Amazon.com Recommendations

Item-to-Item Collaborative Filtering

Greg Linden, Brent Smith, and Jeremy York • *Amazon.com*

Recommendation algorithms are best known for their use on e-commerce Web sites,[1] where they use input about a customer's interests to generate a list of recommended items. Many applications use only the items that customers purchase and explicitly rate to represent their interests, but they can also use other attributes, including items viewed, demographic data, subject interests, and favorite artists.

At Amazon.com, we use recommendation algorithms to personalize the online store for each customer. The store radically changes based on customer interests, showing programming titles to a software engineer and baby toys to a new mother. The click-through and conversion rates — two important measures of Web-based and email advertising effectiveness — vastly exceed those of untargeted content such as banner advertisements and top-seller lists.

E-commerce recommendation algorithms often operate in a challenging environment. For example:

- A large retailer might have huge amounts of data, tens of millions of customers and millions of distinct catalog items.
- Many applications require the results set to be returned in realtime, in no more than half a second, while still producing high-quality recommendations.
- New customers typically have extremely limited information, based on only a few purchases or product ratings.
- Older customers can have a glut of information, based on thousands of purchases and ratings.
- Customer data is volatile: Each interaction provides valuable customer data, and the algorithm must respond immediately to new information.

There are three common approaches to solving the recommendation problem: traditional collaborative filtering, cluster models, and search-based methods. Here, we compare these methods with our algorithm, which we call *item-to-item collaborative filtering*. Unlike traditional collaborative filtering, our algorithm's online computation scales independently of the number of customers and number of items in the product catalog. Our algorithm produces recommendations in realtime, scales to massive data sets, and generates high-quality recommendations.

Recommendation Algorithms

Most recommendation algorithms start by finding a set of customers whose purchased and rated items overlap the user's purchased and rated items.[2] The algorithm aggregates items from these similar customers, eliminates items the user has already purchased or rated, and recommends the remaining items to the user. Two popular versions of these algorithms are *collaborative filtering* and *cluster models*. Other algorithms — including search-based methods and our own item-to-item collaborative filtering — focus on finding similar items, not similar customers. For each of the user's purchased and rated items, the algorithm attempts to find similar items. It then aggregates the similar items and recommends them.

Traditional Collaborative Filtering

A traditional collaborative filtering algorithm represents a customer as an N-dimensional vector of items, where N is the number of distinct catalog items. The components of the vector are positive for purchased or positively rated items and negative for negatively rated items. To compensate for

 Published by the IEEE Computer Society 1089-7801/03/$17.00©2003 IEEE

best-selling items, the algorithm typically multiplies the vector components by the inverse frequency (the inverse of the number of customers who have purchased or rated the item), making less well-known items much more relevant.[3] For almost all customers, this vector is extremely sparse.

The algorithm generates recommendations based on a few customers who are most similar to the user. It can measure the similarity of two customers, A and B, in various ways; a common method is to measure the cosine of the angle between the two vectors:[4]

$$similarity(\vec{A}, \vec{B}) = \cos(\vec{A}, \vec{B}) = \frac{\vec{A} \bullet \vec{B}}{\|\vec{A}\| * \|\vec{B}\|}$$

The algorithm can select recommendations from the similar customers' items using various methods as well, a common technique is to rank each item according to how many similar customers purchased it.

Using collaborative filtering to generate recommendations is computationally expensive. It is $O(MN)$ in the worst case, where M is the number of customers and N is the number of product catalog items, since it examines M customers and up to N items for each customer. However, because the average customer vector is extremely sparse, the algorithm's performance tends to be closer to $O(M + N)$. Scanning every customer is approximately $O(M)$, not $O(MN)$, because almost all customer vectors contain a small number of items, regardless of the size of the catalog. But there are a few customers who have purchased or rated a significant percentage of the catalog, requiring $O(N)$ processing time. Thus, the final performance of the algorithm is approximately $O(M + N)$. Even so, for very large data sets — such as 10 million or more customers and 1 million or more catalog items — the algorithm encounters severe performance and scaling issues.

