The News that Matters to You
Design & Deployment of a Personalized News Service

Mark Stefik
Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, Ca. 94304
stefik@parc.com

Lance Good
Google, 6425 Penn Avenue, Suite 700, Pittsburgh, PA 15206
lance@google.org

Abstract
With the growth of online information, many people are challenged in finding and reading the information most important for their interests. From 2008-2010 we built an experimental personalized news system where readers can subscribe to organized channels of information that are curated by experts. AI technology was employed to radically reduce the work load of curators, and to efficiently present information to readers. The system has gone through three implementation cycles and processed over 16 million news stories from about 12,000 RSS feeds on over 8000 topics organized by 160 curators for over 600 registered readers. This paper describes the approach, engineering and AI technology of the system.

Problem Description
It is hard to keep up on what matters. The limiting factor is not the amount of information available but our available attention (Simon 1971). In the context of news, traditional mainstream media coverage cannot address this issue. Although most people are interested in some of the topics covered in the mainstream, they also have specialized interests from their personal lives, professions, and hobbies that are not popular enough to be adequately covered there. A news service which can optimize our individual information foraging (Pirolli 2007) needs to be personalized to our interests.

A snapshot of a personal information appetite or “information diet” reveals further nuances. Some interests are enduring. Some interests are transient, following the events of life. When we form new interests, we benefit from a topical orientation, to help us to organize our understanding of a new subject area. These issues arise not only in the context of news, but also in other areas. For information analysts and other “sensemakers” they matter for professional productivity. These issues also matter for information providers that want to grow their audience by providing personalized information delivery and targeting groups with focused interests.

Readers and Curators
Our approach is powered by the expertise of curators. Curators or traditional editors set standards for information, both for the quality of sources and the organization of its presentation. In traditional publishing, the number of editors and the scope of subject matter are necessarily limited. Publishers arrange to have enough material to satisfy their audiences and enough curators to vet and organize the material.

We depart from tradition by enabling any user to be a curator, publishing and sharing articles in topically-organized channels. The idea is to reach down the long tail (Anderson 2006) of specialized interests with a growing group of curators. This approach draws on three sources of power that we call the light work of the many (the readers), the hard work of the few (the curators), and the tireless work of the machines (our system).
Supporting Readers and Curators

Our two classes of users, readers and curators, need distinct kinds of support. Readers select subject areas of interest and the system provides current information, vetted and organized. Busy readers want to fit their reading into available moments of their days. Sessions are of variable length and the article presentation should help readers to get the most important information efficiently, whether they have a minute, five minutes, or longer. Our users can access the system using web browsers at their computers or mobile devices. Readers should be able to allocate attention dynamically, getting details or more articles on a topic when it captures their interest. The system should foster information discovery, so that readers can move sideways to related topics and discover new interests.

When articles come from many sources, the main work of curators is finding and organizing them. Automating this work is the main opportunity for supporting curators. Automation requires capturing the relevant expertise of the curators, who are often busy people. A challenge is for the system to acquire their expertise efficiently, enabling them to train the system rather than providing explicit, detailed rules. To support flexibility for subject areas with different needs, the system should enable curators to organize their information in their own folksonomies.

Application Description

Our application is implemented as a web service (www.kiflets.com). Most of our programming is web and distributed systems (“cloud”) technology. About one-fourth is artificial intelligence or information retrieval technology (Jones and Willett, 1997).

System Architecture

Figure 2 shows the system architecture. Users access the system through web browsers. Interactivity is provided by browser programs written in HTML/JavaScript and Adobe Flash. The API to the web server uses REST protocols with arguments encoded in JSON. The web server is outside our firewall and uses Django as the web framework. Code for transactions with the rest of the system are written in Python.

Web and Distributed Systems Technology

Transactions through the firewall are directed to a MySQL database, a Solr (variant of Lucene) server that indexes articles, and a topic search server that we wrote in Java. These servers are for transactions that perform fast, low-latency computations. The computations include user-initiated searches for articles or topics and services for curators who are tuning their topic models and finding new sources. Another server caches query results to reduce load on the database for common queries. All of these servers run on fairly recent mid-class Dell workstations.

Most of the information processing is carried out by Java programs that run on a Hadoop cluster of a dozen workstation-class computers. The main work of the cluster is in crawling curator-specified RSS feeds on the web, collecting and parsing the articles, classifying articles by topic, and clustering related articles from multiple sources. Most of the article information (about 3 terabytes) is stored in HBase, a NoSQL (key-value pair) database that runs.
AI Technology for Robust Topic Identification

In manual curation the most time-consuming part is finding and identifying articles for topics of interest. Our system classifies 20 to 30 thousand articles by topic every day. Many online news systems classify articles automatically by matching a user-supplied Boolean query against articles. However, several common conditions can cause this approach to be unsatisfactory. One issue is that common words often have multiple meanings. Does a search for “mustang” refer to a horse, a car, or something else? User expectations of precision are much higher for automatic article classification than for results of search engines. When someone uses a search engine, they face a trade-off between carefully developing a precise query and spending time foraging through the results. In a search setting, it can be acceptable if 50 percent or more of the results are off topic as long as a satisfactory article appears in the top few results. However, readers perceive such imprecision as unacceptable when the system applies its own query and there are many off-topic articles.

