Hidden Factors and Hidden Topics:
Understanding Rating Dimensions with Review Text

Julian McAuley
Stanford University
jmcauley@cs.stanford.edu

Jure Leskovec
Stanford University
jure@cs.stanford.edu

ABSTRACT

In order to recommend products to users we must ultimately predict how a user will respond to a new product. To do so we must uncover the implicit tastes of each user as well as the properties of each product. For example, in order to predict whether a user will enjoy *Harry Potter*, it helps to identify that the book is about wizards, as well as the user's level of interest in wizardry. *User feedback* is required to discover these latent product and user dimensions. Such feedback often comes in the form of a numeric rating accompanied by review text. However, traditional methods often discard review text, which makes user and product latent dimensions difficult to interpret, since they ignore the very text that *justifies* a user's rating. In this paper, we aim to combine latent *rating* dimensions (such as those of latent-factor recommender systems) with latent review *topics* (such as those learned by topic models like LDA). Our approach has several advantages. Firstly, we obtain highly interpretable textual labels for latent rating dimensions, which helps us to 'justify' ratings with text. Secondly, our approach more accurately predicts product ratings by harnessing the information present in review text; this is especially true for new products and users, who may have too few ratings to model their latent factors, yet may still provide substantial information from the text of even a single review. Thirdly, our discovered topics can be used to facilitate other tasks such as automated genre discovery, and to identify useful and representative reviews.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information Search and Retrieval

Keywords

recommender systems, topic models

1. INTRODUCTION

Recommender systems have transformed the way users discover and evaluate products on the web. Whenever users assess products, there is a need to model *how* they made their assessment, either

to suggest new products they might enjoy [19], to summarize the important points in their reviews [13], or to identify other users who may share similar opinions [24]. Such tasks have been studied across a variety of domains, from hotel reviews [6] to beer [21].

To model how users evaluate products, we must understand the hidden dimensions, or facets, of their opinions. For example, to understand why two users agree when reviewing the movie *Seven Samurai*, yet disagree when reviewing *Casablanca*, it helps to know that these films belong to different genres; these users may have similar preferences toward action movies, but opposite preferences for romantic dramas.

Modeling these hidden factors is key to obtaining state-of-the-art performance on product recommendation tasks [2]. Crucially, recommender systems rely on *human feedback*, which typically comes in the form of a plain-text review and a numeric score (such as a star rating). This type of feedback is used to train machine learning algorithms, whose goal is to predict the scores that users will give to items that they have not yet reviewed [12].

In spite of the wealth of research on modeling *ratings*, the other form of feedback present on review websites—namely, the reviews themselves—is typically ignored. In our opinion, ignoring this rich source of information is a major shortcoming of existing work on recommender systems. Indeed, if our goal is to *understand* (rather than merely predict) how users rate products, we ought to rely on reviews, whose very purpose is for users to explain *why* they rated a product the way they did. As we show later, 'understanding' these factors helps us to justify users' reviews, and can aid us in tasks like genre discovery and identification of informative reviews.

Thus our goal in this paper is to develop statistical models that combine latent dimensions in rating data with topics in review text. This leads to natural interpretations of rating dimensions, for instance we automatically discover that movie ratings are divided along topics like genre, while beer ratings are divided along topics like beer style. Rather than performing *post-hoc* analysis to make this determination, we discover both rating dimensions and review topics in a single learning stage, using an objective that combines the accuracy of rating prediction (in terms of the mean squared error) with the likelihood of the review corpus (using a topic model). We do this using a transform that aligns latent rating and review terms, so that both are determined by a single parameter. Essentially, the text likelihood acts as a 'regulariser' for rating prediction, ensuring that parameters that are good at predicting ratings are also likely in terms of the review text.

Not only does our model help us to 'explain' users' review scores by better understanding rating dimensions, but it also leads to better predictions of the ratings themselves. Traditional models (based on ratings alone) are difficult to apply to new users and products, that have too few ratings to model their many latent dimensions or

factors [14]. In contrast, our model allows us to accurately uncover such factors from even a single review.

Furthermore, by understanding how hidden *rating* dimensions relate to hidden *review* dimensions, our models facilitate a variety of novel tasks. For instance, we apply our models to automatically discover product categories, or 'genres', that explain the variation present in both ratings and reviews. We also apply our models to find 'informative' reviews, whose text is good at explaining the factors that contributed to a user's rating.

1.1 Contributions and Findings

Our main contribution is to develop statistical models that combine latent dimensions in rating data with topics in review text. In practice, we claim the following benefits over existing approaches.

Firstly, the topics we obtain readily explain the variation present in ratings and reviews. For example, we discover that beer style (e.g. light versus dark) explains the variation present in beer rating and review data; platform (e.g. console versus PC) explains the variation present in video game data; country (e.g. Italian versus Mexican) explains the variation present in restaurant data, etc.

Secondly, combining ratings with review text allows us to predict ratings more accurately than approaches that consider either of the two sources of data in isolation. This is especially true for 'new' products: product factors cannot be fit from only a few *ratings*, though we can accurately model them from only a few *reviews*. We improve upon state-of-the-art models based on matrix factorization and LDA by 5-10% in terms of the mean squared error.

Thirdly, we can use our models to facilitate novel tasks. We apply our models to the task of automatic genre discovery, where they are an order of magnitude more accurate than methods based on ratings or reviews alone. We also apply our models to the task of identifying informative reviews; we discover that those reviews whose text best 'explains' the hidden factors of a product are rated as being 'useful' by human annotators on sites like *yelp.com*.

Last, we apply our models to novel corpora consisting of over forty million reviews, from nine million users and three million products. To our knowledge, this is the largest-scale study of public review data conducted to date. Our methods readily scale to datasets of this size.

1.2 Further Related Work

Although many works have studied ratings and review text in isolation, few works attempt to combine the two sources of information. The most similar line of work is perhaps that of *aspect discovery* [6, 25, 29]. Aspects generally refer to *features* that are relevant to all products. Individual users may assign such aspects different weights when determining their overall score; for example a spendthrift hotel reviewer might assign a low weight to 'price' but a high weight to 'service', thus explaining why their overall rating differs from a miserly reviewer who is only interested in price [6].

Though our work shares similarities with such studies, the topics we discover are not similar to aspects. 'Aspects' explain dimensions along which ratings and reviews vary, which is also our goal. Yet while aspects are dimensions common to every individual review, they may not explain the variation present across entire review corpora. For instance, although one may learn that individual users assign different weights to 'smell' and 'taste' when reviewing beers [21], this fails to explain why one user may love the smell of a beer, while another believes the *same* beer smells terrible.

An early work that combines review text and ratings is [6]. They observe that reviews often discuss multiple aspects, and that a user's rating will depend on the importance that they ascribe to each aspect. They find that these dimensions can be recovered from review

Symbol	Description
$r_{u,i}$	rating of item i by user u
$d_{u,i}$	review ('document') of item i by user u
$rec(u,i)$	prediction of the rating for item i by user u
α	global offset term
β_u	bias parameter for user u
β_i	bias parameter for item i
γ_u	K-dimensional latent features for user u
γ_i	K-dimensional latent features for item i
K	number of latent dimensions/topics
θ_i	K-dimensional topic distribution for item i
ϕ_k	word distribution for topic k
ψ_k	unnormalized word distribution for topic k
$w_{u,i,j}$	j^{th} word of user u's review of item i
$z_{u,i,j}$	topic for the jth word of user u's review of item i
N_d	number of words in document d

Table 1: Notation.

text, and that this information can be harnessed for rating prediction. However, their method differs from ours in that their 'aspects' are provided by human annotators (e.g. they use domain knowledge to determine that the 'aspects' of a restaurant are price, service, etc.) for each of which a sentiment classifier must be trained.

