# Inferring Networks of Substitutable and Complementary Products

Julian McAuley
UC San Diego
jmcauley@cs.ucsd.edu

Rahul Pandey
Pinterest
rahul@pinterest.com

Jure Leskovec
Stanford University & Pinterest
jure@cs.stanford.edu

## ABSTRACT

In a modern recommender system, it is important to understand how products relate to each other. For example, while a user is looking for mobile phones, it might make sense to recommend other phones, but once they *buy* a phone, we might instead want to recommend batteries, cases, or chargers. These two types of recommendations are referred to as *substitutes* and *complements*: substitutes are products that can be purchased *instead of* each other, while complements are products that can be purchased *in addition* to each other.

Here we develop a method to infer networks of substitutable and complementary products. We formulate this as a supervised link prediction task, where we learn the semantics of substitutes and complements from data associated with products. The primary source of data we use is the text of product reviews, though our method also makes use of features such as ratings, specifications, prices, and brands. Methodologically, we build topic models that are trained to automatically discover topics from text that are successful at predicting and explaining such relationships. Experimentally, we evaluate our system on the *Amazon* product catalog, a large dataset consisting of 9 million products, 237 million links, and 144 million reviews.

## 1. INTRODUCTION

Recommender systems are ubiquitous in applications ranging from e-commerce to social media, video, and online news platforms. Such systems help users to navigate a huge selection of items with unprecedented opportunities to meet a variety of special needs and user tastes. Making sense of a large number of products and driving users to new and previously unknown items is key to enhancing user experience and satisfaction [2, 14, 15].

While most recommender systems focus on analyzing patterns of interest in products to provide personalized recommendations [14, 30, 34, 36], another important problem is to understand *relationships* between products, in order to surface recommendations that are relevant to a given context [17, 35]. For example, when a user in an online store is examining t-shirts she should receive recommendations for similar t-shirts, or otherwise jeans, sweatshirts, and socks, rather than (say) a movie even though she may very well be interested in it. From these relationships we can construct a *product graph*, where nodes represent products, and edges represent various types of product relationships. Such product graphs facilitate many important applications: Navigation between related products, discovery of new and previously unknown products, identification of interesting product combinations, and generation of better and more context-relevant recommendations.

Despite the importance of understanding relationships between products there are several interesting questions that make the problem of building product graphs challenging: What are the common types of relationships we might want to discover? What data will allow us to reliably discover relationships between products? How do we model the semantics of why certain products are related?—For example, the semantics of why a given t-shirt might be related to a particular pair of jeans are intricate and can only be captured by a highly flexible model. And finally, how do we scale-up our methods to handle graphs of millions of products and hundreds of millions of relations?

**Inferring networks of product relationships.** Here we are interested in inferring networks of relationships between millions of products. Even though our method can be used to learn any type of relationship, we focus on identifying two types of links between products: *substitutes* and *complements* [21]. *Substitutable* products are those that are interchangeable—such as one t-shirt for another, while *complementary* products are those that might be purchased together, such as a t-shirt and jeans.

We design a system titled Sceptre *(Substitute and Complementary Edges between Products from Topics in Reviews)*, that is capable of modeling and predicting relationships between products from the text of their reviews and descriptions. At its core, *Sceptre* combines topic modeling and supervised link prediction, by identifying topics in text that are useful as features for predicting links between products. Our model also handles additional features such as brand, price, and rating information, product category information, and allows us to predict multiple types of relations (*e.g.* substitutes and complements) simultaneously. Moreover, *Sceptre* harnesses the fact that products are arranged in a category hierarchy and allows us to extend this hierarchy to discover 'micro-categories'—fine-grained categories of closely related products.

An example of the output of *Sceptre* is shown in Figure 1. Here, given a query item (a hiking boot), our system identifies a ranked list of potential substitutes (other hiking boots), and complements (heavy-duty socks, shoe polish, etc.).

We train *Sceptre* on a large corpus of 9 million products from *Amazon*, with 237 million connections derived from browsing and co-purchasing data. We evaluate *Sceptre* in terms of its accuracy at link prediction and ranking, where we find it to be significantly

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org
KDD '15 August 11 - 14, 2015, Sydney, NSW, Australia
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-3664-2/15/08 ...$15000.
DOI: http://dx.doi.org/10.1145/2783258.2783381.

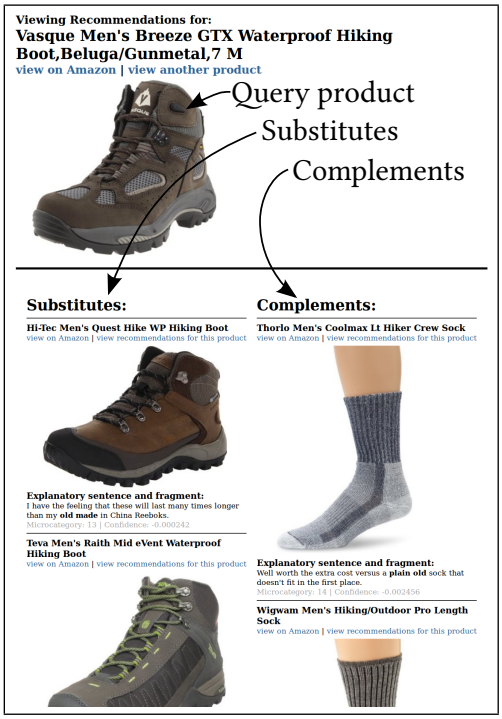


Figure 1: *Sceptre* learns the concept of substitute and complement goods from product information (descriptions, reviews, etc.). Given a query item, *Sceptre* allows us to generate substitute and complementary recommendations as shown above.

more accurate than alternatives. We also use *Sceptre* to build a product graph, where for every product we recommend a list of the most related complementary and substitutable products. Finally, we show that *Sceptre* can be applied in 'cold-start' settings, by using other sources of text when reviews are unavailable. Overall, we find that the most useful source of information to identify substitutes and complements is the text associated with each product (*i.e.*, reviews, descriptions, and specifications), from which we are able to uncover the key features and relationships between products, and also to explain these relationships through textual signals.

We envision several applications of the product graphs produced by our system. Our system can help users to navigate, explore and discover new and previously unknown products. Or, it can be used to identify interesting product combinations, *e.g.* we can recommend outfits by matching a shirt with complementary trousers and a jacket. And, our system can be used as a candidate-generation step in providing better and more context-relevant recommendations.

## 2. RELATED WORK

The basic task of a recommender system is to suggest relevant items to users, based on their opinions, context, and behavior. One component of this task is that of estimating users' ratings or rankings of products [14], *e.g.* by matrix factorization [15] or collaborative filtering [17]. Our goal here is related but complementary to rating estimation as we aim to discover relations between products.

In principle the types of relationships in which we are interested can be mined from behavioral data, such as browsing and co-purchasing logs. For example, *Amazon* allows users to navigate between products through links such as 'users who bought X also bought Y' and 'users who viewed X also viewed Y' [17]. Such a 'co-counting' solution, while simple, has a few shortcomings, for example it may produce noisy recommendations for infrequently-purchased products, and has limited ability to explain the recommendations it provides. More sophisticated solutions have been proposed that make use of browsing and co-purchasing data (*e.g.* [35]), but in contrast to such 'behavioral-based' solutions our goal is to *learn* the semantics of 'what makes products related?' in order to generate new links, adapt to different notions of relatedness, and to understand and explain the features that cause humans to consider products to be related.

