Using Probabilistic Tag Modeling to Improve Recommendations

Beliz U. Gokkaya
Udemy Inc.
600 Harrison St.
San Francisco, California 94107
beliz@alumni.stanford.edu

Larry Wai
Udemy Inc.
600 Harrison St.
San Francisco, California 94107
larry.wai@udemy.com

ABSTRACT

Predicting users' interest in products is at the core of recommender systems. This requires understanding of users' tastes as well as the characteristics of the products. In this study, we focus on automatically discovering latent product features from algorithmically defined tags. Tags not only help understand implicit tastes of users but also facilitate discovery and categorization of products in large inventories. Product tags also provide structured information to refine the similarity between products in a content-based recommender system. A framework is presented for modeling product tags using search query data. User search queries are modeled as an undirected network; nodes represent the search queries and the similarities between nodes are obtained from user feedback. Queries that have similar meanings are clustered using community detection algorithms and the central node in each cluster is used as a tag. We use a probabilistic model to relate products and users with these tags. Tag profiles of products and users are then used with the recommendation engine. The framework is employed to tag courses on Udemy, a global marketplace for learning and teaching online. We demonstrate significant improvements to Udemy's recommendations using this novel method.

CCS CONCEPTS

•Information systems → Personalization; Similarity measures; Recommender systems; *Query intent;*

KEYWORDS

tag models, community detection, graph modeling, recommender systems

ACM Reference format:
Beliz U. Gokkaya and Larry Wai. 2017. Using Probabilistic Tag Modeling to Improve Recommendations. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Halifax, Nova Scotia - Canada, August 13–17, 2017 (KDD'17),* 7 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Recommender systems are typically used in marketplaces and e-commerce sites to surface products that users might be interested in. These systems are built on the premise of understanding user preferences and product characteristics to assist users in discovering preferred products. There are various factors that play a role in how users assess products and, ideally, these implicit and explicit factors are represented in recommender systems. This implies an understanding of the features describing products as well as users. To facilitate user experience in discovery of products and improve user exploratory search, marketplaces and e-commerce sites also face the challenge of categorizing products in their inventory.

A common way to address these concerns is to mark the products with appropriate keywords, also know as tags. Tags not only provide efficient organization products but can also be used with content-based recommender systems as product features that are descriptive of user intent. Collaborative tagging, which enables consumers to tag products, is commonly used in the industry [10, 16]. Content-based techniques are employed for automatic tagging of content [12, 13]. These techniques are also used in tagging of multimedia content [18, 22]. However, the aforementioned methods require a tagging infrastructure as well as consumers willing to provide feedback.

Probabilistic topic modeling [3] can also be used for tagging. Latent Dirichlet Allocation provides a model for discovering topics in text data [4]. Latent topics are discovered from review text data along with latent rating dimensions to improve predicting product ratings [17]. Probabilistic topic modeling is combined with collaborative filtering to recommend scientific articles [24]. It is also used to improve recommendations in auction marketplaces [6]. [11] improved recommendation quality by modeling user profiles using a mixture of latent topics. However, probabilistic topic modeling requires a large collection of text data to analyze the prominent words.

In this paper, we present a statistical framework to algorithmically generate tags from search logs. Products are modeled with these tags to be used as latent product features. Product tags provide structured information to refine the similarity between products in content-based recommender systems. We also model users with tags. User tags facilitate understanding of not only users' implicit taste but also groupings among users.

Our framework is composed of three main parts: 1) Tag Generation, 2) Product Tag Modeling, and 3) User Tag Modeling. The process is illustrated in Figure 1. In the first part, we use search logs to generate the tags. In the second part, we use these tags and probabilistically model products with these tags. Third, we employ a probabilistic approach to model users with tags using historical data on user interactions with products.

