Personalised News and Blog Recommendations based on User Location, Facebook and Twitter User Profiling

Gabriella Kazai
Lumi News
London
gabs@lumi.news

Iskander Yusof
Lumi News
London
iskander@lumi.news

Daoud Clarke
Lumi News
London
daoud@lumi.news

ABSTRACT

This demo presents a prototype mobile app that provides out-of-the-box personalised content recommendations to its users by leveraging and combining the user's location, their Facebook and/or Twitter feed and their in-app actions to automatically infer their interests. We build individual models for each user and each location. At retrieval time we construct the user's personalised feed by mixing different sources of content-based recommendations with content directly from their Facebook/Twitter feeds, locally trending articles and content propagated through their in-app social network. Both explicit and implicit feedback signals from the users' interactions with their recommendations are used to update their interests models and to learn their preferences over the different content sources.

CCS Concepts

•Information systems → Mobile information processing systems; Recommender systems;

Keywords

Lumi News, recommender system, mobile app

1. INTRODUCTION

Online content is being produced at an unprecedented rate, with thousands of news stories, blogs and videos added every day by a wide range of publishers, media outlets and Internet users. At the same time, in this accelerating supply-demand cycle, consumer behaviour is changing at an equally high pace [1, 7, 2]. Users increasingly consume news and media content on the go, using their smartphones, either via dedicated news apps or through social media. According to a survey by Mobiles Republic[1], a global news syndication company, 75% of readers with smartphones and 70% with tablets check the news more than once a day.

[1]`http://www.news-republic.com/infographic2013/`

SIGIR '16, July 17-21, 2016, Pisa, Italy
© 2016 ACM. ISBN 978-1-4503-4069-4/16/07...$15.00
DOI: http://dx.doi.org/10.1145/2911451.2911464

The increasing use of mobiles for rich media consumption is reflected in the growing number of news apps available and their growing user base. For example, news aggregator apps, such as Flipboard, BuzzFeed, Yahoo News, Feedly, News360, Pulse and Apple News present news from a range of publishers, but either provide no personalisation, rely on the user's social networks or on the user selecting topics of interests. Research prototypes that experiment with more advanced recommendation techniques include Focal [4], PEN [3] and others, e.g. [9, 8, 6, 5]. This plethora of news portals and news apps is a clear sign of users' need for news consumption, but also a sign that the question of how to serve this content to users is still very much an open problem.

The goal of mobile news recommender systems is to help users find fresh content that is relevant to them or to their particular context (e.g. location, social) from a constant stream of publications and serve this content in ways that require minimal user interaction, given the limits of a device's form-factors. When considering the user's experience during a reading session, the feed of news stories recommended to the user should also cater for diversity and serendipity so that the user can discover new topics or stay informed on trending, but perhaps less personally relevant, events. This requires a different approach to those in traditional information retrieval, one that goes beyond optimising for relevance.

Another key challenge in recommender systems is the cold-start issue: how to recommend items to users for whom we have very little data. Typical approaches require the user to select from a list of predefined topics (e.g., Flipboard), making use of the user's location information or social network (e.g., Pulse) or learning the user's interests over time through interactions with the recommended content (e.g., [5]).

Our system, Lumi Social News[2], aims to provide users an out-of-the box personalised experience by leveraging both external and internal data and automatically building the user's profile from their location, their Facebook or Twitter feed, while also learning from the user's in-app interactions. The incoming streams of content are matched against the user's evolving profile to generate recommendations, where the scoring considers the item's relevance, popularity and freshness. The user's feed is then composed of items relevant to the user's interests, intermixed with locally trending stories, as well as content from other sources, such as the user's Twitter feed or stories from the user's social network on Lumi. Our approach thus aims to combine temporal, locational, social and preferential information to provide a

