E-commerce in Your Inbox: Product Recommendations at Scale

Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati
Yahoo Labs
701 First Avenue, Sunnyvale, USA
{mihajlo, vladan, nemanja, narayanb}@yahoo-inc.com

Jaikit Savla, Varun Bhagwan, Doug Sharp
Yahoo, Inc.
701 First Avenue, Sunnyvale, USA
{jaikit, vbhagwan, dsharp}@yahoo-inc.com

ABSTRACT

In recent years online advertising has become increasingly ubiquitous and effective. Advertisements shown to visitors fund sites and apps that publish digital content, manage social networks, and operate e-mail services. Given such large variety of internet resources, determining an appropriate type of advertising for a given platform has become critical to financial success. Native advertisements, namely ads that are similar in look and feel to content, have had great success in news and social feeds. However, to date there has not been a winning formula for ads in e-mail clients. In this paper we describe a system that leverages user purchase history determined from e-mail receipts to deliver highly personalized product ads to Yahoo Mail users. We propose to use a novel neural language-based algorithm specifically tailored for delivering effective product recommendations, which was evaluated against baselines that included showing popular products and products predicted based on co-occurrence. We conducted rigorous offline testing using a large-scale product purchase data set, covering purchases of more than 29 million users from 172 e-commerce websites. Ads in the form of product recommendations were successfully tested on online traffic, where we observed a steady 9% lift in click-through rates over other ad formats in mail, as well as comparable lift in conversion rates. Following successful tests, the system was launched into production during the holiday season of 2014.

Categories and Subject Descriptors

H.2.8 [**Database applications**]: Data Mining

Keywords

Data mining; computational advertising; audience modeling

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 ACM 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 10-13, 2015, Sydney, NSW, Australia
© 2015 ACM. ISBN 978-1-4503-3664-2/15/08...$15.00
DOI: http://dx.doi.org/10.1145/2783258.2788627.

1. INTRODUCTION

Hundreds of millions of people around the world visit their e-mail inboxes daily, mostly to communicate with their contacts, although a significant fraction of time is spent checking utility bills, reading newsletters, and tracking purchases. To monetize this overwhelming amount of traffic, e-mail clients typically show display ads in a form of images alongside native e-mail content. Convincing users to exit the "e-mail mode", characterized by a relentless focus on the task of dealing with their mail, in order to enter a mode where they are willing to click on ads is a challenging task. Effective personalization and targeting [11], where the goal is to find the best matching ads to be displayed for each individual user, is essential to tackling this problem, as ads need to be highly relevant to overcome user's inclination to focus narrowly on the e-mail task. In addition to financial gains for online businesses [26], proactively tailoring advertisements to the tastes of each individual consumer also leads to an improved user experience, and can help with increasing user loyalty and retention [2].

Inbound e-mails are still insufficiently explored and exploited for the purposes of ad targeting, while arguably representing a treasure trove of monetizable data. A recent study [13] showed that only 10% of inbound volume represents human-generated e-mails. Furthermore, out of the remaining 90% of traffic more than 22% represents e-mails related to online shopping. Given that a significant percentage of overall traffic has commercial intent, a popular form of targeted advertising is mail retargeting (MRT), where advertisers target users who previously received e-mails from certain commercial web domains. These e-mails present a strong signal useful for targeting campaigns, as they give a broad picture about each customer's interests and relationship with commercial domains. A recent paper [14] proposed to make use of this vast potential and presented a clustering method to generate MRT rules, showing that such rules are more accurate than the ones generated by human experts.

However, in order to go beyond MRT rules that utilize only the fact that users and e-commerce sites communicated, advertisers require more detailed data such as purchased product name and price that are often part of e-mail body. E-mail clients have been working with e-commerce and travel domains to standardize how e-mails are formatted, resulting in schemas maintained by the *schema.org* com-

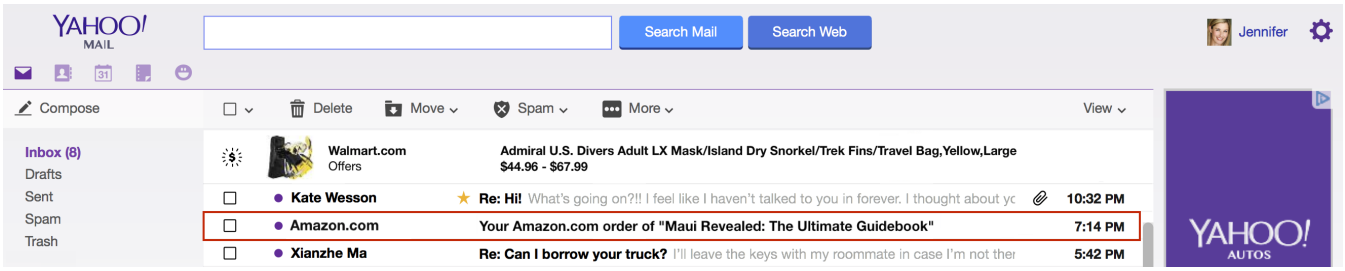


Figure 1: Product recommendations in Yahoo Mail

munity[1]. With more and more e-commerce sites using standard schemas, e-mail clients can provide more personalized user notifications, such as package tracking[2] and flight details[3]. In addition, e-mail receipt extraction brings monetization opportunity through product advertising to users based on their individual purchase history. Availability of purchase data from multiple commercial e-mail domains puts the e-mail provider in the unique position to be able to build better recommendation systems than those based on any one commercial e-mail domain alone. In particular, unlike e-commerce websites that make recommendations of type: "Customers who bought X also bought Y", e-mail providers can make recommendations of type: "Customers who bought X from vendor V_1 also bought Y from vendor V_2", allowing much more powerful and effective targeting solutions.

In this paper we tell the story of an end-to-end development of product ads for Yahoo Mail. The effort included developing a product-level purchase prediction algorithm, capable of scaling to millions of users and products. To this end, we propose an approach that embeds products into real-valued, low-dimensional vector space using a neural language model applied to a time series of user purchases. As a result, products with similar contexts (i.e., their surrounding purchases) are mapped to vectors that are nearby in the embedding space. To be able to make meaningful and diverse suggestions about the next product to be purchased, we further cluster the product vectors and model transition probabilities between clusters. The closest products in the embedding space from the most probable clusters are used to form final recommendations for a product.

