The New York Times

Building the Next New York Times Recommendation Engine

By

August 11, 2015 11:27 am

The New York Times publishes over 300 articles, blog posts and interactive stories a day.

Refining the path our readers take through this content — personalizing the placement of articles on our apps and website — can help readers find information relevant to them, such as the right news at the right times, personalized supplements to major events and stories in their preferred multimedia format.

In this post, I'll discuss our recent work revamping The New York Times's article recommendation algorithm, which currently serves behind the Recommended for You section of NYTimes.com.

History

Content-based filtering

News recommendations must perform well on fresh content: breaking news that hasn't been viewed by many readers yet. Thus, the article data available at publishing time can be useful: the topics, author, desk and associated keyword tags of each article.

Our first recommendation engine used these keyword tags to make recommendations. Using tags for articles and a user's 30-day reading history, the algorithm recommends articles similar to those that have already been read.

Because this technique relies on a content model, it's part of a broader class of

content-based recommendation algorithms.

The approach has intuitive appeal: If a user read ten articles tagged with the word "Clinton," they would probably like future "Clinton"-tagged articles. And this technique performs as well on fresh content as it does on older content, since it relies on data available at the time of publishing.

However, this method relies on a content model that, sometimes, has unintended effects. Because the algorithm weights tags by their rareness within a corpus, rare tags have a large effect. This works well most of the time, but occasionally degrades the user experience. For instance, one reader noted that while she was interested in same-sex pieces, occasionally in the Weddings section, she was being recommended wedding coverage about heterosexual couples. This is because a low-frequency tag, "Weddings and Engagements," was in an article previously clicked, outweighing all other tags that may have been more applicable for that reader.

Collaborative Filtering

To accommodate the shortcomings of the previous method, we tested collaborative filtering. Collaborative filters surface articles based on what similar readers have read; in our case, similarity was determined by reading history.

This approach is also appealing: If one reader's preferences are very similar to another reader's, articles that the first reader reads might interest the second, and vice versa.

However, this approach fails at recommending newly-published, unexplored articles: articles that were relevant to groups of readers but hadn't yet been read by any reader in that group. A collaborative filter might also, hypothetically, cluster reading patterns in narrow viewpoints.

Current Approach

It turns out that straddling both techniques can give us the best of both worlds. We built an algorithm inspired by a technique, Collaborative Topic Modeling (CTM), that (1) models content, (2) adjusts this model by viewing signals from readers, (3) models reader preference and (4) makes recommendations by similarity between preference and

content.

Overview

Our algorithm starts by modeling each article as a mixture of the topics it addresses. We can think of a topic as an unobserved theme, like **Politics** or **Environment**, that affects the words observed in the article. For example, if an article is about the environment, we'd expect words like "tree" or "conservation."

We model each reader based on their topic preferences. We can then recommend articles based on how closely their topics match a reader's preferred topics.

As an example, we run our algorithm supposing that all New York Times articles published in the last month can be represented as a combination of two topics. Under these constraints, the algorithm identifies these topics, roughly, as **Politics** and **Art**. Our algorithm finds an article, America Deepens its Footprint in Iraq Once More, as 100% **Politics**, and a film review by A.O. Scott as 100% **Art**. It labels mixtures, too; for example, an article about art politics, Frick Museum Abandons Contested Renovation Plan, is labeled 50% **Politics**, 50% **Art**.

In this **Politics**–**Art** space, we might describe our articles in the following manner:

Next, suppose that one reader prefers to read about **Art** 60% of the time and **Politics** 40% of the time. We might represent that reader with the red x. The magical part is that they are spatially close to articles that align with their interests, even if they haven't read them yet; we recommend articles that are closest to them in the space.

There are further questions for us to answer. Can this topic space capture ambiguous word usage? And how do we best observe the preferences of our readers? Clicks are, after all, not robust: I'm sure that at some point, you have clicked on something you didn't enjoy and missed something you would have found interesting.

We tested many options carefully; the algorithm we built brings us closer to addressing some of these questions, and gives us a powerful new way to understand The New York Times.