[2]`android.lumi.do`

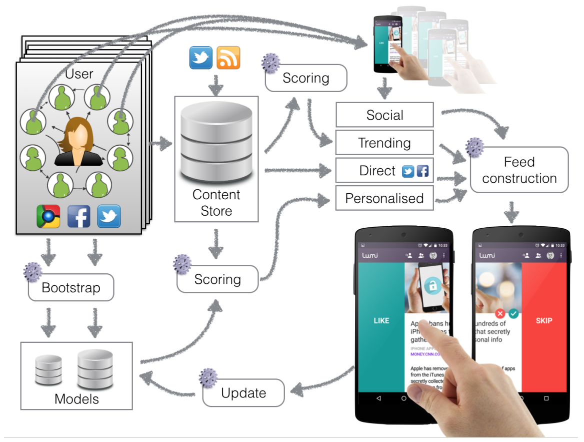

Figure 1: Lumi architecture data flow

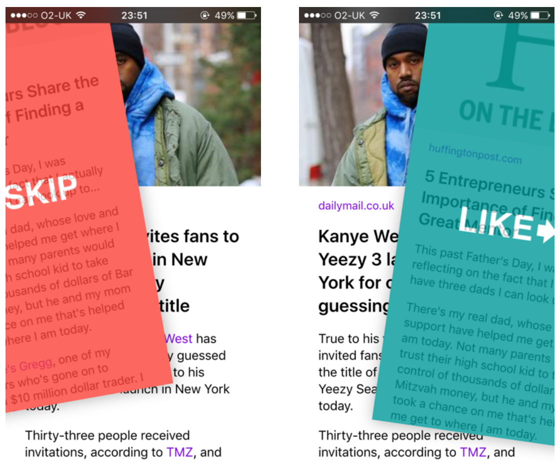
SKIP
LIKE
dailymail.co.uk
huffingtonpost.com
5 Entrepreneurs
Thirty-three people received invitations, according to TMZ, and

Figure 2: Screenshots of user swipe actions to skip or like an article

more fine-grained recommendation strategy.

Lumi is also different from other news recommender systems in that it is not limited to publishers, but presents users with a broad range of blogs and online publications, allowing much greater diversity and specialisation in the content delivered to users. This is achieved by leveraging user data to identify promising sources of content.

This paper presents the implementation of our solution to recommend news and more long tail media from a large crowd-curated content stream in a way that provides the user a diverse reading experience, combining topical, location-based and socially relevant content as well as breaking or trending news. Recommendations are presented to the user through an intuitive and easy-to-use, swiping interface, one story at a time. Users' interactions with their recommendations are then used to update their models and to tune their preferences over the various content sources.

2. SYSTEM FEATURES

Figure 1 shows a schematic overview of the data flow in Lumi. Content is ingested either via RSS or is contributed and curated by the users themselves. For example, users can sign up with Twitter and, in the process, bring the public content of those they follow on Twitter with them into Lumi. This content, after various quality filtering, becomes part of the content pool, from which recommendations can then be generated.

During the bootstrap process, Lumi automatically learns the user's interests from their location, from their public Twitter and/or Facebook feeds and from their in-app actions. The generated user models are then matched against the stream of incoming content and recommended to users based on a combination of their relevance and trending scores.

In addition to the model based recommendations, Lumi also serves content from a range of other sources to the user. In fact, a user's Lumi feed is made up of a mixture of content from any of the following sources, depending on availability for a given user: model-based recommendations, content pulled directly from the user's social media streams, e.g., Twitter/Facebook, as well as trending content with local or global relevance and content from the user's Lumi connections. The preferred composition of the feed across these sources is tuned based on the user's in-app interactions. For example, if a user spends longer on reading their model based recommendations than on content coming directly from their Twitter feed, Lumi updates the source weights and will include more model based recommendations in the user's feed.

Lumi also has a unique user interface in comparison to other news apps. Instead of showing the user a list of recommendations that is typical in recommender systems, Lumi displays only a single recommended item at any given time. This item is picked from the various ingested content streams, where the selection is based on the item's relevance to the user or the user's location as well as its recent popularity on social media, i.e., Twitter and Facebook. Figure 2 shows an example recommended article.

All articles are shown in the same style, providing a cohesive and smooth experience. The header image for an article is picked based on image quality and positioning in the original text, which is followed by the title and the full text of the article. In the case of a video, the video itself is positioned at the top, followed by the title and any textual description if available on the original site. There is a reactions bar at the bottom of each article, which lists users who liked the recommended item, ordered by social closeness or similarity to the user.

To get to the next recommendation, the user needs to either like or skip the current item by swiping left or right, respectively (see Figure 2). The metaphor used is that of a stack of cards, where the user has to flick the top card out of the way to see the next card. This provides a playful element to the reading experience.