The product prediction model was trained using a large-scale purchase data set, comprising more than 280 million purchases made by 29 million users, involving 2.1 million unique products. The model was evaluated on a held-out month, where we tested the effectiveness of recommendations in terms of yield rate. In addition, we evaluated several baseline approaches, including showing popular products to all users, showing popular products in various user groups (called *cohorts*, specified by user's gender, age, and location), as well as showing products that are historically frequently purchased after the product a user most recently bought. To mitigate the cold start problem, popular products in user's cohort were used as back-fill recommendations for users without earlier purchases.

Empirical results show that the proposed product-level model is able to more accurately predict purchases than baseline approaches. In the experimental section we also share results of bucket tests on live traffic, where we compared the performance of the baselines in the same ad slot. Our method substantially improves key business metrics, measured in terms of clicks and conversions. Moreover, our product-prediction technique was successfully implemented at large scale, tested in live buckets, and finally launched in production. The system is able to make near-real-time product recommendations with latency of less than $200ms$. In Figure 1 we show an example of our product recommender, where an offer from *walmart.com* is suggested after related purchase was made on *amazon.com* (marked with red).

2. RELATED WORK

In this section we describe related work in mail-based advertising, as well as neural language models that motivated our product recommendation approach.

2.1 Leveraging e-mail data in advertising

Web environment provides content publishers with a means to track user behavior in much greater detail than in an offline settings, including capturing user's registered information and activity logs of user's clicks, page views, searches, website visits, social activities, and interactions with ads. This allows for targeting of users based on their behavior, which is typically referred to as ad targeting [1]. With the rise of big data applications and platforms, machine learning approaches are heavily leveraged to automate the ad targeting process. Within the past several years, there has been a plethora of research papers that explored different aspects of online advertising, each with the goal of maximizing the benefits for advertisers, content publishers, and users.

For any machine learning technique, features used to train a model typically have a critical influence on the performance of the deployed algorithm. Features derived from user events collected by publishers are often used in predicting the user's propensity to click or purchase [9]. However, these features represent only a weak proxy to what publishers and advertisers are actually interested in, namely, user's purchase intent. On the other hand, commercial e-mails in the form of promotions and purchase receipts convey a strong, very direct purchase intent signal that can enable advertisers to reach a high-quality audience. According to the Direct Marketing Association's "National Client E-mail Report"[4], an e-mail user has been identified as having over 10% more value than the average online customer. Despite these

[1]http://schema.org/email, accessed June 2015
[2]yahoomail.tumblr.com/post/107247385566/track-your-packages-now-in-the-yahoo-mail-app, accessed June 2015
[3]venturebeat.com/2014/10/22/yahoo-mail-now-tells-you-about-upcoming-flights-and-events, accessed June 2015

[4]www.powerprodirect.com/statistics, accessed June 2015

encouraging facts, limited work has been done to analyze the potential of features derived from commercial e-mails. A recent work [14] was among the first attempts to investigate the value of commercial e-mail data. The authors applied Sparse Principal Component Analysis (SPCA) on the counts of received e-mails in order to cluster commercial domains, demonstrating significant promise of e-mail data source.

Outside of the e-mail domain, information about user's purchase history is extensively used by e-commerce websites to recommend relevant products to their users [22]. Recommendation systems predict which products a user will most likely be interested in either by exploiting purchase behavior of users with similar interests (referred to as collaborative filtering [22]) or by using user's historical interaction with other products (i.e., context-based recommendation [29]). Unlike these studies, we are not limited to the data from a single website, as purchases extracted from e-mails enable us to gather information from hundreds of different e-commerce websites that can be exploited to learn better product predictions. To the best of our knowledge, this work represents the first study that offers a comprehensive empirical analysis and evaluation of product predictors using e-mail data of such scale and nature.

2.2 Neural language models

In a number of Natural Language Processing (NLP) applications, including information retrieval, part-of-speech tagging, chunking, and many others, specific objectives can all be generalized to the task of assigning a probability value to a sequence of words. To this end, language models have been developed, defining a mathematical model to capture statistical properties of words and the dependencies among them [3, 20]. Traditionally, language model approaches represent each word as a feature vector using a one-hot representation, where a word vector has the same length as the size of a vocabulary, and the position that corresponds to the observed word is equal to 1, and 0 otherwise. However, this approach often exhibits significant limitations in practical tasks, suffering from high dimensionality of the problem and severe data sparsity, resulting in suboptimal performance.

Neural language models have been proposed to address these issues, inducing low-dimensional, distributed embeddings of words by means of neural networks [5, 8, 28]. Such approaches take advantage of the word order in text documents, explicitly modeling the assumption that closer words in the word sequence are statistically more dependent. Historically, inefficient training of the neural network-based models has been an obstacle to their wider applicability, given that the vocabulary size may grow to several millions in practical tasks. However, this issue has been successfully addressed by recent advances in the field, particularly with the development of highly scalable continuous bag-of-words (CBOW) and skip-gram (SG) language models [23, 24] for learning word representations. These powerful, efficient models have shown very promising results in capturing both syntactic and semantic relationships between words in large-scale text corpora, obtaining state-of-the-art results on a plethora of NLP tasks. More recently, the concept of distributed representations has been extended beyond word representations to sentences and paragraphs [10, 21], relational entities [6, 27], general text-based attributes [19], descriptive text of images [18], nodes in graph structure [25], and other applications.

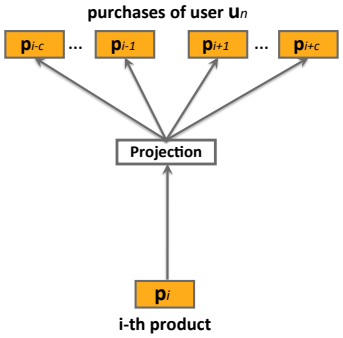


Figure 2: prod2vec skip-gram model

3. PROPOSED APPROACH

In this section we describe the proposed methodology for the task of product recommendations, which leverages information about prior purchases determined from e-mail receipts. To address this task we propose to learn representation of products in low-dimensional space from historical logs using neural language models. Product recommendation can then be performed in the learned embedding space through simple nearest neighbor search.

More specifically, given a set $\mathcal{S}$ of e-mail receipt logs obtained from N users, where user's log $s = (e_1, \ldots, e_M) \in \mathcal{S}$ is defined as an uninterupted sequence of M receipts, and each e-mail receipt $e_m = (p_{m1}, p_{m2}, \ldots p_{mT_m})$ consists of T_m purchased products, our objective is to find D-dimensional real-valued representation $\mathbf{v}_p \in \mathcal{R}^D$ of each product p such that similar products lie nearby in the vector space.