It is possible to partially address these scaling issues by reducing the data size.[4] We can reduce M by randomly sampling the customers or discarding customers with few purchases, and reduce N by discarding very popular or unpopular items. It is also possible to reduce the number of items examined by a small, constant factor by partitioning the item space based on product category or subject classification. Dimensionality reduction techniques such as clustering and principal component analysis can reduce M or N by a large factor.[5]

Unfortunately, all these methods also reduce recommendation quality in several ways. First, if the algorithm examines only a small customer sample, the selected customers will be less similar to the user. Second, item-space partitioning restricts recommendations to a specific product or subject area. Third, if the algorithm discards the most popular or unpopular items, they will never appear as recommendations, and customers who have purchased only those items will not get recommendations. Dimensionality reduction techniques applied to the item space tend to have the same effect by eliminating low-frequency items. Dimensionality reduction applied to the customer space effectively groups similar customers into clusters; as we now describe, such clustering can also degrade recommendation quality.

Cluster Models

To find customers who are similar to the user, cluster models divide the customer base into many segments and treat the task as a classification problem. The algorithm's goal is to assign the user to the segment containing the most similar customers. It then uses the purchases and ratings of the customers in the segment to generate recommendations.

The segments typically are created using a clustering or other unsupervised learning algorithm, although some applications use manually determined segments. Using a similarity metric, a clustering algorithm groups the most similar customers together to form clusters or segments. Because optimal clustering over large data sets is impractical, most applications use various forms of greedy cluster generation. These algorithms typically start with an initial set of segments, which often contain one randomly selected customer each. They then repeatedly match customers to the existing segments, usually with some provision for creating new or merging existing segments.[6] For very large data sets — especially those with high dimensionality — sampling or dimensionality reduction is also necessary.

Once the algorithm generates the segments, it computes the user's similarity to vectors that summarize each segment, then chooses the segment with the strongest similarity and classifies the user accordingly. Some algorithms classify users into multiple segments and describe the strength of each relationship.[7]

Cluster models have better online scalability and performance than collaborative filtering[3] because they compare the user to a controlled number of segments rather than the entire cus-

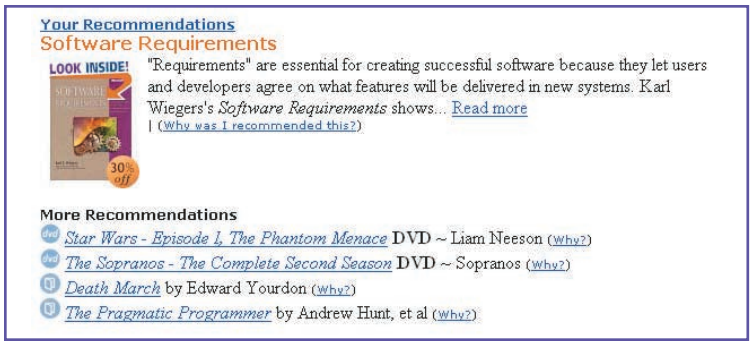

Figure 1. The "Your Recommendations" feature on the Amazon.com homepage. Using this feature, customers can sort recommendations and add their own product ratings.

Figure 2. Amazon.com shopping cart recommendations. The recommendations are based on the items in the customer's cart: The Pragmatic Programmer *and* Physics for Game Developers.

tomer base. The complex and expensive clustering computation is run offline. However, recommendation quality is low.[1] Cluster models group numerous customers together in a segment, match a user to a segment, and then consider all customers in the segment similar customers for the purpose of making recommendations. Because the similar customers that the cluster models find are not the most similar customers, the recommendations they produce are less relevant. It is possible to improve quality by using numerous fine-grained segments, but then online user–segment classification becomes almost as expensive as finding similar customers using collaborative filtering.

Search-Based Methods

Search- or content-based methods treat the recommendations problem as a search for related items.[8] Given the user's purchased and rated items, the algorithm constructs a search query to find other popular items by the same author, artist, or director, or with similar keywords or subjects. If a customer buys the Godfather DVD Collection, for example, the system might recommend other crime drama titles, other titles starring Marlon Brando, or other movies directed by Francis Ford Coppola.