We briefly mention works where aspects are *explicit*, i.e., where users rate individual features of a product in addition to their overall rating [1, 9, 17, 25], though this work differs from ours in that it requires *multiple* ratings per review. We also briefly mention orthogonal work that studies 'interpretations' of ratings using sources other than reviews, e.g. information from a user's social network [24].

A few works have considered the task of automatically identifying review dimensions. Early works discover such dimensions based on frequently occurring noun phrases [7], or more sophisticated grammatical rules [23]. More recent works attempt to address this problem in an unsupervised manner [5, 26, 30], however these sophisticated methods are limited to datasets with only a few thousand reviews. Furthermore, although related to our work, the goal of such studies is typically summarization [5, 13, 18], or feature discovery [23], rather than rating prediction *per se*.

Beyond product reviews, modeling the dimensions of free-text is an expansive topic. The goal of Latent Dirichlet Allocation (LDA) [4], like ours, is to discover hidden dimensions in text. Such models have been applied in recommendation settings, e.g. to recommend scientific articles in citation networks [28]. Although related, such approaches differ from ours in that the dimensions they discover are not necessarily correlated with ratings. Variants of LDA have been proposed whose dimensions *are* related to output variables (such as ratings), for example supervised topic models [3]. This is but one of a broad class of works on *sentiment analysis* [10, 15], whose goal is to model numeric scores from text.

In spite of the similarity between our work and sentiment analysis, there is one critical difference: the goal of such methods—at test time—is to predict sentiment from text. In contrast, in recommendation settings such as our own, our goal at test time is to predict ratings of products that users have *not* reviewed.

2. MODELS OF RATINGS AND REVIEWS

We begin by briefly describing the 'standard' models for latent factor recommender systems and Latent Dirichlet Allocation, before defining our own model. The notation we shall use throughout the paper is defined in Table 1.

Latent-Factor Recommender Systems

The 'standard' latent-factor model [11] predicts ratings $r_{u,i}$ for a user u and item i according to

$$rec(u,i) = \alpha + \beta_u + \beta_i + \gamma_u \cdot \gamma_i, \tag{1}$$

where α is an offset parameter, β_u and β_i are user and item biases, and γ_u and γ_i are K-dimensional user and item factors (respectively). Intuitively, γ_i can be thought of as the 'properties' of the product i, while γ_u can be thought of as a user's 'preferences' towards those properties. Given a training corpus of ratings $\mathcal{T}$, the parameters $\Theta = \{\alpha, \beta_u, \beta_i, \gamma_u, \gamma_i\}$ are typically chosen so as to minimize the Mean Squared Error (MSE), i.e.,

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmin}} \frac{1}{|\mathcal{T}|} \sum_{r_{u,i} \in \mathcal{T}} (rec(u,i) - r_{u,i})^2 + \lambda \Omega(\Theta), \tag{2}$$

where $\Omega(\Theta)$ is a regulariser that penalizes 'complex' models, for example the ℓ_2 norm $\|\gamma\|_2^2$. A variety of methods exist to optimize (eq. 1), for instance alternating least-squares, and gradient-based methods [11].

Latent Dirichlet Allocation

Unlike latent factor models, which uncover hidden dimensions in review *ratings*, Latent Dirichlet Allocation (LDA) uncovers hidden dimensions in review *text*. LDA associates each document $d \in \mathcal{D}$ with a K-dimensional topic distribution θ_d (i.e., a stochastic vector), which encodes the fraction of words in d that discuss each of the K topics. That is, words in the document d discuss topic k with probability $\theta_{d,k}$.

Each topic k also has an associated word distribution, ϕ_k, which encodes the probability that a particular word is used for that topic. Finally, the topic distributions themselves (θ_d) are assumed to be drawn from a Dirichlet distribution.

The final model includes word distributions for each topic ϕ_k, topic distributions for each document θ_d, and topic assignments for each word $z_{d,j}$. Parameters $\Phi = \{\theta, \phi\}$ and topic assignments z are traditionally updated via sampling [4]. The likelihood of a particular text corpus $\mathcal{T}$ (given the word distribution ϕ and topic assignments for each word) is then

$$p(\mathcal{T}|\theta, \phi, z) = \prod_{d \in \mathcal{T}} \prod_{j=1}^{N_d} \theta_{z_{d,j}} \phi_{z_{d,j}, w_{d,j}}, \tag{3}$$

where we are multiplying over all documents in the corpus, and all words in each document. The two terms in the product are the likelihood of seeing these particular topics ($\theta_{z_{d,j}}$), and the likelihood of seeing these particular words for this topic ($\phi_{z_{d,j}, w_{d,j}}$).

2.1 The HFT Model

Our model, which we title 'Hidden Factors as Topics', or HFT for short, attempts to combine these two ideas. Unlike supervised topic models (for example), that learn topics that are correlated with an output variable [3], HFT discovers topics that are correlated with the 'hidden factors' of products and users, γ_i and γ_u.

Topic models operate on *documents*, so first we must define the concept of a 'document' in HFT. We shall derive documents from review text, so an obvious choice is to define each review as a document $d_{u,i}$ (for user u and item i). Alternately, we could define a document d_i as the set of all reviews of an item i, or d_u as the set of all reviews by a user u.

Although we will later consider other alternatives, for the moment, let us consider the set of all reviews of a particular item i as a document d_i. Our reasoning is that when users review products,

they tend to discuss properties of the product more than they discuss their own personal preferences; as we see in Section 4, modeling documents in this way leads to the best performance in practice.

With documents defined in this way, for each item i we learn a topic distribution θ_i (a stochastic vector, i.e., $\theta_i \in \Delta^K$). This vector encodes the extent to which each of K topics is discussed across all reviews for that product. Note that we implicitly assume that the number of rating factors is the same as the number of review topics.

Given our motivation, we do not wish to learn *rating* parameters γ_i and *review* parameters θ_i independently. Rather, we want the two to be linked. Intuitively, rating 'factors' γ_i can be thought of as properties that a product possesses; users will then give the product a high rating if they 'like' these properties according to γ_u. On the other hand, 'topics' θ_i define particular distributions of words that appear in reviews of a product. By linking the two, we hope that if a product exhibits a certain property (high $\gamma_{i,k}$), this will correspond to a particular topic being discussed (high $\theta_{i,k}$).

However, such a transformation is non-trivial, for example, we cannot simply define the two to be equal. Critically, θ_i is a stochastic vector, i.e., each of its entries describes a *probability* that a topic is discussed, while rating factors γ_i can take any value in $\mathbb{R}^K$.