This is a three-part challenge:

Part 1: How to model an article based on its text.
Part 2: How to update the model based on audience reading patterns.
Part 3: How to describe readers based on their reading history.

Part 1: How to model an article based on its text.

First, our algorithm looks at the body of each article and applies Latent Dirichlet Allocation (LDA), a content modeling algorithm. LDA learns the mixture of "topics" in each article: A topic is formally defined as a distribution over a vocabulary. If a document has a certain topic weighted highly, the words seen in the article are more likely to be words weighted highly under this topic.

LDA is quick, accurate for our purposes and capable of online inference (or learning topics in real time as new articles are published). LDA topics tend to be broad (some examples are **Middle East**, **Film** and **Healthcare**), which allows us to relate pieces from differing viewpoints.

LDA is based on a graphical model, which can easily be extended to incorporate new assumptions and information. In our case, we extend our model to model not only the text of an article, but also the specific readers reading that article, described in the next section.

Part 2: How to update the model based on audience reading patterns.

LDA takes words as input, but words are often ambiguous: Context, style and voice can adjust their meaning. For example, if Gail Collins writes a piece with the words "dog," "car" and "roof," can we tell that she's being allegorical, and not just writing about animals and automobiles?

Indeed, a purely LDA-based approach gives weight to travel, placing her piece at the blue point in the diagram below.

However, a large sample of readers reading the piece have also read pieces about Hillary Clinton and Ted Cruz (visualized for illustration at the red x's), so we'd like our algorithm to adjust it to the green point in the **Politics** topic. By adding offsets to

model topic error, as described in the CTM paper, our algorithm incorporates reading patterns on top of content modeling to create a hybrid approach.

The CTM algorithm works by iteratively adjusting offsets and then recalculating reader scores. It runs until there is little change in either. A randomly chosen subset of readers, called our training sample, gives us the information we need.

We tested two methods for calculating offsets: (1) CTM and (2) an approach called Collaborative Poisson Factorization. In online A/B testing, we found CTM to perform better.

Part 3: How to describe readers based on their reading history.

The methods used for adjusting article topics also calculate reader preference, but they can't scale to all our users. Therefore, we needed a quick way to calculate reader preferences, which can occur after finalizing article topics.

A simple approach would be to average together the topics of all articles you've read: If you've clicked on one article that is 40% **Politics**, 60% **Art** and another that is 60% **Politics**, 40% **Art**, you're [. 5, .5] in the **Politics**–**Art** topic space.

However, this assumes that clicks perfectly indicate preference. What if you clicked on an article you didn't like, or missed one you would have? One way to approach this is to back off a bit, and say that you only "90% like" the articles you read, and "10% like" the ones you didn't. This leaves room for mistaken clicks or missed gems.

In this diagram, the green dots represent articles a reader has read, and the red dots represent ones they haven't. The black x might be the reader's preferences calculated using an average of articles read, and the blue x would be using the back-off approach.

The back-off approach makes a more conservative estimate of preferences, allowing us to be more robust to noisy data. It also, we've noticed, brings readers out of niches and exposes them to different, "serendipitous" recommendations.

By implementing some neat algorithmic speed-ups, we were able to calculate

preferences in less than one millisecond per reader, enabling us to scale to all registered users.

Conclusion

By modeling article content and reader preferences with topics, then adjusting based on reading patterns, we've reconceptualized our recommendation engine. Our system is now a successful, large-scale implementation of cutting-edge research in collaborative topic modeling, and it provides significant performance increases when compared with previous algorithms used to make recommendations.

Recommendation systems, we hope, can enable a dynamic New York Times to serve interesting articles at opportune times. They also help shine light on the types of articles we're writing, and who they appeal to.

The New York Times technology team is responsible for building, maintaining and improving NYTimes.com, our mobile apps and products, and the technology that powers them. We're a group of over 250 people, delivering content from one of the largest news organizations in the world. If you're interested in leveraging one of the deepest datasets in the news industry and helping us develop better products, make smarter business decisions and reach larger audiences through our personalization efforts, then we'd love to hear from you.