If the user has few purchases or ratings, search-based recommendation algorithms scale and perform well. For users with thousands of purchases, however, it's impractical to base a query on all the items. The algorithm must use a subset or summary of the data, reducing quality. In all cases, recommendation quality is relatively poor. The recommendations are often either too general (such as best-selling drama DVD titles) or too narrow (such as all books by the same author). Recommendations should help a customer find and discover new, relevant, and interesting items. Popular items by the same author or in the same subject category fail to achieve this goal.

Item-to-Item Collaborative Filtering

Amazon.com uses recommendations as a targeted marketing tool in many email campaigns and on most of its Web sites' pages, including the high-traffic Amazon.com homepage. Clicking on the "Your Recommendations" link leads customers to an area where they can filter their recommendations by product line and subject area, rate the recommended products, rate their previous purchases, and see why items are recommended (see Figure 1).

As Figure 2 shows, our shopping cart recommendations, which offer customers product suggestions based on the items in their shopping cart. The feature is similar to the impulse items in a supermarket checkout line, but our impulse items are targeted to each customer.

Amazon.com extensively uses recommendation algorithms to personalize its Web site to each customer's interests. Because existing recommendation algorithms cannot scale to Amazon.com's tens of millions of customers and products, we developed our own. Our algorithm, item-to-item collaborative filtering, scales to massive data sets and produces high-quality recommendations in real time.

How It Works

Rather than matching the user to similar customers, item-to-item collaborative filtering matches each of the user's purchased and rated items to similar items, then combines those similar items into a recommendation list.[9]

To determine the most-similar match for a given item, the algorithm builds a similar-items table by finding items that customers tend to purchase together. We could build a product-to-product matrix by iterating through all item pairs and computing a similarity metric for each pair. However, many product pairs have no common customers, and thus the approach is inefficient in terms of processing time and memory usage. The following

iterative algorithm provides a better approach by calculating the similarity between a single product and all related products:

```
For each item in product catalog, I1
  For each customer C who purchased I1
    For each item I2 purchased by
       customer C
      Record that a customer purchased I1
       and I2
  For each item I2
    Compute the similarity between I1 and I2
```

It's possible to compute the similarity between two items in various ways, but a common method is to use the cosine measure we described earlier, in which each vector corresponds to an item rather than a customer, and the vector's M dimensions correspond to customers who have purchased that item.

This offline computation of the similar-items table is extremely time intensive, with $O(N^2M)$ as worst case. In practice, however, it's closer to $O(NM)$, as most customers have very few purchases. Sampling customers who purchase best-selling titles reduces runtime even further, with little reduction in quality.

Given a similar-items table, the algorithm finds items similar to each of the user's purchases and ratings, aggregates those items, and then recommends the most popular or correlated items. This computation is very quick, depending only on the number of items the user purchased or rated.

Scalability: A Comparison

Amazon.com has more than 29 million customers and several million catalog items. Other major retailers have comparably large data sources. While all this data offers opportunity, it's also a curse, breaking the backs of algorithms designed for data sets three orders of magnitude smaller. Almost all existing algorithms were evaluated over small data sets. For example, the MovieLens data set[4] contains 35,000 customers and 3,000 items, and the EachMovie data set[3] contains 4,000 customers and 1,600 items.

For very large data sets, a scalable recommendation algorithm must perform the most expensive calculations offline. As a brief comparison shows, existing methods fall short:

- Traditional collaborative filtering does little or no offline computation, and its online computation scales with the number of customers and catalog items. The algorithm is impractical on large data sets, unless it uses dimensionality reduction, sampling, or partitioning — all of which reduce recommendation quality.
- Cluster models can perform much of the computation offline, but recommendation quality is relatively poor. To improve it, it's possible to increase the number of segments, but this makes the online user–segment classification expensive.
- Search-based models build keyword, category, and author indexes offline, but fail to provide recommendations with interesting, targeted titles. They also scale poorly for customers with numerous purchases and ratings.

