Discovering Personalized Semantics for Soft Attributes in Recommender Systems using Concept Activation Vectors*

CHRISTINA GÖPFERT†, Amazon, Germany
ALEX HAIG, Google Research, USA
CHIH-WEI HSU‡, Google Research, USA
YINLAM CHOW‡, Google Research, USA
IVAN VENDROV†, Midjourney, USA
TYLER LU†, Meta AI, USA
DEEPAK RAMACHANDRAN, Google Research, USA
HUBERT PHAM, Google Research, USA
MOHAMMAD GHAVAMZADEH†, Amazon, USA
CRAIG BOUTILIER, Google Research, USA

Interactive *recommender systems* have emerged as a promising paradigm to overcome the limitations of the primitive user feedback used by traditional recommender systems (e.g., clicks, item consumption, ratings). They allow users to express intent, preferences, constraints, and contexts in a richer fashion, often using natural language (including faceted search and dialogue). Yet more research is needed to find the most effective ways to use this feedback. One challenge is *inferring a user's semantic intent* from the open-ended terms or attributes often used to describe a desired item. This is critical for recommender systems that wish to support users in their everyday, intuitive use of natural language to refine recommendation results. Leveraging *concept activation vectors (CAVs)* [26], a recently developed approach for model interpretability in machine learning, we develop a framework to learn a representation that captures the semantics of such attributes and connects them to user preferences and behaviors in recommender systems. One novel feature of our approach is its ability to distinguish objective and *subjective* attributes (both subjectivity of *degree* and of *sense*), and associate *different senses* of subjective attributes with different users. We demonstrate on both synthetic and real-world data sets that our CAV representation not only accurately interprets users' subjective semantics, but can also be used to improve recommendations through *interactive item critiquing*.

*This is an extended version of a paper that appeared at WWW-22: The Web Conference 2022, April 2022, Lyon, France. This version contains further elaboration of our methods, additional experiments, and a detailed evaluation of our methods (see Sec. 6) using human rater data.
†This work conducted while at Google Research.
‡Contact author.

Authors' addresses: Christina Göpfert, Amazon, Germany, chgopfert@gmail.com; Alex Haig, Google Research, 1600 Amphitheatre Parkway, Mountain View, CA, USA, 94043, ahaig@google.com; Chih-wei Hsu, Google Research, 1600 Amphitheatre Parkway, Mountain View, CA, USA, 94043, cwhsu@google.com; Yinlam Chow, Google Research, 1600 Amphitheatre Parkway, Mountain View, CA, USA, 94043, yinlamchow@google.com; Ivan Vendrov, Midjourney, San Francisco, CA, USA, ivendrov@gmail.com; Tyler Lu, Meta AI, Menlo Park, CA, USA, tyler.lu@gmail.com; Deepak Ramachandran, Google Research, 1600 Amphitheatre Parkway, Mountain View, CA, USA, 94043, ramachandrand@google.com; Hubert Pham, Google Research, 1600 Amphitheatre Parkway, Mountain View, CA, USA, 94043, hubertpham@google.com; Mohammad Ghavamzadeh, Amazon, Palo Alto, CA, USA, ghavamza@amazon.com; Craig Boutilier, Google Research, 1600 Amphitheatre Parkway, Mountain View, CA, USA, 94043, cboutilier@google.com.

CCS Concepts: • **Information systems → Personalization**.

Additional Key Words and Phrases: Interactive recommender system, Personalized semantics, Concept activation vectors (CAVs)

ACM Reference Format:
Christina Göpfert, Alex Haig, Chih-wei Hsu, Yinlam Chow, Ivan Vendrov, Tyler Lu, Deepak Ramachandran, Hubert Pham, Mohammad Ghavamzadeh, and Craig Boutilier. 2024. Discovering Personalized Semantics for Soft Attributes in Recommender Systems using Concept Activation Vectors. 1, 1 (March 2024), 36 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The ubiquity of *recommender systems* in mediating the discovery and consumption of content, products and services has not only driven user demand for recommender systems, but has increased the expectation that such systems should more deeply understand their needs and preferences. *Conversational recommenders* [3] have emerged as a promising paradigm to meet such expectations—they can overcome the primitive user feedback admitted by traditional recommender systems (e.g., simple queries, clicks, item consumption, ratings, purchases), and allow users to express their intent, preferences, constraints and contexts in a richer fashion through the use of natural-language-based interaction (e.g., faceted search, or more open-ended dialogue), and thereby dramatically increasing the bandwidth of communication between user and the recommender system. However, interpreting such interactions requires grounding the intended semantics of the user with respect to the recommender system's underlying model of user preferences. For example, if a user expresses a desire for a "funny" movie, this must be translated into an actionable representation of her preferences over the target movie corpus.

When the set of item attributes is well-defined and known *a priori* (e.g., as with the item attributes in a product catalog), existing techniques such as *faceted search* [29, 57] or *example critiquing* [13, 15] can be used directly. But often item attributes are *soft* [2]: there is no definitive "ground truth" source associating such soft attributes with items; the attributes themselves may have imprecise interpretations; and they may be *subjective* in nature (i.e., different users may interpret them differently). For instance, in *collaborative filtering (CF)* tasks such as movie recommendation, side information about movie attributes like 'funny,' 'thought-provoking,' or 'violent' is often available, but it is often ancillary, derived from sparse, noisy user comments, reviews, or tags. Moreover, users may disagree on which movies they consider to be 'violent' (or which are 'too violent' for them). Using soft attributes for item search or recommendation is thus challenging for three main reasons. First, in the absence of ground truth, we must predict which items exhibit such attributes using models built from noisy, incomplete data, a process characterized by considerable uncertainty. Second, these predictions may depend on a *specific* user's usage or "intended semantics," reflecting potential subjectivity in the semantics of such attributes. Third, their open-ended nature means that recommender systems must often take steps to *identify* those soft attributes that are most relevant to their domain and to their users.

Recent work has attempted to *jointly* learn the semantics of soft attributes with user preferences [32, 41, 59], providing some steps toward the question of soft attribute semantics. In this work, we adopt a different perspective: we treat the recommendation task as primary, using standard collaborative filtering models for recommender systems without adjusting them to incorporate soft attributes. Instead, we *infer the semantics of soft attributes using the representation learned by the recommender system model itself* [16, 50]. Our approach has four key advantages over joint methods:

(1) The recommender system's model capacity is directed to predicting user-item preferences without further trying to predict additional side information (e.g., tags), which often does not improve recommender system performance.

(2) The recommender system model can easily accommodate new attributes *without retraining* should new sources of tags, keywords or phrases emerge from which to derive new soft attributes.
(3) Our approach offers a means to test whether specific soft attributes are *relevant* to predicting user preferences. Thus, we are able focus attention on attributes most relevant to capturing a user's intent (e.g., when explaining recommendations, eliciting preferences, or suggesting critiques).
(4) One can learn soft attribute/tag semantics with relatively small amounts of labelled data, in the spirit of pre-training and few-shot learning.

At a high-level, our approach works as follows. we assume we are given: (i) a collaborative filtering-style model (e.g., probabilistic matrix factorization or dual encoder) which embeds items and users in a latent space based on user-item ratings; and (ii) a (small) set of *tags* (i.e., soft attribute labels) provided by a *subset of users* for a *subset of items*. We develop methods that associate with each item the degree to which it exhibits a soft attribute, thus determining that attribute's semantics. We do this by applying *concept activation vectors (CAVs)* [26]—a recent method developed for interpretability of machine-learned models—to the collaborative filtering model to detect whether it *learned a representation of the attribute*. The projection of this CAV in embedding space provides a (local) *directional semantics* for the attribute that can then be applied to items (and users). Moreover, the technique can be used to identify the *subjective nature* of an attribute, specifically, whether different users have different meanings (or tag *senses*) in mind when using that tag. Such a *personalized semantics* for subjective attributes can be vital to the sound interpretation of a user's true intent when trying to assess her preferences.

Our key contributions in this work are as follows:

(1) We propose a novel framework using CAVs to identify the semantics of soft attributes relative to preference prediction or behavioral models in recommender systems *without requiring co-training* of semantics and preference models.
(2) We develop methods to distinguish *objective* and *subjective* attributes (both subjectivity of *degree* and of *sense*) and associate different senses of subjective attributes with different users.
(3) We investigate a simple method that leverages this semantics to elicit preferences via *example critiquing*.

The remainder of the paper is organized as follows. In Section 2 we outline our problem setup and discuss related work. In Section 4, we use CAVs to identify the semantics of (objective) soft attributes by leveraging collaborative filtering user/item representations (linear and nonlinear), while in Section 5 we extend the approach to handle subjective attributes and identify a user's interpretation for different tags. We test our methods on both synthetic data and the MovieLens20M data set [20] in these two sections. In Section 6 we test the ability of CAVs to uncover the semantics of soft attributes relative to semantic assessments of items provided by human raters, using the *SoftAttributes* data set, a rater-generated soft-attribute data set developed by Balog et al. [2]. In Section 7 we illustrate how CAVs can be used to support example critiquing using such soft attributes. We also include appendices that describe some of the details omitted (but summarized) in the main body of the paper—this is done to improve the flow of the main text. To support reproducibility and further research on this topic, we have made available source code (where possible) and the data used by our methods. Links are referenced at appropriate points in the paper, and summarized in App. A.5.

2 PROBLEM FORMULATION

We first outline our problem formulation and data assumptions.

User-item Ratings. We assume a standard collaborative filtering task: users $\mathcal{U}$ offer ratings of items $\mathcal{I}$, with $r_{u,i} \in \mathcal{R}$ denoting the rating of user $u \in \mathcal{U}$ for item $i \in \mathcal{I}$, where $\mathcal{R}$ is the set of possible ratings (e.g., 1–5 stars). Let $n = |\mathcal{U}|$, $m = |\mathcal{I}|$, and $\mathbf{R}$ denote the $m \times n$ (usually sparse) ratings matrix, with $r_{u,i} = 0$ denoting that user u has not rated item i. Let $R = \{(u, i) : r_{u,i} \neq 0\}$ be the set of user-item pairs for which a rating has been given.

Preference Predictions. We assume a collaborative filtering method has been applied to the ratings matrix $\mathbf{R}$ to construct *user and item embeddings*, $\phi_U : \mathcal{U} \mapsto \mathbb{R}^d$ and $\phi_I : \mathcal{I} \mapsto \mathbb{R}^d$, respectively, such that the model's predicted (expected) rating for any user-item pair is $\hat{r}_{i,u} = \phi_U(u)^\top \phi_I(i)$. We let $X \subseteq \mathbb{R}^d$ generically denote the embedding space. Suitable methods for generating such embedding representations include matrix factorization [51] or certain forms of neural collaborative filtering [4, 60]. For concreteness, we assume a *two-tower model* (or *dual encoder*) in which users and items are passed through separate (but co-trained) deep neural nets (DNNs), N_U and N_I, to produce their respective vector embeddings $\phi_U(u)$ and $\phi_I(i)$, which are combined via dot product to predict user-item affinity $\hat{r}_{i,u}$ [60, 61]. We can view $\phi_I(i)$ as a (learned) *latent feature vector* characterizing item i and $\phi_U(u)$ as parameterizing user u's estimated *utility (or preference) function* over these "latent features." Notice that, by construction, this interpretation means user utility is linear with respect to these latent item features. This is a limitation we discuss in some depth in Section 4.3. While we focus on this specific two-tower architecture, our methods apply directly to matrix factorization models and should extend to more general neural collaborative filtering approaches [21].

Soft Attributes & Tags. Collaborative filtering methods are often used to predict user-item affinity in *content* recommender systems (e.g., for movies, music, news, etc.) because *user rating or consumption behavior* is generally far more predictive of user preferences than typical *hard attributes*. Here we use the term to denote the commonly known and "objective" features or properties of (e.g., genre, artist, director) [28]. Despite this, users often *describe* items of interest using *soft attributes* [2, 45], features that are not part of an agreed-upon, formal item specification. For example, movies might be described using terms like 'funny,' 'thought-provoking,' 'violent,' 'cheesy,' etc.

Often soft attributes are articulated by users via terms or *tags* applied to items, which are intended to refer to some underlying attribute or property of the item in question. The use of tags makes it clear why these underlying attributes are soft: the tags are neither applied universally to all items, nor are they applied by all users. Moreover, as we discuss below, these tags may be applied *subjectively* in the sense that users may disagree on the items to which, or the degree to which, these tags apply. Finally, their often open-ended nature makes it challenging to determine which soft attributes are most closely aligned with, or predictive of, user preferences. It is these three properties that require a recommender system to treat the attributes corresponding to tags as soft.

A number of recommender systems support user-supplied tags (see, e.g., the MovieLens data set [20] we use below). In what follows, we will make this assumption; however, tags may also be extracted from user descriptions, reviews or other data sources [22]. For simplicity, we assume a set of k canonical tags $\mathcal{T}$ that users may adopt to describe items.[1] We assume that tags are used *propositionally*, that is, a user who considers some tag for some item simply chooses to apply the tag to an item or not. However, even in this propositional usage, the underlying attribute to which a tag refers may be *ordinal* or *cardinal*, e.g., a tag 'violent' may refer to some degree of 'violence'.[2] *Tag data* comprises a $m \times n \times k$ tensor $\mathbf{T}$ where $t_{u,i,g} = 1$ if user u applies tag g to item i, and 0 otherwise. Let $T = \{(u, i, g) : t_{u,i,g} = 1\}$ be the set of user-item-tag triples in the data set, and $T_g = \{(u, i) : t_{u,i,g} = 1\}$ be the set of user-item pairs for which tag g has been applied. Tags are usually strictly sparser than ratings—that is, if a user u tags an item i with *any* tag g, u has also rated

[1]We set aside the question of aggregating potentially diverse tags into a set of covering prototypes.
[2]Our techniques can be extended in a straightforward way to Boolean (positive and negative application), ordinal or cardinal tags.

i—so we assume $T_g \subseteq R$ for all tags g. Note that user u may apply multiple tags to the same item (e.g., a user may tag a movie as 'violent,' 'thought-provoking' and 'dark'). Let $T_u \subseteq \mathcal{I}$ be the set of items tagged by u (using any tag), $T_{u,g}$ be those tagged by u with g specifically, and $T_{u,\overline{g}} = T_u \setminus T_{u,g}$ be those tagged by u but *not* with g. Our tag-data setup is similar to that used in work on tag recommenders [16, 18, 32].

Elicitation & Critiquing. Collaborative filtering models, in isolation, are ill-suited to recommender systems that aim to naturally interact with users to refine knowledge of their preferences. A CF-based recommender system can actively elicit new ratings or support "more like this" statements at the *item level* [10, 63], but the embedding representation of items, since it is not generally interpretable, does not lend itself to such *attribute-based interaction*. Tags can help alleviate this limitation, and offer a compelling mechanism to help users navigate the item space. A number of preference elicitation and example critiquing methods have been developed that use *hard attributes* (see Related Work below). The challenge in content recommenders is that tags typically correspond to *soft attributes* and are often *subjective* in nature. For example, if a user critiques a movie recommendation by asking for something "more thought-provoking" or "less violent," standard techniques for adjusting the recommender system's model of the user's preference cannot be applied unless we have a concrete semantics that relates the corresponding tag—or more precisely, the underlying soft attribute—to specific items in a way that better reflects her preferences. It is this problem we address in this work.

Concept Activation Vectors. Research on interpretable representations tries to overcome the fact that modern machine-learned models—and deep neural networks (DNNs) in particular—usually learn complex, non-transparent representations of concepts [26, 52]. The *testing CAVs (TCAV)* framework [26] is one such mechanism that tries to find a correspondence between the "state" of a model (e.g., input features, DNN activation patterns) and human-interpretable concepts. For instance, suppose a DNN has been trained to classify animals in images. Using only a small set of images with positive and negative examples of some concept (e.g., "objects with stripes"), TCAV is used to test whether the DNN has learned a representation of that concept in the form of a vector of activation (CAV) that correlates with its presence. Moreover, using the derivative of the classifiers output with respect to the CAV's direction, it measures how important that concept is to the classifier's predictions (e.g., how sensitive a "zebra" classification is to the presence of stripes in an image). In our setting, we use CAVs to translate between latent item representations learned by a collaborative filtering model and the soft attributes that users adopt to describe items and preferences. We provide more detail on the application of CAVs to recommender item representations in the next section.

For recommender systems, we use CAVs to translate between the latent item representations learned by a collaborative filtering model and the soft attributes users adopt to describe items and preferences. We detail the adaptation of key CAV concepts to recommender systems in Section 4 below.

3 RELATED WORK

We briefly discuss selected related work that bear connections to either our problem or some of the techniques we use.

A number of methods exist for finding the semantics of tags and attributes in recommender systems using tag data [18, 32] or reviews [36]. While some learn semantics jointly with ratings prediction, others build attribute models "on top of" a latent factor model built for ratings prediction as we do here. Most related to our approach is that of Gantner et al. [18], who learn semantics for tags or explicit item features as a linear combination of latent features using k-nearest neighbors or linear regression, where the latent features are those of a Bayesian personalized regression (BPR) model [49] (though the technique applies more broadly). Cohen et al. [16] extend this model to incorporate uncertainty and active exploration. This line of work is proposed as a means for solving the cold-start problem. Our work differs primarily in its ability to handle nonlinear representations and subjectivity, and in its focus on conversational and critiquing recommender systems. More generally, the broader literature on tag recommendation bears some connection to our work [30, 31, 34, 50] by modeling the relationship between users, items and tags.