*Topic models* are a fundamental building block of text modeling [3, 4, 5] and form the cornerstone of our model. A variety of works have used topic models within recommender systems (*e.g.* [6, 10, 11, 22, 23, 28, 31, 32]), though generally with the goal of predicting user *ratings* (or opinions) rather than learning relationships between products as we do here. More specifically, our work builds on topic models for networks: Block-LDA [1], topic-link LDA [18], and relational topic models [7] all attempt to identify topics that explain the links in document networks. A promising line of work uses such ideas to model social and citation networks [8, 33, 34]. However, these methods have trouble scaling to large networks, while *Sceptre* scales to corpora with millions of documents (products) and hundreds of millions of links.

Last, a related branch of work aims to enhance e-commerce using browsing data. For example, [12] aims to forecast commercial intent based on query logs; and in [26] the authors use query data to identify attributes that are important to users in order to surface recommendations. While different in terms of the data and problem setting, such works are similar in that they uncover relationships from large sources of weakly-structured data.

## 3. THE SCEPTRE MODEL

In the following we build *Sceptre* gradually, but in such a way that at each step we are specifying a usable model. We highlight the differences between successive iterations of our model in blue. We do this to emphasize the fact that while *Sceptre* makes use of several interacting components, each of these components brings an additional modeling aspect into the framework. Table 1 describes the notation we use throughout the paper.

### 3.1 High-level Overview

We first present a high-level, conceptual view of *Sceptre*, to explain the intuition behind the model before we fully specify it.

**Topic Models.** We use *topic models* [4] to discover topics from product reviews and other sources of text. Conceptually, this means that the text from millions of products can be clustered into a small number of dimensions, so that each product $i$ (and its text) can be represented by a topic vector $\theta_i$ encoding the extent to which reviews/descriptions of a given product discuss each of the topics.

**Link Prediction.** Topic models allow us to represent each product $i$ by a vector $\theta_i$. On top of this we can have a statistical model to predict properties about products. In our case, we use logistic regression to make predictions about *pairs* of items, using features that combine the topics of two products $\theta_i$ and $\theta_j$ simultaneously. The classification task we are interested in is: *does a relationship exist between $i$ and $j$?* Using *pairwise* features of the products, *e.g.* $\psi(i,j) = \theta_j - \theta_i$, we build logistic classifiers such that $\langle \beta, \psi(i,j) \rangle$ takes a positive value if $i$ and $j$ are connected by an edge. We further develop this model so that predicting the *presence* of an edge and the *direction* of an edge are treated as two separate tasks, to account for asymmetries and to help with interpretability.

Importantly, it should be noted that we do not train topic models and *then* perform link prediction, but rather we define a joint

objective such that we discover topics that are informative for our link prediction task. In this way our model uncovers topics that are good at 'explaining' the relationships between products.

**Micro-Categories.** An additional goal of *Sceptre* is to be able to discover micro-categories of closely related products. We achieve this by using sparse representations of very high dimensional topic vectors for each product. We make use of explicit product hierarchies (such as the category tree available from *Amazon*), where each node of the hierarchy has a small number of topics associated with it. The hierarchical nature of the category tree means that topics associated with top-level nodes are general and broad, while topics associated with leaf categories focus on differentiating between subtle product features, which can be interpreted as micro-categories (*e.g.* different styles of running shoes).

**Product graph.** Finally, we have a supervised learning framework to predict relationships between products. Discovering substitutes and complements then depends on the choices of graph we use to train the model, for which we collect several graphs of related products from *Amazon*. For example, a co-purchasing graph such as 'users frequently purchased $a$ and $b$ together' encodes some notion of complements, whereas a graph such as 'users who viewed $a$ eventually purchased $b$' captures the notion of substitutes. Thus, for every product, we predict a list of complementary and substitutable products and collect them into a giant network of related products.

## 3.2 Detailed Technical Description

### 3.2.1 Background: Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA, [4]) uncovers latent structure in document corpora. For the moment, 'documents' shall be the set of reviews associated with a particular product. LDA associates each document in a corpus $d \in \mathcal{T}$ with a $K$-dimensional topic distribution $\theta_d$ (a stochastic vector, *i.e.*, $\sum_k \theta_{d,k} = 1$), which encodes the fraction of words in $d$ that discuss each of the $K$ topics. That is, words in the document $d$ discuss topic $k$ with probability $\theta_{d,k}$.

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

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

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

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

### 3.2.2 Link Prediction with Topic Models

'Supervised Topic Models' [3] allow topics to be discovered that are predictive of an output variable associated with each document. We propose a variant of a supervised topic model that identifies topics that are useful as features for link prediction. We choose an approach based on logistic regression because (1) It can be scaled to millions of documents/products by hundreds of millions of edges, and (2) It can be adapted to incorporate both latent features (topics) and manifest features (such as brand, price, and rating information), as well as arbitrary transformations and combinations of these features. Our goal here is to predict links, that is labels at the level of *pairs* of products. In particular, we want to train logistic classifiers that for each pair of products $(i, j)$ predicts whether they are related ($y_{i,j} = 1$) or not ($y_{i,j} = 0$). For now we will consider the case where we are predicting just a single type of relationship and we will later generalize the model to predict multiple types of relationships (substitutes and complements) simultaneously.

| Symbol | Description |
|---|---|
| $d_i$ | document associated with an item (product) $i$ |
| $\mathcal{T}$ | document corpus |
| $K$ | number of topics |
| $\theta_i$ | $K$-dimensional topic distribution for item $i$ |
| $\phi_k$ | word distribution for topic $k$ |
| $w_{d,j}$ | $j^{\text{th}}$ word of document $d$ |
| $z_{d,j}$ | topic of the $j^{\text{th}}$ word document $d$ |
| $N_d$ | number of words in document $d$ |
| $F(x)$ | logistic (sigmoid) function, $1/(1+e^{-x})$ |
| $\mathcal{E}_g$ | observed edges in graph $g$ |
| $\psi(i,j)$ | pairwise (undirected) features for items $i$ and $j$ |
| $\varphi(i,j)$ | pairwise (directed) features for items $i$ and $j$ |
| $\beta$ | logistic weights associated with $\psi(i,j)$ |
| $\eta$ | logistic weights associated with $\varphi(i,j)$ |

Table 1: Notation.

We want the topics associated with each product to be 'useful' for logistic regression in the sense that we are able to learn a logistic regressor parametrized by $\beta$ that predicts $y_{i,j}$, using the topics $\theta_i$ and $\theta_j$ as features. That is, we want the logistic function

$$F_\beta(\theta_d) = \sigma(\langle \beta, \psi_\theta(i,j) \rangle) \tag{2}$$

to match $y_{i,j}$ as closely as possible, where $\psi_\theta(i,j)$ is a pairwise feature vector describing the two products. We then aim to design features that encode the *similarity* between the two products (documents). The specific choice we adopt is

$$\psi_\theta(i,j) = (1, \theta_{i,1} \cdot \theta_{j,1}, \theta_{i,2} \cdot \theta_{j,2}, \ldots, \theta_{i,K} \cdot \theta_{j,K}). \tag{3}$$

Intuitively, by defining our features to be the elementwise product between $\theta_i$ and $\theta_j$, we are saying that products with *similar* topic vectors are likely to be linked. The logistic vector $\beta$ then determines *which* topic memberships should should be similar (or dissimilar) in order for the products to be related.