We propose several approaches for learning product representations that address specifics of the recommendations from e-mail receipts. We first propose *prod2vec* method that considers all purchased products independently. We then propose novel *bagged-prod2vec* method that takes into account that some products are listed as purchased together in e-mail receipts, which results in better, more useful product representations. Finally, we present product-to-product and user-to-product recommendation models that make use of the learned representations.

3.1 Low-dimensional product embeddings

prod2vec. The prod2vec model involves learning vector representations of products from e-mail receipt logs by using a notion of a purchase sequence as a "sentence" and products within the sequence as "words", borrowing the terminology from the NLP domain (see Figure 2 for graphical representation of the model). More specifically, prod2vec learns product representations using the skip-gram model [24] by maximizing the objective function over the entire set $\mathcal{S}$ of e-mail receipt logs, defined as follows

$$\mathcal{L} = \sum_{s \in \mathcal{S}} \sum_{p_i \in s} \sum_{-c \le j \le c, j \neq 0} \log \mathbb{P}(p_{i+j} | p_i), \tag{3.1}$$

where products from the same e-mail receipt are ordered arbitrarily. Probability $\mathbb{P}(p_{i+j}|p_i)$ of observing a neighboring product p_{i+j} given the current product p_i is defined using the soft-max function,

$$\mathbb{P}(p_{i+j} | p_i) = \frac{\exp(\mathbf{v}_{p_i}^\top \mathbf{v}'_{p_{i+j}})}{\sum_{p=1}^{P} \exp(\mathbf{v}_{p_i}^\top \mathbf{v}'_p)}, \tag{3.2}$$

where $\mathbf{v}_p$ and $\mathbf{v}'_p$ are the input and output vector representations of product p, c is the length of the context for product sequences, and P is the number of unique products in the vocabulary. From equations (3.1) and (3.2) we see that prod2vec models context of product sequence, where products with similar contexts (i.e., with similar neighboring purchases) will have similar vector representations. However, prod2vec does not explicitly take into account that e-mail receipt may contain multiple products purchased at the same time, which we address by introducing a *bagged* version of prod2vec described below.

bagged-prod2vec. In order to account for the fact that multiple products may be purchased at the same time, we propose a modified skip-gram model that introduces a notion of a shopping bag. As depicted in Figure 3, the model operates at the level of e-mail receipts instead at the level of products. Product vector representations are learned by maximizing a modified objective function over e-mail sequences s, defined as follows

$$\mathcal{L} = \sum_{s \in \mathcal{S}} \sum_{e_m \in s} \sum_{-n \le j \le n, j \neq 0} \sum_{k=1,\ldots,T_m} \log \mathbb{P}(e_{m+j}|p_{mk}). \quad (3.3)$$

Probability $\mathbb{P}(e_{m+j}|p_{mk})$ of observing products from neighboring e-mail receipt e_{m+j}, $e_{m+j} = (p_{m+j,1} \ldots p_{m+j,T_m})$, given the k-th product from m-th e-mail receipt reduces to a product of probabilities $\mathbb{P}(e_{m+j}|p_{mk}) = \mathbb{P}(p_{m+j,1}|p_{mk}) \times \ldots \times \mathbb{P}(p_{m+j,T_m}|p_{mk})$, each defined using soft-max (3.2). Note that the third sum in (3.3) goes over receipts, so the items from the same e-mail receipt do not predict each other during training. In addition, in order to capture temporal aspects of product purchases we propose to use the directed language model, where as context we only use future products [12]. The modification allows us to learn product embeddings capable of predicting future purchases.

Learning. The models were optimized using stochastic gradient ascent, suitable for large-scale problems. However, computation of gradients $\nabla\mathcal{L}$ in (3.1) and (3.3) are proportional to the vocabulary size P, which is computationally expensive in practical tasks as P could easily reach millions of products. As an alternative, we used negative sampling approach proposed in [24], which significantly reduces the computational complexity.

3.2 Product-to-product predictive models

Having learned low-dimensional product representations, we considered several possibilities for predicting the next product to be purchased.

prod2vec-topK. Given a purchased product, the method calculates cosine similarities to all other products in the vocabulary and recommends the top K most similar products.

prod2vec-cluster. To be able to make more diverse recommendations, we considered grouping similar products into clusters and recommending products from a cluster that is related to the cluster of previously purchased product. We applied K-means clustering algorithm implemented on the top of Hadoop distributed system where we grouped products based on cosine similarity between their low-dimensional representations. We assume that purchasing a product from any of the C clusters after a purchase from cluster c_i follows a multinomial distribution $Mu(\theta_{i1}, \theta_{i2}, \ldots \theta_{iC})$, where θ_{ij} is the probability that a purchase from cluster c_i is followed by a purchase from cluster c_j. In order to estimate parameters θ_{ij}, for each i and j, we adopted a maximum likelihood approach,

$$\hat{\theta}_{ij} = \frac{\#\text{ of times } c_i \text{ purchase was followed by } c_j}{\text{count of } c_i \text{ purchases}}. \quad (3.4)$$

In order to recommend a new product given a purchased product p, we first identify which cluster p belongs to (e.g., $p \in c_i$). Next, we rank all clusters c_j, $j = 1, \ldots, C$, by the value of θ_{ij} and consider the top ones as top-related clusters to cluster c_i. Finally, products from the top clusters are sorted by their cosine similarity to p, and we use the top K products as recommendations.

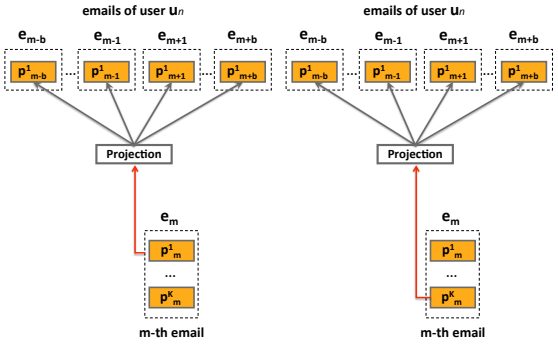


Figure 3: bagged-prod2vec model updates

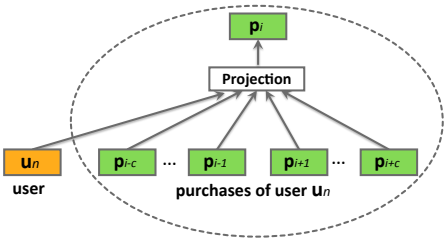


Figure 4: User embeddings for user to product predictions

3.3 User-to-product predictive models

In addition to product-to-product predictions [16, 17], most recommendation engines allow user-to-product predictions as well [15, 30]. Recommendation for a user are typically made considering historical purchases and/or interests inferred using other data sources such as user's online behavior [15], social contacts [30], and others. In this section we propose a novel approach to simultaneously learn vector representations of products and users such that, given a user, recommendations can be performed by finding K nearest products in the joint embedding space.