One of our main motivations is to make soft and subjective attributes interpretable for use in critiquing [15], faceted search [57], and preference elicitation [1, 43, 56] for recommender systems. A wide variety of techniques have been developed in these areas, though we do not survey these here since we do not propose novel elicitation or critiquing methods—our focus is instead on how to incorporate soft, subjective attributes into such methods, which we exemplify using a fairly "vanilla" critiquing model. However, Radlinski et al. [45] develop a methodology for connecting the preferences of users with their use of soft attributes in conversational recommender systems, while Balog et al. [2] develop techniques and data set (which we use below) to assess the semantics of soft attributes.

While little work in elicitation for recommendation addresses subjectivity, one exception is [8, 9] who consider "definitional" subjectivity defined using personalized logical concepts, which is a very different notion from ours. Radlinski et al. [46] provides an overview of various research challenges associated with the use of subjective attributes. Subjectivity has been studied in natural language processing and psycholinguistics. Welch et al. [58] show that training personalized embeddings for certain classes of words (e.g., adverbs, social words, words relating to cognitive processes) increases language modeling performance. Their methodology could be used for a more granular analysis of which words and phrases are most subjective, which can then be used as a prior for analyzing subjectivity in recommender settings. Prototype theory [42] is a popular methodology where subjectivity is related to the similarity of an item to an idealized exemplar. Geometrically, this could be considered analogous to a "ball" semantics of subjectivity while CAVs encapsulate a "line" semantics.

4 FINDING RELEVANT SOFT ATTRIBUTES

Our first contribution is the development of a novel method for identifying the semantics of *relevant soft attributes* with respect to the item embedding representation learned by our collaborative filtering method. Assume a collaborative filtering model $\Phi = (\phi_U, \phi_I)$ trained on ratings data $\mathbf{R}$ with additional tag data $\mathbf{T}$. As above, we will often assume that ϕ_U and ϕ_I are realized by DNNs N_U and N_I, respectively. We use CAVs [26] to discover whether the collaborative filtering model has learned an implicit representation of a soft attribute corresponding to a given tag. If so, that representation can be used to support example critiquing, preference elicitation or item-space navigation (Section 7). Critically, *we do not use the tag data* when training the collaborative filtering model—this is akin to work that builds attribute models on top of embeddings to address the cold-start problem [16, 50], and stands in contrast with approaches that jointly train models to predict both ratings and attributes [32, 59]. Our hypothesis is the following:

> *If a tag is useful for understanding user preferences across a broad swath of the user population, the collaborative filtering model will have learned a representation of the corresponding soft attribute.*

As such, our general approach takes the collaborative filtering model as primary, and works directly with it. Of course, the converse is that if no such representation (or CAV) is uncovered by our approach, the soft attribute in question is of limited value for users expressing their preferences.

Our approach has a several important advantages:

(1) The recommender system model can be developed/trained/used without a pre-commitment to a specific attribute vocabulary—new attributes can be added as needed without retraining the model itself.

(2) Recommender system model capacity is focused on the core task of preference prediction and recommendation, and not used for attribute prediction directly.
(3) Our method can be used to assess the relevance and importance of specific attributes for preference elicitation or critiquing.

In this section, we set aside the possibility of subjectivity in attribute semantics, deferring its treatment to Section 5 (though some implicit *subjectivity of degree* is accounted for here). We begin with an informal elaboration of how CAVs can be applied directly to soft attribute semantics in recommender systems (Section 4.1). We explicate our technical approach by first assuming that soft attributes are linear in user/item embedding space (Section 4.2), then turn to nonlinear representations (Section 4.3). We evaluate our methods empirically in Section 4.4.

4.1 Mapping CAV Notions to Recommender Systems

Before formalizing our approach, we briefly map (and adapt) key CAV concepts to our recommender setting by drawing an analogy with the image classification setting used to explicate CAVs by Kim et al. [26] (and briefly described above). Our DNN collaborative filtering model $\Phi = (\phi_U, \phi_I)$ is trained on user-item ratings, similar to a multi-class image classifier trained on labeled images. A soft attribute in our setting corresponding to some tag g (say, 'violent') is analogous to some image feature (e.g., 'stripes') in the image setting. Our aim is to use CAVs to determine if the trained *item network* N_I has learned some representation of the attribute corresponding to this tag, by direct analogy to the image case, where the goal is to uncover a representation of an image feature like 'stripes' if one exists in the classifier network. Just as in the image setting, where some small set of positive example images (with stripes) and negative examples (non-striped) is used to attempt to identify a CAV, we use a *small* set of positive (tagged) and negative (untagged) items for the same purpose, though we must account for both variability and inconsistency in the tags applied by different users.[3] Fig. 1 offers a graphical depiction of our use of CAVs for recommender systems (including in example critiquing, see Section 7). We refer to Table 1 for a concise list of key concepts and notations.

4.2 Linear Attributes

We adapt CAVs to test whether a collaborative filtering model has learned a representation of a soft attribute corresponding to a tag. We first illustrate our approach by testing whether the *embedding space itself* contains a *linear* representation of the tag's underlying attribute (i.e., an attribute that is linear in item embedding features ϕ_I).[4] We generalize this linear model (whose weaknesses we detail below) in Section 4.3.

Our basic assumptions are as follows. We assume a collaborative filtering model Φ, where each item $i \in \mathcal{I}$ is represented by its embedding $\phi_I(i) \in X$ and each user $u \in \mathcal{U}$ by $\phi_U(u) \in X$ For any user u and tag g, we have access to $T_{u,g}$, the set of items to which u has applied g, and $T_{u,\overline{g}}$, those items not tagged with g (but to which the user has applied at least one other tag). We treat $\phi_I(i)$ for any $i \in T_{u,g}$ as a positive example of the underlying concept (say, violent movies) associated with tag g, and for any $i \in T_{u,\overline{g}}$ as a negative example. Notice that these negatives are "implicit," but they are plausible—the user u has otherwise tagged these items, so they are aware and engaged with any $i \in T_{u,\overline{g}}$, but has not used the tag g; moreover, we only consider negative examples for g if u has applied tag g positively to at

[3]While not our aim in this work, we can also use CAVs to test the sensitivity of various predictions to the presence of this soft attribute: by analogy with testing the sensitivity of a 'zebra' classification to the presence of stripes, we can test the sensitivity of a user's item rating to the item's (degree of), say, violence. In the collaborative filtering setting however, this sensitivity will differ across the user population.
[4]The code used compute CAVs with linear attributes, as well as to evaluate the CAVs using the evalution metrics described below, is available at https://github.com/google-research/google-research/tree/master/attribute_semantics. It is used for linear attribute results as described in Sections 4.4, 5.3, 6.2 and 6.3.

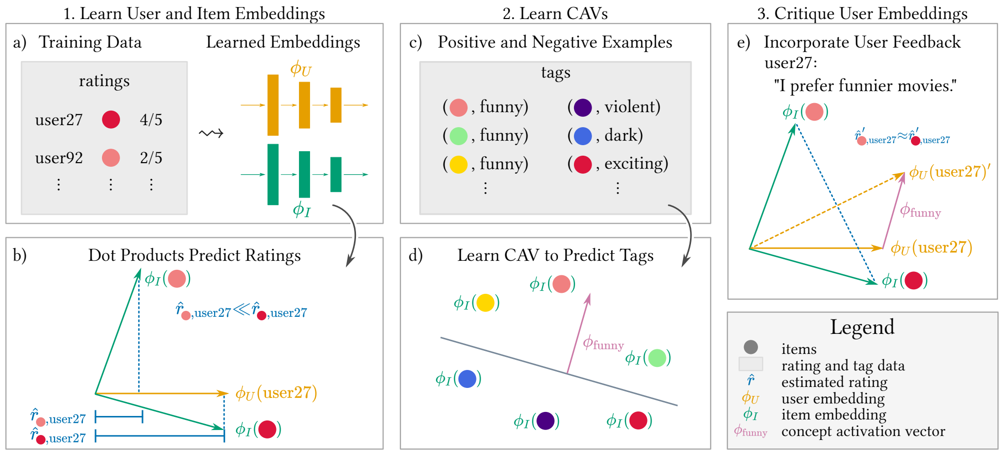
1. Learn User and Item Embeddings
a) Training Data
Learned Embeddings
ratings
user27 4/5
user92 2/5
ϕ_U
ϕ_I
b) Dot Products Predict Ratings
$\hat{r}_{\bullet,\text{user27}} \ll \hat{r}_{\bullet,\text{user27}}$
$\phi_U(\text{user27})$
2. Learn CAVs
c) Positive and Negative Examples
tags
(, funny)
(, violent)
(, dark)
(, exciting)
d) Learn CAV to Predict Tags
ϕ_{funny}
3. Critique User Embeddings
e) Incorporate User Feedback
user27:
"I prefer funnier movies."
$\hat{r}'_{\bullet,\text{user27}} \approx \hat{r}'_{\bullet,\text{user27}}$
$\phi_U(\text{user27})'$
Legend
items
rating and tag data
estimated rating
user embedding
item embedding
concept activation vector

Fig. 1. An overview of the collaborative filtering, CAV learning and critiquing setup used in our work. a) Learn user ϕ_U and item embeddings ϕ_I from ratings data. b) User-item affinity $\hat{r}_{i,u}$ is predicted using dot product of user-item embeddings. c) To learn a CAV for the concept 'funny,' gather positive and negative examples, e.g., from tag data, and use them to d) learn a CAV in item-embedding space. (We explore several methods for CAV training.) e) In one use case, we use the CAV to update a user's embedding given her item-attribute feedback. Figure inspired by [26].

Rating	Measure $r_{u,i}$ of user u's preference for item i.
User/item embedding	Vector representations of users $\phi_U(u)$ and items $\phi_I(i)$ learned using, for example, collaborative filtering methods on ratings data. The estimate of u's rating for i is $\hat{r}_{i,u} = \phi_U(u)^\top \phi_I(i)$. If ϕ_I is represented by a DNN, we denote the activations for item i at the ℓ-th layer by $\phi_{I,\ell}(i)$.
Tag	A term (from set $\mathcal{T}$) propositionally applied by users to describe items. Each tag corresponds to an attribute or *concept*.
Attribute	A specific property of items (or a *concept*). Attributes may be cardinal or ordinal (items can possess more or less of an attribute) or Boolean (items either exhibit that attribute or not).
Concept activation vector (CAV)	A CAV ϕ_g for a tag g is a vector in embedding or activation space that represents a direction based on which items "possess more of" the attribute corresponding to g. In effect, this is the semantics of the underlying attribute. CAVs can be learned using classification or learning-to-rank methods on tag data.
Subjectivity	We distinguish three types of tags: (1) *objective* tags, where users (more or less) agree on whether (or the degree to which) an item satisfies the attribute corresponding to the tag; (2) *degree subjective* tags, where users agree on the relative degree to which items possess the underlying attribute, but may disagree on whether the (boolean) attribute/tag applies; and (3) *sense subjective* tags, where (groups of) users may disagree on which items possess the underlying attribute (or the relative degree).

Table 1. A Recap of Several Key Concepts Used Throughout the Paper.

least one item (i.e., has shown an awareness of, and willingness to use, tag g).[5] Explicit negatives are, of course, more informative, and what follows can be generalized easily to this case. Taking examples over all users $u \in \mathcal{U}$, we train a linear classifier in one of two ways.

[5]Potential bias that may arise since tags may not be "missing at random" [35], but we defer this issue to future work.

Binary Logistic Regression. Our first model assumes that every user u uses a tag g in roughly the same way. As such, we simply aggregate the positive instances over all users to form the multi-set $T_g = \cup_{\mathcal{U}}\{i : i \in T_{u,g}\}$; similarly, we aggregate all negatives to obtain $T_{\overline{g}} = \cup_{\mathcal{U}}\{i : i \in T_{u,\overline{g}}\}$. Since positive tag examples are often sparse, we use *negative sampling* to manage class imbalance [38, 60], sub-sampling the negative data set.[6] Let D_g be the induced "global" data set aggregating the data of all users, with positive examples $\langle\phi_I(i), +1\rangle$ for any $i \in T_g$ and negative examples $\langle\phi_I(i), -1\rangle$ for any $i \in T_{\overline{g}}$. We train a logistic regressor ϕ_g to predict whether an item i "satisfies" the soft attribute corresponding to tag g using (regularized) logistic loss:

$$\mathcal{L}(\phi_g; D_g) = \sum_{(i,y)\in D_g} \log(1 + e^{-y\phi_g^\top \phi_I(i)}) + \frac{\lambda}{2}\phi_g^\top \phi_g. \tag{1}$$

Here, our labels are $y \in \{+1, -1\}$ and λ is our regularization penalty weight. Once trained, we obtain a regressor ϕ_g for tag g that predicts the probability $P(g(i); \phi_g)$ that $g(i)$ holds—i.e., that g applies to an item i—to be $\sigma(\phi_g^\top \phi_I(i))$ (where σ is the sigmoid function, $\sigma(x) = \frac{1}{1+e^{-x}}$). The induced regressor is a vector in embedding space X and serves as our CAV. We refer to this method as *binary logistic regression for CAVs.*

Per-user Pairwise Loss (RankNet). The logistic regression approach—which learns the CAV semantics ϕ_g for tag g by training on the *union* of all users' positive and negative examples—is "global" and lacks some nuance. For instance, if two users disagree on the application of tag g to some item, this global classifier treats the lack of agreement as label noise. An alternative explanation for such a discrepancy is that the two users actually agree on the "direction" of the attribute embodied by tag g, but disagree on the "degree" to which item i exhibits g's underlying soft attribute. For example, given *any* pair of movies, two users may agree on which movie of the pair is more violent; but they may have different *thresholds* or tolerances when actually applying the tag (e.g., they might disagree on "how violent is violent"). For instance, both users may agree that movie i is more violent than movie j, but one may consider neither i nor j to reach the threshold needed for them to label it truly 'violent,' while the other may believe that one or both of i and j warrant that label. Our second model accounts for this by treating each user u as generating a set of *pairwise comparisons*,

$$D_u = \{i >_g j : i \in T_{u,g}, j \in T_{u,\overline{g}}\},$$

drawn from an underlying ranking. Here D_u captures the fact that u considers item i to possess a "greater degree" of attribute g that item j, i.e., $i >_g$, for any i to which she applied the tag and any j to which she has not (assuming j is in u's tagged set T_u at all).

Using this set of comparisons (over all $u \in \mathcal{U}$, with negative sampling), we use a *per-user* pairwise ranking loss to generate a *regressor* over X specifying the *degree* to which items exhibit the soft attribute corresponding to tag g:

$$\mathcal{L}(\phi_g; (D_u)_{u\in\mathcal{U}}) = \sum_u \sum_{\substack{i\in T_{u,g}\\ j\in T_{u,\overline{g}}}} \log(1 + e^{-\phi_g^\top(\phi_I(j)-\phi_I(i))}) + \frac{\lambda}{2}\phi_g^\top \phi_g. \tag{2}$$

This logistic pairwise loss is the same as that used in RankNet [11], hence we refer to this method as RankNet for CAVs.[7] Unlike the global logistic regressor, training with the RankNet loss finds a CAV ϕ_g—that predicts the *cardinal degree* $\phi_g^\top \phi_I(i)$ to which item i possesses attribute g—that attempts to ensure that each (individual) user's usage of tag

[6]For example, in the MovieLens20M data set we use below, there are more than 50 times as many ratings given as there are tags used. We evaluate the ratio of positive to negative samples in Section 6. In what follows, we assume that $T_{\overline{g}}$ has been suitably sub-sampled.
[7]Other loss functions—for instance, LambdaRank [12], exponential loss, or loss hinge [14]—may also be useful. We empirically examine LambdaRank in our experiments.

g is consistent with this degree. That is, if any user u applies g to i but not to j, this encourages the RankNet CAV to attempt to satisfy $\phi_g^\top \phi_I(i) > \phi_g^\top \phi_I(j)$. The regressor ϕ_g obtained serves as our CAV. Notice that ϕ_g is linear in the CF model's item embedding features ϕ_I.

Given a CAV ϕ_g, the degree to which an item i satisfies the induced attribute is given by the score $\phi_g(i) = \phi_g^\top \phi_I(i)$. The *quality* $Q(\phi_g; D)$ of a CAV on data set D is the fraction of the tag applications that it orders "correctly", i.e., if $i \in T_{u,g}$, $j \in T_{u,\overline{g}}$, then $\phi_g(i) \geq \phi_g(j)$:

$$Q(\phi_g; D) = \sum_u |\{(i,j) : i \in T_{u,g}, j \in T_{u,\overline{g}}, \phi_g(i) \geq \phi_g(j)\}| \,/\, |D|. \tag{3}$$

This is closely related to the pairwise loss above and is a key metric we use to evaluate CAVs, and can be used as a measure of a CAV's "usefulness" for, say preference elicitation, as we discuss below.