Finally, as illustrated in Figure 1, the product and user tag models can be used with personalized ranking algorithms to improve recommender systems. In this paper, we employ the framework

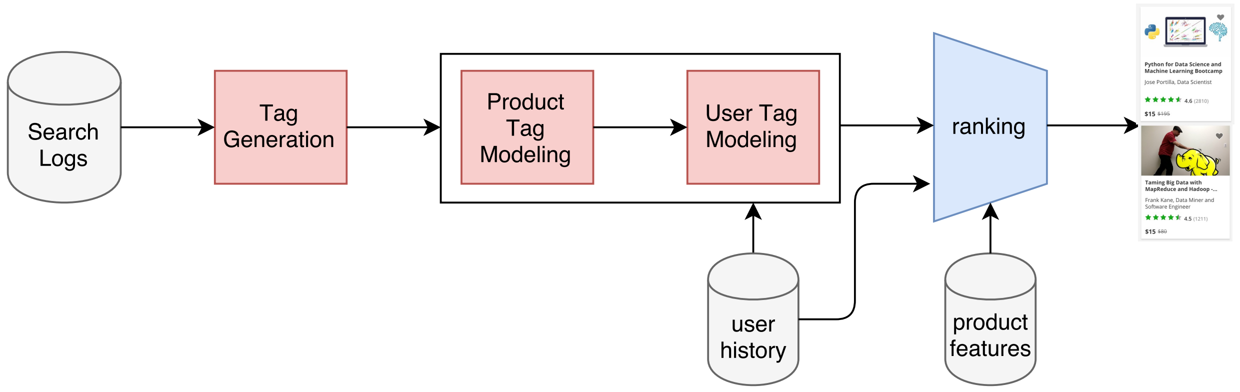


Figure 1: Overview of the methodology

to improve personalization at Udemy [1], a global marketplace for learning and teaching online. The proposed framework is implemented as a part of the recommendation engine to improve course suggestions to over 15 million students.

In the remaining sections of the paper we describe our framework in detail. We present our experiments using Udemy's user and course data. The results from offline and online controlled experiments show that the proposed framework works well for personalized recommendations.

1.1 Contributions

Our main contribution is a statistical framework for automatically discovering product features from algorithmically defined tags and probabilistically associating user behavior to these tags to improve recommendations. These tags put user behavior into context, which is missing in traditional collaborative filtering based methods. They provide essential information to better understand and characterize users and products. Furthermore, the framework has the following advantages:

- The tags are generated using search log data. We reduce the noise in the dataset (i.e. remove queries of similar meaning) by employing clustering techniques and discover the prominent tags in the data.
- Products in an inventory are automatically labeled using the generated tags. We employ a probabilistic approach for modeling products with tags, which results in a ranking of tags for a given product. The tag profile of products also enables similarity computations between products, which is an essential component of a content-based recommender system.
- Users are associated probabilistically with tags through their interactions with the products. From a recommender system perspective, this provides a layer of personalization at tag level.

Finally, the recommendations provided using this framework are assessed using offline and online controlled experiments at Udemy.

[1]This can be found at www.udemy.com.

2 METHODOLOGY

In the following sections, we provide detailed information on the methodology. The sections follow the parts of the methodology outlined in Figure 1.

2.1 Tag Generation

In this step, we are interested in discovering latent dimensions of users and products. These dimensions are named as tags in this study. Search log data is used as data source. This data is modeled as a graph to understand the relationships between search queries. We then use clustering to eliminate similar words and discover the tags to be used in recommendations.

2.1.1 Graph Model for Search Queries. Search engines retrieve products that are relevant to a given search query. User feedback from search data (e.g., impressions, clicks, purchases) provides an indication of the relationship between a given query with the product. It can be assumed that users with the same intent provide feedback on similar products using similar queries. The similar queries represent the users' goal or the topic of interest. Therefore, collective feedback of users provides an estimate of the relationships of search queries.

The relationship between search queries is modeled using an undirected search query graph. In this model, nodes of the graph represent the search queries. If users provide feedback on the same product using two distinct search queries, then the two queries are related. When the collective feedback on similar products is frequent, the relationship between the two queries is stronger. Therefore, the weights of the edges in the graph are based on the strength of the user actions defining the similarity between any two pairs of search queries.

Clicks and purchases are two important sources of user feedback at e-commerce sites. Users of these sites select among the products listed by the search engine and clicks if that course is relevant to their query. Besides clicks, purchases are another source of user feedback showing intent. After clicking on a product through the search results, if the user further makes a purchase, it is another indicator of the relationship between the search query and the product.