If we simply enforced that γ_i was also stochastic, we would lose expressive power in our rating model, yet if we relaxed the requirement that θ_i was stochastic, we would lose the probabilistic interpretation of review topics (which is crucial for sampling). In short, we desire a transform that allows arbitrary $\gamma_i \in \mathbb{R}^K$, while enforcing $\theta_i \in \Delta^K$ (i.e., $\sum_k \theta_{ik} = 1$). We also desire a transform that is *monotonic*, i.e., it should preserve orderings so that the largest values of γ_i should also be the largest values of θ_i. Therefore to link the two we define the transformation

$$\theta_{i,k} = \frac{\exp(\kappa \gamma_{i,k})}{\sum_{k'} \exp(\kappa \gamma_{i,k'})}. \tag{4}$$

Here the exponent in the denominator enforces that each $\theta_{i,k}$ is positive, and the numerator enforces that $\sum_k \theta_{i,k} = 1$. In this way γ_i acts as a natural parameter for the multinomial defined by θ_i [27]. We introduce the parameter κ (which we fit during learning) to control the 'peakiness' of the transformation. As $\kappa \to \infty$, θ_i will approach a unit vector that takes the value 1 only for the largest index of γ_i; as $\kappa \to 0$, θ_i approaches a uniform distribution. Intuitively, large κ means that users only discuss the *most important* topic, while small κ means that users discuss all topics evenly.

Note that in practice, we do not fit both γ and θ, since one uniquely defines the other (in practice we fit only γ). For convenience, we shall still use γ_i when referring to rating parameters, and θ_i when referring to topics.

Our final model is based on the idea that the factors γ_i should accurately model users' ratings (as in eq. 1), but also that the review corpus should be 'likely' when these factors are transformed into topics (as in eq. 3). To achieve this, we define the objective of a corpus $\mathcal{T}$ (ratings and reviews) as

$$f(\mathcal{T}|\Theta, \Phi, \kappa, z) = \sum_{r_{u,i} \in \mathcal{T}} \underbrace{(rec(u,i) - r_{u,i})^2}_{\text{rating error}} - \mu \underbrace{l(\mathcal{T}|\theta, \phi, z)}_{\text{corpus likelihood}}. \tag{5}$$

Recall that Θ and Φ are rating and topic parameters (respectively), κ controls the transform (eq. 4), and z is the set of topic assignments for each word in the corpus $\mathcal{T}$. The first part of this equation is the error of the predicted ratings as in (eq. 1), while the second part is the (log) likelihood of the review corpus as in (eq. 3). μ is a hyperparameter that trades-off the importance of these two effects.

Note that the presence of the corpus likelihood in (eq. 5) is critical, even if our ultimate goal is only to predict ratings. Essentially, the corpus likelihood acts as a regulariser for the rating prediction model, replacing the regulariser Ω from (eq. 1). When few ratings are available for a product i (or user u), the regulariser Ω of (eq. 1) has the effect of pushing γ_i and γ_u toward zero. What this means in practice is that the standard model of (eq. 1) reduces to an offset and bias term for products and users that have few ratings.

Alternately, HFT can accurately predict product factors γ_i even for items with only a few reviews; or, for users with few reviews if we model documents at the level of users. In essence, a small number of *reviews* tells us much more about a product or a user than we can possibly glean from the same number of *ratings*.

In the next section, we describe in detail how we fit the parameters Θ and Φ of the HFT model.

3. FITTING THE HFT MODEL

Our goal is to simultaneously optimize the parameters associated with ratings $\Theta = \{\alpha, \beta_u, \beta_i, \gamma_u, \gamma_i\}$ and the parameters associated with topics $\Phi = \{\theta, \phi\}$. That is, given our review and rating corpus $\mathcal{T}$ our objective is to find

$$\underset{\Theta, \Phi, \kappa, z}{\operatorname{argmin}} f(\mathcal{T}|\Theta, \Phi, \kappa, z). \qquad (6)$$

Recall that θ and γ are linked through (eq. 4), so that both Θ and Φ depend on γ. So, a change in γ modifies *both* the rating error *and* the corpus likelihood in (eq. 5). Thus we cannot optimize Θ and Φ independently.

Typically, recommender parameters like those in (eq. 1) are fit by gradient descent, while those of (eq. 3) would be found by Gibbs sampling [4]. Since HFT consists of a combination of these two parts, we use a procedure that alternates between the two steps.

We define a stochastic optimization procedure consisting of the following two steps, which we describe in more detail below:

$$\text{update } \Theta^{(t)}, \Phi^{(t)}, \kappa^{(t)} = \underset{\Theta, \Phi, \kappa}{\operatorname{argmin}} f(\mathcal{T}|\Theta, \Phi, \kappa, z^{(t-1)}) \qquad (7)$$

$$\text{sample } z_{d,j}^{(t)} \text{ with probability } p(z_{d,j}^{(t)} = k) = \phi_{k, w_{d,j}}^{(t)}. \qquad (8)$$

In the first of these steps (eq. 7) topic assignments for each word (our latent variable z) are fixed. We then fit the remaining terms, Θ, Φ, and κ, via gradient descent. We use L-BFGS, a quasi-Newton method for non-linear optimization of problems with many variables [22]. This is similar to 'standard' gradient-based methods [11], but for the presence of the additional terms arising from (eqs. 3 and 4), whose gradients are easily computed.

The second step (eq. 8) iterates through all documents d and all word positions j and updates their topic assignments. As with LDA, we assign each word to a topic (an integer between 1 and K) randomly, with probability proportional to the likelihood of that topic occurring with that word. Recalling that each item i has a K-dimensional topic distribution θ_i, for each word $w_{u,i,j}$ (in user u's review of item i), we set $z_{u,i,j} = k$ with probability proportional to $\theta_{i,k}\phi_{k,w_{u,i,j}}$. This expression is the probability of the topic k being used for this product ($\theta_{i,k}$), multiplied by the probability of the particular word $w_{u,i,j}$ being used for the topic k ($\phi_{k,w_{u,i,j}}$).

To ensure that ϕ_k (the word distribution for topic k) is a stochastic vector ($\sum_w \phi_{k,w} = 1$), we introduce an additional variable ψ and define

$$\phi_{k,w} = \frac{\exp(\psi_{k,w})}{\sum_{w'} \exp(\psi_{k,w'})}. \qquad (9)$$

As with (eq. 4), we can then optimize $\psi_k \in \mathbb{R}^D$ to determine $\phi_k \in \Delta^D$ (where D is the size of the dictionary, i.e., the num-

ber of unique words in the corpus). That is, $\psi_k \in \mathbb{R}^D$ acts as a natural parameter for the multinomial $\phi_k \in \Delta^D$.

The above procedure (eq. 8) is analogous to updating topic assignments using LDA. The critical difference is that in HFT, topic proportions θ are not sampled from a Dirichlet distribution, but instead are determined based on the value of Θ^t found during the previous step, via (eq. 4). What this means in practice is that only a single pass through the corpus is required to update z. We only sample new topic assignments z once Θ^t has been updated by (eq. 7).

Finally, the two steps (eqs. 7 and 8) are repeated until convergence, i.e., until the change in Θ and Φ between iterations is sufficiently small. Although our objective (like other 'standard' recommender system objectives) is certainly non-convex [12] and subject to local minima, in our experience it yields similar topics (though possibly permuted) when restarting with different initial conditions.