4.3 Nonlinear Attributes

Recall that any user u's representation $\phi_U(u)$ induced by the CF model is such that u's predicted rating for item i is $\hat{r}_{i,u} = \phi_U(u)^\top \phi_I(i)$. In other words, u's utility function is linear in the latent embedding space $X \subseteq \mathbb{R}^d$. A limitation of the linear approaches detailed in Section 4.2 is that the CAV ϕ_g for any tag g is also linear in the embedding space X. This implies that every user's utility for an attribute g—i.e., the degree to which utility changes as the degree of attribute g changes—is must be linear in X. For example, if the CAV for the tag 'violent' is linear in X, it implies the preference for any user u must be such that she prefers either *maximally* or *minimally* violent movies; she cannot prefer movies that are only "somewhat violent" and disprefer both movies that are more and less violent. For many soft attributes, this will not be the case—real-world preferences are often nonlinear (e.g., saturating [17]) and even non-monotone (e.g., single-peaked [40, 48]) with respect to many natural attributes. Such attributes are unlikely to be adequately represented linearly in embedding space X.

Fortunately, CAVs can also be applied to nonlinear DNN representations. We assume a two-tower/dual-encoder model with DNN N_I denoting the subnetwork that generates item embeddings. To remove the limitation of linear attribute utility, we extract CAVs *from the hidden layers* of the item DNN. For ease of exposition, we follow Kim et al. [26] by assuming that relevant concepts, if learned, can be uncovered within a *single* hidden layer of the (trained) deep collaborative filtering model, specifically, from N_I.[8] To find these CAVs, we assume the same positive and negative labels as in Section 4.2, but we change the example item representation: for item i, we use i's activation $\phi_{I,\ell}(i)$ at the ℓ'th layer of N_I as item i's training input representation instead of the item embedding $\Phi_I(i)$. Otherwise, the regressor is trained exactly as in Section 4.2, using either logistic regression or RankNet. The validity of the induced CAV can be tested in the same fashion as well.

The result is a regressor $\phi_{g,\ell}$ that can be applied to any item's representation in the intermediate "activation space" X_I^ℓ, where $\phi_{g,\ell}^\top \phi_{I,\ell}(i)$ captures the degree to which item i satisfies the attribute corresponding to tag g. The projection of this regressor through the last $L - \ell$ layers of N_I generates a (nonlinear) manifold in embedding space X. This gives considerably more flexibility to the relation between user utilities and soft attributes. In our nonlinear CAV experiments, we treat ℓ as a tunable hyper-parameter, and our reported results are based on the best CAVs extracted from these layers.

[8]Kim et al. [26] argue that TCAV is typically robust to this choice. We can apply TCAV to multiple layers simultaneously as well, a method we we employ in some of our experiments below.

4.4 Empirical Assessment of CAV Quality

We first evaluate our approach on synthetic data, which allows strict control over the generative process and direct access to ground truth labels (i.e., the degree of any attribute expressed by an item), as we explain below. We then test it on real-world data. For linear soft attributes, we train a collaborative filtering model $\Phi = (\phi_U, \phi_I)$ using *weighted alternating least squares (WALS)* [23], with the following regularized objective:

$$(\phi_U^*, \phi_I^*) \in \arg\min \sum_{u,i} c_{u,i}(\hat{r}_{u,i} - r_{u,i})^2 + \kappa(||\phi_U||^2 + ||\phi_I||^2). \tag{4}$$

Here $c_{u,i}$ is a confidence weight for the predicted rating $\hat{r}_{u,i} = \phi_U^\top(u;\theta_U)\phi_I(i;\theta_I)$, and $\kappa > 0$ is a regularization parameter. We select embeddings (ϕ_U^*, ϕ_I^*) using validation loss and an item-oriented confidence weight $c_{u,i} \propto n - \sum_u r_{u,i}$ (i.e., higher weight for less-frequently or lower-rated items).[9] For nonlinear attributes, we train a two-tower DNN embedding model with SGD/Adam [27]. Further details on synthetic data generation (A.1), the data set we use (A.2), and training methods we adopt (A.3) are provided in the appendix.

Synthetic Data. We refer to Appendix A.1 for a detailed description of the generative model used to construct our synthetic data sets.[10] At a high level, to generate synthetic data a generative model outputs both user-item ratings $\mathbf{R}$ and tag data $\mathbf{T}$ for $n = 25,000$ users and $m = 10,000$ items. Users and items are each represented by $d = 25$-dimensional embedding vectors, sampled from pre-defined mixture distributions to induce correlation in the data. For linear utility, user ratings are generated by first sampling items (giving a sparse ratings matrix $\mathbf{R}$) and then their ratings (noise added to the user/item dot product). For nonlinear utility, utility is the sum of nonlinear sub-functions (one per dimension) peaked at some (random) point and dropping as the item moves away from that peak. In both cases, users are more likely to rate items with higher utility.

To generate tags, five of the 25 latent item dimensions are treated as user-interpretable or “taggable,” each with a different tag. Intuitively, these correspond to dimensions in the latent space that are both “perceivable” and “interpretable” by users. Note, however, that the collaborative filtering model does not have access to the structure or details of the underlying generative process; it only sees the ratings and tag data so-generated. Each user u has a random *propensity to tag*, which influences the probability with which they will tag an item they have rated. Users are also more likely to tag higher-rated items—given their fixed tagging propensity, u is more likely to tag a item they have rated highly. For a given tag g, there is a *fixed* (non-subjective) threshold τ_g against which u evaluates an item to be tagged, and noisily applies g if that threshold is met.

CAV Accuracy. We evaluate CAV accuracy—how well the learned regressors capture the underlying soft attribute—using standard metrics of prediction quality with respect to user tag usage on held-out test data. The synthetic model also allows evaluation relative to the *ground-truth item representations and attribute levels of each tag* since we have access (for purposes of evaluation) to the *true latent attribute values of each item* as generated by the model. Indeed, this is the main reason we include synthetic results since it provides a definite assessment of our techniques. We evaluate three training methods, binary logistic regression, RankNet [11], and LambdaRank [12]. We split the tag data $\mathbf{T}$ into training and test sets and use the following metrics to measure the accuracy of CAVs on the test set:

[9]CAVs on MovieLens20M data with linear attributes use embeddings that were learned (via WALS) using internal production code, which is not releasable. However, we have made available the resulting WALS embeddings. (While results may vary with different embeddings, our methods are agnostic to the precise method used to generate them). These can be found at https://storage.googleapis.com/gresearch/attribute_semantics/cf_embeddings.npz.
[10]The code for the generative model used is available at https://github.com/google-research/recsim_ng/tree/master/recsim_ng/applications/cav_synthetic_model. It is used at several other points in the paper.

Utility CAV Model	**Linear** **Lin-Emb** Accur	 Sprmn	**Linear** **Lin-Emb** Accur	 Sprmn	**NonLinear** **NL-Emb** Accur	 Sprmn
Log. Regr.	0.906	0.569	0.889	0.565	0.922	0.577
RankNet	**0.968**	0.674	0.943	**0.670**	**0.978**	**0.686**
LambdaRank	0.961	**0.679**	**0.947**	0.666	0.974	0.680
PITF	0.683	0.056	0.707	0.070	N/A	N/A

Table 2. CAV Evaluation, Synthetic Data (Non-subjective)

CAV Model	**Lin-Emb** Accur	**NL-Emb** Accur
Log. Regr.	0.727	0.745
RankNet	0.803	**0.820**
LambdaRank	**0.804**	0.818
PITF	0.715	N/A

Table 3. CAV Evaluation, MovieLens

(i) *Accur*, the mean accuracy of the logistic model or *quality* $Q(\phi_g; D)$ of the ranking model;

(ii) *Sprmn*, the *Spearman rank correlation coefficient* between predicted and ground-truth attribute values.[11]

The latter compares the *ranking* of all items by their *ground-truth* attribute value with the ranking induced by the attribute's *learned CAV*. This metric applies equally to "nonlinear" latent attributes, since the CAV still provides a precise ordering of all items according to the attribute.

Synthetic Results. Table 2 shows the predictive performance of our CAVs on synthetic data for three different combinations of user data model and general CAV approach: (i) user utility is linear and we train linear CAVs; (ii) user utility is nonlinear but we train linear CAVs (Lin-Emb); and (iii) user utility is nonlinear and we train nonlinear CAVs (NL-Emb). We list the accuracy metrics above, where CAVs are trained using either logistic regression, RankNet or LambdaRank, with results averaged over the five tags. We bold the best value within each of the three settings. We see that the CAVs predict user tagging behaviors reasonably accurately as indicated by *Accur*, and reliably order test items according to the ground-truth value of the underlying soft attribute, as evidenced by *Sprmn*, despite the noise in the tagging process. The ranking methods, RankNet and LambdaRank, dominate logistic regression with respect to both *Accur* and *Sprmn*, which suggests that accounting for variation in user tagging behavior is important (we discuss this further when describing subjective results in the next section). For nonlinear utilities, we also compare the best "linear" CAV (extractable from the output embedding) with the best nonlinear CAV (extractable from DNN hidden layers). While linear CAVs may be viewed as conceptually simpler, they are mismatched to the underlying attributes when user utility is nonlinear. Unsurprisingly, nonlinear CAVs outperform their linear counterparts, demonstrating the value of seeking nonlinear (or "distributed") attribute representations within the DNN, and the power of TCAV to help interpret them. We also include a baseline tag recommender *PITF (pairwise interaction tensor factorization)* [50], which uses tensor decomposition to model pairwise interactions between users, items, and tags (see more details in the appendix)—note that it produces linear representations. Its tag prediction accuracy is worse than that of the CAV approaches. *PITF* does not use ratings which may be valuable in training personalized representations with sparse tag data.

MovieLens Results. We also evaluate our methods on the more realistic MovieLens20M data set [20]. In contrast to the synthetic results—which are most useful for evaluating the *effectiveness* of our methods in controlled settings—evaluation on real-world data sets like MovieLens20M, and the rater-based evaluation in Section 6, are more suitable for testing their *applicability* "in the wild." MovieLens20M comprises 20M movie ratings (1–5 scale with half-star increments) and 465K tag-instances applied to 27K movies by 138K users. Tags are user-generated strings that describe various types of movie attributes. These can include attributes such as genres like 'sci-fi', 'comedy', 'action'; descriptions like 'emotional', 'funny', 'atmospheric'; and themes like 'zombies', 'world war 2', 'cyberpunk'. We split rating and tag data into train and test sets such that *all examples* for any specific user-item pair are present in exactly one of these subsets (i.e., if u rated and tagged i, the rating and *all tags g applied by u to i* are all in either the training set or all in the

[11]We also computed precision and recall results, but these are omitted since they simply corroborate the findings generated by these metrics.

test set).[12] We generate $d = 50$-dimensional user and item embeddings: we use WALS as our collaborative-filtering method in the case of linear CAVs, and train two-tower DNNs for non-linear CAVs.

Positive examples for tag g are user-item pairs to which g has been applied; negatives are those tagged by that user, but not with g (recall that we use negative examples of g for user u *only if* u has positively applied g to at least one item). Given the sparseness of tags—of 30745 unique tags, more than 28000 are applied to fewer than 10 unique movies—to ensure that the analyzed tags are relevant to sufficiently many user-item interactions, we train CAVs only for those tags that are among the 250 most frequently applied both by unique users and to unique items, which results in 164 tags. Table 3 shows test accuracy for linear and nonlinear CAVs. As in the synthetic settings, the ranking methods outperform logistic regression, which again hints at some subjectivity (see next section). While we cannot measure Spearman correlation (since we have no ground truth ranking), nor control the form of user utility, we see that nonlinear CAVs perform slightly better than linear CAVs, suggesting that user preferences for at least some MovieLens tags are nonlinear in their embedding-space representation. Again, as in the synthetic settings, our CAV methods consistently outperform PITF.

5 IDENTIFYING SUBJECTIVE ATTRIBUTES

If users largely agree on the usage of tags, it is reasonable to treat the semantics of a tag as a *single* soft attribute or CAV as we do above. But in many cases, different users may have different "senses" in mind when they apply a tag. For example, one user may use the term 'funny' to describe movies that are silly, involving, say, physical or slapstick humor, while another may use the same term to refer to dry, political satire.[13] While correlated, these two *tag senses* will order movies quite differently. Such *sense subjectivity* may hinder our ability to produce an accurate CAV and understand a user's true intent. We now turn to this issue and show how to learn *personalized semantics* for tags.

5.1 Subjectivity of Degree

As discussed above, *degree subjectivity* is likely to emerge quite naturally. The use of *intra-user pairwise comparisons* with a ranking loss in CAV training (see Eq. (2)) ensures that the induced CAVs are robust to this form of subjectivity. However, since two users may use a tag g differently if they have different thresholds for applying g, interpreting user u's usage (e.g. statements such as "that's not funny" or "I'm in the mood for something funny") requires a *personalized semantics* that is sensitive to her threshold. Let g be a tag that is degree subjective, ϕ_g be g's CAV and $\phi_g(i)$ be the degree to which item i satisfies ϕ_g. A *user-specific threshold* $\tau_g^u \in \mathbb{R}$ determines a user-specific semantics for g: tag g applies (typically, noisily) to item i for user u only if $\phi_g(i) \geq \tau_g^u$. Equivalently, this can be viewed as a *personal* linear separator for u in X, but constrained to be orthogonal to the direction ϕ_g induced from the *population* labels. The (estimated) *optimal personal threshold* τ_g^u minimizes the number of misclassifications with respect to u's usage of g:

$$\tau_g^u \in \arg\min_{\tau} |\{i \in T_{u,g} : \phi_g(i) \geq \tau\} \cup \{i \in T_{u,\bar{g}} : \phi_g(i) < \tau\}|. \tag{5}$$

Among the continuum of such minimizers, the threshold τ_g^u that maximizes the margin between the nearest bracketing positive and negative items is a natural choice for u's personal semantic threshold. Of course, since tag usage by an individual user is typically extremely sparse, these thresholds are likely to be noisy—this is one reason that we do not compute CAVs themselves are on a per-user basis. However, we can refine u's semantics with well-chosen queries, e.g.,

[12]The resulting data set, with our train-test splits, is available at https://github.com/google-research/google-research/tree/master/attribute_semantics/data.
[13]Some users may offer more "refined" tags to distinguish their intended sense, but many will not. For this reason, we still require the ability to disambiguate a user's intended meaning in the general case.

"Do you consider movie m to be violent?" This can be used to implement a loose binary search for an approximate threshold (possibly made robust to account for noisy responses), but we defer this to future research. If usage is correlated within user sub-populations, generalization of thresholds across users is also viable, as we discuss in the case of sense subjectivity below.

5.2 Subjectivity of Sense

We now turn to *sense subjectivity*. We can readily detect sense subjectivity and assign a *personalized semantics* for a tag g to different (groups of) users, using *distinct CAVs* for each tag sense. We assume that g has *at most* s_g distinct senses $g[1], \ldots, g[s_g]$, for some small positive integer s_g, where each sense denotes a different soft attribute (we discuss their relation below). Moreover, each user adopts exactly *one* such sense of g.[14] We propose a method to discover whether a tag has multiple senses, and to uncover suitable CAVs for each sense if so.

Intuitively, if the CAV quality metric $Q(\phi_g; D)$ is high, then CAV ϕ_g does a good job of explaining usage of tag g among all users in data set D. If not, then model Φ is unlikely to have learned a good representation for g. This could be due to g being poorly correlated with user ratings (hence, user preferences), or because g has multiple senses. In the latter case, if g has s senses, there should be some *user-partitioning* of D into subsets $D_1, \ldots, D_s$ of users such that there is a CAV $\phi_{g,k}$ with high quality $Q(\phi_{g,k}; D_k)$ for each $k \leq s$. We first propose a simple scheme to find a good set of CAVs for a fixed s, then discuss determination of a suitable number of senses $s \leq s_g$.

Assume a fixed number of target senses s and a given data set D. We propose a simple scheme for generating s CAVs corresponding to maximally distinct senses for tag g. Let $\Sigma = \{\sigma_1, \ldots, \sigma_s\}$ be a partitioning of users into s clusters, where σ_k is the set of users that, we assume for the time being, adopt a common sense for g. Let D_k be the restriction of D to tag data generated by users $u \in \sigma_k$. For a fixed Σ, we can readily generate a CAV $\phi_{g,k}$ for each data set D_k capturing the corresponding sense, and measure its quality $Q(\phi_{g,k}; D_k)$. Of course, this quality depends on whether the partitioning Σ is sensible (i.e., whether most users in each cluster use g similarly). If the quality of these CAVs is low, one can repartition users by "assigning" each user u to the cluster in Σ whose CAV best explains her tag usage:

$$k_u^* = \arg\max_k |\{(i,j) : i \in T_{u,g},\ j \in T_{u,\overline{g}},\ \phi_{g,k}(i) \geq \phi_{g,k}(j)\}|. \tag{6}$$

This leads to an EM-like alternating optimization procedure [5] for finding a good clustering that repeatedly applies the following steps:

(a) Learns a CAV $\phi_{g,k}$ for tag g for each current (user) cluster k using one of our methods above;
(b) Reconstructs the clusters by assigning each user to the cluster whose new CAV $\phi_{g,k}$ best explains her usage of tag g per Eq. 6; i.e., $D_k \longleftarrow \{u \in \mathcal{U} : k_u^* = k\}$.