Using the Jaccard similarity, the similarity of queries q_i and q_j is computed using click data ($sim_C(q_i, q_j)$) as follows:

$$sim_C(q_i, q_j) = \frac{P_C(q_i) \cap P_C(q_j)}{P_C(q_i) \cup P_C(q_j)} \tag{1}$$

where $P_C(q_i)$ denotes the set of products clicked following the query q_i and $P_C(q_i) \cap P_C(q_j)$ denotes the intersection of sets of products clicked using q_i and q_j, and $P_C(q_i) \cup P_C(q_j)$ denotes the union of sets of products clicked using q_i and q_j.

Similarity of queries q_i and q_j is computed using purchase data ($sim_P(q_i, q_j)$) as follows:

$$sim_P(q_i, q_j) = \frac{P_P(q_i) \cap P_P(q_j)}{P_P(q_i) \cup P_P(q_j)} \tag{2}$$

where $P_P(q_i)$ denotes the set of products purchased following query q_i.

Equations 1 and 2 are combined in a weighted fashion to compute similarity as defined below:

$$sim(q_i, q_j) = \alpha \times sim_P(q_i, q_j) + \beta \times sim_C(q_i, q_j) \tag{3}$$

where α and β denote the weights of similarity based on purchases and clicks, respectively and $\alpha + \beta = 1$. These weights are calibrated empirically using historical data.

2.1.2 Community Detection on Search Query Graph. Users who are interested in the same topic or have the same goal might use different search queries. Even though these queries have similar information, they will be represented as different queries. In this study, search queries are used as tags to describe the products. Understanding the relationship of search queries and their topics increases the reliability of the tags. Since the underlying relationships between search queries are revealed by a graph model, clustering techniques can be employed on the search query graph to identify the groups of search queries having similar meanings.

Clustering search data has shown to be an effective way of finding topics in search queries. [2] use clustering on a bipartite graph to find similar queries and similar URLs using user-feedback data from a search engine. [26] also use a directed bipartite graph to model click behavior in Yahoo web search logs. Maximal bipartite cliques are extracted to form clusters of queries which are showing similar user information needs. [25] adopt a query-clustering approach for finding similar search queries to identify Frequently Asked Queries in a search engine. They use user feedback and query content to define the similarity between queries.

Clustering reveals the communities in graphs; within a community the nodes are densely connected within each other and have fewer links to other nodes in the graph [9]. There are many algorithms to conduct community detection in graphs and we refer to [8] for a comprehensive review of these methods.

In this study, we use Walktrap community detection algorithm [20] since it provides a computationally efficient method for finding the community structure in graphs. The algorithm runs in $O(mn^2)$ in time and $O(n^2)$ in space, where n denotes the number of nodes and m denotes the number of edges in the graph.

Walktrap algorithm defines the distance between two nodes, as well as two communities, using properties of random walks on graphs. Probability of going from one node to another is computed using a Markov chain approach based on the adjacency matrix of the graph. The algorithm then employs an agglomerative hierarchical clustering approach. In this approach, the graph is initially partitioned in to n communities and pairwise distances between all communities are computed. The most adjacent communities are merged into one community and distances between communities are updated. This is repeated until all nodes are merged into one community. The result of the hierarchical clustering is a dendogram. The dendogram provides a tree-like structure representing the hierarchy with the clusters.

2.1.3 Tag Extraction using PageRank. Groups of search queries having similar information are determined using community detection on the search query graph. These groups are then analyzed using link analysis. The link analysis is required to: 1) Understand the high level information in these groups. The most central node in the group is assumed to represent the high level information within the group of search queries and it is used as the "tag" of that cluster. 2) Extract relative importance of queries within a group. In mapping products to tags, queries are weighted differently depending on significance of the query.

We use PageRank algorithm [19] for link analysis. The links within a subgraph, which corresponds to a cluster of search queries, are analyzed using this algorithm to understand the relative importance of search queries within a cluster. PageRank scores provide a measure of node significance by incorporating the degree of connectedness of nodes in the graph. Using the PageRank scores of each search query, we obtain a ranking for each term. The search query having the highest PageRank is deemed most important and used as the tag of the cluster.