4. EXPERIMENTS

In this section, we show that:

1. Our model leads to more accurate predictions (in terms of the MSE) on standard recommendation tasks (Sec. 4.4)
2. HFT allows us to address the 'cold-start' problem. HFT's ability to fuse review text and rating information allows for more accurate recommendations for products with few ratings (Sec. 4.5)
3. HFT allows us to automatically discover product categories or 'genres' (Sec. 4.7)
4. HFT can be used to automatically identify representative reviews, which are considered 'useful' by human annotators (Sec. 4.8)

4.1 Datasets

We collect review data from a variety of public sources. Our primary source of data is *Amazon*, from which we obtain approximately 35 million reviews. To obtain this data, we started with a list of 75 million asin-like strings (Amazon product identifiers) obtained from the Internet Archive;[1] around 2.5 million of them had at least one review. We further divide this dataset into 26 parts based on the top-level category of each product (e.g. books, movies). This dataset is a superset of existing publicly-available Amazon datasets, such as those used in [8, 16, 19] and [21].

We also consider 6 million beer and wine reviews previously studied in [19, 21], though in addition we include *pub* data from *ratebeer.com*, and we consider restaurant data from *citysearch.com*, previously used in [6]. Finally, we include 220 thousand reviews from the recently proposed *Yelp Dataset Challenge*.[2]

A summary of the data we obtain is shown in Table 2. In total, we obtain 42 million reviews, from 10 million users and 3 million items. Our datasets span a period of 18 years and contain a total of 5.1 billion words. Data and code are available online.[3]

4.2 Baselines

We compare HFT to the following baselines:
(a) Offset only: Here we simply fit an offset term (α in eq. 1), by taking the average across all training ratings. In terms of the MSE, this is the best possible constant predictor.
(b) Latent factor recommender system: This is the 'standard' latent factor model, i.e., the model of (eq. 1). We fit all parameters using L-BFGS [22].

---

[1] http://archive.org/details/asin_listing/
[2] https://www.yelp.com/dataset_challenge/
[3] http://snap.stanford.edu/data/

dataset	#users	#items	#reviews	#words	av. words	since
Amazon (total)	9,505,488	2,490,432	35,276,583	4.63B	90.90	Jun. 1995
Pubs (Ratebeer)	13,957	22,418	140,359	9.89M	70.52	Jan. 2004
Beer (Ratebeer)	29,265	110,369	2,924,163	154.01M	52.67	Apr. 2000
Pubs (Beeradvocate)	15,268	16,112	112,184	21.85M	194.79	Mar. 2002
Beer (Beeradvocate)	33,388	66,055	1,586,614	195.31M	123.09	Aug. 1996
Wine (Cellartracker)	35,235	412,666	1,569,655	60.02M	38.24	
Citysearch	5,529	32,365	53,122	3.94M	74.18	May. 2002
Yelp Phoenix	45,981	11,537	229,907	29.88M	129.98	Mar. 2005
Total	9,684,111	3,161,954	41,892,587	5.10B		

Table 2: Dataset statistics (number of users; number of items; number of reviews; total number of words; average number of words per review, time of oldest review).

dataset	(a) offset	(b) +lat. factors	(c) LDA	(d) HFT ($\theta \sim \gamma_u$)	(e) HFT ($\theta \sim \gamma_i$)	improvement e vs. b	e vs. c
Amazon (average)	1.774 (0.00)	1.423 (0.00)	1.410 (0.00)	1.329 (0.00)	1.325 (0.00)*	6.89%	6.03%
Pubs (Ratebeer)	0.699 (0.01)	0.477 (0.00)	0.483 (0.01)	0.456 (0.00)*	0.457 (0.00)	4.13%	5.44%
Beer (Ratebeer)	0.701 (0.00)	0.306 (0.00)	0.306 (0.00)	0.301 (0.00)*	0.302 (0.00)	1.18%	1.24%
Pubs (Beeradvocate)	0.440 (0.00)	0.331 (0.00)	0.332 (0.00)	0.311 (0.00)*	0.311 (0.00)	6.16%	6.31%
Beer (Beeradvocate)	0.521 (0.00)	0.371 (0.00)	0.372 (0.00)	0.367 (0.00)	0.366 (0.00)*	1.50%	1.61%
Wine (Cellartracker)	0.043 (0.00)	0.029 (0.00)	0.029 (0.00)	0.028 (0.00)	0.027 (0.00)*	4.84%	4.03%
Citysearch	2.022 (0.04)	1.873 (0.03)	1.875 (0.03)	1.728 (0.03)*	1.731 (0.03)	7.56%	7.66%
Yelp Phoenix	1.488 (0.01)	1.272 (0.01)	1.282 (0.01)	1.225 (0.01)	1.224 (0.01)*	3.78%	4.53%
Average MSE	1.577	1.262	1.253	1.181	1.178	6.70%	5.98%
Average MSE with $K = 10$	1.577	1.260	1.253	1.180	1.176	6.64%	6.11%

Table 3: Results in terms of the Mean Squared Error for $K = 5$ (the best performing model on each dataset is starred; the standard error is shown in parentheses). Due to the large size of our datasets, all reported improvements are significant at the 1% level or better. HFT is shown with topics tied to user parameters (column d), and with topics tied to product parameters (column e).

(c) Product topics learned using LDA: Finally, as a baseline that combines text with product features, we consider Latent Dirichlet Allocation [4]. By itself, LDA learns a set of topics and topic proportions (stochastic vectors) for each document. By treating each 'document' as the set of reviews for a particular product, LDA generates a stochastic vector per product (θ_i), which we use to set γ_i. With γ_i fixed, we then fit α, β_u, β_i, and γ_u using L-BFGS. In other words, we fit a 'standard' recommender system, except that γ_i is set using the per-document topic scores produced by LDA. Like HFT, this baseline benefits from review text at training time, but unlike HFT its topics are not chosen so as to explain variation in ratings.

We compare these baselines to two versions of HFT:
(d) HFT, user topics In this version of HFT, topics in review text are associated with *user* parameters γ_u.
(e) HFT, item topics In this version of HFT, topics in review text are associated with *item* parameters γ_i.

4.3 Evaluation Procedure

We randomly subdivide each of the datasets in Table 2 into training, validation, and test sets. We use 80% of each dataset for training, up to a maximum of 2 million reviews. The remaining data is evenly split between validation and test sets. Offset and bias terms α, β_u, and β_i are initialized by averaging ratings and residuals; other parameters are initialized uniformly at random. Parameters for all models are fit using L-BFGS, which we run for 2,500 iterations. After every 50 iterations, topic assignments are updated (for HFT), and the Mean Squared Error (MSE) is computed. We report the MSE (on the test set) for the model whose error on the validation set is lowest.

Experiments were run on commodity machines with 32 cores and 64gb of memory. Our largest dataset fits on a single machine, and our update equations (eqs. 7 and 8) are naïvely parallelizable. Using L-BFGS on *Beeradvocate* data (for example), our algorithm required 30 minutes to fit, compared to 11 minutes for a 'standard' latent-factor recommender system. Our largest model (Amazon books) required around one day to fit. Although our optimization procedure yields local optima, we observed less than 3% deviation in MSE and energy across ten random initializations.

4.4 Rating Prediction

Results in terms of the Mean Squared Error are shown in Table 3. HFT achieves the best performance on 30 of the 33 datasets we consider (columns d and e). Due to space constraints, we only show average results across the 26 Amazon product categories.[4]

On average, across all datasets with $K = 5$ topics, a latent factor model achieves an MSE of 1.262; latent dirichlet allocation achieves a similar MSE of 1.253. HFT (with item topics) achieves on average an MSE of 1.178. Increasing the number of topics to $K = 10$ improves the performance of all models by a small amount, to 1.262 (latent factor model), 1.253 (LDA), and 1.176 (HFT). With $K = 10$ HFT achieves the best performance on 32 out of 33 datasets.