The iterative process proceeds until the partitioning Σ no longer changes or quality improvements become sufficiently small. Since we use a hard clustering scheme and only change Σ when the new clusters yield strictly better quality, it is easy to see that the EM procedure terminates in a finite number of steps. If we assign each u by minimizing the logistic or ranking loss incurred by u, using convergence properties of the standard k-means algorithm (e.g., [7]), one can show that the procedure converges to a local minimum and generates s distinct CAV senses.[15]

Assuming that the complexity of the regression/ranking sub-problem is linear in number of examples N, with L iterations of EM, the complexity of this approach is $O(LNs)$. Empirically the EM method converges very quickly ($L \leq 5$)

[14] We defer to future work the possibility of *mixtures* of senses.
[15] A number of variants of this general approach can be explored (but are beyond the scope of this work): various criteria for initialization (e.g., using ICA for initial clustering); hard vs. soft clustering; various termination criteria; etc.

Utility CAV Model	**Linear** **Lin-Emb**		**NonLinear** **Linear-Emb**		**NonLinear** **NL-Emb**	
	Accur.	Sprmn	Accur.	Sprmn	Accur.	Sprmn
Log. Regr.	0.872	0.566	0.860	0.523	0.886	0.548
RankNet	0.960	**0.671**	**0.947**	**0.660**	**0.961**	0.680
LambdaNet	**0.962**	0.669	0.938	0.653	0.958	**0.684**
PITF	0.700	0.064	0.708	0.068	N/A	N/A

Table 4. CAV Evaluation, Synthetic Data (Degree subjectivity)

	Subjective Tags						**Objective Tags**					
Utility CAV Model	**Linear** **Lin-Emb**		**NonLinear** **Lin-Emb**		**NonLinear** **NL-Emb**		**Linear** **Lin-Emb**		**NonLinear** **Lin-Emb**		**NonLinear** **NL-Emb**	
	Accur.	Sprmn	Accur.	Sprmn	Accur.	Sprmn	Accur.	Sprmn	Accur.	Sprmn	Accur.	Sprmn
Log. Regr.	0.643	0.039	0.634	0.026	0.616	0.036	0.936	0.634	0.926	0.642	0.922	0.635
EM Log. Regr.	0.833	**0.478**	0.796	0.419	0.828	0.476	0.937	0.631	0.926	**0.637**	0.922	0.635
RankNet	0.615	0.042	0.606	0.035	0.603	0.022	**0.992**	**0.636**	**0.987**	0.636	**0.982**	**0.636**
EM RankNet	**0.922**	0.466	**0.915**	**0.468**	**0.908**	**0.468**	**0.992**	0.633	**0.987**	0.633	**0.982**	**0.636**
LambdaRank	0.615	0.028	0.606	0.036	0.604	0.009	**0.992**	0.634	**0.987**	0.632	**0.982**	0.634
EM LambdaRank	0.920	0.458	**0.915**	0.465	**0.908**	0.465	**0.992**	0.631	**0.987**	0.630	**0.982**	0.631
PITF	0.672	0.063	0.711	0.070	N/A	N/A	0.640	0.050	0.675	0.074	N/A	N/A

Table 5. CAV Evaluation, Synthetic Data (Sense Subjectivity): Subjective Tags (left half), Objective Tags (right half).

in both our MovieLens and the synthetic data experiments below, so the computational cost of the EM algorithm is minimal (and recall that these are offline computations).

We can search for the appropriate number of senses—effectively implementing a form of *model selection* [5]—by starting with an initial (single) CAV, and applying the procedure above to gradually increasing number of clusters $s = 2, 3, \ldots, s_g$, and terminating once the improvement in average quality, $\sum_k (|D_k|/|D|) Q(\phi_{g,k}; D_k)$ is negligible.[16]

Notice that our approach to disentangling tag senses is in the style of top-down "disaggregative" or *divisive* [33]. We do this since bottom-up agglomerative clustering [33] is likely to be very noisy—the tag set of any individual user is extremely sparse, so attempts to produce a CAV for very small groups of users will generally be unreliable. Once we find the appropriate number of senses and corresponding CAVs, in practice, we can continue to update them as new users, items and tagging data arises. For example, given a new user u who has applied tags to items, we can associate a "personal" semantics for u with a given tag g by assigning u to the g-cluster whose CAV best fits u's usage of data by computing k^*_u.

5.3 Empirical Assessment

As above, we test our methods for subjective soft attribute identification on both synthetically generated data with ground truth labels, and on the MovieLens20M data set.

Synthetic Results. We again test our approach on synthetic data to exploit access to ground truth CAV semantics for our tags for purposes on evaluation. We generate several data sets incorporating both degree and sense subjectivity, as well as linear and nonlinear soft attributes. We refer to App. A.1 for a detailed description of the generative model used to construct these data sets with $n = 25,000$, $m = 10,000$, and $d = 25$. The model is similar to that used in Section 4.4, differing only in the addition of subjectivity to the user tagging behavior.

To test degree subjectivity, we use five tags as above, but with each user's personal tagging threshold sampled from a mixture distribution with two components. To test sense subjectivity, we introduce a subjective tag "tag-S" with three

[16] A number of other more complex approaches can be considered.

senses, each reflecting one of three (of the five) taggable dimensions. As above, each of these is associated with a specific semantics in embedding space. Each user u adopts exactly one of these three senses; specifically, when applying tag-S to any item, they assess that it based on the specific sense of that tag (of the three) to which they have been assigned. The remaining two tags are objective. As above, ratings and tags are applied in a noisy fashion to generate the synthetic data set.

We evaluate the three CAV training methods used in Section 4.4, applying each to a linear (WALS) model and a nonlinear two-tower model as needed. For sense subjectivity, we test our EM-like algorithm with each training method. Further details on data generation and the experimental set up can be found in the appendix.

Table 4 summarizes performance of the CAVs under degree subjectivity, using the same methods and models as in Section 4.4 (results averaged over the five degree-subjective tags). In contrast to the non-subjective case in Section 4.4, where users have the same threshold for each tag, here the per-user ranking-based methods (RankNet and LambdaRank) significantly outperform logistic regression, demonstrating the need to be sensitive to a user's degree subjectivity.

Table 5 summarizes results for sense subjectivity, showing CAV accuracy for our baseline methods both with and without our EM-based approach for distinguishing senses. The left side of the table shows results for the sense-subjective tag-S, demonstrating that our EM procedure can dramatically improve CAV accuracy by reliably disentangling the three distinct senses of tag-S. This demonstrates that treating a subjective concept as if it were objective can be problematic, leading to erroneous interpretations of any given user's intent when using that tag. Note also that our ranking methods perform better than logistic regression (*EM-RankNet* and *EM-LambdaRank* perform similarly). The right side of the table shows CAV accuracy on the two *objective* tags. We see that the EM and non-EM methods perform almost identically. This is important since it suggests our techniques are unlikely to identify spurious subjective senses of a tag if in fact tag usage is objective. Somewhat intriguingly, we see that the use of nonlinear CAVs does not offer much improvement over linear CAVs with ranking methods, though nonlinear CAVs perform better when trained using logistic regression. The performance of the PITF baseline is worse than that of the CAV approaches in both the degree and sense subjectivity experiments.

MovieLens Results. We also evaluate our subjective CAV methods on MovieLens20M. To assess degree subjectivity, we select 13 tags that we expect to exhibit various degrees of subjectivity and compare the accuracy of different CAV methods for each (the tags are listed in the top row of Table 6). Since MovieLens data has no ground truth with respect to possible subjectivity, Spearman rank correlation cannot be measured, hence we focus on CAV predictive accuracy only. Table 6 shows Accuracy measures for the linear and nonlinear variants of our three primary methods for the 13 MovieLens tags. Generally, our ranking methods outperform logistic regression, suggesting that real users exhibit some variation in their thresholds (degree subjectivity). We also see that some tags (e.g., *sci-fi*) have much higher agreement and CAV-predictability across users than others (e.g., *funny*), hence arguably less degree subjectivity. In particular, of the 13 tags, across all models, *funny*, *surreal*, *quirky* and *atmospheric* seem to be most subjective, while *sci-fi*, *action* and *classic* are the most objective.

We also observe differences in the degree of improvement offered by nonlinear CAVs vs. linear CAVs across the tags: those with larger improvements (e.g., *dark comedy*, *dystopia*) suggest that user utility may often be nonlinear in the degree of that attribute (so extreme degrees may not be preferred); while those where nonlinear CAVs perform no better, or even worse (e.g., *sci-fi*, *action*, *funny*), may be most preferred at their maximum or minimum degree.

We have no labeled data in MovieLens20M with which to assess sense subjectivity. So to evaluate this form of subjectivity, we construct two types of *artificial tags* from MovieLens data. First are *objective tags*, capturing four

	sci-fi	atmospheric	surreal	twist ending	action	funny	classic	dark comedy	quirky	psychology	dystopia	stylized	thought-provoking
Log. Regr.	0.831	0.721	0.739	0.705	0.823	0.689	0.818	0.714	0.725	0.715	0.737	0.753	0.764
RankNet	0.905	0.793	0.811	0.817	**0.899**	**0.775**	0.877	0.822	**0.840**	0.812	0.850	0.866	0.834
LambdaRank	**0.906**	0.784	0.788	0.869	0.876	0.605	0.838	0.830	0.821	0.788	0.867	0.809	0.839
NL. Log. Regr.	0.831	0.725	0.742	0.711	0.812	0.681	0.821	0.705	0.751	0.704	0.807	0.811	0.764
NL. RankNet	0.893	0.838	**0.827**	0.854	0.890	0.754	**0.888**	0.868	0.833	**0.837**	0.882	0.856	0.844
NL. LambdaRank	0.891	**0.843**	0.807	**0.875**	0.880	0.744	0.865	**0.891**	0.825	0.796	**0.921**	**0.898**	**0.856**

Table 6. CAV Accuracy Evaluation, 13 Possible Subjective Concepts in MovieLens

genres (comedy, horror, fantasy and romance). For a random subset of user-item pairs in the tag data set, we add the corresponding user-item-genre triple to the set if the item's meta-data lists that genre. This ensures that the new "genre tag" data replicates natural tagging patterns. We also add a synthetic tag *odd year*—was a movie was released in an even or odd year—to 50% of user-item pairs. This (presumably) *preference-irrelevant* attribute acts as a baseline for which no good CAV should be discoverable. These tags are "objective"—their presence does not depend on a user's interpretation of tags (though they may depend on a user's inclination to tag certain types of movies).

The second artificial tag type are *sense-conflated tags*, constructed by coalescing several related "ground" tags into a single "meta-tag" then replacing each ground tag with that meta-tag. Each ground tag in the group can be viewed as a subjective sense of the meta-tag. We test our ability to "disentangle" the different senses of the meta-tag relative to the ground truth. We introduce four meta-tags:

- Meta-tag *monsters*: groups ground tags *zombies*, *ghosts* and *vampires*;
- Meta-tag *funny*: groups ground tags *parody*, *satire*, and *dark humor*;
- Meta-tag *intrigue*: groups ground tags *corruption*, *conspiracy*, and *politics*; and
- Meta-tag *relationship*: groups ground tags *family*, *friendship*, and *love story*.

For each user and meta-tag, we choose *exactly one* ground tag from the group as that user's *designated sense* and add the meta-tag to the data set for each user-item-tag triple that uses the ground tag. For example, some users have all of their *friendship* tags replaced with the meta-tag *relationship*, while others have all of their *love story* tags replaced this way. In this way, we have access to subjective tags that follow the natural tag application patterns in the MovieLens data, but with a ground-truth partitioning of users and (partial) item labeling according to the different senses of the meta-tag.

Table 7 summarizes the accuracy of our trained CAVs for these two types of artificial tags. For the four "objective" *genre* tags, the ranking-based methods outperform logistic regression. However, adding EM to our ranking methods provides only modest incremental benefit (especially relative to lift it offers for the "subjective" sense-conflated tags below). This implies that *genres* exhibit at most modest, if any, subjectivity of sense, as expected.[17] The 'horror' and 'fantasy' tags are the easiest to learn, suggesting they are more "objective" and "linear."

None of our methods uncovers a good CAV for the artificial tag *odd year*—their predictive accuracy is barely above random. This is an important finding, since it corroborates our hypothesis that CAVs are useful for identifying *preference-related* attributes/tags. On the other hand, results on our four *sense-conflated* tags clearly demonstrate the ability of our EM-style approach to disentangle the distinct subjective or personal senses of each of the meta-tags—this is true for each baseline algorithm—thus greatly improving tag prediction accuracy.

[17]EM will often provide some improvement in accuracy, even if distinct senses do not exist, by allowing some overfitting. In this case, EM may also overcome the limitations of the linear CAVs if user utilities for these objective genres are non-linear.

	Odd Year	Comedy	Horror	Fantasy	Romance	Monsters	Funny	Intrigue	Relationship
Log. Regr., Lin. Emb.	0.519	0.521	0.770	0.685	0.693	0.671	0.658	0.669	0.662
EM Log. Regr., Lin. Emb.	0.532	**0.685**	0.759	0.744	0.704	0.831	0.769	0.712	0.730
RankNet, Lin. Emb.	0.505	0.620	0.790	0.778	0.730	0.718	0.705	0.660	0.634
EM RankNet, Lin. Emb.	**0.593**	0.676	0.833	**0.824**	0.749	**0.892**	**0.874**	0.834	0.840
LambdaRank, Lin. Emb.	0.533	0.609	0.809	0.779	0.716	0.719	0.718	0.661	0.623
EM LambdaRank, Lin. Emb.	0.582	0.670	**0.838**	0.819	**0.762**	0.883	0.870	**0.836**	**0.847**

Table 7. CAV Accuracy Evaluation, Artificial MovieLens Tags (5 objective, 4 sense-conflated)

6 CAV EVALUATION WITH RATER DATA

Our evaluation of CAV semantics in the preceding sections used two distinct approaches. Evaluation using synthetic data allows assessment relative to fully known, ground-truth labels (i.e., the degree of a soft attribute exhibited by each item is known), but does not capture directly how real users of recommender systems use or interpret the terms or tags associated with these attributes. Our second approach, using MovieLens20M tagging behavior, does reflect actual user tag usage, but requires that we make some assumptions (e.g., negative tag imputation) about tag usage to draw specific inferences about the semantics of the underlying soft attributes. While both allow meaningful conclusions to be reached, neither reflect the ideal evaluation in which users explicitly tell us whether they believe one item exhibits more or less of a soft attribute than another.

In this section, we evaluate CAVs using precisely such rater data. Specifically, we use the *SoftAttributes* data set [2],[18] in which raters explicitly compare movies based on the degree to which they exhibit specific attributes (we outline the precise nature of this data set below). This provides explicit pairwise comparisons by human raters against which to evaluate the CAVs learned from tagging or comparison data, and does not require any assumptions about usage, or any form of imputation or down-sampling. We first describe the *SoftAttributes* data set in Section 6.1. In Section 6.2, we use this rater data to evaluate the predictive accuracy of the CAVs we trained (see above) using MovieLens tags to provide an independent assessment of those CAVs that does not rely on the assumptions used to train the model. In Section 6.3, we train a new set of CAVs directly on the *SoftAttributes* data set itself to determine if CAVs trained using *explicit* (rather than imputed) soft-attribute comparisons have greater predictive accuracy. This latter evaluation also demonstrates the application of CAVs to tagging/attribute data generated in a different fashion (since the *SoftAttributes* tag vocabulary is extracted from user reviews rather than explicit tagging of items).

6.1 The *SoftAttributes* Data Set

Balog et al. [2] provide one of the first detailed investigations of the semantics of soft attributes in recommender systems, with an emphasis on the weaknesses of typical binary tagging-based approaches, including the fact that items often exhibit attributes to varying degrees, as well as the contextual, subjective nature of attribute usage. Apart from algorithmic contributions (which we use as a benchmark below), Balog et al. [2] also created the *SoftAttributes* data set.

The data set itself is generated using an intuitive process designed to extract robust (and personal) soft-attribute comparisons from raters. A rater r is first asked to indicate a subset of "familiar" movies (i.e., movies they have seen) from a pool of popular movies (namely, the 300 most popular movies in MovieLens20M). The rater r is then presented with a specific soft attribute/tag g (e.g., 'violent'), an anchor movie $m_{r,g}$ and a set of 10 candidate movies $C_{r,g} = \{c_{i,r,g} : i \leq 10\}$, where the anchor and candidates are all drawn from r's familiar set. Rater r is then asked, using an intuitive graphical interface, to specify, for each candidate $c_{i,r,g}$, whether it is more, less, or about the same as the anchor $m_{r,g}$ w.r.t. g (e.g., whether $c_{i,r,g}$ is more violent than $m_{r,g}$, less violent than $m_{r,g}$ or about the same as $m_{r,g}$ in terms of degree of violence).

[18]See `https://github.com/google-research-datasets/soft-attributes`

This divides the 11 movies $C_{r,g} \cup \{m_{r,g}\}$ into three equivalence classes: those movies $L_{r,g}$ that are "less g" than the anchor $m_{r,g}$, those $M_{r,g}$ that are "more g," and those $S_{r,g}$ that are about the same. This is repeated for each of 60 soft attributes across 100 raters.[19] The resulting data set consists of approximately 250K pairwise comparisons of movies over these 60 soft attributes.

