Amazon Search: The Joy of Ranking Products

Daria Sorokina
A9.com
130 Lytton, Palo Alto, CA, US
dariasor@a9.com

Erick Cantú-Paz
A9.com
130 Lytton, Palo Alto, CA, US
cantupaz@a9.com

ABSTRACT

Amazon is one of the world's largest e-commerce sites and Amazon Search powers the majority of Amazon's sales. As a consequence, even small improvements in relevance ranking both positively influence the shopping experience of millions of customers and significantly impact revenue. In the past, Amazon's product search engine consisted of several hand-tuned ranking functions using a handful of input features. A lot has changed since then. In this talk we are going to cover a number of relevance algorithms used in Amazon Search today. We will describe a general machine learning framework used for ranking within categories, blending separate rankings in All Product Search, NLP techniques used for matching queries and products, and algorithms targeted at unique tasks of specific categories — books and fashion.

1. RANKING MODELS

Ranking models are responsible for a function that, given a customer's query, returns a sorted list of products in a match set. A single ranking model usually covers a combination of a category and a marketplace, e.g., Books in Japan.

For training the ranking models we use labels based on customers actions, such as purchases, add-to-basket, or clicks.

We use the search engine to collect our training sets. Several times per day we compute the unique set of keywords issued for each context of interest. The context can be a combination of marketplace, category, and some user features. Then we re-issue these queries in their context requesting feature values for all items in the match set. This feature collection runs regularly, so that the feature vector collected is as close as possible to the one observed when the query was originally issued by customers.

To train ranking models, we construct training, validation, and test sets by collecting data from several days of customer traffic. Test sets are constructed from dates after the training set dates. We choose impressions that resulted in either click or purchase as positive examples. There are two types of negative example impressions: seen, corresponding to items which were displayed to a customer, and unseen, corresponding to items which matched the query terms but were never shown due to pagination. To manage the size of the training set, we sample unseen examples.

Gradient boosted trees[1] are our method of choice in ranking because they can discover complex feature interactions, can handle categorical and real-valued features, are robust in the presence of missing values, and work well even without significant tuning. For ranking problems, we use models trained with pairwise objectives and nDCG as the default objective function.

We do feature selection in two stages. First, with fixed values of tree depth and learning rate, we train a "kitchen sink" model allowing all features available. We then discard the features which end up being ranked lower than random features and use the remaining features to grow the final feature set in a forward selection process. Once the feature set is chosen, we perform a grid search over the pairs of values for tree depth and learning rate, choosing the model which has the highest offline score on the validation set.

Finally, the model is evaluated through an A/B test. We look at a large variety of metrics, including the number of converting customers, the number of products they purchase and the overall revenue.

2. BEHAVIORAL FEATURES

When training ranking models we use many features. Some of them measure intrinsic properties of products (e.g., sales, reviews). Others reflect properties of the queries or the context in which the query is issued (e.g., query specificity, customer status). Other features provide different measures of textual similarity. However in product search, hundreds of products might share very similar descriptions and seem to be equally relevant to a particular query. But some of those products are more popular than others and should be ranked higher. That's why behavioral features drive the rankings in Amazon Search to a much larger extent than they do in Web Search. Typically, they account for most of the variance reduction in gradient-boosted trees.

It is well-known that users tend to click more often on results on the top of a search result page. To correct for that, we have tried to use classical versions of click-over-expected-clicks features[3]. However, since more relevant documents tend to appear higher in the ranking, the observed click-through rate at a given position captures not only the position bias, but also the typical relevance at this position. This problem is more pronounced for a product

SIGIR '16 July 17-21, 2016, Pisa, Italy

ACM ISBN 978-1-4503-4069-4/16/07.
DOI: http://dx.doi.org/10.1145/2911451.2926725

search engine, compared to web search or advertising, as more products on average are relevant to a single query, and more clicks on average happen at higher position. To correct for that, we developed a scheme that adapts the bias correction from day to day.