user2vec. The user2vec model simultaneously learns vector representations of products and users by considering the user as a "global context", motivated by paragraph2vec algorithm [21]. The architecture of such model is illustrated in Figure 4. The training data set was derived from user purchase sequences $\mathcal{S}$, which comprised users u_n and their purchased products ordered by the time of purchase, $u_n = (p_{n1}, p_{n2}, \ldots p_{nU_n})$, where U_n denotes number of items purchased by user u_n. During training, user vectors are updated to predict the products from their e-mail receipts, while product vectors are learned to predict other products in their context. For simplicity of presentation and w.l.o.g.

in the following we present non-bagged version of the language model, however we note that it is straightforward to extend the presented methodology to use the bagged version.

More specifically, objective of user2vec is to maximize the log-likelihood over the set $\mathcal{S}$ of all purchase sequences,

$$\mathcal{L} = \sum_{s \in \mathcal{S}} \Big(\sum_{u_n \in s} \log \mathbb{P}(u_n | p_{n1} : p_{nU_n}) + \sum_{p_{ni} \in u_n} \log \mathbb{P}(p_{ni} | p_{n,i-c} : p_{n,i+c}, u_n) \Big) \tag{3.5}$$

where c is the length of the context for products in purchase sequence of the n-th user. The probability $\mathbb{P}(p_{ni}|p_{n,i-c} : p_{n,i+c}, u_n)$ is defined using a soft-max function,

$$\mathbb{P}(p_{ni}|p_{n,i-c} : p_{n,i+c}, u_n) = \frac{\exp(\bar{\mathbf{v}}^\top \mathbf{v}'_{p_{ni}})}{\sum_{p=1}^{V} \exp(\bar{\mathbf{v}}^\top \mathbf{v}'_p)}, \tag{3.6}$$

where $\mathbf{v}'_{p_{ni}}$ is the output vector representation of p_{ni}, and $\bar{\mathbf{v}}$ is averaged vector representation of the product context including corresponding u_n, defined as

$$\bar{\mathbf{v}} = \frac{1}{2c+1} (\mathbf{v}_{u_n} + \sum_{-c \le j \le c, j \neq 0} \mathbf{v}_{p_{n,i+j}}), \tag{3.7}$$

where $\mathbf{v}_p$ is the input vector representation of p. Similarly, the probability $\mathbb{P}(u_n|p_{n1} : p_{nU_n})$ is defined as

$$\mathbb{P}(u_n|p_{n1} : p_{nU_n}) = \frac{\exp(\bar{\mathbf{v}}_n^\top \mathbf{v}'_{u_n})}{\sum_{p=1}^{V} \exp(\bar{\mathbf{v}}_n^\top \mathbf{v}'_p)}, \tag{3.8}$$

where $\mathbf{v}'_{u_n}$ is the output vector representation of u_n, and $\bar{\mathbf{v}}_n$ is averaged input vector representation of all the products purchased by user u_n,

$$\bar{\mathbf{v}}_n = \frac{1}{U_n} \sum_{i=1}^{U_n} \mathbf{v}_{p_{ni}}. \tag{3.9}$$

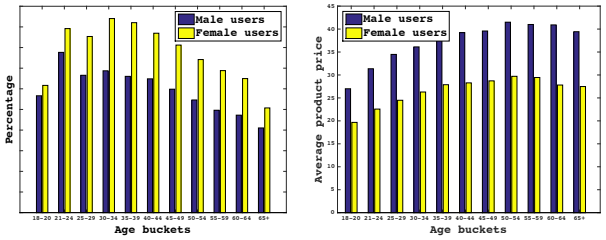
One of the main advantages of the *user2vec* model is that the product recommendations are specifically tailored for that user based on his purchase history. However, disadvantage is that the model would need to be updated very frequently. Unlike product-to-product recommendations, which may be relevant for longer time periods, user-to-product recommendations need to change often to account for the most recent user purchases.

4. EXPERIMENTS

The experimental section is organized as follows. We first describe data set used in development of our product-to-product and user-to-product predictors. Next we present valuable insights regarding purchase behavior of different age and gender groups in different US states. This information can be leveraged to improve demo- and geo-targeting of user groups. This is followed by a section on effectiveness of recommendation of popular products in different user cohorts, including age, gender, and US state. Finally, we show comparative results of various baseline recommendation algorithms. We conclude with the description of the system implementation at a large scale and bucket results that preceded our product launch.

4.1 Data set

Our data sets included e-mail receipts sent to users who voluntarily opted-in for such studies, where we anonymized user IDs. Message bodies were analyzed by automated systems. Product names and prices were extracted from e-mail messages using an in-house extraction tool.



(a) Percentage of purchasing users among all online users (b) Average product price

Figure 5: Purchasing habits for different demographics

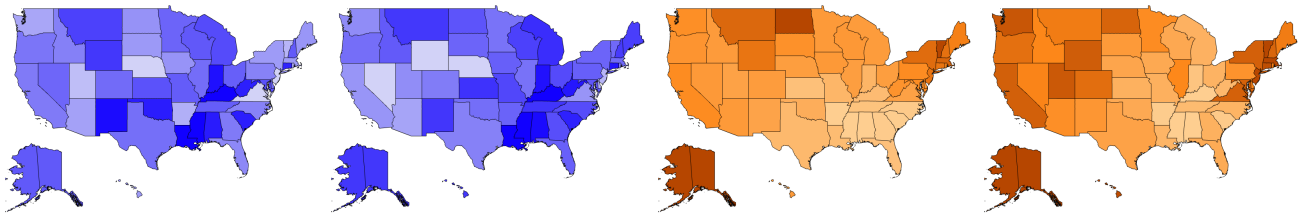
Training data set collected for purposes of developing product prediction models comprised of more than 280.7 million purchases made by $N = 29$ million users from 172 commercial websites. The product vocabulary included $P = 2.1$ million most frequently purchased products priced over $5.