Skilled searchers and query writers can address this issue to a degree by writing more complex queries. We have found, however, that complex queries are prone to errors and refining them is often beyond the skill and patience of our curators.

One way that we have addressed query complexity is by developing a machine learning approach to create optimal queries. In this approach a curator marks articles as on-topic (positive training examples) or off-topic (negative training examples). The system searches for the simplest query that matches the positive examples and does not match the negative ones.

Because we have reported on this approach earlier, we describe it here only briefly. Our system employs a hierarchical generate-and-test method (Stefik 1995) to generate and evaluate queries. The queries are generated in a Lisp-like query language and compiled into Java objects that call each other to carry out a match. The articles themselves are encoded as arrays of stemmed words represented as unique integers. With query-matching operations implemented as operations on numeric arrays, the system is able to evaluate tens of thousands of candidate queries in a few seconds. This is fast enough for interaction with a curator.

The query terms are chosen from the training examples, focusing on words that have high TFIDF ratios, that is, words whose frequencies in the training examples are substantially higher than their frequencies in a baseline corpus. The generated query relationships are conjunctions, disjunctions, ngrams, and recursive compositions of these. Candidate queries are scored according to matching of the positive and negative training examples and structural simplicity.

Although the optimal query generator automates writing queries, this approach alone does not get around fundamental problems with using queries alone to classify articles. For example, it does not distinguish cases where articles match a query incidentally, such as when article web pages contain advertisements or short descriptions provided by a publisher to draw a reader to unrelated articles. From the perspective of article classification, this information on a web page is noise. The query approach also does not distinguish articles that are mainly on-topic from articles that are mainly off-topic, but which contain tangential references to a topic. For this reason, we characterize the query approach as having high precision and high vulnerability to noise.

To reduce noise vulnerability, we incorporate a second approach to classifying articles. The second approach complements query matching and has opposite characteristics. In contrast to the query approach, it has low vulnerability to noise but also low precision.

The second approach considers an article as a whole, rather than focusing on just the words and phrases in a query. It represents an article as a term vector, pairing basis words with their relative frequencies in the article. We compute the similarity of the term vector for an article to a term vector for the topic as derived from its training examples. This is a variant of standard similarity approaches from information retrieval. With a cosine similarity metric, the score approaches one for a highly similar article and zero for a dissimilar article. A similarity score of about 0.25 is a good threshold for acceptability.

In summary, our system combines two topic models with opposite characteristics to provide a robust classification of articles by topic. An article is classified as on-topic if it matches the query for a topic and has a high enough similarity score. This combined method has proven precise enough for topics and robust against the noise found in most articles. It requires that curators identify good examples of on-topic and off-topic articles. The curator knowledge is captured from the training examples that they select. For most topics, three to six training examples of each type are enough for satisfactory results.

AI Technology for Multi-level Topic Presentation

Readers expect articles to be classified and well organized in sections corresponding to topics. For example, in a channel covering hard core national news, there are currently over 300 topics for articles drawn from several hundred sources. The topic tree has eight top-level topics including “Crime and the Courts,” “Economy and Trade,” “Health and Safety,” “Politics,” and “War and Terrorism.” Eighty to a hundred articles are collected and classified over Hadoop’s distributed file system. Hadoop also runs other jobs that pre-compute information for the news presentations.
each day for this channel. Figure 2 gives examples of three articles promoted from subtopics of “Health and Safety”. The first article comes from the leaf topic “Snow”. Its full topic trail is “USA > Health and Safety > natural disasters > Storms > Snow”.

Health and Safety

Major winter storm expected to hit Great Plains, eastern states
[feeds.reuters.com] 09:28AM Jan 3, 2011 (CT 52)
USA > Storms > Snow
off topic different topic

CHICAGO (Reuters) - A massive storm system bringing heavy snow, sleet, and freezing rain could potentially impact 100 million people as it slams the Rockies, Plains, and Midwest regions early this week before traveling to the eastern seaboard Wednesday.... All 2 stories like this

Clinton: US has no plans to suspend aid to Haiti (AP)
USA > natural disasters > Earthquakes
off topic different topic

AP - The United States has no plans to halt aid to earthquake-ravaged Haiti in spite of a crisis over who will be the nation's next leader but does insist that the president's chosen successor be dropped from the race, U.S. Secretary of State Hillary...