Our goal then is to simultaneously optimize both topic distributions $\theta_d$ and logistic parameters $\beta$ to maximize the joint likelihood of topic memberships and relationships in the product graph:

$$L(y, \mathcal{T}|\beta, \theta, \phi, z) = \overbrace{\prod_{d \in \mathcal{T}} \prod_{j=1}^{N_d} \theta_{z_{d,j}} \phi_{z_{d,j}, w_{d,j}}}^{\text{corpus likelihood}} \underbrace{\prod_{(i,j) \in \mathcal{E}} F_\beta(\psi_\theta(i,j)) \prod_{(i,j) \in \bar{\mathcal{E}}} (1 - F_\beta(\psi_\theta(i,j)))}_{\text{logistic likelihood of the observed graph}}. \tag{4}$$

This expression says that the review corpus should have high likelihood according to a topic model, but also that those topics should be useful as predictors in a logistic regressor that uses their similarity as features. In this way, we will intuitively discover topics that correspond to some concept of document 'relatedness'.

This idea of jointly training topic and regression models is closely related to the model of [22], where topics were discovered that are useful as parameters in a latent-factor recommender system. Roughly, in the model of [22], a user would give a high rating to a product if their latent user parameters were similar to the topics discovered from reviews of that item; topics were then identified that were good at predicting users' ratings of items. The basic model of (eq. 4) is similar in the sense that we are coupling parameters $\theta$ and $\beta$ in a joint likelihood in order to predict the output variable $y$.

**Directed vs. Undirected Graphs.** So far we have shown how to train topic models to predict links between products. However, the feature vector of (eq. 3) is symmetric ($\psi_\theta(i,j) = \psi_\theta(j,i)$), meaning that it is only useful for predicting *undirected* relationships. However, none of the relationships we want to predict are necessarily symmetric. For example $y$ may be a good substitute for $x$ if $y$ is a similar product that is cheaper and better rated, but in this case $x$ would be a poor substitute for $y$. Or, while a replacement battery may be a good complement for a laptop, recommending a laptop to a user already purchasing a battery makes little sense. Thus we ought to account for such asymmetries in our model.

We model such asymmetries by first predicting whether two products are *related*, and *then* predicting in which direction the relation flows. That is, we predict

$$p(a \text{ has an edge toward } b) =$$
$$p(a \text{ is related to } b) \times p(\text{edge flows from } a \text{ to } b \mid a \text{ is related to } b),$$

which we denote

$$p((a,b) \in \mathcal{E}) = \underbrace{p(a \leftrightarrow b)}_{\text{'are they related?'}} \overbrace{p(a \rightarrow b | a \leftrightarrow b)}^{\text{'does the edge flow in this direction?'}}, \quad (5)$$

where relations $(a,b) \in \mathcal{E}$ are now *ordered* pairs (that may exist in both directions). We model relations in this way since we expect different types of language or features to be useful for the two tasks—relatedness is a function of what two products have *in common*, whereas the direction the link flows is a function of how the products *differ*. Indeed, in practice we find that the second predictor $p(a \rightarrow b | a \leftrightarrow b)$ tends to pick up qualitative language that explains why one product is 'better than' another, while the first tends to focus on high-level category specific topics. Our objective now becomes

$$L(y, \mathcal{T} | \beta, \eta, \theta, \phi, z) =$$
$$\overbrace{\prod_{(i,j) \in \mathcal{E}} F^{\leftrightarrow}_{\beta}(\psi_\theta(i,j)) F^{\rightarrow}_{\eta}(\varphi_\theta(i,j))(1 - F^{\rightarrow}_{\eta}(\varphi_\theta(j,i)))}^{\text{positive relations } (F^{\leftrightarrow}) \text{ and their direction of flow } (F^{\rightarrow})}$$
$$\underbrace{\prod_{(i,j) \in \bar{\mathcal{E}}} (1 - F^{\leftrightarrow}_{\beta}(\psi_\theta(i,j)))}_{\text{non-relations}} \underbrace{\prod_{d \in \mathcal{T}} \prod_{j=1}^{N_d} \theta_{z_{d,j}} \phi_{z_{d,j}, w_{d,j}}}_{\text{corpus likelihood}}. \quad (6)$$

Here $F^{\leftrightarrow}$ is the same as in the previous section, though we have added $F^{\rightarrow}_{\eta}(\varphi_\theta(i,j))$ to predict edge directedness; this includes an additional logistic parameter vector $\eta$, as well as an additional feature vector $\varphi_\theta(i,j)$. The specific feature vector we use is

$$\varphi_\theta(i,j) = (1, \theta_{j,1} - \theta_{i,1}, \ldots, \theta_{j,K} - \theta_{i,K}), \quad (7)$$

*i.e.* the direction in which an edge flows between two items is a function of the *difference* between their topic vectors.

**Incorporating Other Types of Features.** We can easily incorporate manifest features into our logistic regressors, which simply become additional dimensions in $\varphi_\theta(i,j)$. We include the difference in price, difference in average (star-) rating, and an indicator that takes the value 1 if the items were manufactured by different companies, allowing the model to capture the fact that users may navigate towards (or away from) cheaper products, better rated products, or products from a certain brand.

Our entire model ultimately captures the following simple intuition: (1) Users navigate between *related* products, which should have similar topics ("what do $a$ and $b$ have in common?"), and (2) The *direction* in which users navigate should be related to the difference between topics ("what does $b$ have that $a$ doesn't?"). Ultimately, all of the above machinery has been designed to discover topics and predictors that capture this intuition.

**Learning Multiple Graphs.** Next we must generalize our approach to simultaneously learn multiple types of relationships. In our case we wish to discover a graph of products that users might purchase *instead* (substitute products), as well as a graph of products users might purchase *in addition* (complementary products). Naturally, one could train models independently for each type of relationship. But then one would have two sets of topics, and two predictors that could be used to predict links in each graph.

Instead we decide to extend the model from the previous section so that it can predict multiple types of relations simultaneously. We do this by discovering a single set of topics that work well with *multiple* logistic predictors. This is a small change from the previous model of (eq. 6):

$$L(y, \mathcal{T} | \beta, \eta, \theta, \phi, z) = \overbrace{\prod_{d \in \mathcal{T}} \prod_{j=1}^{N_d} \theta_{z_{d,j}} \phi_{z_{d,j}, w_{d,j}}}^{\text{corpus likelihood}}$$
$$\underbrace{\prod_{g \in G} \left\{ \prod_{(i,j) \in \mathcal{E}_g} F^{\leftrightarrow}_{\beta_g}(\psi_\theta(i,j)) F^{\rightarrow}_{\eta_g}(\varphi_\theta(i,j))(1 - F^{\rightarrow}_{\eta_g}(\varphi_\theta(j,i))) \prod_{(i,j) \in \bar{\mathcal{E}}_g} (1 - F^{\leftrightarrow}_{\beta_g}(\psi_\theta(i,j))) \right\}}_{\text{accuracy of the predictors } \beta_g \text{ and } \eta_g \textit{ for the graph } g}. \quad (8)$$

where each graph $g \in G$ contains all relations of a particular type.