More formally, data set $\mathcal{D}_p = \{(u_n, s_n), n = 1, ..., N\}$ was derived by forming e-mail receipt sequences s_n for each user u_n, along with their timestamps. Specifically, $s_n = \{(e_{n1}, t_{n1}), \ldots, (e_{nM_n}, t_{nM_n})\}$, where e_{nm} is a receipt comprising one or more purchased products, and t_{nm} is receipt timestamp. Predictions were evaluated on a held-out month of user purchases D_p^{ts}, formed in the same manner as D_p. We measured the effectiveness of recommendations using prediction accuracy. In particular, we measured the number of product purchases that were correctly predicted, divided by the total number of purchases. For all models, the accuracy was measured separately on each day, based on the recommendations calculated using prior days. We set a daily budget of distinct recommendations each algorithm is allowed to make for a user to $K = 20$, based on an estimate of optimal number of ads users can effectively perceive daily [7].

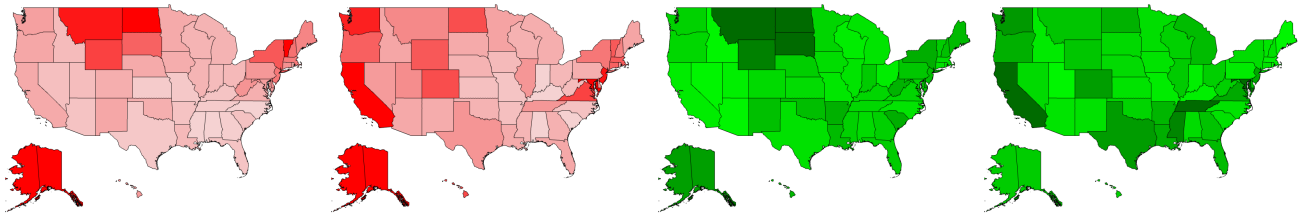
4.2 Insights from purchase data

To get deeper insights into behavior of online users based on their demographic background and geographic location, we segregated users into cohorts based on their age, gender, and location, and looked at their purchasing habits. Such information can be very valuable to marketers when considering how and where to spend their campaign budget. We only considered users from the US, and computed statistics per each state separately. First, we were interested in differences between male and female shopping behavior for different age ranges. In addition, we were interested in the following aspects of purchasing behavior: 1) percentage of online users who shop online; 2) average number of products bought; 4) average spent per user; and 4) average price of a bought item. Figures 5 and 6 illustrate the results.

In Figure 5 for each gender and age group we show the percentage of online users who shop online, as well as the average prices of bought items. Expectedly, we see that male and female demographics exhibit different shopping patterns. Throughout the different age buckets percentage of female shoppers is consistently higher than for males, peaking in the 30-34 age range. On the other hand, percentage of male shoppers reaches its maximum in the 21-24



(a) ages 18-20 (b) ages 40-44 (c) ages 18-20 (d) ages 40-44



(e) ages 18-20 (f) ages 40-44 (g) ages 18-20 (h) ages 40-44

Figure 6: Purchasing behavior for different cohorts, dark color encodes higher values: (a, b) Percentage of shoppers among online users; (c, d) Average number of purchases per user; (e, f) Average amount spent per user; (g, h) Average product price

bucket, and from there drops steadily. In addition, we observe that male users buy more expensive items on average.

Furthermore, in Figure 6 we show per-state results, where darker color indicates higher values. Note that we only show results for different locations and age ranges, as we did not see significant differences between male and female purchasing behavior across states. First, in Figures 6(a) and 6(b) we show the percent of online shoppers in each state, for younger (18-20) and medium-age (40-44) populations, where we can see there exist significant differences between the states. In Figures 6(c) and 6(d) we show the average number of purchased products per state. Here, for both age buckets we see that the southern states purchase the least number of items. However, there is a significant difference between younger and older populations, as in northern states younger populations purchase more products, while in the northeast and the western states this holds true for the older populations. If we take a look at Figures 6(e) and 6(f) where we compare the average amount of money spent per capita, we can observe similar patterns, where in states like Alaska and North Dakota younger populations spend more money than peers from other states, and for older populations states with highest spending per user are California, Washington, and again Alaska. Interestingly, users from Alaska are the biggest spenders, irrespective of the age or metric used. Similar holds when we consider average price of a purchased item, shown in Figures 6(g) and 6(h).

4.3 Recommending popular products

In this section we evaluate predictive properties of popular products. Recommending popular products to global population is a common baseline due to its strong performance, especially during holidays (e.g., Christmas, Thanksgiving). Such recommendations are intuitive, easy to calculate and implement, and may serve in a cold start scenarios for newly registered users when we have limited information.

We considered how far back we need to look when calculating popularity and how long popular products stay relevant.

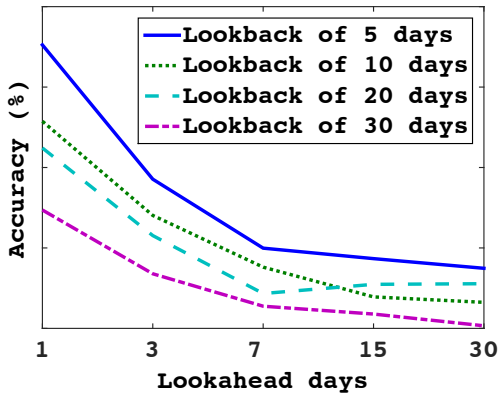


Figure 7: Prediction accuracy and longevity of popular products with different lookbacks

In Figure 7 we give results for popular products calculated in the previous $5, 10, 20$, and 30 days of training data D_p, evaluated on the first $1, 3, 7, 15$, and 30 days of test data D_p^{ts}. To account for ever-changing user purchase tastes, the results indicate that popular products need to be recalculated at least every 3 days with lookback of at most 5 days.