2.2 Product Tag Modeling

The products are mapped to the auto-generated list of tags using a probabilistic approach. The probability of $Product_i$ belonging to Tag_j is represented as $P(Product_i|Tag_j)$ and it is computed as given below:

$$P(Product_i|Tag_j) = \sum_{Query_k \in Tag_j} P(Product_i|Query_k) P(Query_k|Tag_j) \tag{4}$$

The equation follows directly from the law of total probability. We assumed conditional independence of $Product_i$ and Tag_j given $Query_k$. $P(Product_i|Query_k)$ denotes the proportion of user actions on $Product_i$ given $Query_k$, which uses user feedback (i.e. impressions, clicks, enrollments) from search logs. $P(Query_k|Tag_j)$ denotes the PageRank score of $Query_k$ in cluster Tag_j. The summation is performed over the queries that are associated with Tag_j since our community detection method does not result in overlapping communities. Therefore, $P(Query_k|Tag_j)$ is zero for $Query_k \notin Tag_j$.

The probabilistic approach has the following advantages:

- For a given product, the main and secondary tags can be distinguished based on $P(Product_i|Tag_j)$.
- The tag profile of a product is extracted, which can be used in product similarity computations as shown in the later sections.

2.3 User Tag Modeling

Finally, given the inherent uncertainty in user modeling, we model user interest in the tags using a probabilistic model. The probability that $User_k$ will be interested in Tag_j is represented as $P(Tag_j|User_k)$ and it is computed as given below:

$$P(User_k|Tag_j) = \sum_i P(User_k|Product_i)P(Product_i|Tag_j) \quad (5)$$

where $P(Product_i|Tag_j)$ is computed using Equation 4. $P(User_k|Product_i)$ represents the probability that $User_k$ will be interested in $Product_i$. This is obtained using models that are calibrated using historical user feedback data from multiple sources. The summation is performed over all products i.

In order to compute a user's interest in a given tag, Equation 5 performs a sum over the products the user is interested in weighted by the products' relation to the tag. This approach has the following advantages:

- For a given user, the main and secondary tags they will be interested in can be distinguished based on $P(User_k|Tag_j)$.
- The tag profile of a user is extracted, which can be used in user similarity computations.

2.4 Product Similarity

Content based recommendation systems recommend products based on similarity measures between products [15]. Tags can be used as product features and we can measure the similarity in product tag profiles. The similarity between $Product_i$ and $Product_j$ is defined as $s(i, j)$ and it is computed as given in Equation 6.

$$sim(i,j) = \langle P(Product_i|Tag.)\,, \; P(Product_j|Tag.)\rangle \quad (6)$$

where $P(Product_i|Tag.)$ denotes the normalized vector of product-tag relations of Equation 4 and $\langle\cdot\,,\cdot\rangle$ denotes an inner product.

3 EXPERIMENTS

3.1 Udemy's Recommender System

Udemy is the world's online learning marketplace, where 15 million+ students are taking courses in everything from programming to yoga to photography and much more. Each of Udemy's 45,000+ courses is taught by an expert instructor, and Udemy has built the marketplace so that students directly impact the kind of content available, allowing the platform to grow and evolve over time.

At its heart, the recommendation problem at Udemy is to match the right students with the right course. The recommender system aims to find the courses that the users will likely to purchase, consume and enjoy from a large selection of courses. Udemy has vast amount of data describing its users' discovery and learning experience. This data is constantly used to improve the recommender system to provide a seamless course discovery experience.

Udemy's homepage is designed to facilitate exploratory search of courses. An example homepage is shown in Figure 2. The recommended courses are displayed in a matrix-like layout. This layout enables each row, termed as "units" in the following discussions, to have similar courses in a given topic or theme. The units as well as recommended courses within units are tailored based on user tastes and actions.

Udemy's recommender system provides a three-step procedure. The first step is to generate course candidates that will be displayed in a given unit. We have different algorithms for making this initial selection of courses powering different unit. For example, for candidate course generation, we use course-course similarity values for the "Because You viewed" and "Because You enrolled" units. Based on the user's last clicked and enrolled courses, these units retrieve candidate courses that are similar to the seed course. In the second step, we rank the courses within a unit using a personalized model. This model provides a score for each course for a given user targeting impression normalized enrollment weighted Net Promoter Score. These scores are used to rank courses within a unit. The final step is to rank the units to construct the personalized homepage. We refer to [23] for more information on Udemy's recommender system.

3.2 Dataset and Implementation

We apply the aforementioned method to Udemy's historical user interaction data. Our dataset spans a period of 2 years and contains impression level information of 45,000+ courses by 15 million+ students.