On several categories, such as 'clothing' and 'shoes', we gain improvements of up to 20% over state-of-the-art methods. Arguably, these categories where HFT is the best performing are the most 'subjective'; for example, if a user dislikes the style of a shirt, or the size of a shoe, they are arguably revealing as much about

---

[4] Refer to our extended version for complete tables [20].

themselves as they are about the product. By using review text, HFT is better able to 'tease apart' the objective qualities of a product and the subjective opinion of the reviewer.

It is worth briefly addressing the fact that we obtain only small benefits with $K > 10$ topics, contrary to established wisdom that traditional latent-factor models often use many more [12]. We note that while there may be many factors that influence a user's rating, only a few of these factors may be *discussed* in a typical review. A natural extension of HFT would be to augment it with additional latent factors, some of which are *not* constrained to be related to review text.

Modeling User Features with Text

We have primarily focused on how HFT can be used to align product features γ_i with review topics θ_i. However, it can also discover topics associated with *users*. This simply means treating the set of all reviews by a particular user u as a document, with an associated topic vector θ_u; HFT then associates γ_u with θ_u as in (eq. 4).

Results for HFT, trained to model user features with text, are shown in Table 3 (column d). Both models exhibit similar performance; their MSEs differ by less than 1%. A few individual datasets show greater variation, e.g. item features are more informative in the 'music' category, while user features are more informative in the 'office' category (see our extended version [20]).

This high level of similarity indicates that there is not a substantial difference between user and product topics. This seems surprising in light of the fact that our product topics tend to correspond with product categories (as we show later); we would not necessarily expect *product* categories to be similar to *user* categories. However a simple explanation is as follows: users do not review all products with the same likelihood, but rather they have a preference towards certain categories. Thus, while a 'dark beer' topic emerges from reviews of dark beers, a 'dark beer' topic *also* emerges from users who preferentially review dark beers. In both cases, the topics we discover are qualitatively similar.

4.5 Recommending New Products

The 'cold-start' problem is a common issue in recommender system. In particular, when a product or a user is new and one does not have enough rating data available, it is very hard to train a recommender system and make predictions.

In particular, with latent factor recommender systems, as with HFT, each user and product is associated with $K + 1$ parameters (K-dimensional latent factors and a bias term). As a consequence, when given only a few ratings for a particular user u or product i (e.g. fewer than K ratings), a latent factor recommender system cannot possibly estimate γ_u or γ_i accurately due to lack of data. The presence of the regulariser $\Omega(\Theta)$ in (eq. 1) pushes γ_u and γ_i towards zero in such cases, meaning that such users and products are modeled using only their bias terms.

However, *review text* provides significant additional information and our hope is that by including review text the relationship between products and ratings can be more accurately modeled. Even a single review can tell us many of a product's properties, such as its genre. Our hypothesis is that when training data for a product is scarce (e.g. when a new product appears in the corpus), the benefit gained from using review text will be greatest.

In Figure 1 we compare the amount of training data (i.e., the number of training reviews available for a particular product, x-axis) to the improvement gained by using HFT (y-axis). Specifically, we report the MSE of a latent-factor recommender system minus the MSE of HFT; thus a positive value indicates that HFT has better performance. Results are shown for $K = 5$ on our

three largest datasets (Amazon books, movies, and music). Figure 1 (left) shows the absolute improvement in MSE when using HFT compared to a latent factor recommender system. Notice that our model gains significant improvement when training data is particularly scarce. The improvement on movie data is the largest, while for books the 'standard' model is on par with HFT once ratings are abundant. Similar results are obtained for new *users*.

4.6 Qualitative Analysis

Some of the topics discovered by HFT are shown in Table 4. Topics are shown with $K = 5$ for beer, musical instruments, video game, and clothing data, and for $K = 10$ for Yelp data. We find that the most common words (in all topics) are stop words ('the'/'and'/'it' etc.), so to visualize each topic, we first compute averages across all topics for each word w: $b_w = \frac{1}{K} \sum_k \psi_{k,w}$. This generates a 'background' distribution that includes words that are common to all topics. Subtracting this value from $\psi_{k,w}$ gives us topics with common words and stop words removed. These values (i.e., $\psi_{k,w} - b_w$) are shown in Table 4. Note that we show *all* K topics, i.e., none are hand-selected or excluded.

The topics we discover are clean and easy to interpret. Discovered topics are similar to genres, or categories of products. For instance, beer topics include pale ales, lambics (sour Belgian beers), dark beers, and wheat beers, with an additional topic describing spices. From a modeling perspective, there is a simple explanation as to why HFT discovers 'genre-like' topics. Firstly, users are likely to rate products of the same genre similarly, so separating products into genres explains much of the variation in rating data. Secondly, different language is used to describe products from different genres, so genres also explain much of the variation in review data. As we see in the next section, if our goal is to *discover* genres, combining these two sources of data (ratings and reviews) leads to much better performance than using either in isolation.

4.7 Genre Discovery

We noted that the 'topics' discovered by HFT are similar to product 'categories', or 'genres'. Now, we confirm this quantitatively, and investigate the possibility of using HFT to automatically discover such categories. For brevity we focus on the Yelp Phoenix dataset, which contains detailed product category information.

Each of the models compared in the previous section outputs a K-dimensional vector per product, where K is the number of latent product/user dimensions (θ_i in the case of LDA, γ_i in the case of the latent factor model and HFT). We define the category of a product, which we denote c_i, to be the latent dimension with the highest weight, i.e.,

$$c_i = \underset{k}{\mathrm{argmax}}\, \gamma_{i,k} \qquad (10)$$

(or $\mathrm{argmax}_k\, \theta_{i,k}$ for LDA). Thus each product has a 'category' between 1 and K. We then compute the best alignment between these K categories and the K most popular (i.e., most frequently occurring) categories in the Yelp dataset. We denote by C_k the set of products whose predicted category is k (i.e., $\{i|c_i = k\}$), and by C_k^* the set of products whose *true* category is k. The score of the optimal correspondence is then

$$\min_f \frac{1}{K} \sum_{k=1}^{K} F_1(C_k, C_{f(k)}^*), \qquad (11)$$

where $f(\cdot)$ is a one-to-one correspondence (computed by linear assignment) between predicted product categories k and ground-truth categories $f(k)$; (eq. 11) is then the average F_1 score between the predicted and the true product categories.
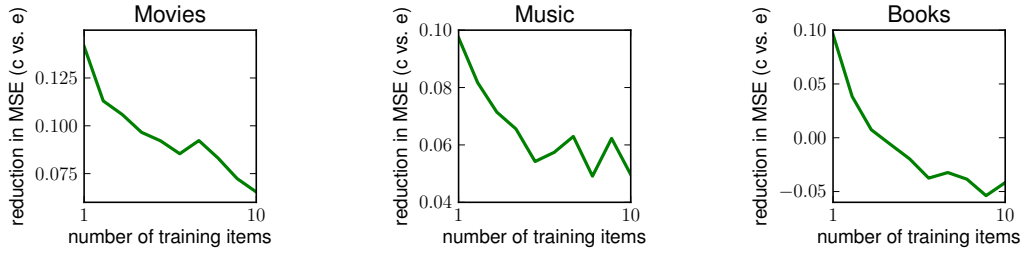
Figure 1: When training data is scarce, the benefit gained from modeling review text is the greatest. Improvement in MSE compared to a latent factor model on three Amazon datasets.