Next, we evaluated the prediction accuracy of popular products computed for different user cohorts, and compared to the accuracy of globally popular products. In Figure 8 we compare accuracies of popular products in different user cohorts, with the lookback fixed at 5 days. We can observe that popular products in gender cohorts give bigger lift in prediction accuracy than popular products in age or state cohorts. Further, jointly considering age and gender when calculating popular products outperforms the popular gender products. Finally, including geographic dimension contributes to further accuracy lift. Overall, the results indicate that in the cold start scenario the best choice of which popular products to recommend are the ones from the user's (age,

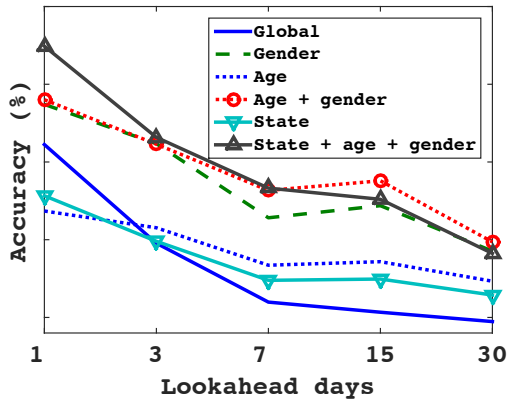


Figure 8: Prediction accuracy of popular products for different user cohorts

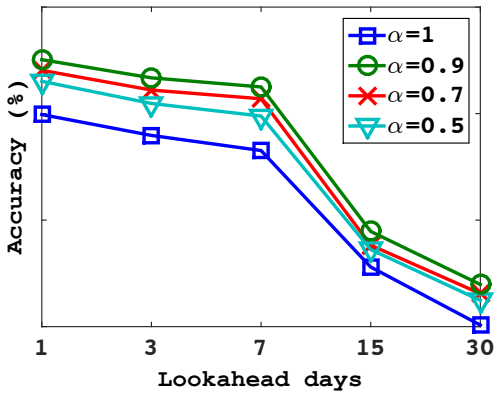


Figure 9: prod2vec accuracy with different decay values

gender, location) cohort. The results also suggest that popular products should be recalculated often to avoid decrease in accuracy with every passing day.

4.4 Recommending predicted products

In this section we experiment with recommending products to users based on neural language models described in Section 3. Specifically, we compare the following algorithms:

1) prod2vec-topK was trained using data set D_p, where product vectors were learned by maximizing log-likelihood of observing other products from sequences s, as proposed in (3.1). Recommendations for a given product p_i were given by selecting the top K most similar products based on the cosine similarity in the resulting vector space.

2) bagged-prod2vec-topK was trained using D_p, where product vectors were learned by maximizing log-likelihood of observing other products from e-mail sequences s as proposed in (3.3). Recommendations for a given product p_i were given by selecting the top K most similar products based on the cosine similarity in the resulting vector space.

3) bagged-prod2vec-cluster was trained similarly to the bagged-prod2vec model, followed by clustering the product vectors into C clusters and calculating transition probabilities between them. After identifying which cluster p_i belongs to (e.g., $p_i \in c_i$), we rank all clusters by their transition probabilities with respect to c_i. Then, the products from top clusters are sorted by cosine similarity to p_i, where top K_c from each cluster are used as recommendations ($\sum K_c = K$). Example predictions of bagged-prod2vec-cluster compared to predictions of bagged-prod2vec are shown in Table 2. It can be observed that predictions based on the clustering approach are more diverse.

4) user2vec was trained using data set D_p where product vectors and user vectors were learned by maximizing log-likelihood proposed in (3.5) (see Figure 4). Recommendations for a given user u_n were given by calculating cosine similarity between the user vector u_n and all product vectors, and retrieving the top K nearest products.

5) co-purchase. For each product pair (p_i, p_j) we calculated frequency $F_{(p_i,p_j)}, i = 1,\ldots,P, j = 1,\ldots,P$, with which product p_j was purchased immediately after product p_i. Then, recommendations for a product p_i were given by sorting the frequencies $F_{(p_i,p_j)}, j = 1,\ldots,P$, and retrieving the top K products.

Since user u_n may have multiple products purchased prior to day t_d, separate predictions need to reach a consensus in order to choose the best K products to be shown on that day. To achieve this we propose time-decayed scoring of recommendations, followed by choice of top K products with the highest score. More specifically, given user's products purchased prior to t_d along with their timestamps, $\{(p_1,t_1),\ldots(p_{U_n},t_{U_n})\}$, for each product we retrieve top K recommendations along with their similarity scores, resulting in the set $\{(p_j, sim_j), j = 1,\ldots,KU_n\}$, where sim denotes cosine similarity. Next, we calculate a decayed score for every recommended product,

$$d_j = sim_j \cdot \alpha^{(t_d - t_i)}, \tag{4.1}$$

where $(t_d - t_i)$ is a difference in days between current day t_d and the purchase time of product that led to recommendation of p_j, and α is a decay factor. Finally, the decayed scores are sorted in descending order and the top K products are chosen as recommendations for day t_d.

Training details. Neural language models were trained using a machine with 96GB of RAM memory and 24 cores. Dimensionality of the embedding space was set to $d = 300$, context neighborhood size for all models was set to 5. Finally, we used 10 negative samples in each vector update. Similarly to the approach in [24], most frequent products and users were subsampled during training. To illustrate the performance of the language models, in Table 1 we give examples of product-to-product recommendations computed using bagged-prod2vec, where we see that the neighboring products are highly relevant to the query product (e.g., for "despicable me" the model retrieved similar cartoons).

Evaluation details. Similarly to how popular product accuracy was measured, we assumed a daily budget of $K = 20$ distinct product recommendations per user. Predictions for day t_d are based on prior purchases from previous days, and we did not consider updating predictions for day t_d using purchases that happened during that day.

Results. In the first set of experiments we evaluated performance of prod2vec for different values of decay factors. In Figure 9 we show prediction accuracy on test data D_p^{ts} when looking $1, 3, 7, 15$, and 30 days ahead. Initial prod2vec predictions were based on the last user purchase in the training data set D_p. The results show that discounting of old predictions leads to improved recommendation accuracy, with decay factor of $\alpha = 0.9$ being an optimal choice.

Table 1: Examples of product recommendations made by the bagged-prod2vec model

despicable me	first aid for the usmle step 1	disney frozen lunch napkins
monsters university	usmle step 1 secrets 3e	disneys frozen party 9 square lunchdinner plates
the croods	first aid basic sciences 2e	disneys frozen party 9oz hotcold cups
turbo	usmle step 1 qbook	disneys frozen party 7x7 square cakedessert plates
cloudy with a chance of meatballs	brs physiology	disneys frozen party printed plastic tablecover
hotel transylvania	rapid review pathology with student consult	disneys frozen party 7 square cakedessert plates
brave	lippincotts microcards microbiology flash cards	disney frozen beverage napkins birthday party supplies
the smurfs	first aid cases for the usmle step 2	disney frozen 9 oz paper cups
wreckit ralph	highyield neuroanatomy	frozen invitation and thank you card
planes	lange pharmacology flash cards third edition	disneys frozen party treat bags