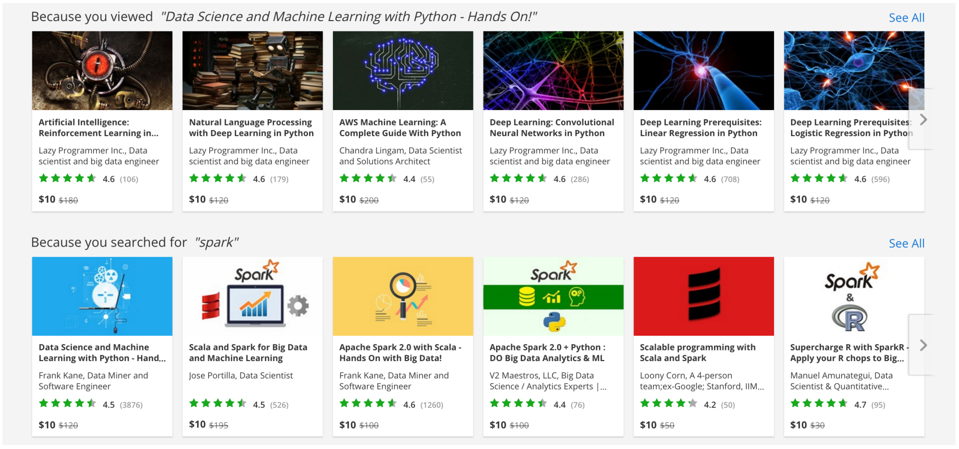
We first construct the search query graph. Search logs are preprocessed to remove the queries having typos and are normalized to remove capitalizations and punctuations. We then select approximately the top 9,000 unique search queries as the nodes of the graph with approximately 9M edges connecting the nodes. Clicks and course enrollments are used as the source user feedback to define the relationships between search queries. In Figure 3, we show a low level view of the resulting search query graph.

In order to extract groups of queries having similar meanings, Walktrap community detection is applied to the search query graph. By optimizing modularity, the method resulted in approximately 1700 clusters. These clusters are then analyzed using PageRank analysis. An example query cluster is shown in Table 1. The search query "photography" is determined to have the highest PageRank score and therefore is used as the tag defining this cluster.

In order to model users and products with the tags, we use historical user interaction data. To illustrate product tag modeling, a popular course on Udemy is shown in Figure 4. This course provides educational content around web development focusing on design and coding of websites. The top algorithmically generated tags that are associated with this course are listed in Table 4. These tags are observed to match closely with the topic of the course.

3.3 Offline Experiments

The similarity between Udemy's courses are computed following Equation 6. We perform offline testing to evaluate the performance of the product similarity computations using historical data. Our aim in offline testing is to make predictions for the courses the user has not seen yet. We therefore split the dataset into training and test sets based on the timestamp of user actions. The training set consists of user actions made in a 90 day period and the test set consists of the records belonging to the next day following the


Because you viewed *"Data Science and Machine Learning with Python - Hands On!"* See All

Artificial Intelligence: Reinforcement Learning in... Lazy Programmer Inc., Data scientist and big data engineer 4.6 (106) **$10** ~~$180~~

Natural Language Processing with Deep Learning in Python Lazy Programmer Inc., Data scientist and big data engineer 4.6 (179) **$10** ~~$120~~

AWS Machine Learning: A Complete Guide With Python Chandra Lingam, Data Scientist and Solutions Architect 4.4 (55) **$10** ~~$200~~

Deep Learning: Convolutional Neural Networks in Python Lazy Programmer Inc., Data scientist and big data engineer 4.6 (286) **$10** ~~$120~~

Deep Learning Prerequisites: Linear Regression in Python Lazy Programmer Inc., Data scientist and big data engineer 4.6 (708) **$10** ~~$120~~

Deep Learning Prerequisites: Logistic Regression in Python Lazy Programmer Inc., Data scientist and big data engineer 4.6 (596) **$10** ~~$120~~

Because you searched for *"spark"* See All

Data Science and Machine Learning with Python - Hand... Frank Kane, Data Miner and Software Engineer 4.5 (3876) **$10** ~~$120~~

Scala and Spark for Big Data and Machine Learning Jose Portilla, Data Scientist 4.5 (526) **$10** ~~$195~~

Apache Spark 2.0 with Scala - Hands On with Big Data! Frank Kane, Data Miner and Software Engineer 4.6 (1260) **$10** ~~$100~~

Apache Spark 2.0 + Python : DO Big Data Analytics & ML V2 Maestros, LLC, Big Data Science / Analytics Experts |... 4.4 (76) **$10** ~~$100~~

Scalable programming with Scala and Spark Loony Corn, A 4-person team;ex-Google; Stanford, IIM... 4.2 (50) **$10** ~~$50~~

Supercharge R with SparkR - Apply your R chops to Big... Manuel Amunategui, Data Scientist & Quantitative... 4.7 (95) **$10** ~~$30~~

Figure 2: An example homepage at Udemy. The matrix-like layout enables each row to have similar recommendations in a given context.