Beer (Beeradvocate)

pale ales	lambics	dark beers	spices	wheat beer
ipa	funk	chocolate	pumpkin	wheat
pine	brett	coffee	nutmeg	yellow
grapefruit	saison	black	corn	straw
citrus	vinegar	dark	cinnamon	pilsner
ipas	raspberry	roasted	pie	summer
piney	lambic	stout	cheap	pale
citrusy	barnyard	bourbon	bud	lager
floral	funky	tan	water	banana
hoppy	tart	porter	macro	coriander
dipa	raspberries	vanilla	adjunct	pils

Musical instruments (Amazon)

drums	strings	wind	microphones	software
cartridge	guitar	reeds	mic	software
sticks	violin	harmonica	microphone	interface
strings	strap	cream	stand	midi
snare	neck	reed	mics	windows
stylus	capo	harp	wireless	drivers
cymbals	tune	fog	microphones	inputs
mute	guitars	mouthpiece	condenser	usb
heads	picks	bruce	battery	computer
these	bridge	harmonicas	filter	mp3
daddario	tuner	harps	stands	program

Video games (Amazon)

fantasy	nintendo	windows	ea/sports	accessories
fantasy	mario	sims	drm	cable
rpg	ds	flight	ea	controller
battle	nintendo	windows	spore	cables
tomb	psp	xp	creature	ps3
raider	wii	install	nba	batteries
final	gamecube	expansion	football	sonic
battles	memory	program	nhl	headset
starcraft	wrestling	software	basketball	wireless
characters	metroid	mac	madden	controllers
ff	smackdown	sim	hockey	component

Clothing (Amazon)

bags	winter	formal	pants	bras
backpack	vest	scarf	pants	bra
bag	jacket	cards	jeans	bras
jansport	fleece	shirt	pair	support
costume	warm	shirts	dickies	cup
books	columbia	suit	these	cups
hat	coat	silk	levis	underwire
laptop	sweatshirt	wallet	waist	supportive
bags	russell	belt	pairs	breasts
backpacks	gloves	leather	socks	sports
halloween	sweater	tie	they	breast

Yelp Phoenix

theaters	spas	mexican	vietnamese	snacks	italian	medical	donuts	coffee	seafood
theater	massage	mexican	pho	cupcakes	pizza	dr	donuts	coffee	sushi
movie	spa	salsa	vietnamese	cupcake	crust	stadium	donut	starbucks	dish
harkins	yoga	tacos	yogurt	hotel	pizzas	dentist	museum	books	restaurant
theaters	classes	chicken	brisket	resort	italian	doctor	target	latte	rolls
theatre	pedicure	burrito	beer	rooms	bianco	insurance	subs	bowling	server
movies	trail	beans	peaks	dog	pizzeria	doctors	sub	lux	shrimp
dance	studio	taco	mojo	dogs	wings	dental	dunkin	library	dishes
popcorn	gym	burger	shoes	frosting	pasta	appointment	frys	espresso	menu
tickets	hike	carne	froyo	bagel	mozzarella	exam	tour	stores	waiter
flight	nails	food	zoo	bagels	pepperoni	prescription	bike	gelato	crab

Table 4: Top ten words from each of $K = 5$ topics from five of our datasets (and with $K = 10$ from Yelp). Each column is labeled with an 'interpretation' of that topic. Note we display all the topics (and not only the 'interpretable' ones). All topics are clean and easily interpretable.

	(b)	(c)	(e)	improvement	
K	lat. factors	LDA	HFT	e vs. b	e vs. c
5	0.166	0.205	0.412*	148.15%	100.23%
10	0.097	0.169	0.256*	163.54%	51.16%
20	0.066	0.091	0.165*	151.34%	81.82%
50	0.042	0.047	0.199*	369.14%	317.58%

Table 5: Genre discovery (on Yelp data). Values shown are average F_1 score between the predicted and the ground-truth product categories (higher is better).

This score is shown in Table 5 for different values of K. Neither latent factor models nor LDA produce latent dimensions that are similar to genres, while HFT recovers these genres accurately; with 50 genres HFT outperforms latent factor models and LDA by over 300%. It is worth noting that $K = 50$ is far larger than what we found beneficial when predicting ratings in Section 4.4. We note that when K is small, 'similar' genres tend to be combined in a single topic, e.g. 'yoga' and 'massage' in Table 4. Pleasingly, this implies that even after further topics no longer improve rating prediction accuracy, HFT continues to generate meaningful topics.

4.8 Identifying Useful Reviews

In addition to modeling how users rate products, HFT can identify reviews that users are likely to find 'useful'. As we have shown, for each product i, HFT generates a topic vector θ_i, which determines the distribution of topics, and consequently the distribution of *words*, that are likely to be used when describing that product.

We use this topic distribution to identify 'representative' reviews. Specifically, we identify reviews $r_{u,i}$ whose language matches the topic distribution θ_i closely. Recall that in HFT, 'topics' are learned in order to explain the variation present in product ratings. Thus, in order for a reviewer to adequately explain their rating, they ought to discuss each of these topics in proportion to their importance.

Recall that for each word $w_{u,i,j}$ (jth word in user u's review of item i), HFT estimates a topic assignment $z_{u,i,j}$ for that word (i.e., each word belongs to a topic from 1 to K). Given the full set of words in a review $d_{u,i}$, we can then compute the proportion of each topic in that review. Specifically, we compute

$$\vartheta_{u,i,k} = \frac{1}{N_{d_{u,i}}} \sum_{j=1}^{N_{d_{u,i}}} \delta(z_{u,i,j} = k). \quad (12)$$

Here we are simply counting the number of times each topic k occurs in a review $d_{u,i}$, and normalizing so that the entries of $\vartheta_{u,i}$
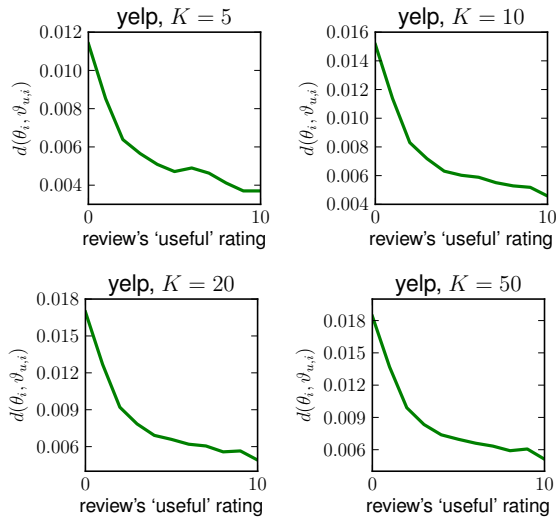
Figure 2: Reviews that are 'useful' (according to Yelp users) are those whose choice of language matches the topic distribution learned by HFT.

sum to 1. In this way, we have a topic distribution for each product, θ_i, and a topic distribution for each individual review, $\vartheta_{u,i}$ (both of which are stochastic vectors).

Our definition of a 'representative' review is one for which $\vartheta_{u,i}$ is similar to θ_i, i.e., a review that discusses those topics that explain the variation in product i's ratings. We identify this by computing the distance

$$d(\theta_i, \vartheta_{u,i}) = \|\theta_i - \vartheta_{u,i}\|_2^2. \tag{13}$$

We hypothesize that reviews with small $d(\theta_i, \vartheta_{u,i})$ will be 'representative' of the product i. We evaluate this quantitatively by comparing $d(\theta_i, \vartheta_{u,i})$ to 'usefulness' ratings on Yelp.