We use the data set in a similar fashion to Balog et al. [2] with three classes of comparisons. For any fixed rater r and tag g, we say:

- $c \succ^{s}_{r,g} c'$ if $c \in M_{r,g}$ and $c' \in L_{r,g}$. These are *strong differences*, since c is more g than $m_{r,g}$ and c' is less g, so c is "two levels" more g than c'.
- $c \succ^{w}_{r,g} c'$ if $c \in M_{r,g}$ and $c' \in S_{r,g}$; or if $c \in S_{r,g}$ and $c' \in L_{r,g}$ These are *weak differences*, since c is only "one level" more g than c'.
- $c \approx^{s}_{r,g} c'$ if $c, c' \in M_{r,g}$ or $c, c' \in S_{r,g}$ or $c, c' \in L_{r,g}$. These are *indifferences*, since c and c' are roughly indistinguishable w.r.t. g.

Notice that these ordinal differences may be sensitive to the choice of anchor and candidates, hence one might ignore the distinction between weak and strong differences. However, since we compare our results to those of Balog et al. [2], who use this distinction, we do the same in this section.

The resulting *SoftAttributes* data set consists of approximately 250K pairwise comparisons of movies over these 60 soft attributes and has a number of appealing properties. Of note is the fact that, since each rater is asked to evaluate the same 60 attributes, it does not suffer from tag popularity bias, in contrast to MovieLens20M. However, subjectivity may be present in some attributes as raters may disagree on their usage.

Of the 60 attributes used in the *SoftAttributes* data set, only 36 are found (frequently) in the MovieLens20M data set. When evaluating CAVs trained using MovieLens tags, we do so only on these 36 in-common attributes. [20]

6.2 Evaluation of CAVs using MovieLens Data

We begin by evaluating how well CAVs, trained using the MovieLens20M data set, can predict the assessments of *SoftAttributes* raters. These CAVs are trained exactly as in the preceding sections, using collaborative filtering on ratings to generate user and item embeddings, and then using tag data to infer CAVs with our negative-label imputation scheme. We use the same procedures as above, namely, two-tower collaborative filtering and various methods for CAV training (logistic regression, RankNet and LambdaRank) using both straightforward objective methods and EM-based methods for sense subjectivity. The only difference with our earlier models is that our user/item embedding dimension is $d = 128$ in the collaborative filtering model. We use the larger embedding space to provide somewhat better recommendation performance and more nuance in the constructed item embeddings.

We evaluate the accuracy of our CAVs on each of the 36 in-common soft attributes/tags on the movies in the *SoftAttributes* data set. Apart from evaluating CAVs using more precise human-rater data, this also provides an informal assessment of the degree to which CAVs trained on tag data generated by one user population under specific conditions generalize to attribute usage by a different population under different conditions.

Given some attribute g with its CAV ϕ_g, and an arbitrary pair of movies (i, j) with item embeddings $\phi_I(i)$ and $\phi_I(j)$, respectively, the sign of $\cos(\phi_I(i), \phi_g) - \cos(\phi_I(j), \phi_g)$ determines the CAV assessment of the relative degree of attribute

[19]We refer to Balog et al. [2] for details on how their soft attributes are generated and selected. Of note is the fact that they are extracted from movie reviews in an Amazon reviews corpus rather than from explicit tagging of items.

[20]The in-common attributes are: animated, artsy, believable, big budget, bizarre, boring, cheesy, complicated, confusing, dramatic, entertaining, factual, funny, gory, harsh, incomprehensible, intense, interesting, long, original, over the top, overrated, pointless, predictable, realistic, romantic, scary, unrealistic, violent, cartoonish, exaggerated, light-hearted, mainstream, mindless, terrifying, and sappy.

g for this pair, with a positive sign indicating $i \succ_g^{\phi_g} j$ and a negative sign indicating $j \succ_g^{\phi_g} i$.[21] Following [2], we compare these predictions against the rater assessments in *SoftAttributes* using the *extended Goodman-Kruskal gamma rank correlation coefficient* [19] G'. Specifically, for any attribute g, let a *weak (resp., strong) disagreement* with the CAV ϕ_g be any r, i, j triple where $i \succ_g^{\phi_g} j$ but $j \succ_{r,g}^{w} i$ (resp., $j \succ_{r,g}^{s} i$). Weak and strong agreements are defined analogously. Let N_s be the the number of weak agreements of ϕ_g with the data (w.r.t. g, we suppress g in the notation), N_{ss} the number of strong agreements, N_d the number of weak disagreements, and N_{dd} the number of strong disagreements. The extended Goodman-Kruskal gamma rank correlation coefficient (for a given attribute g) is defined as:

$$G'_g = \frac{N_s - N_d + 2(N_{ss} - N_{dd})}{N_s + N_d + 2(N_{ss} + N_{dd})} \ .$$

Note that G'_g is always between -1 (perfect anti-correlation) and $+1$ (perfect correlation). The "double" weight of strong agreements and disagreements is a simple extension of the Goodman-Kruskal metric adopted by Balog et al. [2]. We use G' to denote aggregate rank correlation across *all attributes* where the agreement and disagreement counts are aggregates (effectively, a weighted average of G'_g over all g).

When building models that incorporate sense subjectivity, we apply our EM procedures as above, computing distinct CAVs for each uncovered sense of a specific attribute g. We limit the number of senses for any attribute g to 10, but use a simple model selection criterion to select the precise number of senses. Specifically, we increase the number of senses from one (no subjectivity) by adding one additional sense at a time: if the improvement in average CAV quality when increasing from k to $k+1$ senses falls below a fixed threshold, we terminate with k senses, but stop at a maximum of $k = 10$ senses if this threshold is not met at any stage. Since the users who apply tags in the MovieLens data differ from the raters evaluating attributes in the *SoftAttributes* data set, we assign each *SoftAttributes* rater to the attribute sense that best explains their ranking data for that attribute.

For reference, we also report the results of two methods proposed by Balog et al. [2] for determining the semantics of soft attributes, weakly-supervised weighted dimensions+term based-item-centric (WWD+TB-IC) and weakly-supervised weighted dimensions+term based-item-centric (WWD+TB-RC). While these methods are trained using different data sets than our use of MovieLens tags—they aggregate Amazon movie reviews joined with MovieLens [62] to obtain attribute labels for movies—they are similar in these sense that they also use binary attribute labels rather than rater comparison data (as we discuss in the next section). Both methods fit a logistic regression model using item embeddings of dimension $d = 25$. We refer to the original paper for further details.

Table 8 shows the performance of CAVs trained with MovieLens tag data using various techniques, specifically, the Goodman-Kruskal rank correlation with *SoftAttributes* rater data. CAVs generated by all 12 methods examined exhibit a positive correlation with raters' attribute assessments of the 36 attributes, showing reasonable generalization of the semantics across the two different groups (and types) of users/raters. Our logistic regression methods (with both linear and nonlinear attribute embeddings) attain somewhat higher accuracy than both WWD+TB-IC and WWD+TB-RC. Apart from better performance by CAVs, this also suggests that the use of the tag data and our technique for negative data imputation has value for learning the semantics of soft attributes. Setting aside the EM methods designed for sense subjectivity for the moment, we see that each method trained using nonlinear embeddings achieves a slightly better result than its linear counterpart, a result consistent with those in Table 3 above, suggesting that the preferences of raters for at least some soft attributes are nonlinear in their embedding-space representations.

[21] We find that the use of cosine similarity gives a slightly better results than the use of dot products due the inherent normalization it provides. But results using dot products are qualitatively similar.

CAV Training Method	Gamma Rank Correlation
WWD+TB-IC	0.194 [2]
WWD+TB-RC	0.200 [2]
Logistic Regression, Lin. Emb.	0.226
EM Logistic Regression, Lin. Emb.	0.336
RankNet, Lin. Emb.	0.199
EM RankNet, Lin. Emb.	0.388
LambdaRank, Lin. Emb.	0.191
EM LambdaRank, Lin. Emb.	0.379
Logistic Regression, NL Emb.	0.238
EM Logistic Regression, NL Emb.	0.329
RankNet, NL Emb.	0.216
EM RankNet, NL Emb.	0.339
LambdaRank, NL Emb.	0.203
EM LambdaRank, NL Emb.	0.371

Table 8. CAV Accuracy Evaluation, Training with MovieLens

CAV Training Method	Number of Senses (Count, Average)					
	1	2	3	4	5	Average
EM Logistic Regression, Lin. Embs.	21	7	5	0	3	1.806
EM RankNet, Lin. Emb.	4	18	8	6	0	2.444
EM LambdaRank, Lin. Emb.	5	14	12	3	2	2.528
EM Logistic Regression, NL Embs.	20	6	4	4	2	1.944
EM RankNet, NL Emb.	10	18	6	2	0	2.0
EM LambdaRank, NL Emb.	5	19	5	5	2	2.444

Table 9. Number of CAV Senses Discovered by Various EM Methods (MovieLens 20M). Displayed for each method: (a) how many of the 36 attributes have been assigned the specified number of senses (1–5), and (b) the average number of senses over all 36 attributes.

When assessing the role of subjectivity, we first note that (non-EM) logistic regression performs as well as our (non-EM) ranking methods. This suggests that degree subjectivity may play a rather small role in rater assessments in the *SoftAttributes* data set.[22] By contrast, the EM methods used to identify sense subjectivity do a significantly better job of predicting the attribute assessments of raters—overall, they seem to disentangle the distinct senses used by different raters, leading to much better rank correlation than objective (single-sense) methods. With EM, nonlinear embeddings and linear embeddings give similar results, suggesting some nonlinearity in attribute semantics can be captured by sense subjectivity. Table 9 shows the number of senses discovered by the EM methods across all 36 attributes. We see that logistic regression with EM rarely uncovers more than one sense for a majority of attributes—21 (resp., 20) of the 36 attributes have a single sense in the case of linear (resp., nonlinear) CAVs. and 20 with nonlinear embedding). On average, the EM ranking methods do discover a greater number of senses than logistic regression models.

In Table 10 we show the results of an ablation study to test the effect of negative sampling on various CAV training methods. When training the CAV for a specific attribute, we select from one to five negative exemplars randomly for each positive example (where negatives are defined as in Section 4.2). We see that results are not especially sensitive to

[22]Note that the results for ranking methods vs. logistic regression in Table 3 are based on different training/test sets.

No. Negative Samples	LogRegr	EM LogRegr	RankNet	EM RankNet	LambdaRank	EM LambdaRank
1	0.205	0.288	0.199	0.388	0.191	0.379
2	0.234	0.319	0.223	0.390	0.195	0.370
3	0.213	0.331	0.207	0.359	0.211	0.346
4	0.226	0.336	0.197	0.382	0.215	0.342
5	0.218	0.307	0.213	0.338	0.194	0.340

Table 10. Comparison of negative sampling with different numbers of samples for the linear embedding case.

ratio of negative to positive examples. With logistic regression, 2–4 negatives works best providing an improvement of about 0.03 (resp., 0.05) in gamma rank correlation for the non-subjective (resp., subjective) case. Our ranking methods show negligible improvement when adding more than one negative sample. We suspect this is so because logistic regression is trying to separate negative and positive examples, thus additional negative samples assists the model in finding a better classification boundary, though too many puts too much weight on the noisy negative samples relative to ground-truth positive examples. On the other hand, in ranking only the relative ordering matters, so additional negative samples (which are inherently more noisy since they are imputed) adds little value and may even be detrimental. Given these results, we use a negative sampling rate of 4:1 for logistic regression and 1:1 for ranking in all following experiments.

6.3 Evaluation of CAVs Trained using *SoftAttributes* Rater Data

In contrast to the evaluation above—where CAVs were trained using MovieLens tags and used to predict the attribute assessments of *SoftAttributes* raters—we now train CAVs directly on rater assessments of soft attributes and evaluate their predictive quality. Since rater assessments are ordinal rather than binary, we train CAVs using RankNet and LambdaRank, but not logistic regression.

We train and test the CAV ϕ_g for attribute g using the pairwise comparisons induced by each rater's assessment of g, specifically, applying RankNet and LambdaRank to the data set consisting of per-rater pairwise comparisons. In training, we set the instance weight to two for comparisons with strong differences and one for weak differences. When considering sense subjectivity, we use EM variants of our two approaches to "cluster raters" based on their usage as we do above when training with MovieLens data. The number of senses for any specific g is selected in the same fashion as above, using a specific threshold on performance to stop increasing the number of different senses. We do not use nonlinear embeddings in this assessment, since results above (see Table 8) suggests that nonlinear embeddings provide only marginal improvements over linear embeddings when coupled with sense subjective modeling (or EM). We also report the results of the Supervised Weighted Dimensions (SWD) method used by Balog et al. [2] for reference. SWD uses a linear ranking support vector machine [25] induced using the same raters' pairwise comparisons as those used to train CAVs (we refer to the original paper for additional details).

We report CAV results using a 5-fold testing procedure. Specifically, to evaluate the performance of each CAV method on attribute g, we split raters into five groups of 20 at random, train the CAVs on each subset of four groups (80% of the data) and evaluate the CAV on the fifth (20% of the data), and average results over the five train/test splits—we do this to account for the potential noise in using only 20 raters for testing. For evaluation, we use extended Goodman-Kruskal gamma rank correlation to assess how well CAV predictions conform to the item rankings of test raters. As above we report the aggregate coefficient G' (i.e., weighted average results over all 60 attributes).

CAV Training Method	Gamma Rank Correlation
SWD	0.485 [2]
RankNet, Lin. Emb.	0.523
EM RankNet, Lin. Emb.	0.667
LambdaRank, Lin. Emb.	0.522
EM LambdaRank, Lin. Emb.	0.672

Table 11. Evaluation of Accuracy of CAVs Trained using *SoftAttributes* data set. Average Aggregate G' (extended Goodman-Kruskal gamma rank correlation) with test set (5-fold).

CAV Training Method	[2.5, 3.5)	[3.5, 4.5)	[4.5, 5.5)	[5.5, 6.5)	Average Number of Senses
EM RankNet, Lin. Emb.	11	31	18	0	4.153
EM LambdaRank, Lin. Emb.	10	26	21	3	4.223

Table 12. Number of CAV Senses Discovered by Various EM Methods (*SoftAttributes* data set). Displayed for each method: (a) how many of the 60 attributes have been assigned the specified number of senses—since we use 5-fold testing, we use the *average* number of senses across the five test sets for each attribute (bucketed around integer values 3 through 6); and (b) the average number of senses over all 60 attributes.

Table 11 shows the performance of CAVs trained on the *SoftAttributes* data set. The CAV methods show highly positive correlation between the learned CAVs and rater assessment of attibute values. This suggests that the ability to learn the semantics of soft attributes directly from ground-truth user comparison of a small set of items of w.r.t. the attribute in question can dramatically improve results relative to CAVs trained indirectly using tags (see Table 8). Both RankNet and LambdaRank outperform SWD, while the difference between RankNet and LambdaRank is minimal. The EM ranking methods outperform non-EM techniques, suggesting that sense subjectivity is a major factor in rater assessments in the *SoftAttributes* data, though the difference between EM RankNet and EM LambdaRank is negigible. Table 12 shows the number of distinct senses (averaged over 5-fold train/test) uncovered by the EM methods. The two methods differ very little, EM LambdaRank discovering only slightly more senses than EM RankNet.

7 USING CAVS FOR EXAMPLE CRITIQUING

While preference elicitation in recommender systems often uses attributes [6, 44, 55], an important question is the extent to which such methods can be adapted to handle soft attributes (see, e.g., [45]). The sheer variety of approaches to preference elicitation means we cannot consider this question in its full depth in this work. Instead, we focus on one particular methods by which users can express their preferences explicitly in recommenders, namely *example critiquing* [15], and examine how the CAV semantics for tags can be used for this purpose. In lieu of live experiments, we adopt a stylized but plausible *user response model* in which a user's critiques are driven by her underlying utility function and her personal tag/attribute semantics.[23] While other response models are possible (e.g., models that are fit to real user interaction data), this model suffices to demonstrate the value of CAVs for critiquing. We run synthetic experiments in which we have access to the ground truth utility and semantics for each user, and a MovieLens experiment, for which we propose a novel method for generating utilities and responses.

[23]Synthetic user models are commonly used to evaluate recommender systems [24, 64].

Recommender System Interaction Model. We assume a predefined list of critiquable tags and adopt a simple interactive recommender system that supports user critiques. We provide a brief description of our set up here but refer to Appendix A.4 for further details, parameter values used, etc. In the interactive system, each interaction with user u has the recommender present a slate S of k items to u. User u can *accept* one of the recommended items, at which point the session terminates. Otherwise, u can *critique* S using a tag g and a desired direction ('more,' 'less'). For instance, given a slate of k movies from which to choose, u might offer a critique like "more funny" or "less violent," intending that the recommender system makes an improved slate of next recommendations at the next iteration that reflects this preference. The recommender system then updates its *user representation* given this response and generates the next recommended slate. The process repeats until the user terminates by accepting a recommendation or the recommender system reaches the maximum critiquing steps T. We describe below the processes by which the recommender selects slates and updates its user representation, and by which the user responds to recommended slates.