Note that we learn separate predictors $\beta_g$ and $\eta_g$ for each graph $g$, but we learn a *single* set of topics ($\theta$) and features ($\psi$ and $\varphi$) that work well for all graphs simultaneously. We adopt this approach because it provides a larger training set that is more robust to overfitting compared to training two models separately. Moreover it means that both logistic regressors operate in the same feature space; this means that by carefully constructing our labeled training set (to be described in the following section), we can train the model not only to predict substitute and complementary relationships, but also to *differentiate* between the two, by training it to identify substitutes as non-complements and vice versa.

### 3.2.3 *Sparse Topics via Product Hierarchies*

Our goal is to learn topic models on corpora with millions of products and hundreds of topics. However, training models with hundreds of topics per product is not practical, nor is it realistic from a modeling perspective. Rather, each product should draw from a small number of topics, which can be encoded using a sparse representation [13]. To achieve this we enforce sparsity through a specific type of hierarchical topic representation that makes use of

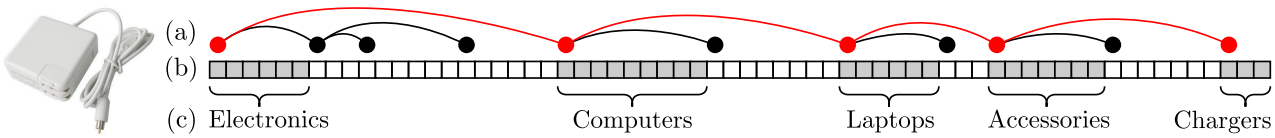


Figure 3: A demonstration of our topic hierarchy. A product (left) is shown with its associated topics (right). (a) the category tree (b) the topic vector (c) the product's ground-truth category. The product's position in the category tree is highlighted in red, and the set of topics that are 'activated' is highlighted in gray.

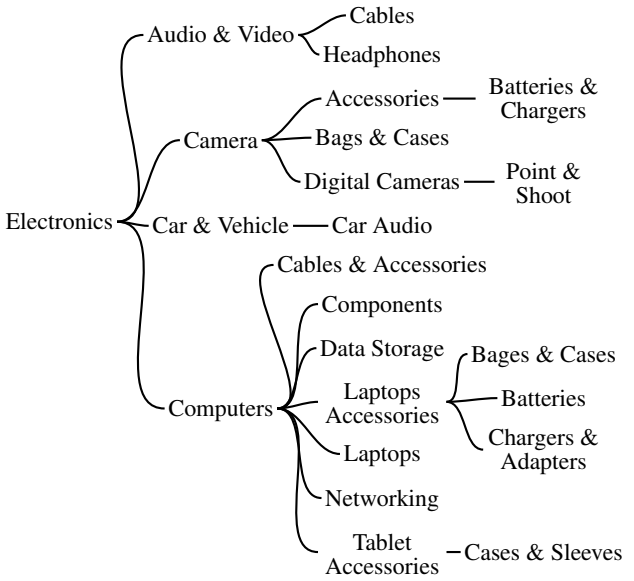


Figure 2: Part of the product hierarchy for *Amazon* Electronics products (the complete tree, even for Electronics alone, is too large to include here).

an explicit category tree, such as the one available from *Amazon*.

An example of the product hierarchy we obtain from *Amazon* is shown in Figure 2. The category of each product is a node in this tree (though not necessarily a leaf node); some products may also belong to multiple categories simultaneously.

We build our topic representation using the following scheme: First, each product is represented by a path, or more simply a set of nodes, in the category tree. For products belonging to multiple categories, we take the union of those paths. Second, each topic is associated with a particular node in the category tree. Every time we observe, say, a thousand instances of a node, we associate an additional topic with that node, up to some maximum. In this way we have many topics associated with popular categories (like 'Electronics') and fewer topics associated with more obscure categories. Third, we use a sparse representation for each product's topic vector. Specifically, if a product occupies a particular path in the category tree, then it can only draw words from topics associated with nodes in that path. In this way, even though our model may have hundreds of topics, only around 10-20 of these will be 'active' for any particular product. This is not only necessary from a scalability standpoint, but it also helps the model quickly converge to meaningful topic representations.

This process is depicted in Figure 3. Here a product like a laptop charger draws from 'generic' topics that apply to all electronics products, as well as topics that are specific to laptops or chargers; but it cannot draw from topics that are specific to mobile phones or laptop cases (for example). Thus all products have some high-level categories in common, but are also assumed to use their own unique sub-category specific language. Then, at the lowest level, each leaf node in the category tree is associated with multiple topics; thus we might learn several 'microcategories' of laptop chargers, *e.g.* for different laptop types, price points, or brands. We present some examples of the types of microcategory we discover in Section 4.6.

## 3.3 Optimization and Training

Optimizing an objective such as the one in (eq. 8) is a difficult task, for instance it is certainly non-convex.[1] We solve it using the following EM-like procedure, in which we alternate between optimizing the model parameters $\Theta = (\beta, \eta, \theta, \phi)$ and topic assignments (latent variables) $z$:

$$\text{update } \Theta^{(t)} = \underset{\Theta}{\operatorname{argmax}}\, l(y, \mathcal{T}|\beta, \eta, \theta, \phi, z^{(t-1)}) \quad (9)$$

$$\text{sample } z_{d,j}^{(t)} \text{ with probability } p(z_{d,j}^{(t)} = k) = \theta_{d,k}\phi_{k,w_{d,j}}^{(t)}, \quad (10)$$

where $l(y, \mathcal{T}|\beta, \eta, \theta, \phi, z)$ is the log-likelihood from (eq. 8). To generate initial values for $\Theta$ and $z$ we initialize continuous parameters and topics uniformly at random (continuous parameters are sampled from $[0, 1)$).

In the first step (eq. 9), topic assignments for each word ($z$) are fixed. We fit the remaining terms, $\beta$, $\eta$, $\theta$, and $\phi$, by gradient ascent. We use the Hybrid LBFGS solver of [19], a quasi-Newton method for non-linear optimization of problems with many variables [25]. Computing the partial derivatives themselves, while computationally expensive, is naïvely parallelizable over edges in $\mathcal{E}$ and documents (*i.e.*, products) in $\mathcal{T}$.

The second step iterates through all products $d$ and all word positions $j$ and updates topic assignments. As with LDA, we assign each word to a topic (an integer between 1 and $K$) randomly, with probability proportional to the likelihood of that topic occurring with that word. The expression $\theta_{d,k}\phi_{k,w_{d,j}}$ is the probability of the topic $k$ for the product $d$ ($\theta_{d,k}$), multiplied by the probability of the word at position $j$ ($w_{d,j}$) being used in topic $k$ ($\phi_{k,w_{d,j}}$).

# 4. EXPERIMENTS

Next we evaluate *Sceptre*. We first describe the data as well as the baselines and then proceed with experimental results.