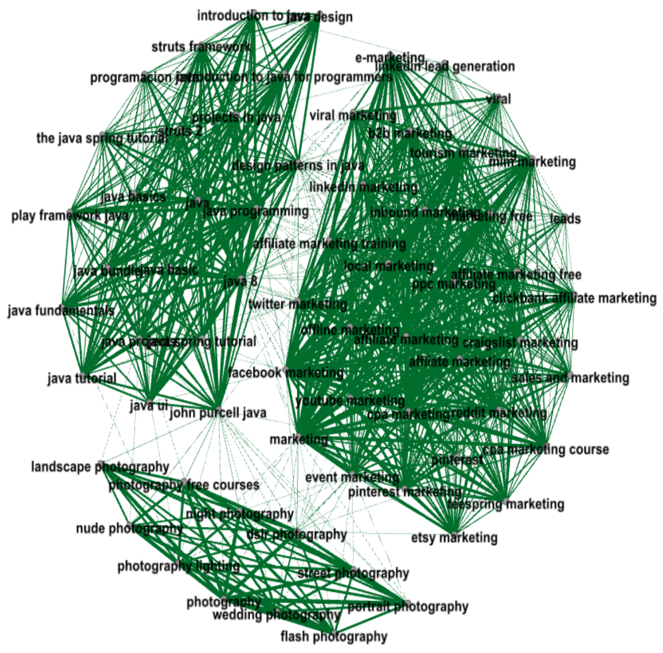


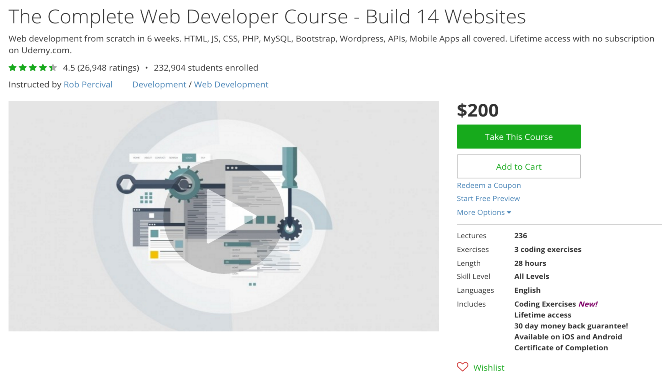
Figure 3: Low level view of the Udemy’s search query graph. This graph illustrates the relationship using a subset of the 9,000 search queries.

training set. We denote our prediction for course i by 0.user a as $p_{a,i}$ and compute it as given below:

$$p_{a,i} = \frac{\sum_j s(i,j) u_{a,j}}{\sum_j s(i,j)} \tag{7}$$

Table 1: An example search query cluster with PageRank scores

query	$PR(query)$
photography	0.11
portrait photography	0.08
photography lighting	0.07
flash photography	0.07
wedding photography	0.07
dslr photography	0.07
landscape photography	0.07
macro photography	0.07
street photography	0.07
night photography	0.06
nude photography	0.06
nikon photography	0.05
newborn photography	0.05
photography free courses	0.04
karl taylor photography	0.04
nikon	0.01


The Complete Web Developer Course - Build 14 Websites

Web development from scratch in 6 weeks. HTML, JS, CSS, PHP, MySQL, Bootstrap, Wordpress, APIs, Mobile Apps all covered. Lifetime access with no subscription on Udemy.com.

