heterogeneous corpora like the web. In *Proceedings of the Second Workshop on Human-Computer Interaction and Information Retrieval (HCIR'08)*.

TINTAREV, N. AND MASTHOFF, J. 2007. Effective explanations of recommendations: User-centered design. In *Proceedings of the ACM Conference on Recommender Systems (RecSys'07)*. ACM, New York, NY, 153–156.

TSO-SUTTER, K. H. L., MARINHO, L. B., AND SCHMIDT-THIEME, L. 2008. Tag-aware recommender systems by fusion of collaborative filtering algorithms. In *Proceedings of the ACM Symposium on Applied Computing (SAC'08)*. ACM, New York, NY, 1995–1999.

ULGES, A., SCHULZE, C., KEYSERS, D., AND BREUEL, T. M. 2008. A system that learns to tag videos by watching YouTube. In *Proceedings of the 6th International Conference on Computer Vision Systems (ICVS'08)*. Springer-Verlag, Berlin, 415–424.

VIG, J., SEN, S., AND RIEDL, J. 2009. Tagsplanations: Explaining recommendations using tags. In *Proceedings of the 13th International Conference on Intelligent User Interfaces (IUI'09)*. ACM, New York, NY, 47–56.

VIG, J., SEN, S., AND RIEDL, J. 2011. Navigating the tag genome. In *Proceedings of the 15th International Conference on Intelligent User Interfaces (IUI'11)*. ACM, New York, NY.

WU, L., YANG, L., YU, N., AND HUA, X.-S. 2009. Learning to tag. In *Proceedings of the 18th International Conference on the World Wide Web (WWW'09)*. ACM, New York, NY, 361–370.

YEE, K.-P., SWEARINGEN, K., LI, K., AND HEARST, M. 2003. Faceted metadata for image search and browsing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'03)*. ACM, New York, NY, 401–408.

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

ZHANG, J. AND PU, P. 2006. A comparative study of compound critique generation in conversational recommender systems. In *Proceedings of Adaptive Hypermedia'06*. Springer, 234–243.

Received October 2011; revised March 2012; accepted April 2012