Alpha in $3.5bn deal for Massey Energy
[bbc.co.uk] 07:00AM Jan 30, 2011 (CT 41)
USA > occupational safety > mining disasters
off topic different topic

Alpha Natural Resources buys Massey Energy in $8.5bn deal that makes further consolidation of the industry

Older articles...

Figure 2. Displaying articles promoted from subtopics.

In an early version of the system, all of the new articles for a channel were presented at once. This was overwhelming for readers even though articles were accurately classified by topic. In a second version, users had to click through the levels of the tree to find articles, which was too much work and caused readers to miss important stories. In the current version, articles are presented at a level at a time and a rationed number of articles from the leaf topics are selectively bubbled up the topic tree through their parents. The question is which articles should be selected to propagate upwards through each intermediate topic? The considerations and AI techniques for this approach are the subject of another paper. For this reason, we describe the approach very briefly here.

Our system combines several factors in promoting articles through levels. Articles are favored if they are central to a topic, that is, if their term vector is similar to the composite term vector for a topic or close to one of its individual training examples. Articles from a topic are favored if the topic is hot, meaning that the number of articles on the topic is dramatically increasing with time. Story coverage in a parent topic is allocated to have some balance across competing subtopics.

These computations are carried out in parallel across topics using Hadoop as a job scheduler on the back-end of the system with the results saved in the MySQL database. This enables the front end of the system to present the best articles for the particular interests of each user without executing expensive queries in real time for each topic.

AI Technology for Detecting Duplicate Articles

Busy news readers expect a news system to help them to satisfy their interests efficiently. They can be annoyed if duplicate articles appear under a topic. Exact duplicates of articles can be collected by the system when curators include multiple feeds that carry the same articles under different URLs. Reader perception of duplication, however, is more general than exact duplication and includes articles that are just very similar. Similar articles might come together in the same topic or via the article promotion process to high level topics. The challenge is finding an effective and efficient way to detect duplicates.

Our approach begins with simple heuristics for detecting identical wording. The main method uses clustering. Since the number of clusters of similar articles is not known in advance we developed a variant of agglomerative clustering. By employing a greedy algorithm with a fixed minimum threshold for similarity, we found that two passes through the candidate clusters for a topic is almost always enough to cluster the duplicate articles. The found clusters are dated and recorded in the database. An example of a clustering result is shown below the first article in Figure 2 in the link to “All 2 stories like this.”

Other AI Technology for Information Processing

Most of the programming in the system is for system tasks such as job scheduling, data storage and retrieval, and user interactions. Nonetheless, AI techniques have been essential for those parts of the system that need to embody knowledge or heuristics. Here are some examples:

- A hot-topics detector prioritizes topics according to growth rates in editorial coverage across sources, identifying important breaking news.
- A related-topic detector helps users discover additional channels for their interests.
- A near-misses identifier finds articles that are similar to other articles that match a topic, but which fail to match the topic’s query. The near-miss articles can be inspected by curators and added as positive examples to broaden a topic.
A source recommender looks for additional RSS feeds that a curator has not chosen, but which deliver articles that are on topic for a channel.

Interweaving Development and Evaluation

This project was inspired by “scent index” research (Chi, Hong, Heiser, Card, and Bumbrecht 2007) for searching the contents of books. That research returned links to page numbers as search results organized by categories from a back-of-the-book index. For example, a search query like “Ben Bederson” in an HCI book returned results organized by topics corresponding to Bederson’s research projects and institutional affiliations. We thought it would be exciting to extrapolate from a given index to organize web search results.

The key technological uncertainty was whether a machine learning approach could accurately model index topics. A one-person internal project was started that built and developed the first version of an optimal query generator. After a few months we showed that it could quickly generate queries that accurately matched all 900 index entries in a book, essentially reproducing results of the original index (but finding errors in it).

Alpha and Beta Testing

In April 2008 we created a two-person team to explore the application of this technology. The initial business objective was to create a prototype product suitable for an advertising-based business delivering personalized news. Later the objective evolved to provide information processing services for news organizations.

In October 2008 we opened our first prototype to alpha-testing by a dozen users. We had a flash-based interface for curators and a simple web interface for readers. Each of the curators built a sample index and used it for a few weeks. Three more people joined the team, focusing on release testing, user interviews, design issues, and fund raising.

Although the system was able to collect and deliver articles when we built the channels, it became clear that curation was too difficult for our first curators. They had difficulty finding RSS feeds and did not completely grasp the requirements of curation. Extensive interviews and observation session helped us to identify key usability issues. We came to understand that the system would not be a commercial success unless it went viral, and making it much easier to use was needed.