All liked stories are saved by the system and can be accessed through the user's profile area and can also be read offline. The explicit feedback of the like/skip actions as well as implicit signals, such as the time the user spent on an article, are used to continually update the user's model, thus learning more about the user's evolving interests. The feedback is also applied to tune the user's preferences over the various sources that compose the user's feed.

3. SYSTEM OVERVIEW

The system is built using a microservices architecture, with separate services, for example, to perform ingestion of new content via Twitter, Facebook or RSS, content quality assessment (to determine whether to ingest a page), or generation of recommendations.

The front-end is implemented for Android phones in Java, which connects to the back-end via a restful API. User models and recommendations are generated using scikit-learn libraries and in-house developed services and methods.

Most components are deployed using Docker together with Amazon EC2 Container Service.

3.1 Ingestion

Ingestion components process a number of incoming streams of content from RSS, Twitter and Facebook. The content is rendered (HTML content of URLs downloaded), passed through a series of quality filters and subsequently a range of features and media are extracted. Exact duplicates, e.g., when the same article is tweeted by multiple people are removed at this stage. A fuzzy, content based filter is used later on in the pipeline, for example, to identify when the same story is published by multiple outlets, albeit with different details and in different editorial styles. Logistic Regression classifiers are used to assign topic labels, e.g., business, technology, football, etc., to the ingested articles. Content is then indexed and stored in Elasticsearch[3]. For each ingested page, we maintain our own social media share and like counts, tracking their popularity on the Web.

At the time of writing, with just over 100k installs, the system processes around 200 million tweets a day, resulting in over 100k articles curated by the Lumi users (after quality filtering and deduping), which then make it into the recommendable content pool.

3.2 Bootstrap

At signup, the public pages in a new user's Twitter or Facebook feeds (i.e., public pages published by those the user is following) are analysed and an initial user model is built. As some feeds may be noisy or not necessarily representative of the user's interests, we impose quality thresholds before a model can be built. While the model building is taking place in the background, the user is shown local and global trending stories. Stories explicitly liked by the user (swipe right) and stories that the user spent at least thirty seconds reading are also used as positive samples when training the user's model. A key stage of the model building is the feature extraction. We extract a wide range of features, including standard word features as well as categorical data and combine these within a single model in order to support users' topical interests and source preferences. We employ feature hashing to keep model sizes small. Once a model is built, the cache of stories already on the device is flushed and the new feed, containing personalised recommendations is sent to the device.

3.3 User model update

Users' respective models are updated based on their ongoing online activities, e.g. Twitter or Facebook, as well as based on their in-app actions, e.g. when they read, swipe to like or swipe to skip a story. For scalability, we follow an incremental model updating approach. Model updating happens as a background process, which loops through users in a staggered way, prioritising more active users first. This ensures that users ,who use Lumi multiple times a day always have up to date models and fresh recommendations.

[3]`https://www.elastic.co/products/elasticsearch`

3.4 Trending content

Trending content is identified by monitoring social media, e.g. likes on Facebook and tweets on Twitter. Locally trending stories in a given country or region are identified through content that is popular with users who share the same location. A particular challenge with trending content detection is that most social media share counts are heavily biased to US and UK interests, so we built our own location specific trending score method, which biases towards the sites and topics that are relevant to the user's location.

3.5 Model based recommendations

For each ingested page, each user's model is used to generate relevance scores, which are then stored in Amazon DynamoDB against the user. This is run as a background process, which generates recommendations for users even when they are not online, so that they don't miss out on interesting content. Relevance scores are calculated on batches of incoming pages, against all the user models, starting with more active users, similarly to how model updating is performed.

3.6 Social content

Lumi users can follow each other and discover interesting content that was liked by those they follow. Recommendations on who to follow are based on the relevance of the suggested user's liked items to the current user's interests as well as based on existing social links in the user's Twitter and Facebook networks.

When users follow other Lumi users, they also get to see stories that were liked by those they follow. The selection of these stories combines factors such as the user's closeness to the followed users, e.g., reciprocal relationship, similarity of interests, as well as the popularity of the liked story and its freshness. Similarly, stories liked by the user are shared to the user's followers in their Lumi social network. The reactions of their followers on these shared stories are then fed back to the user.

3.7 Direct Twitter/Facebook content

Similarly to social content, articles from the user's Twitter or Facebook feed are selected based on timeliness, social factors and popularity and are then mixed into the user's feed.