## 4.1 Data

We use data crawled from *Amazon.com*, whose characteristics are shown in Table 2. This data was collected by performing a breadth-first search on the user-product-review graph until termination, meaning that it is a fairly comprehensive collection of English-language product data. We split the full dataset into top-level categories, *e.g.* Books, Movies, Music. We do this mainly for practical reasons, as it allows each model and dataset to fit in memory on a

[1] It is smooth, and multiple local minima can be found by permuting the order of topics and logistic parameters.

| Category | Users | Items | Reviews | Edges |
|---|---|---|---|---|
| Men's Clothing | 1.25M | 371K | 8.20M | 8.22M |
| Women's Clothing | 1.82M | 838K | 14.5M | 17.5M |
| Music | 1.13M | 557K | 6.40M | 7.98M |
| Movies | 2.11M | 208K | 6.17M | 4.96M |
| Electronics | 4.25M | 498K | 11.4M | 7.87M |
| Books | 8.20M | 2.37M | 25.9M | 50.0M |
| All | 21.0M | 9.35M | 144M | 237M |

Table 2: Dataset statistics for a selection of categories on *Amazon*.

single machine (requiring around 64GB RAM and 2-3 days to run our largest experiment). Note that splitting the data in this way has little impact on performance, as there are few links that cross top-level categories, and the hierarchical nature of our model means that few parameters are shared across categories.

To obtain ground-truth for pairs of substitutable and complementary products we also crawl graphs of four types from *Amazon*:

1. 'Users who viewed $x$ also viewed $y$'; 91M edges.
2. 'Users who viewed $x$ eventually bought $y$'; 8.18M edges.
3. 'Users who bought $x$ also bought $y$'; 133M edges.
4. 'Users frequently bought $x$ and $y$ together'; 4.6M edges.

We refer to edges of type 1 and 2 as *substitutes* and edges of type 3 or 4 as *complements*, though we focus on 'also viewed' and 'also bought' links in our experiments, since these form the vast majority of the dataset. Note the minor differences between certain edge types, *e.g.* edges of type 4 indicate that two items were purchased as part of a single basket, rather than across sessions.

## 4.2 Experimental Setting

We split our training data ($\mathcal{E}$ and $\bar{\mathcal{E}}$) into 80% training, 10% validation, 10% test, discarding products with fewer than 20 reviews. In all cases we report the error on the test set. The iterative fitting process described in (eqs. 9 and 10) continues until no further improvement is gained on the validation set.

**Sampling Non-Edges.** Since it is impractical to train on *all* pairs of non-links, we start by building a balanced dataset by sampling as many non-links as there are links (*i.e.*, $|\mathcal{E}| = |\bar{\mathcal{E}}|$).

However, we must be careful about how non-links (*i.e.*, negative examples) are sampled. Sampling random pairs of unrelated products makes negative examples very 'easy' to classify; rather, since we want the model to be able to *differentiate* between edge types, we treat substitute links as negative examples of complementary edges and vice versa. Thus, we explicitly train the model to identify substitutes as non-complements and vice versa (in addition to a random sample of non-edges). This does not make prediction 'easier', but it helps the model to learn a better separation between the two edge types, by explicitly training it to learn distinct notions of the two concepts.

In the following, we consider both link prediction and ranking tasks: (1) to estimate for a pair of products whether they are related, and (2) for a given query, rank those items that are most likely to be related. We first describe the baselines we compare against.

## 4.3 Baselines

**Random.** Link probabilities $F_\beta$ and $F_\eta$ are replaced with random numbers between 0 and 1. Note that since both predictors have to 'fire' to predict a relation, random classification identifies 75% of directed edges as non-links; imbalance in the number of positive vs. negative relations of each type (due to our sampling procedure described above) explains why the performance of random classification is slightly different across experiments.

**LDA + logistic regression (LDA).** Rather than training topic models and logistic parameters simultaneously, this baseline *first* trains a topic model and *then* trains logistic classifiers on the pre-trained topics. This baseline assesses our claim that *Sceptre* learns topics that are 'good as' features for edge prediction, by comparing it to a model whose topics were *not* trained specifically for this purpose. We used *Vowpal Wabbit* to pre-train the topic model, and fit models with $K = 100$ topics for each *Amazon* category.

We also experimented with a baseline in which features were defined over *words* rather than topics. That is, topics $\theta_i$ for each product are replaced by TF-IDF scores for words in its reviews [20]. Logistic parameters $\beta$ and $\eta$ are then trained to determine which tf-idf-weighted words are good at predicting the presence or absence of edges. This baseline was uniformly weaker than our other baselines, so we shall not discuss its performance further.

**Category-Tree (CT).** Since we make use of *Amazon*'s category tree when building our model, it is worth questioning the extent to which the performance of our model owes to our decision to use this source of data. For this baseline, we compute the co-counts between categories $c_1 \rightarrow c_2$ that are observed in our training data. Then we predict that an edge exists if it is among the 50$^{\text{th}}$ percentile of most commonly co-occurring categories. In other words this baseline 'lifts' links to the level of categories rather than individual products.[2]

**Item-to-Item Collaborative Filtering (CF).** In 2003 *Amazon* reported that their own recommendation solution was a collaborative-filtering approach, that identified items that had been browsed or purchased by similar sets of users [17]. This baseline follows the same procedure, though in lieu of actual browsing or purchasing data we consider sets of users who have *reviewed* each item. We then proceed by computing for each pair of products $a$ and $b$ the cosine similarity between the set of users who reviewed $a$ and the set of users who reviewed $b$. Sorting by similarity generates a ranked list of recommendations for each product. Since this method is not probabilistic we only report its performance at ranking tasks.

## 4.4 Link Prediction and Ranking

**Link Prediction.** Our first goal is to predict for a given pair of products $(a, b)$, and a graph type $g$, whether there is a link from $a$ to $b$ in $\mathcal{E}_g$. We optimize exactly the objective in (eq. 8). Note that a prediction is correct when

- for each test edge (in each graph): $a \rightarrow b$, $F_\theta^{\leftrightarrow}(\psi(a,b),\beta) > 0$ ***and*** $F_\theta^{\rightarrow}(\varphi(a,b),\eta) > 0$
- for each *non*-edge $a \not\rightarrow b$, $F_\theta^{\leftrightarrow}(\psi(a,b),\beta) \leq 0$ ***or*** $F_\theta^{\rightarrow}(\varphi(a,b),\eta) \leq 0$,

in other words the model must correctly predict both that the link exists and its direction.

Results are shown in Table 3 for each of the datasets in Table 2. We also show results from 'Baby' clothes, to demonstrate that performance does not degrade significantly on a (relatively) smaller dataset (43k products). 'Substitute' links were unavailable for the vast majority of products from Music and Movies in our crawl, so results are not shown. We summarize the main findings from this table as follows:

1. *Sceptre* is able to accurately predict both substitute and complement links across all product categories, with performance

[2]We experimented with several variations on this theme, and this approach yielded the best performance.

being especially accurate for clothing and electronics products. Accuracy is between 85.57-96.76% for the binary prediction tasks we considered.