3. MATCHING PRODUCTS AND QUERIES

A major problem in understanding queries in product search is determining whether a word or phrase in the query refers to a product type. For example, recognizing that "dress" in the query "casual dress" refers to a product type and not a modifier as in "dress shoes" or "dress socks" allows the systematic exclusion of non-dress products from the search results.

We treat each query as a noun phrase and consider the head of the noun phrase to be the core product type and all other words in the query to be modifiers. This is expressed as a probabilistic context-free grammar (PCFG), which is trained in an unsupervised manner using Variational Bayes. Limited supervision is inserted into the model by using brands from the Amazon catalog as well as other common modifiers such as color and gender. The set of product types observed in the training set under this model is extended to incorporate multiword expressions which encompass the product types (like "t shirt"), and the model is retrained with this augmented set of possible product types.

The second part of the task is to assign each product all the product-type expressions that properly describe it. For example, if the product is a tshirt, we want to label it "tshirt","shirt","clothing". This problem is naturally framed as a multilabel classification problem and the predictor can be trained as a series of logistic regression models. We represent the multilabel structure by a hypernym-synonym graph over the various product types and use it to enforce consistency constraints on the final predictions.

We use detected product types in queries and product descriptions to create powerful features in ranking models.

4. FINDING PREFERRED EDITION

Some of the product categories present unique problems. For example, books come in a variety of different editions (e.g. hardcover, paperback, digital, audiobook, etc). In search results we surface only a single, representative preferred edition in order to avoid overwhelming the user. Customers can then select a specific variation of the product on the details page. We use a separate combination of gradient boosted tree models to predict the face out edition for each book. These models are using only query-independent features, such as publication date or binding format, and most of the score is precalculated offline.

5. RANKING IN FASHION

Ranking in Amazon Fashion presents several unique challenges due to a large and varied product catalog, users with diverse fashion preferences, and business requirements to make the store appear fashionable and fresh.

We found that optimizing for an individual target (such as click, add-to-cart, or purchase) does not achieve the desired outcome. For example, while customers tend to purchase discounted items, they often browse and click on the items that are high-end fashion. We discovered that optimizing for a fused target that combines purchases with clicks often satisfies users' information need better in the Fashion Store.

An interesting challenge in the Fashion Store is the discrepancy between what the majority of customers actually buy and what they want to see on top of the page. The item most commonly bought for the query "diamond ring" might be a cheap zirconium ring. However, if we show the zirconium ring as a first result, our search will be perceived as broken. Besides, our Fashion Store would look like a flea market, instead of a classic department store where the latest collections meet you at the entrance.

To approach this problem, we identify strategic categories of fashionable customers — customers who bought or added to cart fashion brand products — and significantly amplify their influence while designing the training set.

6. BLENDING ACROSS CATEGORIES

Ranking models provide rankings within categories, but customers often use the "Search All" option on the search box. One of the biggest challenges in this case is to provide a blend of results from different categories. For example, for the query "game of thrones" customers may be looking for a book, a DVD, or a board game. The ordering of results returned by any ranking model is not changed, we only interleave results from individual categories based on an estimate of the probability that a given query is associated with a particular category. For the most common queries, this probability is based on observed user actions following that query. For tail queries, we predict category intent using an n-gram based language model. We use all queries with clicks on products from a given category over the last 90 days as text corpus for this category. Then, the affinity for a given query and a category is the probability of the query text to be observed in the category corpus. We use trigram language model with Modified Kneser-Ney smoothing[2] for this task.

7. ACKNOWLEDGEMENTS

This talk provides description of a number of past and current projects in Amazon Search. As such, this paper has a large number of authors, including, but probably not limited to, **Praveen Arasada, William Headden, François Huet, Anish Nair, Anil Sewani, Sheng Peng, Stefan Schroedl, Chris Varano, Bing Yin**.

8. REFERENCES

[1] J. Friedman. Greedy Function Approximation: a Gradient Boosting Machine. *Annals of Statistics*, 29:1189–1232, 2001.

[2] F. James. Modified Kneser-Ney Smoothing of N-gram Models. Technical report, 2000.

[3] W. V. Zhang and R. Jones. Comparing click logs and editorial labels for training query rewriting. In *WWW'07 workshops*.