The key to item-to-item collaborative filtering's scalability and performance is that it creates the expensive similar-items table offline. The algorithm's online component — looking up similar items for the user's purchases and ratings — scales independently of the catalog size or the total number of customers; it is dependent only on how many titles the user has purchased or rated. Thus, the algorithm is fast even for extremely large data sets. Because the algorithm recommends highly correlated similar items, recommendation quality is excellent.[10] Unlike traditional collaborative filtering, the algorithm also performs well with limited user data, producing high-quality recommendations based on as few as two or three items.

Conclusion

Recommendation algorithms provide an effective form of targeted marketing by creating a personalized shopping experience for each customer. For large retailers like Amazon.com, a good recommendation algorithm is scalable over very large customer bases and product catalogs, requires only subsecond processing time to generate online recommendations, is able to react immediately to changes in a user's data, and makes compelling recommendations for all users regardless of the number of purchases and ratings. Unlike other algorithms, item-to-item collaborative filtering is able to meet this challenge.

In the future, we expect the retail industry to more broadly apply recommendation algorithms for targeted marketing, both online and offline. While e-commerce businesses have the easiest vehicles for personalization, the technology's increased conversion rates as compared with traditional broad-scale approaches will also make it compelling to offline retailers for use in postal mailings, coupons, and other forms of customer communication.

ADVERTISER / PRODUCT INDEX

JANUARY / FEBRUARY 2003

Advertiser / Product	Page Number
John Wiley & Sons	Inside Back Cover
CTIA Wireless	Back Cover

Advertising Personnel

Marion Delaney
IEEE Media, Advertising Director
Phone: +1 212 419 7766
Fax: +1 212 419 7589
Email: md.ieeemedia@ieee.org

Marian Anderson
Advertising Coordinator
Phone: +1 714 821 8380
Fax: +1 714 821 4010
Email: manderson@computer.org

Sandy Brown
IEEE Computer Society,
Business Development Manager
Phone: +1 714 821 8380
Fax: +1 714 821 4010
Email: sb.ieeemedia@ieee.org

Debbie Sims
Assistant Advertising Coordinator
Phone: +1 714 821 8380
Fax: +1 714 821 4010
Email: dsims@computer.org

Advertising Sales Representatives

Mid Atlantic (product/recruitment)
Dawn Becker
Phone: +1 732 772 0160
Fax: +1 732 772 0161
Email: db.ieeemedia@ieee.org

Midwest (product)
David Kovacs
Phone: +1 847 705 6867
Fax: +1 847 705 6878
Email: dk.ieeemedia@ieee.org

New England (product)
Jody Estabrook
Phone: +1 978 244 0192
Fax: +1 978 244 0103
Email: je.ieeemedia@ieee.org

Southwest (product)
Royce House
Phone: +1 713 668 1007
Fax: +1 713 668 1176
Email: rh.ieeemedia@ieee.org

Northwest (product)
John Gibbs
Phone: +1 415 929 7619
Fax: +1 415 577 5198
Email: jg.ieeemedia@ieee.org

Southern CA (product)
Marshall Rubin
Phone: +1 818 888 2407
Fax: +1 818 888 4907
Email: mr.ieeemedia@ieee.org

Midwest (product)
Dave Jones
Phone: +1 708 442 5633
Fax: +1 708 442 7620
Email: dj.ieeemedia@ieee.org
Will Hamilton
Phone: +1 269 381 2156
Fax: +1 269 381 2556
Email: wh.ieeemedia@ieee.org
Joe DiNardo
Phone: +1 440 248 2456
Fax: +1 440 248 2594
Email: jd.ieeemedia@ieee.org

Southeast (product/recruitment)
C. William Bentz III
Email: bb.ieeemedia@ieee.org
Gregory Maddock
Email: gm.ieeemedia@ieee.org
Sarah K. Wiley
Email: sh.ieeemedia@ieee.org
Phone: +1 404 256 3800
Fax: +1 404 255 7942

Midwest/Southwest recruitment)
Tom Wilcoxen
Phone: +1 847 498 4520
Fax: +1 847 498 5911
Email: tw.ieeemedia@ieee.org