In Figure 2 we compare Yelp 'useful' ratings (non-negative integers) to $d(\theta_j, \vartheta_{u,i})$. Results are shown for $K \in \{5, 10, 20, 50\}$. We observe a clear relationship between the two quantities. When $K = 5$ (for example), reviews with even two useful votes are *half* the distance from θ_i compared to reviews with no useful votes; reviews with 10 useful votes are *one third* the distance from θ_i. This experiment suggests that in addition to predicting ratings, HFT can also be used to identify 'useful', or 'representative' reviews.

5. CONCLUSION

We have presented HFT, a model that combines ratings with review text for product recommendations. HFT works by aligning hidden factors in product *ratings* with hidden topics in product *reviews*. Essentially, these topics act as regularisers for latent user and product parameters. This allows us to accurately fit user and product parameters with only a few *reviews*, which existing models cannot do using only a few *ratings*. We evaluated HFT on large, novel corpora consisting of over forty million product reviews. In addition to more accurately predicting product ratings, HFT discovers highly interpretable product topics, that can be used to facilitate tasks such as genre discovery and to suggest informative reviews.

Acknowledgements

6. REFERENCES

[1] S. Baccianella, A. Esuli, and F. Sebastiani. Multi-facet rating of product reviews. In *ECIR*, 2009.

[2] J. Bennett and S. Lanning. The Netflix prize. In *KDD Cup and Workshop*, 2007.

[3] D. Blei and J. McAuliffe. Supervised topic models. In *NIPS*, 2007.

[4] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *JMLR*, 2003.

[5] S. Brody and N. Elhadad. An unsupervised aspect-sentiment model for online reviews. In *ACL*, 2010.

[6] G. Ganu, N. Elhadad, and A. Marian. Beyond the stars: Improving rating predictions using review text content. In *WebDB*, 2009.

[7] M. Hu and B. Liu. Mining and summarizing customer reviews. In *KDD*, 2004.

[8] N. Jindal and B. Liu. Opinion spam and analysis. In *WSDM*, 2008.

[9] Y. Jo and A. Oh. Aspect and sentiment unification model for online review analysis. In *WSDM*, 2011.

[10] S.-M. Kim and E. Hovy. Determining the sentiment of opinions. In *COLING*, 2004.

[11] Y. Koren and R. Bell. Advances in collaborative filtering. In *Recommender Systems Handbook*. Springer, 2011.

[12] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 2009.

[13] K. Lerman, S. Blair-Goldensohn, and R. McDonald. Sentiment summarization: evaluating and learning user preferences. In *EACL*, 2009.

[14] A. Levi, O. Mokryn, C. Diot, and N. Taft. Finding a needle in a haystack of reviews: cold start context-based hotel recommender system. In *RecSys*, 2012.

[15] C. Lin and Y. He. Joint sentiment/topic model for sentiment analysis. In *CIKM*, 2009.

[16] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 2003.

[17] B. Lu, M. Ott, C. Cardie, and B. Tsou. Multi-aspect sentiment analysis with topic models. In *Workshop on SENTIRE*, 2011.

[18] Y. Lu, C. Zhai, and N. Sundaresan. Rated aspect summarization of short comments. In *WWW*, 2009.

[19] J. McAuley and J. Leskovec. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *WWW*, 2013.

[20] J. McAuley and J. Leskovec. Hidden Factors and Hidden Topics: Understanding Rating Dimensions with Review Text. Extended version http://i.stanford.edu/~julian/pdfs/recsys_extended.pdf

[21] J. McAuley, J. Leskovec, and D. Jurafsky. Learning attitudes and attributes from multi-aspect reviews. In *ICDM*, 2012.

[22] J. Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 1980.

[23] A. Popescu and O. Etzioni. Extracting product features and opinions from reviews. In *HLT*, 2005.

[24] A. Sharma and D. Cosley. Do social explanations work? Studying and modeling the effects of social explanations in recommender systems. In *WWW*, 2013.

[25] I. Titov and R. McDonald. A joint model of text and aspect ratings for sentiment summarization. In *ACL*, 2008.

[26] I. Titov and R. McDonald. Modeling online reviews with multi-grain topic models. In *WWW*, 2008.

[27] M. Wainwright and M. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 2008.

[28] C. Wang and D. Blei. Collaborative topic modeling for recommending scientific articles. In *KDD*, 2011.

[29] H. Wang, Y. Lu, and C. Zhai. Latent aspect rating analysis on review text data: a rating regression approach. In *KDD*, 2010.

[30] W. Zhao, J. Jiang, H. Yan, and X. Li. Jointly modeling aspects and opinions with a MaxEnt-LDA hybrid. In *EMNLP*, 2010.

7. APPENDIX

7.1 Complete Tables

dataset	#users	#items	#reviews	#words	av. words	since
arts (Amazon)	24071	4211	27980	2.05M	73.55	Apr. 1998
books (Amazon)	2589009	929273	12886506	1.86B	144.86	Jun. 1995
gourmet foods (Amazon)	112544	23476	154635	10.95M	70.83	Jun. 1998
movies (Amazon)	1318175	235042	8567727	1.25B	146.98	Jun. 1995
video games (Amazon)	228570	21025	463669	66.06M	142.48	Feb. 1997
automotive (Amazon)	133256	47577	188728	13.67M	72.45	Oct. 1998
health (Amazon)	311636	39539	428781	34.72M	80.99	Jul. 1998
musical instruments (Amazon)	67007	14182	85405	7.93M	92.88	Apr. 1998
shoes (Amazon)	73590	48410	389877	23.82M	61.11	Apr. 2000
watches (Amazon)	62041	10318	68356	5.77M	84.43	Dec. 1998
baby (Amazon)	123837	6962	184887	18.17M	98.31	Feb. 1999
clothing (Amazon)	128794	66370	581933	35.13M	60.37	Jan. 1999
home and kitchen (Amazon)	644509	79006	991794	86.10M	86.81	Oct. 1995
music (Amazon)	1134704	556845	6396690	847.35M	132.46	Aug. 1995
software (Amazon)	68464	11234	95084	12.23M	128.69	Nov. 1997
electronics (Amazon)	884175	96643	1371574	148.54M	108.30	Nov. 1996
industrial and scientific (Amazon)	29590	22622	137042	7.11M	51.90	Aug. 1998
office (Amazon)	112348	14972	140517	12.03M	85.62	Jun. 1997
sports and outdoors (Amazon)	329234	68295	510995	40.79M	79.83	Jul. 1999
beauty (Amazon)	167725	29004	252056	18.29M	72.59	Jan. 1997
furniture (Amazon)	166912	19577	206338	18.14M	87.92	Nov. 1998
jewelery (Amazon)	40594	18794	58621	3.12M	53.26	Feb. 1999
pet supplies (Amazon)	160496	17523	217170	19.38M	89.26	Apr. 2000
tools and home improvement (Amazon)	290100	53377	419778	37.15M	88.51	Jul. 1998
kitchen (Amazon)	13394	2555	14444	1.24M	86.38	Apr. 2000
toys (Amazon)	290713	53600	435996	36.05M	82.70	Aug. 1996
Pubs (Ratebeer)	13,957	22,418	140,359	9.89M	70.52	Jan. 2004
Beer (Ratebeer)	29,265	110,369	2,924,163	154.01M	52.67	Apr. 2000
Pubs (Beeradvocate)	15,268	16,112	112,184	21.85M	194.79	Mar. 2002
Beer (Beeradvocate)	33,388	66,055	1,586,614	195.31M	123.09	Aug. 1996
Wine (Cellartracker)	35,235	412,666	1,569,655	60.02M	38.24	
Citysearch	5,529	32,365	53,122	3.94M	74.18	May. 2002
Yelp Phoenix	45,981	11,537	229,907	29.88M	129.98	Mar. 2005
Total	9,684,111	3,161,954	41,892,587	5.10B		