User Response Model. Each behavior is driven by an underlying user model which is identical in form to the models used in our synthetic generative model above. Specifically, a user u has (i) a (personal) ground truth utility function over items (i.e., u can assess the utility of any item i that is recommended) and (ii) a (personal) ground-truth semantics for attributes (i.e., u can assess, for any recommender item i, i's value of the soft attribute corresponding to any of the critiquable tags). User interactions assume a *user response model.* User u also has a rough estimate of the maximum and minimum levels any tag/attribute can attain in the item corpus—this information is used to guide her critiquing behavior without assuming she has unrealistic knowledge of the item corpus (see Appendix A.4). When presented with slate S, u accepts an item i if its utility is sufficiently large, i.e., exceeds some threshold. Otherwise, u critiques slate S using the *most salient tag* g with respect to utility improvement of S; in particular, u critiques using the tag $g = \operatorname{argmax}_g \delta_u^T w_g$, where

$$\delta_u = (\phi_I(i_u^*) - \frac{1}{|S|}\sum_{i\in S}\phi_I(i)) \odot \phi_U(u)$$

is the utility difference vector between user u's estimated ideal item i_u^* and her average utility vector over the k items in S, and w_g is u's interpretation of g. $\odot$ refers to the Hadamard product.

Recommendation Strategy. We assume item embeddings $\phi_I(i)$ are fixed and use a *simple heuristic recommendation strategy* for incorporating critiques. We emphasize that the recommendation strategy used and method for incorporating critiques are fairly generic and are not intended to reflect the state-of-the-art, since our goal is to measure the ability to exploit learned CAVs. More elaborate strategies for updating user embedding are possible, including the use of Bayesian updates relative to a prior over the user embedding [54]. Here instead we adopt a simple heuristic, based on [32], to focus attention on the CAV semantics itself.

Given a user embedding $\phi_U(u)$, the recommender system scores all items i in the corpus with respect to utility $r_{i,u} = \phi_I(i)^T\phi_U(u)$, and presents the slate S of the top k scoring items. If the user critiques S with a tag g (and a specific direction), the system updates the user embedding as follows:

$$\phi_U(u) \leftarrow \phi_U(u) + \alpha_t(g)\cdot\phi_g$$

, where ϕ_g is g's CAV, $\alpha_t(g)$ is a tag-specific step size at iteration t (the tth interaction with u), and the sign of $\alpha_t(g)$ reflects the direction ('more,' 'less') of the critique. We treat the magnitude of $\alpha_t(g)$ as a hyper-parameter selected to optimize the utilities of the resulting recommendations.[24] The system then recommends the top k items given the

[24]Ultimately, this heuristic adjustment should be tuned to the specifics of a user response model generated from live experiments.

updated user embedding, and the process repeats. If the tag g is sense-subjective, the critique is interpreted relative to the systems's estimate of u's sense (or cluster) based on past usage.

Synthetic Data set Results. We first analyze critiquing with CAVs using the same synthetic models described above, where a user's critiques are generated with the user's ground truth utility and tag semantics. By contrast, the recommender system uses its *estimated user embedding* and the *tag's CAV* to interpret a user's critique (and not the ground truth). In our experiment, the recommended slate size is $k = 10$, and the maximum number of critiquing steps is $T = 25$. We evaluate how recommendation quality improves as the number of critiques increases by measuring the user's utility for the sequence of recommended (top-k) slates. We measure this "slate utility" using two different measures:

- *User max utility* of the top-k slate: $UMU(S) = \max_{i \in S} U(i)$, where $U(i)$ is u's *true* utility for i. This reflects the utility of a user who is able to select their most preferred item from S.
- *User average utility* of the top-k slate: $UAU(S) = \text{avg}_{i \in S} U(i)$. This captures the utility of a user who randomly selects an item from the recommended slate.

Fig. 2 presents the interactive critiquing results from three experiments using different synthetic data sets. The first has no subjectivity in user tag usage and assumes linear utility. The second also uses no subjectivity, but user item utility is nonlinear in the soft attributes underlying the tags. Finally, the third allows both degree subjectivity and nonlinear utility. We see from Fig. 2 that in all experiments, user utility, both *UMU* and *UAU*, improves with additional critiquing steps, eventually converging to a steady-state value. These results corroborate our hypothesis that, since CAVs quite accurately represent soft attributes in embedding space, they can be used to effectively update the recommender system's beliefs about user preferences as the user critiques recommended items, which in turn improves recommendation quality.

Furthermore, we recall from Sections 4 and 5 that CAVs trained with logistic regression generally have lower accuracy than those trained with RankNet or LambdaRank. While our CAV algorithms are not optimized to support critiquing, more accurate CAVs learned using ranking methods give rise to better interpretations of user critiques by the recommender—this observation is reflected by both the faster improvement and the greater steady-state values for both *UAU* and *UMU* in our experiments. This is most likely due to the fact that more accurate ranking-based CAVs better capture a user's intended semantics during critiquing. Similarly, the improved accuracy of nonlinear CAVs when utility is nonlinear manifests in the improved critiquing performance (in both the objective and degree-subjectivity tests, and for both *UAU* and *UMU*). Again, this performance improvement is likely due to the better CAV representation uncovered from the intermediate layers of the DNN.

Figure 3 shows interactive critiquing results using synthetic data in which the critiquable tags exhibit *sense subjectivity* under three different experimental settings: (i) user utility is linear w.r.t. soft attributes; (ii) user utility is nonlinear but the trained CAVS are linear CAVs; and (iii) both user utility and the trained CAVs are nonlinear. As above, both *UMU* and *UAU* improve with a increasing number of critiquing steps, eventually converging to a steady-state value. Recall that only three of the 25 utility-relevant dimensions correspond to the "conflated" senses of a single tag. Still, in the case of nonlinear utility, EM-LambdaRank, even with linear CAVs, outperforms LambdaRank without EM. This suggests that disentangling sense subjectivity is more critical than having an "accurate" but incorrectly-assumed objective single CAV. These results also show that EM Logistic Regression can be improved with the use of nonlinear CAVs as suggested in Table 5.

MovieLens20M Results. To evaluate critiquing with soft attributes using the MovieLens20M data set, we propose a novel method for hypothesizing "ground-truth" user utility. We first train a collaborative filtering model with all users and items, then train (non-subjective) CAVs for 164 tags (as described in Section 4.4). We then construct a small set of *test users*, each of whom has rated at least 50 movies. We use the learned embedding $\phi_U(u)$ for each test user u as if it were their *ground truth utility*. Since u has rated a large number of movies, we expect this ground truth utility to be reasonably stable and accurate. We then run the interactive critiquing recommender system by *forgetting* each test user—i.e., forgetting both their ratings and tags—and treating them as a "cold start" user, who is given a generic prior embedding.[25] This user u then generates critiques of the recommended slates using $\phi_U(u)$ as their true utility. Since we have no ground truth tag semantics, each u treats the recommender system's *learned CAV* as her semantics (admittedly giving the recommender system some advantage when interpreting critiques). Otherwise, the user response model is exactly as in the synthetic case. We evaluate as in the synthetic case, but user utility improvements are *estimates* using the learned embedding $\phi_U(u)$. Because of this we also assess some additional metrics (see below).

Fig. 4 shows critiquing results with MovieLens data. In addition to $UAU(S)$, we also report normalized discounted cumulative gain (NDCG) [53], mean reciprocal rank (MRR) [37], and average binarized rating[26] [47] of slates S generated during critiquing. We compare critiquing results generated by four sets of CAVs, trained with the following methods: RankNet, nonlinear RankNet, nonlinear LambdaRank, and nonlinear logistic regression (as these methods usually generate more accurate CAVs as discussed above). While all methods perform similarly with respect to UAU, nonlinear LambdaRank outperforms the others with respect to the other three metrics. This again provides evidence that: (i) incorporating CAVs in recommender systems to capture user-critiquing behavior can improve recommendation quality (both utilities and ratings); and (ii) the performance of critiquing-based recommender systems tends to improve with the quality of the learned CAVs.

8 CONCLUSIONS

We have presented a novel methodology for discovering the semantics of soft attribute/tag usage in recommender systems using concept activation vectors. Its benefits include: (i) using a collaborative filtering representation to identify attributes of greatest relevance to the recommendation task; (ii) distinguishing objective and subjective tag usage; (iii) identifying personalized, user-specific semantics for subjective attributes; and (iv) relating attribute semantics to preference representations, thus allowing interactions using soft attributes/tags in example critiquing and other forms of preference elicitation.

A number of future directions can provide additional value to the use of CAVs for elicitation and critiquing in recommender systems. While our empirical results suggest that CAVs are useful for subjective attributes and critiquing, additional study with real user critiques is critical, as is developing real-world data sets with ground truth utility and personal semantics (e.g., via survey instruments or controlled experiments).

In our current formulation, CAVs are learned offline under the assumptions that the data set contains either sufficient tagging data reflecting the personal preferences and semantics of diverse users or results of *pairwise tag-comparison queries* from sufficiently diverse users. When this is not that case, a useful extension is to consider an interactive methods in which the system can actively elicit a user's personal semantics, for example, using *pairwise tag-comparison queries* to identify a user's tag interpretation. For example, a few well-chosen queries based on offline data like "Which

[25]We use the average of all learned user embeddings as a prior.
[26]We set the binarized rating to 1 if the numeric rating exceeds 3, and 0 otherwise. This is a binary precision measure with respect to highly-rated items.

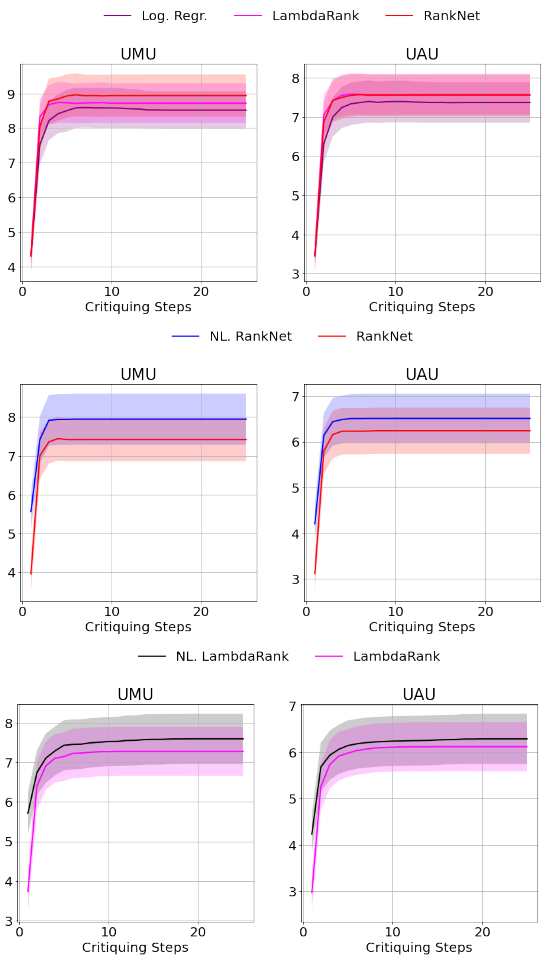

Fig. 2. Results of Interactive Critique with Synthetic Data. Top two: No Subjectivity Linear Utility with All Methods; Middle two: No Subjectivity Nonlinear Utility with RankNet; Bottom two: Degree Subjectivity Nonlinear Utility with LambdaRank

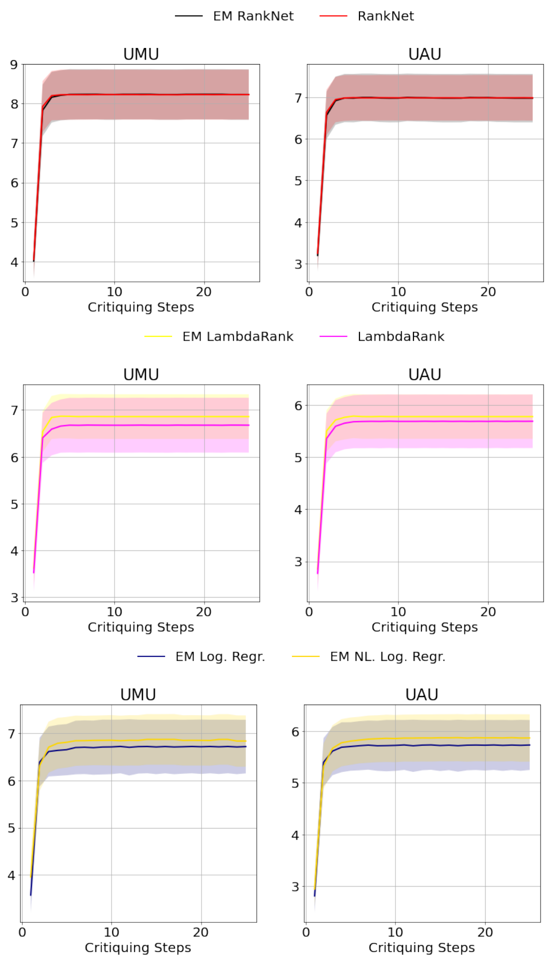

Fig. 3. Results of Interactive Critique with Synthetic Data. Top two: Sense Subjectivity Linear Utility with RankNet; Middle two: Sense Subjectivity Nonlinear Utility with LambdaRank, Lin-Emb; Bottom two: Sense Subjectivity Nonlinear Utility with Log. Regr., NL-Emb.

of these two books is more thought-provoking?” or “Do you consider this song to be upbeat?” could help identify a user’s personal semantics (both degree thresholds and tag senses).

Our focus in this work has been on soft attribute usage in an item-space that is characterized entirely by latent attributes (e.g., as is common in content recommendation). Other domains, such as product recommendation, are traditionally characterized by hard attributes (e.g., price, color, weight, efficiency, capacity, and other product-specific features). However, the use of CAVs to allow the more flexible use of soft and subjective attributes (e.g., cozy, comfortable, vibrant, stylish, compact, inexpensive, etc.) to navigate such domains is of great interest. This raises interesting questions of how to integrate soft and hard attributes, including the possibility of exploiting the ground-truth semantics of hard attributes to facilitate the discovery of soft-attribute semantics (e.g., relating (possibly subjective) terms like “compact” to product dimensions or weight, or “inexpensive” to price).

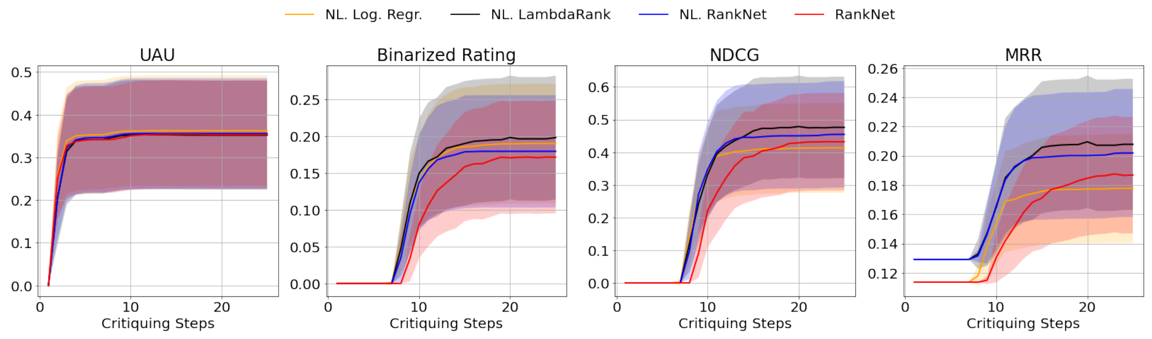

Fig. 4. Results of Interactive Critique with MovieLens Data

Finally, it would be of great value for critiquing-based recommender systems to have CAV-learning algorithms that not only maximize concept accuracy, but whose representations are more directly tuned to support preference elicitation. and to more directly compare CAVs to alternative ways of understanding soft and subjective attributes.

ACKNOWLEDGEMENTS

Thanks to Krisztian Balog, Walid Krichene, Dima Kuzmin, Nicolas Mayoraz, Filip Radlinski, Steffen Rendle, Tania Bedrax-Weiss, and Li Zhang for helpful discussions and feedback, as well as the anonymous reviewers (including those of an earlier version of the manuscript).

REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering* 17, 6 (2005).

[2] Krisztian Balog, Filip Radlinski, and Alexandros Karatzoglou. 2021. On Interpretation and Measurement of Soft Attributes for Recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21)*. 890–899.

[3] G. Bang, G. Barash, R. Bea, J. Cali, M. Castillo-Effen, X. Chen, N. Chhaya, R. Cummings, R. Dhoopar, S. Dumanci, H. Espinoza, E. Farchi, F. Fioretto, R. Fuentetaja, C. Geib, O. E. Gundersen, J. Hernández-Orallo, X. Huang, K. Jaidka, S. Keren, S. Kim, M. Galley, X. Liu, T. Lu, Z. Ma, R. Mallah, J. McDermid, M. Michalowski, R. Mirsky, S. Ó hÉigeartaigh, D. Ramachandran, J. Segovia-Aguas, O. Shehory, A. Shaban-Nejad, V. Shwartz, S. Srivastava, K. Talamadupula, J. Tang, P. Van Hentenryck, D. Zhang, and J. Zhang. 2020. The Association for the Advancement of Artificial Intelligence 2020 Workshop Program. In *AI Magazine*. 100–114.