New England (recruitment)
Barbara Lynch
Phone: +1 401 738 6237
Fax: +1 401 739 7970
Email: bl.ieeemedia@ieee.org

Connecticut (product)
Stan Greenfield
Phone: +1 203 938 2418
Fax: +1 203 938 3211
Email: greenco@optonline.net

Northwest (recruitment)
Mary Tonon
Phone: +1 415 431 5333
Fax: +1 415 431 5335
Email: mt.ieeemedia@ieee.org

Southern CA (recruitment)
Tim Matteson
Phone: +1 310 836 4064
Fax: +1 310 836 4067
Email: tm.ieeemedia@ieee.org

Japan
German Tajiri
Phone: +81 42 501 9551
Fax: +81 42 501 9552
Email: gt.ieeemedia@ieee.org

Europe (product)
Hilary Turnbull
Phone: +44 131 660 6605
Fax: +44 131 660 6989
Email: impress@impressmedia.com

References

1. J.B. Schafer, J.A. Konstan, and J. Reidl, "E-Commerce Recommendation Applications," *Data Mining and Knowledge Discovery*, Kluwer Academic, 2001, pp. 115-153.
2. P. Resnick et al., "GroupLens: An Open Architecture for Collaborative Filtering of Netnews," *Proc. ACM 1994 Conf. Computer Supported Cooperative Work*, ACM Press, 1994, pp. 175-186.
3. J. Breese, D. Heckerman, and C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," *Proc. 14th Conf. Uncertainty in Artificial Intelligence*, Morgan Kaufmann, 1998, pp. 43-52.
4. B.M. Sarwarm et al., "Analysis of Recommendation Algorithms for E-Commerce," *ACM Conf. Electronic Commerce*, ACM Press, 2000, pp.158-167.
5. K. Goldberg et al., "Eigentaste: A Constant Time Collaborative Filtering Algorithm," *Information Retrieval J.*, vol. 4, no. 2, July 2001, pp. 133-151.
6. P.S. Bradley, U.M. Fayyad, and C. Reina, "Scaling Clustering Algorithms to Large Databases," *Knowledge Discovery and Data Mining*, Kluwer Academic, 1998, pp. 9-15.
7. L. Ungar and D. Foster, "Clustering Methods for Collaborative Filtering," *Proc. Workshop on Recommendation Systems*, AAAI Press, 1998.
8. M. Balabanovic and Y. Shoham, "Content-Based Collaborative Recommendation," *Comm. ACM*, Mar. 1997, pp. 66-72.
9. G.D. Linden, J.A. Jacobi, and E.A. Benson, *Collaborative Recommendations Using Item-to-Item Similarity Mappings*, US Patent 6,266,649 (to Amazon.com), Patent and Trademark Office, Washington, D.C., 2001.
10. B.M. Sarwar et al., "Item-Based Collaborative Filtering Recommendation Algorithms," *10th Int'l World Wide Web Conference*, ACM Press, 2001, pp. 285-295.

Greg Linden was cofounder, researcher, and senior manager in the Amazon.com Personalization Group, where he designed and developed the recommendation algorithm. He is currently a graduate student in management in the Sloan Program at Stanford University's Graduate School of Business. His research interests include recommendation systems, personalization, data mining, and artificial intelligence. Linden received an MS in computer science from the University of Washington. Contact him at Linden_Greg@gsb.stanford.edu.

Brent Smith leads the Automated Merchandising team at Amazon.com. His research interests include data mining, machine learning, and recommendation systems. He received a BS in mathematics from the University of California, San Diego, and an MS in mathematics from the University of Washington, where he did graduate work in differential geometry. Contact him at smithbr@amazon.com.

Jeremy York leads the Automated Content Selection and Delivery team at Amazon.com. His interests include statistical models for categorical data, recommendation systems, and optimal choice of Web site display components. He received a PhD in statistics from the University of Washington, where his thesis won the Leonard J. Savage award for best thesis in applied Bayesian econometrics and statistics. Contact him at jeremy@amazon.com.