Table 2: Product recommendations for product *cressi supernova dry snorkel*

bagged-prod2vec-topK	bagged-prod2vec-cluster	cluster ID
jaws quick spit antifog 1 ounce	cressi neoprene mask strap	
cressi neoprene mask strap	cressi frameless mask	cluster 1
cressi frameless mask	cressi scuba diving snorkeling freediving mask snorkel set	
akona 2 mm neoprene low cut socks	akona 2 mm neoprene low cut socks	
tilos neoprene fin socks	tilos neoprene fin socks	cluster 2
cressi scuba diving snorkeling freediving mask snorkel set	mares equator 2mm dive boots	
mares cruise mesh duffle bag	jaws quick spit antifog 1 ounce	
us divers island dry snorkel	aqua sphere kayenne goggle with clear lens clear black regular	cluster 3
us divers trek travel fin	aqua sphere kayenne goggle with clear lens black regular	
us divers proflex ii diving fins	nikon coolpix aw120 161 mp wi-fi and waterproof digital camera	
mares cruise backpack mesh bag	olympus stylus tg ihs digital camera with 5x optical zoom	cluster 4
water gear fin socks	nikon coolpix aw110 wi fi and waterproof digital camera with gps	

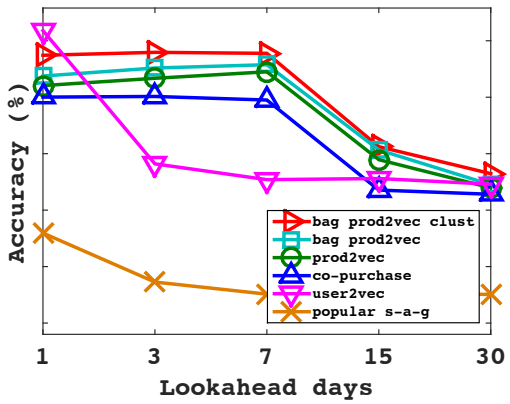


Figure 10: Prediction accuracy of different algorithms

Next, we evaluate different product-to-product and user-to-product methods and compared them to the popular products approach found in Figure 8 to perform the best, namely recommending popular products computed separately for each (state, age, gender) cohort. Results are summarized in Figure 10. We can observe that all neural-based prediction algorithms outperformed method that recommends popular products. Further, even though user2vec model had the best overall accuracy on day 1, its prediction power quickly drops after 3 days. On the other hand, variants of the prod2vec model did not exhibit such behavior, and their performance remained steady across the first 7 days. Of all prod2vec models, the bagged-prod2vec-cluster model achieved the best performance, indicating that diversity in predictions leads to better results. Finally, predictions based on co-purchases did not perform as well as neural language models, supporting the "don't count, predict" claim from [4] where the authors suggest that simple co-occurence approaches are suboptimal.

Table 3: Results from live A/B testing of product recommendations on Yahoo Mail

Metric	Control (5% traffic)	Popular (5% traffic)	Predicted (5% traffic)
CTR	-	+8.33%	+9.81 %
YR		-	+7.63 %

4.5 Bucket results

Following offline evaluation of different product prediction methods, we conducted additional A/B testing experiments on live Yahoo Mail traffic. We ran two buckets with two different recommendation techniques, both on 5% of Yahoo Mail users. In the first bucket, for users with prior purchases the recommendations were based on the bagged-prod2vec-cluster model, while for users without prior purchases they were based on globally popular products. In the second bucket all users were recommended globally popular products. For purposes of a fair bucket test, both models were retrained and refreshed with the same frequency of 7 days.

Both test buckets were evaluated against a control bucket in which products ads were replaced with standard ads from Yahoo Ad serving platform. All three buckets had significant amount of users with prior purchases. Evaluation was done based on the click-through rate (CTR) computed as $\text{CTR} = \frac{\#\text{clicks}}{\#\text{impressions}}$. In particular, we measured the number of clicks on product ads that occurred after the targeted recommendation, divided by the total number of shown (or impressed) product ads. In the control bucket, we computed the same metric for standard ads. For the two product ad buckets we additionally measured the yield rate, calculated as $\text{YR} = \frac{\#\text{conversions}}{\#\text{impressions}}$, where conversion refers to an actual purchase of the recommended product. Conversions were attributed to impression for up to 48 hours since the time users clicked or observed a product ad.

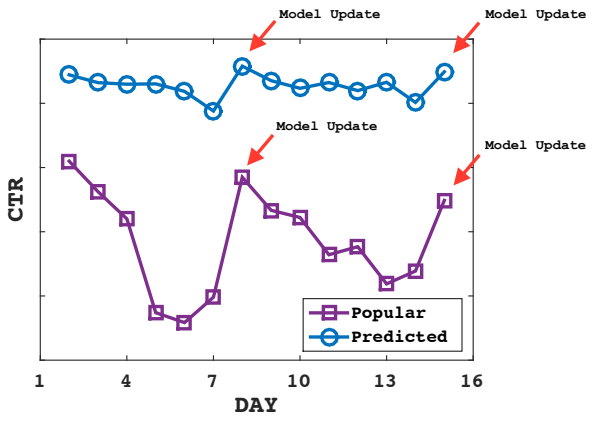


Figure 11: CTR of predicted versus popular recommendations in live bucket test over time

The results are presented in Table 3. We can make several observations. First, both popular and predicted bucket showed better CTR numbers than the control bucket, indicating that the users prefer product ads over standard ads. Second, the prediction bucket achieved slightly better CTR rates than the popular bucket. Finally, the prediction bucket achieved significantly better YR than the popular bucket. This indicates that many times users click on popular products out of curiosity and do not end up buying the product, whereas clicks on targeted products lead to more purchases as they better capture user interests. Overall, the presented results strongly suggest benefits of the proposed approach for the task of product recommendations.

In addition, in Figure 11 we show how CTR rates change over time. Similarly to our offline experiments, it can be observed that popular recommendations become stale much faster than predicted recommendations, as indicated by the steeper CTR drop. They are also more susceptible to novelty bias following model updates, as seen by larger increase in CTR. Another useful finding was that 7 day updates are not sufficient, confirming findings from Figure 7.

5. SYSTEM DEPLOYMENT

In this section we cover the details on our system implementation that led to final product deployment.

5.1 Implementation details

Due to product requirements for near-real-time predictions, we chose to use product-to-product recommendations, with the bagged-prod2vec-cluster model. The model is updated every 5 days with the most recent purchase data. The product vectors are stored on Hadoop Distributed File System (HDFS), and updated via training procedure implemented on Hadoop[5], where we heavily leverage parallel processing with MapReduce. We form purchase sequences using the newest purchased products extracted from e-mail receipts, and incrementally update the product vectors instead of training them from scratch.