[4] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H. Chi. 2018. Latent Cross: Making Use of Context in Recurrent Recommender Systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM-18)*. Marina Del Rey, CA, 46–54.

[5] Christopher M Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer, New York.

[6] Edwin V. Bonilla, Shengbo Guo, and Scott Sanner. 2010. Gaussian Process Preference Elicitation. In *Advances in Neural Information Processing Systems 23 (NIPS-10)*. Vancouver, 262–270.

[7] Léon Bottou and Yoshua Bengio. 1994. Convergence Properties of the K-Means Algorithms. In *Advances in Neural Information Processing Systems 7 (NIPS-94)*. 585–592.

[8] Craig Boutilier, Kevin Regan, and Paolo Viappiani. 2009. Online Feature Elicitation in Interactive Optimization. In *Proceedings of the Twenty-sixth International Conference on Machine Learning (ICML-09)*. Montreal, 73–80.

[9] Craig Boutilier, Kevin Regan, and Paolo Viappiani. 2010. Simultaneous Elicitation of Preference Features and Utility. In *Proceedings of the Twenty-fourth AAAI Conference on Artificial Intelligence (AAAI-10)*. Atlanta, 1160–1167.

[10] Craig Boutilier, Richard S. Zemel, and Benjamin Marlin. 2003. Active Collaborative Filtering. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI-03)*. Acapulco, 98–106.

[11] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to Rank using Gradient Descent. In *Proceedings of the Twenty-second International Conference on Machine Learning (ICML-05)*. 89–96.

[12] Christopher JC Burges. 2010. From RankNet to LambdaRank to LambdaMART: An Overview. *Learning* 11, 23–581 (2010), 81.

[13] Robin Burke. 2002. Interactive Critiquing for Catalog Navigation in E-Commerce. *Artificial Intelligence Review* 18, 3–4 (2002), 245–267.

[14] Yunbo Cao, Jun Xu, Tie-Yan Liu, Hang Li, Yalou Huang, and Hsiao-Wuen Hon. 2006. Adapting ranking SVM to document retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. 186–193.

[15] Li Chen and Pearl Pu. 2012. Critiquing-based Recommenders: Survey and Emerging Trends. *User Modeling and User-Adapted Interaction* 22, 1 (2012), 125–150.

[16] Deborah Cohen, Michal Aharon, Yair Koren, Oren Somekh, and Raz Nissim. 2017. Expediting Exploration by Attribute-to-feature Mapping for Cold-start Recommendations. In *Proceedings of the 11th ACM Conference on Recommender Systems (RecSys17)*. Como, Italy, 184–192.

[17] Simon French. 1986. *Decision Theory*. Halsted Press, New York.

[18] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, Steffen Rendle, and Lars Schmidt-Thieme. 2010. Learning Attribute-to-Feature Mappings for Cold-Start Recommendations. In *2010 IEEE International Conference on Data Mining (ICDM-10)*. 176–185.

[19] Leo A. Goodman and William H. Kruskal. 1954. Measures of Association for Cross Classifications. *J. Amer. Statist. Assoc.* 49, 268 (1954), 732–764.

[20] F. Maxwell Harper and Joseph A. Konstan. 2016. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems* 5, 4 (2016), 19:1–19:19.

[21] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web (WWW-17)*. Perth, Australia, 173–182.

[22] Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 168–177.

[23] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*. 263–272.

[24] Guangda Huzhang, Zhen-Jia Pang, Yongqing Gao, Yawen Liu, Weijie Shen, Wen-Ji Zhou, Qianying Lin, Qing Da, An-Xiang Zeng, Han Yu, Yang Yu, and Zhi-Hua Zhou. 2021. AliExpress Learning-To-Rank: Maximizing online model performance without going online. *IEEE Transactions on Knowledge and Data Engineering* (2021). https://doi.org/10.1109/TKDE.2021.3098898

[25] Thorsten Joachims. 2002. Optimizing Search Engines using Clickthrough Data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-02)*. Edmonton, AB, 133–142.

[26] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory Sayres. 2018. Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV). In *Proceedings of the Thirty-fifth International Conference on Machine Learning (ICML-18)*. Stockholm, 2668–2677.

[27] Diederik Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the Third International Conference on Learning Representations (ICLR-15)*. San Diego, CA.

[28] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. 1997. GroupLens: Applying Collaborative Filtering to Usenet News. *Commun. ACM* 40, 3 (1997), 77–87.

[29] Jonathan Koren, Yi Zhang, and Xue Liu. 2008. Personalized Interactive Faceted Search. In *Proceedings of the 17th International Conference on World Wide Web (WWW-08)*. Beijing, 477–486.

[30] Ralf Krestel, Peter Fankhauser, and Wolfgang Nejdl. 2009. Latent Dirichlet Allocation for Tag Recommendation. In *Proceedings of the 3rd ACM Conference on Recommender Systems (RecSys09)*. 61–68.

[31] Artus Krohn-Grimberghe, Lucas Drumond, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2012. Multi-relational matrix factorization using bayesian personalized ranking for social network data. In *Proceedings of the Fifth International Conference on Web Search and Web Data Mining, WSDM 2012*. 173–182.

[32] Kai Luo, Scott Sanner, Ga Wu, Hanze Li, and Hojin Yang. 2020. Latent Linear Critiquing for Conversational Recommender Systems. In *Proceedings of The Web Conference 2020*. 2535–2541.

[33] Christopher Manning, Prabhakar Raghavan, and Hinrich Schutze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

[34] Leandro Balby Marinho, Christine Preisach, and Lars Schmidt-Thieme. 2009. Relational Classification for Personalized Tag Recommendation. In *Proceedings of ECML PKDD (The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases) Discovery Challenge 2009*, Vol. 497. 7–15.

[35] Benjamin M. Marlin and Richard S. Zemel. 2007. Collaborative Filtering and the Missing at Random Assumption. In *Proceedings of the Twenty-third Conference on Uncertainty in Artificial Intelligence (UAI-07)*. Vancouver, 50–54.

[36] Julian McAuley, Jure Leskovec, and Dan Jurafsky. 2012. Learning Attitudes and Attributes from Multi-aspect Reviews. In *12th International Conference on Data Mining (ICDM-12)*. 1020–1025.

[37] Brian McFee and Gert RG Lanckriet. 2010. Metric learning to rank. In *ICML*.

[38] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems-Volume 2*. 3111–3119.

[39] Martin Mladenov, Chih wei Hsu, Vihan Jain, Eugene Ie, Christopher Colby, Nicolas Mayoraz, Hubert Pham, Dustin Tran, Ivan Vendrov, and Craig Boutilier. 2020. Demonstrating Principled Uncertainty Modeling for Recommender Ecosystems with RecSim NG. In *RecSys 2020: Fourteenth ACM Conference on Recommender Systems, Virtual Event, Brazil, September 22-26, 2020*. 591–593.

[40] Hervé Moulin. 1980. On Strategy-proofness and Single Peakedness. *Public Choice* 35, 4 (1980), 437–455.

[41] Preksha Nema, Alexandros Karatzoglou, and Filip Radlinski. 2021. Disentangling Preference Representations for Recommendation Critiquing with β-VAE. In *30th ACM International Conference on Information and Knowledge Management (CIKM 2021)*. New York.

[42] Daniel N. Osherson and Edward E. Smith. 1981. On the Adequacy of Prototype Theory as a Theory of Concepts. In *Cognition*.

[43] Bilih Priyogi. 2019. Preference Elicitation Strategy for Conversational Recommender System. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 824–825.

[44] Pearl Pu and Li Chen. 2008. User-involved Preference Elicitation for Product Search and Recommender Systems. *AI Magazine* 29, 4 (2008), 93–103.

[45] Filip Radlinski, Krisztian Balog, Bill Byrne, and Karthik Krishnamoorthi. 2019. Coached Conversational Preference Elicitation: A Case Study in Understanding Movie Preferences. In *Proceedings of the Annual SIGDial Meeting on Discourse and Dialogue*.

[46] Filip Radlinski, Craig Boutilier, Deepak Ramachandran, and Ivan Vendrov. 2022. Subjective Attributes in Conversational Recommendation Systems: Challenges and Opportunities. In *Proceedings of the Thirty-sixth AAAI Conference on Artificial Intelligence (AAAI-22)*. 12287–12293.

[47] Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K Lam, Sean M McNee, Joseph A Konstan, and John Riedl. 2002. Getting to know you: learning new user preferences in recommender systems. In *Proceedings of the 7th international conference on Intelligent user interfaces*. 127–134.

[48] Michel Regenwetter, Bernard Grofman, A. A. J. Marley, and Ilia Tsetlin. 2006. *Behavioral Social Choice: Probabilistic Models, Statistical Inference, and Applications*. Cambridge University Press, Cambridge.

[49] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the Twenty-fifth Conference on Uncertainty in Artificial Intelligence (UAI-09)*. Montreal, 452–461.

[50] Steffen Rendle and Lars Schmidt-Thieme. 2010. Pairwise Interaction Tensor Factorization for Personalized Tag Recommendation. In *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining (WSDM-10)*. 81–90.

[51] Ruslan Salakhutdinov and Andriy Mnih. 2007. Probabilistic Matrix Factorization. In *Advances in Neural Information Processing Systems 20 (NIPS-07)*. Vancouver, 1257–1264.

[52] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic Attribution for Deep Networks. In *International Conference on Machine Learning*. 3319–3328.

[53] Hamed Valizadegan, Rong Jin, Ruofei Zhang, and Jianchang Mao. 2009. Learning to Rank by Optimizing NDCG Measure.. In *NIPS*, Vol. 22. 1883–1891.

[54] Ivan Vendrov, Tyler Lu, Qingqing Huang, and Craig Boutilier. 2020. Gradient-based optimization for Bayesian preference elicitation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 10292–10301.

[55] Paolo Viappiani and Craig Boutilier. 2010. Optimal Bayesian Recommendation Sets and Myopically Optimal Choice Query Sets. In *Advances in Neural Information Processing Systems 23 (NIPS)*. Vancouver, 2352–2360.

[56] Paolo Viappiani, Boi Faltings, and Pearl Pu. 2006. Preference-based Search using Example-Critiquing with Suggestions. *Journal of Artificial Intelligence Research* 27 (2006), 465–503.

[57] Bifan Wei, Jun Liu, Qinghua Zheng, Wei Zhang, Xiaoyu Fu, and Boqin Feng. 2013. A Survey of Faceted Search. *Journal of Web Engineering* 12, 1&2 (2013), 041–064.

[58] Charles Welch, Jonathan K. Kummerfeld, Verónica Pérez-Rosas, and Rada Mihalcea. 2020. Exploring the Value of Personalized Word Embeddings. In *Proceedingsof the 28th International Conference on Computational Linguistics*.

[59] Ga Wu, Kai Luo, Scott Sanner, and Harold Soh. 2019. Deep language-based critiquing for recommender systems. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 137–145.

[60] Ji Yang, Xinyang Yi, Derek Zhiyuan Cheng, Lichan Hong, Yang Li, Simon Xiaoming Wang, Taibai Xu, and Ed H Chi. 2020. Mixed Negative Sampling for Learning Two-tower Neural Networks in Recommendations. In *Proceedings of the Web Conference (WWW-20)*. Taipei, 441–447.

[61] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed Chi. 2019. Sampling-bias-corrected Neural Modeling for Large Corpus Item Recommendations. In *Proceedings of the Thirteenth ACM Conference on Recommender Systems (RecSys19)*. Copenhagen, 269–277.

[62] Yury Zemlyanskiy, Sudeep Gandhe, Ruining He, Bhargav Kanagal, Anirudh Ravula, Juro Gottweis, Fei Sha, and Ilya Eckstein. 2021. DOCENT: Learning Self-Supervised Entity Representations from Large Document Collections. In *Proceedings of EACL*. 2540–2549.

[63] Xiaoxue Zhao, Weinan Zhang, and Jun Wang. 2013. Interactive Collaborative Filtering. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management (CIKM-13)*. 1411–1420.

[64] Hao Zou, Peng Cui, Bo Li, Zheyan Shen, Jianxin Ma, Hongxia Yang, and Yue He. 2020. Counterfactual Prediction for Bundle Treatment. In *Advances in Neural Information Processing Systems 33 (NeurIPS-20)*. 19705–19715.

A APPENDIX

We provide additional details on various aspects of this work. We fully specify the synthetic data generation process used in App. A.1, describe our precise use of MovieLens set in App. A.2, elaborate on our training methods (incl. model architectures, parameter values, etc.) in App. A.3, and elaborate on our example-critiquing set up in App. A.4.

A.1 Synthetic Data Generation with RecSim

We use a stylized, but structurally realistic generative model to produce synthetic ratings and tag data for some of our experiments. Its purpose is twofold. First, it offers various parameters or "knobs" that can be used to generate data sets that can increase/decrease the level of difficulty faced by methods designed to extract the semantics of soft or subjective attributes. Second, synthetic data generation provides us with a "ground truth" against we can test (i) the quality of our learned CAV representations of soft attributes and (ii) the effectiveness of our elicitation methods at using soft, subjective attributes to improve recommendations.

The generative user-response model is implemented using RecSim NG [39], a platform for simulating user behavior when interacting with recommender systems that supports the authoring of structured, graphical and causal models of agent behavior or learning such behavior from data. (We use the former capability.)

We start by describing the process for generating ratings and tags for "non-subjective" tags, where users have linear utility for the corresponding soft attributes. We then describe mild modifications of this core generative model to allow for subjective tags (both degree and sense subjectivity) and nonlinear attribute utility.

Non-subjective, linear utility model. The generative process proceeds in the following stages: we first generate items (with their latent and soft attribute values and other properties); then users (with their utility functions and other behavioral characteristics); then user-item ratings; and finally user-item tags. The model reflects realistic characteristics such as item and user "clustering," popularity bias, not-missing-at-random ratings, the sparsity of ratings, the relative sparsity of tags compared to ratings, etc.

Specifically each item i is characterized by an *attribute vector* $\mathbf{v}(i) \in [0,1]^D$, where $D = L + S$: L dimensions correspond to latent item features and S to soft attributes. For a soft dimension $L < s \leq L + S$, $v^s(i)$ captures the degree to which i exhibits soft attribute s. We sample m items from a mixture of K D-dimensional Gaussian distributions (truncated on $[0,1]^D$) $\mathcal{N}(\mu_k, \sigma_k)$, $k \leq K$, with mean vector $\mu_k \in [0,1]^D$ and (diagonal) covariance σ_k. We set D to 25 and K to 100 in our experiments. For simplicity, we assume all σ_k are identical to 0.5, and sample means uniformly. Mixture weights are sampled uniformly at random and normalized. For each item i, we also randomly generate a popularity bias b_i from $[0,1]$ (whose purpose is described below).

Each user u has a *utility vector* $\mathbf{w}(u) \in [0,1]^D$ reflecting its utility for items. We sample n users from the above K-mixture-of-Gaussian distribution similar to that for items. Here the means and variances of these distributions are same as the ones used for item distributions, but the mixture weights are fully resampled. This step ensures that the generated samples of users and items are distributed in different parts of the latent "topic space".

Next, we generate the user-item ratings with the following steps:

(i) For each u, we draw Num_u samples from a Zipf (or zeta) distribution with a power parameter $a = 1.05$ to reflect the natural power law over the number of ratings provided by users. Parameter a is chosen in a way that the average number of rated items by each user is approximately 100. We also set the maximum number of ratings by each user to $1,000$.

(ii) To generate the candidate items to be rated by each user u, we generate a set of $Rated_u$ items, by sampling them without replacement from the overall set of items via a multinomial logit (or softmax) choice model, where the probability associated with each item i is proportional to $e^{\tau \cdot (\mathbf{w}_u \mathbf{v}_i + b_i)}$. Here τ is a temperature parameter that controls the degree of randomness in choosing the item wrt greatest affinity, and we use $\tau = 1$ in our experiments.

(iii) For each user u and item $i \in Rated_u$, the rating r_{ui} is generated as follows. We denote by $s(u,i) := \mathbf{w}_u \mathbf{v}_i + \varepsilon$ be the score of item i, where ε is a small, zero-mean random noise. We then discretize all the scores provided by user u into 5 equally sized sub-intervals in $[\min_u, \max_u]$, where $\min_u = \min\{s(u,i) : i \in Rated_u\}$ and $\max_u = \max\{s(u,i) : i \in Rated_u\}$ and assign a 1 to 5 rating to each item $i \in Rated_u$ accordingly.

For each soft attribute s we assume there is a unique tag g_s that users can apply when referring to that attribute. For each generic tag g, we denote by $s(g)$ the corresponding soft attribute (so $g = g_{s(g)}$). To complete the data-generation procedure we also generate user-item tags with the following steps.

(i) For each user u, we generate PT_u, the probability of tagging an item, from a mixture of two distributions: (a) a Dirac distribution at 0 with weight $0 < x < 1$; and (b) a Uniform distribution over $[p_-, p_+]$, where $0 < p_- \leq p_+$, with weight $1 - x$. This reflects the fact that a large fraction of users never use tags, and among those who do, some users tag much more frequently than others. In our experiments, we set x to 0.8, p_- to 0.1, and p_+ to 0.5.