Table 6: Dataset statistics (number of users; number of items; number of reviews; total number of words; average number of words per review, time of oldest review).

dataset	(a) offset	(b) +lat. factors	(c) LDA	(d) HFT ($\theta \sim \gamma_u$)	(e) HFT ($\theta \sim \gamma_i$)	improvement e vs. b	e vs. c
arts (Amazon)	1.785 (0.05)	1.565 (0.04)	1.574 (0.05)	1.433 (0.04)	1.388 (0.04)*	11.30%	11.79%
books (Amazon)	1.455 (0.00)	1.107 (0.00)	1.107 (0.00)*	1.141 (0.00)	1.135 (0.00)	-2.54%	-2.54%
gourmet foods (Amazon)	1.617 (0.02)	1.515 (0.02)	1.492 (0.02)	1.434 (0.02)	1.431 (0.02)*	5.56%	4.11%
movies (Amazon)	1.678 (0.00)	1.118 (0.00)	1.117 (0.00)*	1.119 (0.00)	1.119 (0.00)	-0.08%	-0.17%
video games (Amazon)	1.825 (0.01)	1.610 (0.01)	1.607 (0.01)	1.513 (0.01)	1.511 (0.01)*	6.15%	5.99%
automotive (Amazon)	1.795 (0.02)	1.570 (0.01)	1.586 (0.02)	1.428 (0.01)*	1.428 (0.01)	9.02%	9.97%
health (Amazon)	1.865 (0.01)	1.613 (0.01)	1.608 (0.01)	1.544 (0.01)	1.528 (0.01)*	5.28%	4.95%
musical instruments (Amazon)	1.585 (0.02)	1.506 (0.02)	1.519 (0.02)	1.397 (0.02)	1.396 (0.02)*	7.34%	8.12%
shoes (Amazon)	1.365 (0.01)	0.273 (0.00)	0.279 (0.00)	0.224 (0.00)*	0.226 (0.00)	17.37%	19.16%
watches (Amazon)	1.583 (0.03)	1.533 (0.03)	1.517 (0.03)	1.493 (0.03)	1.486 (0.03)*	3.02%	2.02%
baby (Amazon)	1.903 (0.01)	1.615 (0.01)	1.599 (0.01)	1.439 (0.01)*	1.442 (0.01)	10.69%	9.76%
clothing (Amazon)	1.586 (0.01)	0.392 (0.00)	0.405 (0.00)	0.328 (0.00)	0.327 (0.00)*	16.55%	19.06%
home and kitchen (Amazon)	1.990 (0.00)	1.626 (0.00)	1.610 (0.00)	1.530 (0.00)	1.527 (0.00)*	6.06%	5.15%
music (Amazon)	1.261 (0.00)	0.957 (0.00)*	0.957 (0.00)	0.980 (0.00)	0.969 (0.00)	-1.23%	-1.20%
software (Amazon)	2.775 (0.01)	2.408 (0.02)	2.215 (0.02)	2.179 (0.02)*	2.197 (0.02)	8.75%	0.79%
electronics (Amazon)	2.092 (0.00)	1.829 (0.00)	1.821 (0.00)	1.722 (0.00)*	1.724 (0.00)	5.73%	5.28%
industrial and scientific (Amazon)	1.328 (0.02)	0.461 (0.01)	0.462 (0.01)	0.359 (0.01)	0.357 (0.01)*	22.54%	22.69%
office (Amazon)	2.097 (0.02)	1.813 (0.02)	1.796 (0.02)	1.670 (0.02)*	1.680 (0.02)	7.32%	6.45%
sports and outdoors (Amazon)	1.603 (0.01)	1.219 (0.01)	1.224 (0.01)	1.139 (0.00)	1.136 (0.00)*	6.74%	7.12%
beauty (Amazon)	1.809 (0.01)	1.400 (0.01)	1.408 (0.01)	1.358 (0.01)	1.347 (0.01)*	3.78%	4.38%
furniture (Amazon)	2.061 (0.01)	1.845 (0.01)	1.828 (0.01)	1.680 (0.01)*	1.686 (0.01)	8.65%	7.76%
jewelery (Amazon)	1.496 (0.03)	1.257 (0.03)	1.279 (0.03)	1.190 (0.03)	1.178 (0.02)*	6.21%	7.84%
pet supplies (Amazon)	1.908 (0.01)	1.700 (0.01)	1.699 (0.01)	1.592 (0.01)	1.582 (0.01)*	6.94%	6.87%
tools and home impr. (Amazon)	1.887 (0.01)	1.600 (0.01)	1.608 (0.01)	1.505 (0.01)	1.499 (0.01)*	6.29%	6.74%
kitchen (Amazon)	2.098 (0.06)	1.996 (0.06)	1.949 (0.06)	1.781 (0.06)	1.777 (0.06)*	10.95%	8.80%
toys (Amazon)	1.674 (0.01)	1.467 (0.01)	1.393 (0.01)	1.370 (0.01)	1.366 (0.01)*	6.86%	1.97%
Pubs (Ratebeer)	0.699 (0.01)	0.477 (0.00)	0.483 (0.01)	0.456 (0.00)*	0.457 (0.00)	4.13%	5.44%
Beer (Ratebeer)	0.701 (0.00)	0.306 (0.00)	0.306 (0.00)	0.301 (0.00)*	0.302 (0.00)	1.18%	1.24%
Pubs (Beeradvocate)	0.440 (0.00)	0.331 (0.00)	0.332 (0.00)	0.311 (0.00)*	0.311 (0.00)	6.16%	6.31%
Beer (Beeradvocate)	0.521 (0.00)	0.371 (0.00)	0.372 (0.00)	0.367 (0.00)	0.366 (0.00)*	1.50%	1.61%
Wine (Cellartracker)	0.043 (0.00)	0.029 (0.00)	0.029 (0.00)	0.028 (0.00)	0.027 (0.00)*	4.84%	4.03%
Citysearch	2.022 (0.04)	1.873 (0.03)	1.875 (0.03)	1.728 (0.03)*	1.731 (0.03)	7.56%	7.66%
Yelp Phoenix	1.488 (0.01)	1.272 (0.01)	1.282 (0.01)	1.225 (0.01)	1.224 (0.01)*	3.78%	4.53%
Average MSE	1.577	1.262	1.253	1.181	1.178	6.70%	5.98%
Average MSE with $K = 10$	1.577	1.260	1.253	1.180	1.176	6.64%	6.11%

Table 7: Results in terms of the Mean Squared Error for $K = 5$ (the best performing model on each dataset is starred; the standard error is shown in parentheses). Due to the large size of our datasets, all reported improvements are significant at the 1% level or better. HFT is shown with topics tied to user parameters (column d), and with topics tied to product parameters (column e).