2. Prediction of 'substitute' links is uniformly more accurate than 'complement' links for all methods, both in absolute (left two columns) and relative (right two columns) terms. This matches our intuition that substitute links should be 'easier' to predict, as they essentially correspond to some notion of similarity, whereas the semantics of complements are more subtle.
3. The performance of the baselines is variable. For substitute links, our LDA baseline obtains reasonable performance on Books and Electronics, whereas the Category Tree (CT) baseline is better for Clothing. In fact, the CT baseline performs surprisingly well at predicting substitute links, for the simple reason that substitutable products often belong to the same category as each other.
4. None of the baselines yield satisfactory performance when predicting complement links. Thus we conclude that neither the topics uncovered by a regular topic model, nor the category tree alone are capable of capturing the subtle notions of what makes items complementary.

Ultimately we conclude that each of the parts of *Sceptre* contribute to its accurate performance. Category information is helpful, but alone is not useful to predict complements; and simultaneous training of topic models and link prediction is necessary to learn useful topic representations.

**Ranking.** In many applications distinguishing links from non-links is not enough as for each product we would like to recommend a limited number of substitutes and complements. Thus, it is important that relevant items (*i.e.*, relevant relations) are ranked higher than irrelevant ones, regardless of the likelihood that the model assigns to each recommendation.

A standard measure to evaluate ranking methods is the precision@k. Given a set of recommended relations of a given type $rec$, and a set of known-relevant products $rel$ (*i.e.*, ground-truth links) the *precision* is defined as

$$\text{precision} = |rel \cap rec|/|rec|, \tag{11}$$

*i.e.*, the fraction of recommended relations that were relevant. The precision@k is then the precision obtained given a fixed budget, *i.e.*, when $|rec| = k$. This is relevant when only a small number of recommendations can be surfaced to the user, where it is important that relevant products appear amongst the first few suggestions.

Figure 4 reports the precision@k on Men's and Women's clothing. Note that we naturally discard candidate links that appeared during training. This leaves only a small number of relevant products for each query item in the corpus—the random baseline (which up to noise should be flat) has precision around $5 \times 10^{-5}$, indicating that only around 5 in 100,000 products are labeled as 'relevant' in this experiment. This, in addition to the fact that many relevant items may not be labeled as such (there are presumably *thousands* of pairs of substitutable pants in our corpus, but only 30 or so are recommended for each product) highlights the incredible difficulty of obtaining high precision scores for this task.

Overall, collaborative filtering is one to two orders-of-magnitude more accurate than random rankings, while *Sceptre* is an order of magnitude more accurate again (our LDA and TF-IDF baselines were less accurate than collaborative filtering and are not shown).[3]

[3]Note that collaborative filtering is done here at the level of *reviewed* products, which is naturally much sparser than the purchase and browsing data used to produce the ground-truth.

| | | Accuracy | | Error reduction vs. random | |
|---|---|---|---|---|---|
| Category | Method | Subst. | Compl. | Subst. | Compl. |
| Men's Clothing | Random | 60.27% | 57.70% | 0.0% | 0.0% |
| | LDA | 70.62% | 65.95% | 26.05% | 19.50% |
| | CT | 78.69% | 61.06% | 46.38% | 7.946% |
| | *Sceptre* | 96.69% | 94.06% | 91.67% | 85.97% |
| Women's Clothing | Random | 60.35% | 56.67% | 0.0% | 0.0% |
| | LDA | 70.70% | 64.80% | 26.11% | 18.75% |
| | CT | 81.05% | 69.08% | 52.21% | 28.63% |
| | *Sceptre* | 95.87% | 94.14% | 89.59% | 86.47% |
| Music | Random | - | 50.18% | - | 0.0% |
| | LDA | - | 52.39% | - | 4.428% |
| | CT | - | 57.02% | - | 13.71% |
| | *Sceptre* | - | 90.43% | - | 80.78% |
| Movies | Random | - | 51.22% | - | 0.0% |
| | LDA | - | 54.26% | - | 6.235% |
| | CT | - | 66.34% | - | 30.99% |
| | *Sceptre* | - | 85.57% | - | 70.42% |
| Electronics | Random | 69.98% | 55.67% | 0.0% | 0.0% |
| | LDA | 89.90% | 61.90% | 66.35% | 14.06% |
| | CT | 87.26% | 60.18% | 57.57% | 10.17% |
| | *Sceptre* | 95.70% | 88.80% | 85.69% | 74.74% |
| Books | Random | 69.93% | 55.35% | 0.0% | 0.0% |
| | LDA | 89.91% | 60.59% | 66.47% | 11.75% |
| | CT | 87.80% | 66.28% | 59.42% | 24.49% |
| | *Sceptre* | 93.76% | 89.86% | 79.25% | 77.29% |
| Baby Clothes | random | 62.93% | 52.47% | 0.0% | 0.0% |
| | LDA | 75.86% | 54.73% | 34.89% | 4.75% |
| | CT | 79.31% | 64.56% | 44.18% | 25.43% |
| | *Sceptre* | 92.18% | 93.65% | 78.91% | 86.65% |
| Average | *Sceptre* | 94.83% | 90.23% | 85.02% | 80.33% |

Table 3: Link prediction accuracy for substitute and complement links (the former are not available for the majority of Music/Movies products in our dataset). Absolute performance is shown at left, reduction in error vs. random classification at right.

Examples of recommendations generated by *Sceptre* are shown in Figure 6.

## 4.5 Cold-Start Prediction without Reviews

Although it is promising that we are able to recommend substitutes and complements from the text of product reviews, an obvious question remains as to what can be done for *new* products, that do not yet have any reviews associated with them, known as the 'cold-start' problem in recommender systems [16, 24, 27, 30, 36]. To address this problem, we note that *Sceptre* merely requires that we have some source of text associated with each linked item in order to learn a model of which products are likely to be related.

To evaluate the possibility of using sources of text other than reviews, we collected descriptive text about each item in our *Amazon Books* catalog, including blurbs, 'about the author' snippets, and editorial reviews. We also collected manufacturer's descriptions for a subset of our Electronics data. Training on these sources of data, *Sceptre* obtained accuracies between 91.28% and 93.67% at predicting substitutes and complements (see Table 4). This result implies that training is possible on diverse sources of text beyond just

| Category | Accuracy | | Error reduction vs. random | |
|---|---|---|---|---|
| | Subst. | Compl. | Subst. | Compl. |
| Electronics, cold-start | 91.28% | 93.22% | 70.95% | 84.71% |
| Books, cold-start | 96.76% | 93.67% | 89.22% | 85.81% |

Table 4: Link prediction accuracy using cold-start data (manufacturer's and editorial descriptions).