(ii) For each user u with non-zero tag probability, i.e., $PT_u > 0$, we first generate the set $Tagged_u$, which represents the items that are tagged by u. Here $Tagged_u$ is a subset of rated items $Rated_u$ such that each rated item will be tagged with (independent) probability PT_u. This reflects the fact that a user will not tag an unrated movie, but may leave some rated movies untagged.[27] For any item $i \notin Tagged_u$, the corresponding indicator value $t_{u,i,g} = 0$ for every tag g, which means that no tag is applied by user u on item i.

(iii) For every (non-subjective) tag g, we use a user-independent threshold $\tau_g = 0.5$ indicating the degree to which an item must possess attribute $s(g)$ to be tagged with tag g by a user.

(iv) For every item $i \in Tagged_u$ and tag g, we set the indicator value $t_{u,i,g} = 1$ (i.e., user u applied tag g to item i) if $v^{s(g)}(i) \geq \tau_g + \varepsilon$ (where ε is a small, zero-mean random noise drawn independently from $\mathcal{N}(0, 0.01)$ for each (u,i,g)). Otherwise the indicator value $t_{u,i,g}$ remains at 0.

Subjective model. The generative model above is modified in a straightforward way to handle subjective attributes. For degree subjectivity, we generate user-dependent tag-application thresholds τ_g^u for each user-tag pair, rather than user-independent thresholds τ_g. In general, to allow for some "commonality" across user sub-populations, we draw these thresholds from mixture distributions with a small number of components and small variance. Threshold distributions that are widely dispersed can be viewed as "fully" subjective, while for objective attributes their thresholds (as above) are special cases for which each of their distributions follow a Dirac at τ_g. In our experiments, for each degree-subjective tag the threshold is randomly chosen between 0.5 and 0.7.

For sense subjectivity, the model is as above with the following modification. We maintain S_{obj} soft attributes of the form above—which we now call *objective*—each $s \in S_{obj}$ corresponds to one item dimension and to a specific *objective (in sense)* tag g_s. In addition, we have S_{subj} *subjective* soft attributes, partitioned into *tag groups*, $S^1, \ldots, S^J$ under the following condition: (a) $S^i \cap S^j = \emptyset$ for $i \neq j$; (b) $\cup_{j \leq J} S^j = S_{subj}$; and (c) $|S^j| > 1$ for all $j \leq J$. Each tag group S^j is associated with a single tag g^j, with each $s \in S^j$ reflecting a different *sense* for g^j.

For each tag group S^j, each user u is randomly assigned to exactly one such sense $s(u,j) \in S^j$. This has two implications. First, when user u considers applying tag g^j to an item, it is evaluated according to soft attribute $s(u,j)$. This means that u uses that specific sense when applying that tag. Second, the utility vector $\mathbf{w}(u)$ of user u is such that its s^{th} component is zero for each $s \in S^j$ except for $s(u,j)$. This implies that u assesses her utility for an item using only her designated attribute (or sense) from each of the tag groups.

Nonlinear utility model. There are many forms of nonlinearity. We propose one especially simple form that allows one to test whether our CAV method (or any other soft-attribute method) can identify such attributes. Specifically,

[27] One could also allow, if desired, the propensity to tag to vary with the tag g, and/or bias the application of tags to higher-rated items.

we consider the simple single-peaked utility function, and for simplicity we only describe the utility function for non-subjective attributes. Extending it to the subjective case is straightforward.

Assume the domain of any arbitrary attribute $1 \leq a \leq D$ is $[0, 1]$, and the user utility is *additive-independent* across attributes, i.e., the utility for an item is the sum of "local utilities" for each attribute (the dot-product model satisfies this trivially). A user u's utility function is *single-peaked* with respect to s if u has an ideal point $p_{u,a} \in [0, 1]$ such that u's local utility for the attribute $l_{u,a}(x)$ is maximized at $p_{u,a}$ and decreases monotonically as x moves away from $p_{u,a}$. The functional form of a single-peaked utility can be arbitrary with the simplest form being a piecewise linear function. Here we use $l_{u,a}(x) = p_{u,a} - |x - p_{u,a}|$, where $x = \mathbf{w}_{u,a}\mathbf{v}_{i,a}$. As both vectors $\mathbf{w}_u$ and $\mathbf{v}_i$ are in $[0, 1]^D$, it is immediate to see that $x \in [0, 1]$. We sample $p_{u,a}$ from a uniform distribution $U(L_a, 1)$, where L_a is a user-specified per-attribute parameter. In the special case when $L_a = 1$ then $U(L_a, 1)$ becomes the linear utility. In our experiments, we set L_a to 0.3 for sense-subjective tags and L_a to 0.5 for the rest.

A.2 MovieLens20M Data

The MovieLens20M data set consists of 20 million movie ratings on a scale from 1 to 5 and 465,000 free form tag applications for 27,000 movies by 138,000 users. For our analysis, we transform all tags to lowercase and filter tag-and-rating data to only include the user-item-tags whose corresponding ratings are at least 4. This results in around 235,000 user-item-tag triples, which comprise 20,068 unique tags. Many of these are applied to only a few movies or by a few users: 11,145 tags are only applied by a single user, and just 268 tags are applied to at least 50 unique movies. Since the CAVs are trained in 50-dimensional latent space, we restrict the CAV training data to the top 250 tags in terms of unique tagged movies. Inspection of the tag data further shows that tags that are applied by only a few users tend to be overly-specific or overly-generic rather than descriptive of the particular movies tagged. For example, the tags 'memasa's movies', 'on dvr' and 'bd-video' were applied by single users to 216, 210 and 192 movies, respectively. The tags 'vhs' and 'owned' were applied to 194 and 155 movies by 20 individual users. To exclude these types of tags, we further filter the data to include only the top 250 tags in terms of unique users who have used the tag at least once. This leaves us with a total of 164 tags for evaluation. User-item-tag triples are then split into train-test with a roughly $(0.75, 0.25)$ split, with *all examples* for any specific user-item pair present in exactly one of these subsets, i.e., if user u rated item i, the all tags g applied by u to i (if any) will only be in either train or test data set.

A.3 More Details on Training Procedures

We provide additional details on the training methods used in our experiments.

A.3.1 *Weighted Alternating Least Squares.*

We learn user and item embeddings using collaborative filtering. We consider two approaches, matrix factorization using *weighted alternating least squares* (WALS) [23], which we describe in this subsection and a two-tower DNN (next subsection). Collaborative filtering typically generates a low-rank approximation of the user-item ratings matrix, which models both users and items in the low-dimensional (latent) feature space. Both users and items are represented by feature vectors in $X \subset \mathbb{R}^d$, where we let ϕ_U (resp., ϕ_I) be the mapping from users (resp., items) to features.

In our experiments with linear utility models, we learn (ϕ_U, ϕ_I) using WALS, with the following regularized objective:

$$(\phi_U^*, \phi_I^*) \in \arg\min \sum_{u,i} c_{u,i}(\hat{r}_{u,i} - r_{u,i})^2 + \kappa(||\phi_U||^2 + ||\phi_I||^2), \tag{7}$$

where $c_{u,i}$ is the confidence weight of the predicted rating $\hat{r}_{u,i}$ and $\kappa > 0$ is the regularization parameter. We pick the feature vectors (ϕ_U^*, ϕ_I^*) based on the best validation loss and train with an item-oriented confidence weight, i.e., $c_{u,i} \propto m - \sum_u r_{u,i}$, which assigns lower weight to less-frequently rated or lower-rated items. Example confidence weights include: (i) uniform, i.e., $c_{u,i} = \delta$ for some $\delta > 0$ for missing entries (u, i) and $c_{u,i} = 1$ otherwise; (ii) user-oriented, i.e., $c_{u,i} \propto \sum_i r_{u,i}$ which assigns higher confidence to users with more ratings, assuming he/she has greater item familiarity; and (iii) item-oriented, i.e., $c_{u,i} \propto m - \sum_u r_{u,i}$, which assigns lower weight to less-frequently rated or lower-rated items. In our experiments with linear utility models, we train a collaborative filtering model $\Phi = (\phi_U, \phi_I)$ using *WALS* given ratings $\mathbf{R}$, in order to obtain user and item feature vectors. We set the regularization parameter κ to 250 for MovieLens and 1 for the synthetic data. The number of WALS iterations is 100. We train with the third option for confidence scores and pick the feature vectors (ϕ_U^*, ϕ_I^*) based on the best validation loss.

A.3.2 DNN Training. When training nonlinear CAVs (i.e., when user utility is nonlinear with respect to the soft attribute in question), user and item embeddings ϕ_U and ϕ_I are generated by DNNs, N_U and N_I, respectively, and (as above) predicted ratings are generated by taking dot products, using a "two-tower" architecture. following deep neural network (DNN) models. Letting θ_U, θ_I denote the parameters of N_U, N_I,[28] similar to the linear case, we train the two-tower model by minimizing the regularized (squared) RMSE loss:.

$$(\theta_U^*, \theta_I^*) \in \arg\min_{\theta_U, \theta_I} \ \mathrm{RMSE}^2(\theta_U, \theta_I) + \kappa(||\phi_U(\cdot; \theta_U)||^2 + ||\phi_I(\cdot; \theta_I)||^2) + \rho(||\theta_U||^2 + ||\theta_I||^2), \tag{8}$$

where $\mathrm{RMSE}(\theta_U, \theta_I) = \sqrt{\frac{1}{N} \sum_{u,i} (\hat{r}_{u,i}(\theta_U, \theta_I) - r_{u,i})^2}$, the predicted rating is $\hat{r}_{u,i}(\theta_U, \theta_I) = \phi_U^\top(u; \theta_U)\phi_I(i; \theta_I)$, $\kappa > 0$ is the activity regularization constant, and $\rho > 0$ is the weight regularization constant. Since the above objective function is nonlinear and nonconvex, we simply optimize this objective function with stochastic gradient descent based approaches, e.g., ADAM [27].

We use one-hot encodings of users and items as inputs to N_U and N_I, respectively. For simplicity, we use the same architecture for both the user DNN and item DNN, which is a feed-forward network with ℓ layers, in which the first layer has no bias term and has a dimension of d—we let d have the same dimension as the linear embedding space (25 for synthetic data, 50 for MovieLens). The main motivation for this architectural choice is to allow weight initalization of the first layer using the linear user and item embeddings as a form of warm start. In our experiments, we use a DNN architecture with $\ell = 3$ fully connected layers, all of size (width) d, and ReLU activations. As in the linear case, we set the activity regularization parameter $\kappa = 250$ for MovieLens and $\kappa = 1$ for the synthetic data. For weight regularization, we set $\rho = 1$ in all the experiments. Given the trained DNNs, the activation vectors for training (item) CAVs are simply the ℓ-th intermediate activation layer $N_{I,\ell}(i)$, extracted from the trained item DNN. We treat the choice of intermediate layer ℓ as a tunable hyper-parameter chosen to optimize the downstream task (CAV prediction).

A.3.3 PITF Training. When evaluating CAVs for their ability to predict tag usage, we use the *PITF (pairwise interaction tensor factorization)* algorithm [50] as a natural baseline, since it is a method designed to predict (or recommend) tags for users. We train PITF using tag triples $t_{u,i,t}$ with the same train-test split as in CAV training. Notice that positive tags are generally more sparse in all data sets, to balance the portion of positive and negative samples for PITF training we also use the same negative sampling methodology as in CAV training. For fairer evaluation, we use the same embedding dimension as in CAV for PITF training (50 for MovieLens, 25 for synthetic data), but notice that the resulting PITF model is still much larger than the CAV model since PITF learns an embedding for each user, item, tag-user and tag-item,

[28]We remove dependence of various terms (e.g., $\phi_U, \phi_I, \hat{r}_{u,i}$) on the DNN parameters θ_U, θ_N to unclutter the notation when this dependence is obvious.

while CAV embedding is user-independent. In evaluating the accuracy of PITF, we do not adopt the F-score over Top-N items as in [50]. but instead we check if $sign(y_{u,i,g}) = t_{u,i,g}$, where $y_{u,i,g}$ is the prediction of PITF. In addition, we evaluate with *Spearman rank correlation coefficient* which is translation-invariant. We employ the implementation at `https://github.com/yamaguchiyuto/pitf/` to train the PITF embedding. For hyper-parameters, we set the learning rate (α in [50]) to 0.001 for MovieLens and 0.0002 for synthetic data, the regularization parameter (λ in [50]) to 0.01 for MovieLens and 0.00005 for synthetic data. These parameters are tuned with a validation set. The number of PITF training iterations is set to 100.

A.4 Additional Critiquing Details

We fill in a few additional details of the critiquing experiments in Section 7. We emphasize that our aim is not to evaluate state-of-the-art critiquing and elicitation methods, but to demonstrate the use of CAV-based soft-attribute semantics to allow users to effectively interact with the recommender and refine recommendation results.

In the synthetic data experiment, we construct each user's *estimated ideal item* using the knowledge of her ground-truth utility function, which is either linear or single-peaked linear in each (latent or soft-attribute) dimension. We note that a user's (actual or estimated) " ideal" item may not actually exist in the item corpus $\mathcal{I}$. Insisting that the user's ideal item exist is unrealistic, since it requires a dense item space. Moreover, the user generally does not know the identify of the actual best item in $\mathcal{I}$, since this would assume too great a state of knowledge for most users. Instead, we assume each u has a rough estimate of the maximum and minimum levels any tag/attribute can attain in the item corpus and uses this to drive her critiques. (For example, we truncate sampled items to $[0, 1]^D$.) Note that when user utility is linear, the ideal item must occur at the boundary of item space, so the estimates inform her estimated ideal. In the MovieLens experiment, we use the ratings data to derive an estimated ground-truth utility for each *test user* (who must have rated sufficiently many items as described in the main text).

During the critiquing process, the recommender system updates its user embedding based on the user's response.[29] We assume item embeddings $\phi_I(i)$ are fixed, and use a *simple heuristic recommendation strategy* for incorporating critiques.[30] Given a user embedding $\phi_U(u)$, the recommender system scores all items i in the corpus with respect to utility $r_{i,u} = \phi_I(i)^T\phi_U(u)$, and presents the slate S of the k top scoring items.

Suppose at step $t > 0$ the user critiques S with a specific tag g (and a specific direction, *more* or *less*). The recommender system updates the user embedding in response to this critique using a simple heuristic update function: $\phi_U(u) \leftarrow \phi_U(u) + \text{Sgn} \cdot \alpha_t(g) \cdot \phi_g$. Here Sgn $\in \{+1, -1\}$ indicates the direction of the move (+1 more, −1 less), and $\alpha_t(g)$ is a step size that controls the scale of the move of the user embedding in the direction of tag g's CAV. For each g, we decay the step size $\alpha_t(g)$ with each critique using g, $\alpha_t(g) = \alpha_0(g)/(1 + t)$, where α_0 is g's initial step size. This ensures that the updating process converges to a stable point (and does not cycle or repeat slates of items) given the coarse control mechanism offered to the user. If the tag g is sense-subjective, the critique is interpreted relative to the recommender system's estimate of u's sense (or user cluster) based on past usage.

[29]We use the average of all learned user embeddings as the systems's prior.

[30]We emphasize that the recommendation strategy used and method for incorporating critiques are fairly generic and are not intended to reflect the state-of-the-art, since our goal is to measure the ability to exploit learned CAVs. More elaborate strategies for updating user embedding are possible, including the use of Bayesian updates relative to a prior over the user embedding [54]. Here instead we adopt a simple heuristic, based on [32], to focus attention on the CAV semantics itself.

In our experiments, $\alpha_0(g)$ is constant (α_0) across all tags, and we treat it as a tunable hyper-parameter selected to optimize user utility metrics such as *UMU* and *UAU* utilities.[31] The critiquing results we report are based on the set of hyper-parameters optimized using a validation set.

A.5 Available Source Code and Training Data

Please see the following for source code and/or data set releases:

- To allow reproducibility (and extension) of our empirical results evaluating the use of CAVs with linear attributes, we have released both the input data and the source code used to compute CAVs as well as our evaluation metrics. Given relevant input data (see below), CAV computation and evaluation (i.e., metrics computation) code for linear attributes can be applied directly, and is available at https://github.com/google-research/google-research/tree/master/attribute_semantics. For example, see results in Sections 4.4, 5.3, 6.2 and 6.3.
- For the synthetic data set described in Appendix A.1 (and used for evaluation at various points in the paper), the code for the generative model used is available at https://github.com/google-research/recsim_ng/tree/master/recsim_ng/applications/cav_synthetic_model.
- In Appendix A.2 we describe various schemes to preprocess and filter the MovieLens20M data set. The resulting data set, with our train-test splits, is available at https://github.com/google-research/google-research/tree/master/attribute_semantics/data.
- CAVs applied to MovieLens20M data are computed using relevant movie embeddings. The linear embeddings were learned (via WALS, Appendix A.3.1) using internal production code, which is not releasable. However, we have made available the resulting embeddings on which our CAVs our trained (while results may vary with different embeddings, our methods are agnostic to the method used to generate them). These can be found at https://storage.googleapis.com/gresearch/attribute_semantics/cf_embeddings.npz.

[31]Ultimately, this heuristic adjustment should be tuned to the specifics of a real-world user response models.