4.5 (26,948 ratings) • 232,904 students enrolled

Instructed by Rob Percival Development / Web Development

$200

Take This Course

Add to Cart

Redeem a Coupon

Start Free Preview

More Options

Lectures	**236**
Exercises	**3 coding exercises**
Length	**28 hours**
Skill Level	**All Levels**
Languages	**English**
Includes	**Coding Exercises** ***New!*** **Lifetime access** **30 day money back guarantee!** **Available on iOS and Android** **Certificate of Completion**

Wishlist

Figure 4: An example course on web development at Udemy

Table 2: Top tags generated for the course given in Figure 4

Web development
Web
Wordpress
Personal Development
PHP
JavaScript
HTML

where $u_{a,j}$ is the user a's feedback on product j. The sum is performed over user a's feedback on courses similar to i weighted by the similarity of the item to i [5].

The predictions, $p_{a,i}$ are then compared with $u_{a,i}$. Two metrics are used to evaluate performance, namely area under the receiver operating characteristics curve (AUROC) [7] and area under the precision-recall curve (AUPRC) [21]. These metrics are benchmarked against an item-based collaborative filtering algorithm implemented using Apache Mahout [1]. In the collaborative filtering algorithm, similarity is defined through Tanimoto coefficients, which are calibrated using historical data on course enrollments. Using the product similarity computations outlined in this paper, AUROC and AUPRC are improved by 2.1% and 4%, respectively, in predicting Udemy's user course enrollments compared to collaborative filtering predictions.

3.4 Online Experiments

The algorithm is used to improve the recommendations on Udemy's homepage and landing pages. In order to monitor user engagement and the impact on business metrics, we design randomized, controlled experiments (A/B tests) [14]. We create experimental variants and power recommendations in each group by different algorithms. We then randomly assign visitors to each experiment variant and monitor users' actions in each group until we can detect differences in the experiment groups with statistical confidence. Finally, we quantify the differences in experiment metrics, which include in-session clicks, purchases, revenue and video consumption.

The first online experiment using the algorithm is performed on the search-based recommendation unit ("Because You Searched") in the homepage. This unit provides recommendations based on the user's last searched query. We perform controlled experiments with a change in the underlying algorithm for this unit. In the control variant, the unit operates as follows: For a given query, our search engine is used to retrieve the most relevant course. This course is then used as a seed course to our collaborative filtering algorithm to retrieve similar courses. In the experimental variant, we use the search query itself to find the tag it belongs. We then retrieve the most relevant courses for a given tag. This approach has the benefit of using user-feedback for query-course relations compared to using results directly from the search engine. This experiment is run over two months. The final experiment resulted in the following differences in metrics at 95% confidence:

- In-session revenue from this unit is increased by 170%. From side-by-side analysis, we hypothesize that this lift is due to improved relevancy of recommendations.
- Overall sitewide video consumption is increased by 6%, which points to the higher level of user engagement.

We also perform online experiments using the click-based recommendation unit ("Because You Viewed") on homepage and landing pages. This unit provides recommendations based on the user's last clicked course. In the control variant the unit operates as follows: For a given seed course, item-based collaborative filtering algorithm is used to retrieve similar courses. In the experimental variant, we use the product similarity computations outlined in this paper to retrieve similar courses to the seed course. This experiment is run approximately three months. With 95% confidence, the final experiment results indicated that the overall in-session revenue from this unit is increased by 13%.

4 CONCLUSION

In this paper, we present a statistical framework for automatically discovering product features from algorithmically defined tags, probabilistically modeling users and products with these tags and using the tag modeling to improve recommendation engines. The framework is based on a graph model of search queries. The graph is analyzed using community detection algorithms to retrieve tags. The products and users are then mapped to the tags probabilistically. This enables product and user similarity computations, which can be used with recommender systems.

We use this framework to improve personalized recommendations at Udemy. The framework is illustrated using Udemy's historical user history data. We then conduct offline and online experiments to evaluate the performance of the recommendations. The results are promising; online experiments showed significant lifts on key business metrics.

Our future work will explore on using the framework to power different parts of the recommendations at Udemy. User tag modeling will be combined with personalized course scores to add a layer of tag interest to our recommendations. We will also explore on improving the search query graph using keyword extraction on course metadata.

5 ACKNOWLEDGEMENTS

The authors would like to thank Keeyong Han for valuable guidance and support. Erol Aran, Heval Azizoglu, Cagatay Calli, Melda Dadandi, James Hou, Okan Kahraman, Ahmet Korkmaz, Ozgen Muzac, Gulsen Kutluoglu and Ibrahim Tasyurt implemented key components of the infrastructure. Sara Hooker, Imeh Williams and Mars Williams contributed detailed feedback.