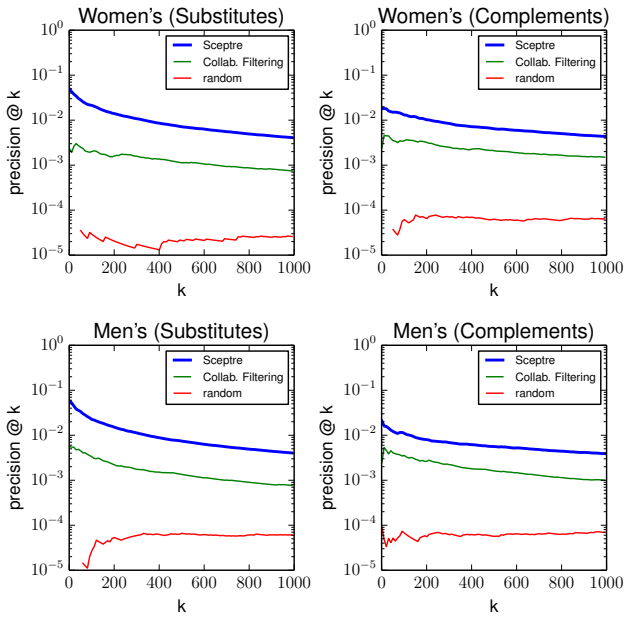


Figure 4: Precision@k for Women's and Men's clothing.

product reviews, and that *Sceptre* can be applied in cold-start settings, even when no reviews are available.[4]

## 4.6 Topic Analysis

Next we analyze the types of topics discovered by *Sceptre*. As we recall from Section 3.2.3, each topic is associated with a node in *Amazon*'s category tree. But, just as a high-level category such as clothing can naturally be separated into finer-grained categories like socks, shoes, hats, pants, dresses (etc.), we hope that *Sceptre* will discover even subtler groupings of products that are not immediately obvious from the hand-labeled category hierarchy.

Table 5 shows some of the topics discovered by *Sceptre*, on two *Amazon* categories: Electronics and Men's Clothing. We pruned our dictionary by using adjectives, nouns, and adjective-noun pairs (as determined by WordNet [9]), as well as any words appearing in the 'brand' field of our crawled data. For visualization we compute the 10 highest-weight words from all topics, after first subtracting a 'background' topic containing the average weight across all topics. That is for each topic $k$ we report the 10 highest values of

$$\phi_k - \underbrace{\frac{1}{K}\sum_{k'}\phi_{k'}}_{\text{background word distribution}}. \quad (12)$$

By doing so, stopwords and other language common to all topics appears only in the background distribution.

[4]Note that this is not directly comparable to review-based results since different subsets of our corpus have reviews vs. descriptions.

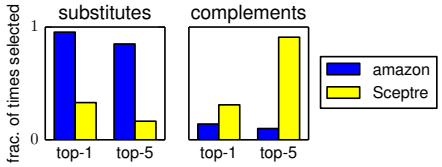


Figure 5: Results of our user study. Users were asked to select which recommendations (ours or Amazon's) were preferable substitutes/complements (users could also select neither or both).

The topics we obtain are closely aligned with categories from *Amazon* (*e.g.* electronics topic e111, or clothing topic c110), though this is to be expected since our topic model is built on top of an explicit hierarchy as in Figure 3. However, we note that finer-grained 'microcategories' are discovered that are not explicitly present in the hierarchy, *e.g.* high-end headphones are distinguished from cheaper models (e89 and e99), and running shoes are distinguished based on distinct running styles (c133, c134, and c156).

We also note that brand words predominate in several topics, *e.g.* high-end headphones can be identified by words like 'Sennheiser', 'AKG' etc. (e89), and high-end t-shirts can be identified by words like 'Ralph Lauren' and 'Geoffrey Beene' (c52). At the extreme end, a *single product* may have its own topic, *e.g.* the popular 'three-wolf moon' shirt (c49), whose reviews have already inspired academic discussion [29]. Here the product's high popularity and unique word distribution means that dedicating it an entire topic substantially increases the corpus likelihood in (eq. 8). Note that we only show a fraction of the full set of topics discovered by our model; other common camera brands (etc.) are covered among the large range of topics not shown here.

Finally, while some topics are highly specific, like those referring to individual products or brands, others are more generic, such as clothing topics c9 and c24. Such topics tend to appear toward the top of the category hierarchy (see Fig. 3), for example the topic c9 is associated with the 'Clothing' node, whereas c24 is associated with its child, 'Clothing: Men', of which all other topics in Table 5 are descendants. Intuitively, these are much like 'background' distributions, containing words that are relevant to the majority of clothing products, like durability, fit, warmth, color, packaging, etc.

## 4.7 User Study

Finally we perform a user study to evaluate the quality of the recommendations produced by *Sceptre*. Naturally we would not expect that a fully-supervised algorithm would produce predictions that were somehow 'better' than the ground-truth used to train it. However, we hope *Sceptre* may correct for some noise in the ground-truth, since while users may often buy multiple pairs of jeans together (for example) we are explicitly training the system to identify complementary items that would *not* be substitutable.

We used Mechanical Turk to compare *Sceptre*'s recommendations to *Amazon*'s 'also viewed' and 'also bought' suggestions, for a random sample of 200 Clothing items. Human judges identified which recommendations they felt were acceptable substitutes and complements (surfaced in a random order without labels; a screenshot is shown in Fig. 6d). Judges evaluated the top recommendation, and top-5 recommendations separately, yielding results shown in Figure 5.

We see here that Amazon's 'also viewed' links generate preferable substitutes, indicating that large volumes of browsing data yield acceptable substitute relationships with relatively little noise. On the other hand, *Sceptre*'s complement links are overwhelmingly

| Electronics | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| e111 | e92 | e75 | e79 | e78 | e50 | e69 | e85 | e96 | e89 | e99 |
| cameras | portable speakers | cases | Samsung cases | heavy-duty cases | styli | batteries | portable radios | car radios | high-end headphones | budget headphones |
| camera | little speaker | leather | Galaxy | Otterbox | pen | batteries | radio | radio | bass | bass |
| zoom | bose | case | elastic | Defender | tip | battery | weather | Pioneer | Sennheiser | Skullcandy |
| pictures | portable speaker | soft | magnets | protection | Bamboo | charged | crank | factory | Bose | sound |
| Kodak | small speaker | Roocase | Samsung | bulky | Wacom | rechargeable | solar | Metra | Shure | bud |
| Canon | sound | velcro | leather | kids | styli | oem | Eton | Ford | Beats | outside noise |
| flash | iHome | closed | closed | shell | gloves | Sanyo | Baofeng | dash | Koss | another pair |
| digital | bass | material | auto | Survivor | Friendly Swede | Lenmar | radio reception | Honda | Akg | comfortable |
| optical | wireless speaker | snug | closing | protected | pencil | alkaline | miles | Jeep | music | gym |
| taken | great speaker | protection | elastic strap | safe | capacitive | Energizer | fm | wiring | classical | Beats |
| picture | mini speaker | standing | cover | protective | precise | full charge | alert | deck | Klipsch | head |

| Men's clothing | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| c44 | c107 | c75 | c49 | c52 | c110 | c156 | c134 | c133 | c24 | c9 |
| dress shirts | dress shoes | dress pants | three-wolf shirt | polo shirts | boots | minimalist running | athletic running | sports shoes | generic clothing | generic clothing |
| sleeves | leather | expandable | wolf | Polo | Bates | running | Balance | court | dry | same |
| arms | sole | expandable waist | moon | Lauren | Red Wing | trail | New | play | cold | durable |
| neck | dress | Dockers | three | Ralph | leather | barefoot | wide | Nike | working | store |
| shoulders | brown | iron | power | Beene | good boot | Altra | running | running shoe | short | different |
| dress shirt | dress shoe | khaki | trailer | nice shirt | casual boot | running shoe | series | running | hot | two |
| dress | polish | stretch waist | hair | Geoffrey | dress boot | minimalist | feet | games | weather | brand |
| jacket | brown pair | hidden | man | great shirt | right boot | zero drop | usa | light shoe | tight | comfort |
| long sleeve | toe | ironed | short-sleeve | quality shirt | motorcycle | road | cross training | great shoe | cool | fine |
| iron | looking shoe | dress pant | magic | white shirt | Wings | glove | athletic shoe | support | down | tight |
| tucked | formal | elastic waist | powerful | fitted shirt | Rangers | run | cross | miles | regular | another pair |

Table 5: A selection of topics from Electronics and Men's Clothing along with our labels for each topic. Top 10 words/bigrams from each topic are shown after subtracting the background distribution. Capital letters denote brand names (Bamboo, Wacom, Red Wing, etc.).

preferred, suggesting that our decision to model complements as non-substitutes qualitatively improves performance.