Popular products were used as back-fill for users who do not have prior purchase history. Following the experimental results, we recalculated popular products every 3 days, with a lookback of 5 days. More specifically, we extracted 100 most frequently purchased products for every state-age-gender cohort, and showed them in randomized order to users without purchase history.

We implemented a multi-tier architecture in order to productionize product recommendation capability on Yahoo Mail. For data storage we used a custom, high-performance distributed key-value store (similar to Cassandra[6]), which persists user profiles and product-to-product prediction model. The user profile store utilizes user identifier as a key and stores multiple columns representing user prior purchases as values. User profiles are updated hourly with new products, extracted from e-mail receipts. Each purchase record persists in memory with time-to-live (TTL) set to 60 days, after which period it is discarded.

Product-to-product prediction model is stored in a similar key-value store, where purchased product is used as a key and values are multiple columns each representing predicted products ordered by score. Both user and product-to-product stores are updated in batch without impacting real traffic. Separate process performs all processing of querying user purchases and asynchronously retrieves relevant predictions and offers. It then returns selected product ad back to presentation process implemented in JavaScript and HTML which renders it in the browser. The retrieval process is configured such that it can fetch offers from multiple e-commerce websites (affiliate partners). The system runs with a service-level agreement of 500ms and can be improved further by caching interactions with affiliate partners.

Given predictions for a certain user, once the user logs into the e-mail client we show a new recommendation after every user action, including clicking on folders, composing e-mail, and searching the inbox. Recommendation are circled in the order of decayed prediction score, as described in Section 4.4. The product ads are implemented in the so-called "pencil" ad position, just above the first e-mail in the inbox (Figure 1).

6. CONCLUSIONS AND FUTURE WORK

We presented the details of a large-scale product recommendation framework that was launched on Yahoo Mail in the form of product ads. We discussed the recommendation methodology, as well as the high-level implementation details behind our system. To perform recommendations, we described how we employed neural language models capable of learning product embeddings for product-to-product predictions, as well as user embeddings for user-to-products predictions. Several variants of the prediction models were tested offline and the best candidate was chosen for an online bucket test. Following the encouraging bucket test results, we launched the system in production. In our ongoing work, we plan to utilize implicit feedback available through ad views, ad clicks, and conversions to further improve the performance of our product recommendation system.

7. REFERENCES

[1] A. Ahmed, Y. Low, M. Aly, V. Josifovski, and A. J. Smola. Scalable distributed inference of dynamic user interests for behavioral targeting. In *KDD*, pages 114–122, 2011.

[2] J. Alba, J. Lynch, B. Weitz, C. Janiszewski, R. Lutz, A. Sawyer, and S. Wood. Interactive home shopping:

[5]https://hadoop.apache.org, accessed June 2015

[6]http://cassandra.apache.org/, accessed June 2015

consumer, retailer, and manufacturer incentives to participate in electronic marketplaces. *The Journal of Marketing*, pages 38–53, 1997.

[3] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.

[4] M. Baroni, G. Dinu, and G. Kruszewski. DonÕt count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 238–247, 2014.

[5] Y. Bengio, H. Schwenk, J.-S. Senécal, F. Morin, and J.-L. Gauvain. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer, 2006.

[6] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795, 2013.

[7] C.-H. Cho and H. J. Cheon. Why do people avoid advertising on the internet? *Journal of advertising*, 33(4):89–97, 2004.

[8] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

[9] N. Djuric, V. Radosavljevic, M. Grbovic, and N. Bhamidipati. Hidden conditional random fields with distributed user embeddings for ad targeting. In *IEEE International Conference on Data Mining*, Dec 2014.

[10] N. Djuric, H. Wu, V. Radosavljevic, M. Grbovic, and N. Bhamidipati. Hierarchical neural language models for joint representation of streaming documents and their content. In *International World Wide Web Conference (WWW)*, 2015.

[11] D. Essex. Matchmaker, matchmaker. *Communications of the ACM*, 52(5):16–17, 2009.

[12] M. Grbovic, N. Djuric, V. Radosavljevic, and N. Bhamidipati. Search retargeting using directed query embeddings. In *International World Wide Web Conference (WWW)*, 2015.

[13] M. Grbovic, G. Halawi, Z. Karnin, and Y. Maarek. How many folders do you really need?: Classifying email into a handful of categories. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 869–878. ACM, 2014.

[14] M. Grbovic and S. Vucetic. Generating ad targeting rules using sparse principal component analysis with constraints. In *international conference on World Wide Web*, pages 283–284, 2014.

[15] D. J. Hu, R. Hall, and J. Attenberg. Style in the long tail: discovering unique interests with latent variable models in large scale social e-commerce. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1640–1649. ACM, 2014.

[16] J. Katukuri, T. Konik, R. Mukherjee, and S. Kolay. Recommending similar items in large-scale online marketplaces. In *Big Data (Big Data), 2014 IEEE International Conference on*, pages 868–876. IEEE, 2014.

[17] J. Katukuri, R. Mukherjee, and T. Konik. Large-scale recommendations in a dynamic marketplace. In *Workshop on Large Scale Recommendation Systems at RecSys*, volume 13, 2013.

[18] R. Kiros, R. Zemel, and R. Salakhutdinov. Multimodal neural language models. In *Proceedings of the 31th International Conference on Machine Learning*, 2014.

[19] R. Kiros, R. S. Zemel, and R. Salakhutdinov. A multiplicative model for learning distributed text-based attribute representations. *arXiv preprint arXiv:1406.2710*, 2014.

[20] V. Lavrenko and W. B. Croft. Relevance based language models. In *SIGIR*, pages 120–127. ACM, 2001.

[21] Q. V. Le and T. Mikolov. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*, 2014.

[22] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, Jan. 2003.

[23] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[24] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[25] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. *arXiv preprint arXiv:1403.6652*, 2014.

[26] D. Riecken. Personalized views of personalization. *Communications of the ACM*, 43(8):27–28, 2000.

[27] R. Socher, D. Chen, C. D. Manning, and A. Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934, 2013.

[28] J. Turian, L. Ratinov, and Y. Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics, 2010.

[29] Y. Zhang and M. Pennacchiotti. Predicting purchase behaviors from social media. In *WWW*, pages 1521–1532, 2013.

[30] X. W. Zhao, Y. Guo, Y. He, H. Jiang, Y. Wu, and X. Li. We know what you want to buy: a demographic-based system for product recommendation on microblogs. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1935–1944. ACM, 2014.