We began learning about lean start-up practices and became obsessed with meeting customer needs. We followed a ruthless development process that divided user engagement into four stages: trying the system, understanding it, being delighted by it, and inviting friends. We divided possible system improvements into a track for curators and a track for readers. We built performance metrics into the system and monitored user engagement with Google Analytics. In 2010 we measured 1300 unique visitors per month with about 8900 page views. The average user stayed for about eight minutes, which was high. Every month we interviewed some users. Every morning we met for an hour to prioritize and coordinate the day’s development activities.

The development and deployment of AI technology was driven by the goal of meeting user needs. For example, when article classification began failing excessively due to noisy articles from the web, we combined our symbolic query-based approach with the statistical similarity-based approach. For another example, multi-level topic presentation was developed to improve user experience on big channels. Other additions such as the source recommender were prioritized when they became the biggest obstacles to user satisfaction.

Over time we came to understand user and curator habits more deeply. For example, when we recognized that curators wanted to tune their topic models while they were reading their daily news, we eliminated the separate “wizard” for curators and incorporated curation controls into the news reading interface. This required changing how the machine learning algorithms were triggered. They went from being requested explicitly in a curation session to being requested implicitly when articles were added to topics (positive examples) or when articles were marked as off-topic during reading. We did not always gain our biggest insights through user interviews and metrics. Some of our insights came from being heavy users ourselves.

Performance Tuning

In early 2009 we began beta-testing with about 60 users. The system load from users and articles increased to a level where we had to prioritize scaling and robustness issues. The first version of the system began to stagger when we reached 100 thousand articles. A recurring theme was to reduce the I/O in processes, since that dominated running time in most computations. For example, an early version of the system would read in arrays representing articles whenever queries changed and then use our optimized matching code to detect matches. Recognizing that most of the time was going into I/O, we computed indexes for articles when they were first collected using Solr. The system could then fetch identifiers for matching articles without re-reading their contents.

We started a NoSQL database for article contents to support the millions of articles that the system now held. We periodically re-worked slow queries and found more ways to pre-compute results on the back-end in order to reduce database delays for users. In June of 2010, we started an open beta process by which any user could come
to the system and try it without being previously invited. By August, the system had over 600 users and was able to run for several months without crashing.

Conclusions

This project was inspired by another project that returned results from searching the contents of a book. To generalize that concept to the ever-expanding web, we needed to develop an effective method for extrapolating from an explicit index over a fixed corpus to a model-based evergreen index. Figure 4 gives an example of our system doing this when this paper was written.

Figure 4. Overview page of reading interface.

The figure shows a set of curated channels selected by the user. Users can subscribe to channels that cover main stream subject areas like the channels “USA” or “World News” in this example. They can also subscribe to channels in more specialized areas, such as “Sustainable Living” or “Future of Journalism”. Given enough time, they can scroll down the page to see top articles from each of the channels. For more, they can drill down selectively to get more topics and more articles.

We did not start out with a goal of using artificial intelligence technology. Rather we used the technology of choice at each stage of trying to satisfy user requirements.

Our system follows the “knowledge is power” logic of earlier AI systems on knowledge systems in that it depends on the expertise of its curators. We acquired curator expertise using a machine learning approach where curators can select sources that they trust (sometimes given some source recommendations by the system), organize topics in a topic tree folksonomy according to how they make sense of the subject matter, and train the topic models with example articles. Using this information the system automatically creates evergreen channels of information every day.

A major challenge was in making curation easy and reliable given the limited time that curators have available. It is easy for us to train curators in a couple of short sessions. It is more challenging to attract people on the web to try the system, to understand what it does, and to invest in becoming a curator.

At this time about one user in three creates a channel and about one in four of those creates a complex channel. We do not know ultimately what fraction of users might be expected to become curators. Many users create very simple channels without bothering to set up topics. We believe that there are very interesting pivots to make on new mobile devices and in engagements with online communities. There are also other applications of the classification technology beyond personalized news.

Further development on this project depends on finding external funders or investors. The news business is increasingly undergoing rapid change and economic challenges. It is changing on several fronts, including how news is delivered (mobile devices), how it is being reported (citizen journalists & content farms), how it is paid for (subscription services, pay walls, and advertising). This project opens a further dimension of change: how news can be curated.

Kiffets was designed, implemented, and deployed by two people over two and a half years. Other project members worked on evaluation, channel development, user experience, release testing, and business development.

Acknowledgments

This research was sponsored by the Palo Alto Research Center. We thank Barbara Stefk, Ryan Viglizzo, Sanjay Mittal and Priti Mittal for their contributions to design, release testing, user studies and great channels. Thank you to Lawrence Lee and Markus Fromherz for their guidance and support. Thank you to Jim Pitkow for advice on lean management. Thanks to our Kiffets users for much feedback.

References