# 5. BUILDING THE PRODUCT GRAPH

Having produced ranked lists of recommended relationships, our final task is to surface these recommendations to a potential user of our system in a meaningful way.

While conceptually simple, comparing all products against all others quickly becomes impractical in a corpus with millions of products. Our goal here is to rank all links, and surface those which have the highest likelihood under the model. That is, for each graph type $g$ we would like to recommend

$$rec_g(i) = \underset{S \in (\mathcal{T} \setminus \{i\})^R}{\operatorname{argmax}} \sum_{j \in S} F^{\leftrightarrow}_{\beta_g}(\psi_\theta(i,j)) F^{\rightarrow}_{\eta_g}(\varphi_\theta(i,j)), \quad (13)$$

where $S \in (\mathcal{T} \setminus \{i\})^R$ is a set of $R$ products other than $i$ itself.

While computing the score for a single candidate edge is very fast ($O(K)$ operations), on a dataset with millions of products this already results in an unacceptable delay when ranking all possible recommendations. Similar to [17] we implemented two modifications that make this enumeration procedure feasible (on the order of a few milliseconds). The first is to ignore obscure products by limiting the search space by some popularity threshold; we consider the hundred-thousand most popular products per-category when generating new recommendations. The second is to cull the search space using the category tree explicitly; *e.g.* when browsing for running shoes we can ignore, say, camera equipment and limit our search to clothing and shoes. Specifically, we only consider items belonging to the same category, its parent category, its child categories, and its sibling categories (in other words, its 'immediate family'). It is very rare that the top-ranked recommendations belong to distant categories, so this has minimal impact on performance.

Another issue is that of adding new products to the system. Naturally, it is not feasible to re-train the system every time a new product is added. However, this is thankfully not necessary, as the introduction of a small number of products will not fundamentally change the word distribution $\phi$. Thus it is simply a matter of estimating the product's topic distribution under the existing model, as can be done using LDA [4].

When assembling our user interface (see Figs. 1 and 6) we use the discovered topics from Section 4.6 to 'explain' recommendations to users, by selecting sentences whose language best explains why the recommended product was predicted. Specifically, we highlight sentences whose words yield the largest response to $F^{\rightarrow}_{\eta_g}$.

**Reproducing Sceptre.** All data and code used in this paper, as well as the interface from Figure 6 is available on the first author's webpage: `http://cseweb.ucsd.edu/~jmcauley/`.

# 6. CONCLUSION

A useful recommender system must produce recommendations that not only match our preferences, but which are also *relevant* to our current topic of interest. For a user browsing a particular product, two useful notions of relevant recommendations include *substitutes* and *complements*: products that can be purchased *instead of* each other, and products that can be purchased *in addition to* each other. In this paper, our goal has been to *learn* these concepts from product features, especially from the text of their reviews.

We have presented *Sceptre*, a model for predicting and understanding relationships between linked products. We have applied this to the problem of identifying substitutable and complementary products on a large collection of *Amazon* data, including 144 million reviews and 237 million ground-truth relationships based on browsing and co-purchasing logs.

(a) Men's clothing (b) Women's clothing (c) Electronics (d) mturk interface

Figure 6: (a,b,c) Examples of recommendations produced by *Sceptre*; the top of each subfigure shows the query product, the left column shows substitutes recommended by *Sceptre*, and the right column shows complements. (d) Interface used to evaluate *Sceptre* on Mechanical Turk; Turkers are shown lists of items suggested by *Amazon* (*i.e.*, the ground-truth) and *Sceptre* and must identify which lists they prefer.

## References


[1] R. Balasubramanyan and W. Cohen. Block-LDA: Jointly modeling entity-annotated text and entity-entity links. In *SDM*, 2011.

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

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

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

[5] D. M. Blei, T. Griffiths, M. Jordan, and J. Tenenbaum. Hierarchical topic models and the nested Chinese restaurant process. In *NIPS*, 2003.

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

[7] J. Chang and D. Blei. Relational topic models for document networks. In *AIStats*, 2009.

[8] J. Chang, J. Boyd-Graber, and D. Blei. Connections between the lines: augmenting social networks with text. In *KDD*, 2009.

[9] C. Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

[10] M. Gamon, A. Aue, S. Corston-Oliver, and E. Ringger. Pulse: Mining customer opinions from free text. In *IDA*, 2005.

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

[12] S. Jagabathula, N. Mishra, and S. Gollapudi. Shopping for products you don't know you need. In *WSDM*, 2011.

[13] B. Kanagal, A. Ahmed, S. Pandey, V. Josifovski, J. Yuan, and L. Garcia-Pueyo. Supercharging recommender systems using taxonomies for learning user purchase behavior. *VLDB*, 2012.

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

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

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

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

[18] Y. Liu, A. Niculescu-Mizil, and W. Gryc. Topic-link LDA: joint models of topic and author community. In *ICML*, 2009.

[19] Y. Liu, W. Wang, B. Lévy, F. Sun, D.-M. Yan, L. Lu, and C. Yang. On centroidal Voronoi tessellation – energy smoothness and fast computation. *ACM Trans. on Graphics*, 2009.

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

[21] A. Mas-Colell, M. Whinston, and J. Green. *Microeconomic Theory*. Oxford University Press, 1995.

[22] J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*, 2013.

[23] S. Moghaddam and M. Ester. On the design of LDA models for aspect-based opinion mining. In *CIKM*, 2012.

[24] S. Moghaddam and M. Ester. The FLDA model for aspect-based opinion mining: Addressing the cold start problem. In *WWW*, 2013.

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

[26] D. Panigrahi and S. Gollapudi. Result enrichment in commerce search using browse trails. In *WSDM*, 2011.

[27] S.-T. Park and W. Chu. Pairwise preference regression for cold-start recommendation. In *RecSys*, 2009.

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

[29] A. Reyes and P. Rosso. Mining subjective knowledge from customer reviews: A specific case of irony detection. In *HLT*, 2011.

[30] A. Schein, A. Popescul, L. Ungar, and D. Pennock. Methods and metrics for cold-start recommendations. In *SIGIR*, 2002.

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

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

[33] D. Vu, A. Asuncion, D. Hunter, and P. Smyth. Dynamic egocentric models for citation networks. In *ICML*, 2011.

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

[35] J. Zheng, X. Wu, J. Niu, and A. Bolivar. Substitutes or complements: another step forward in recommendations. In *EC*, 2009.

[36] K. Zhou, S.-H. Yang, and H. Zha. Functional matrix factorizations for cold-start recommendation. In *SIGIR*, 2011.