REFERENCES

[1] Apache Mahout 2017. Apache Mahout, http://mahout.apache.org. (2017). http://mahout.apache.org
[2] Doug Beeferman and Adam Berger. 2000. Agglomerative clustering of a search engine query log. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 407–416.
[3] David M Blei and John D Lafferty. 2009. Topic models. *Text mining: classification, clustering, and applications* 10, 71 (2009), 34.
[4] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3, Jan (2003), 993–1022.
[5] Fidel Cacheda, Víctor Carneiro, Diego Fernández, and Vreixo Formoso. 2011. Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. *ACM Transactions on the Web (TWEB)* 5, 1 (2011), 2.

[6] Konstantinos Christidis and Gregoris Mentzas. 2013. A topic-based recommender system for electronic marketplace platforms. *Expert Systems with Applications* 40, 11 (2013), 4370–4379.

[7] Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern recognition letters* 27, 8 (2006), 861–874.

[8] Santo Fortunato. 2010. Community detection in graphs. *Physics reports* 486, 3 (2010), 75–174.

[9] Michelle Girvan and Mark EJ Newman. 2002. Community structure in social and biological networks. *Proceedings of the national academy of sciences* 99, 12 (2002), 7821–7826.

[10] Scott A Golder and Bernardo A Huberman. 2006. Usage patterns of collaborative tagging systems. *Journal of information science* 32, 2 (2006), 198–208.

[11] Negar Hariri, Bamshad Mobasher, and Robin Burke. 2013. Query-driven context aware recommendation. In *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 9–16.

[12] Paul Heymann, Georgia Koutrika, and Hector Garcia-Molina. 2008. Can social bookmarking improve web search?. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*. ACM, 195–206.

[13] Paul Heymann, Daniel Ramage, and Hector Garcia-Molina. 2008. Social tag prediction. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 531–538.

[14] Ron Kohavi, Randal M Henne, and Dan Sommerfield. 2007. Practical guide to controlled experiments on the web: listen to your customers not to the hippo. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 959–967.

[15] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. 2014. *Mining of massive datasets*. Cambridge University Press.

[16] Cameron Marlow, Mor Naaman, Danah Boyd, and Marc Davis. 2006. HT06, Tagging Paper, Taxonomy, Flickr, Academic Article, to Read. In *Proceedings of the Seventeenth Conference on Hypertext and Hypermedia (HYPERTEXT '06)*. ACM, New York, NY, USA, 31–40. DOI: https://doi.org/10.1145/1149941.1149949

[17] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 165–172.

[18] Milind Naphade, John R Smith, Jelena Tesic, Shih-Fu Chang, Winston Hsu, Lyndon Kennedy, Alexander Hauptmann, and Jon Curtis. 2006. Large-scale concept ontology for multimedia. *IEEE multimedia* 13, 3 (2006), 86–91.

[19] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The PageRank citation ranking: bringing order to the web. *Stanford InfoLab* (1999).

[20] Pascal Pons and Matthieu Latapy. 2005. Computing communities in large networks using random walks. In *International Symposium on Computer and Information Sciences*. Springer, 284–293.

[21] Badrul M Sarwar, Joseph A Konstan, Al Borchers, Jon Herlocker, Brad Miller, and John Riedl. 1998. Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system. In *Proceedings of the 1998 ACM conference on Computer supported cooperative work*. ACM, 345–354.

[22] Börkur Sigurbjörnsson and Roelof Van Zwol. 2008. Flickr tag recommendation based on collective knowledge. In *Proceedings of the 17th international conference on World Wide Web*. ACM, 327–336.

[23] Larry Wai. 2016. Data Science at Udemy: Agile Experimentation with Algorithms. *arXiv preprint arXiv:1602.05142* (2016).

[24] Chong Wang and David M Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 448–456.

[25] Ji-Rong Wen, Jian-Yun Nie, and Hong-Jiang Zhang. 2001. Clustering user queries of a search engine. In *Proceedings of the 10th international conference on World Wide Web*. acm, 162–168.

[26] Jeonghee Yi and Farzin Maghoul. 2009. Query clustering using click-through graph. In *Proceedings of the 18th international conference on World wide web*. ACM, 1055–1056.