3.8 Feed construction

In order to provide diverse content, a user's feed is made up of content from the model-based recommendations, from local or global trending news, content that is directly streamed from their Twitter and/or Facebook feeds and social content that has been shared on Lumi by users that the current user is following. When a mobile client requests new recommendations, the top ranking items are returned from all the different sources that are available for the given user. The ranking function takes into account both the relevance of an item to the user's interests and its local or global popularity.

The preferences over the different sources are learnt and continually tuned based on the user's interactions. For this we use a Bayesian update process, normalised across the user's actions and across other users. Diversity across the recommended content is ensured by optimising the order of stories from different sources in the feed as well as using the calculated similarity between articles. This is where the fuzzy deduping takes place, making sure that the user is recommended an article on a given story of interest from their preferred outlet, while ensuring that they are not recommended too similar articles related to the same story from other outlets.

4. EVALUATING RECOMMENDATIONS

Our ultimate goal is to make an app that people want to use every day: we want to optimise for user retention. However it takes a long time to measure the impact on retention, and we have found a metric that correlates closely with it, namely, the number of recommendations that a user spends over thirty seconds reading (from now on we refer to this just as "reading"). This is a natural metric to use for a news app, and it allows us to iterate much faster when trying out different recommendation algorithms.

To account for natural variations in the user-base, and different users' propensity for reading, we use split testing on individual recommendations, so that each user gets recommendations from a variety of sources or algorithms. This allows us to see how many users prefer one source or algorithm to another, giving a reliable way to identify when recommendations are working or not. If we were to split the recommendations across users so that some users got one type of recommendation and some users got another, it would be harder to trust our results because some users naturally read more than others.

We use a Bayesian formula due to Evan Miller[4] to compare two types of recommendations. Specifically, we assume that there is some probability p_A that a user will read a page of type A. For a new type of recommendation, B, we then estimate $P(p_B > p_A)$, the probability that the user reads a page of type B is greater than the probability that they read a page of type A. Given that the user has read r_A pages of type A and been presented with s_A pages of the same type that they have not read, then this probability can be estimated by

$$\sum_{i=1}^{r_B+1} \frac{B(r_A+i, s_A+s_B+2)}{(s_B+i)B(i, s_B+1)B(r_A+1, s_A+1)}$$

where B is the beta function. We evaluate this probability for each user, and look at the proportion of users where this probability exceeds a threshold of 0.9. If the proportion is high, we can be fairly certain that recommendations of type B are preferred by users.

5. CONCLUSIONS

Personalised recommendation of news and media content in general paves the way for solving the information overload and attention scarcity problem, especially in the face of the recent increase in content publishing, including citizen journalism and other user generated content. This demo paper presents the development of a personalised mobile news recommender system.

The presented system is the first in its class to provide personalised feeds by combining a number of different sources and model-based recommendations over a crowd curated content pool. The system non-intrusively learns users' interests from their Twitter or Facebook activities and responds to their feedback actions when reading recommended items.

It is a complex system with many moving parts that build on a combination of machine learning methods as well as information extraction and retrieval technologies to provide users with an experience of diverse, fresh, relevant content feed in a simple swiping user interface. The assessment of how recommender systems may perform in this setting with noisy and sparse data and online user feedbacks is a key challenge to deliver this service.

For future work, we will investigate combining topic based content streams into a user's feed to increase coverage across user's interest silos. In addition, we will experiment with various visualisations of the user's profile (their liked stories), for example, grouping liked stories by topics.

[4]`http://www.evanmiller.org/bayesian-ab-testing.html`

6. ACKNOWLEDGMENTS

Lumi is a result of the efforts of a great team, see `https://lumi.do/about/team`.

7. REFERENCES

[1] M. Constantinides, J. Dowell, D. Johnson, and S. Malacria. Exploring mobile news reading interactions for news app personalisation. MobileHCI '15, pages 457–462, New York, NY, USA, 2015. ACM.

[2] C. Esiyok, B. Kille, B.-J. Jain, F. Hopfgartner, and S. Albayrak. Users' reading habits in online news portals. In *Proceedings of the 5th Information Interaction in Context Symposium*, IIiX '14, pages 263–266, New York, NY, USA, 2014. ACM.

[3] F. Garcin and B. Faltings. Pen recsys: A personalized news recommender systems framework. NRS '13, pages 3–9. ACM, 2013.

[4] F. Garcin, F. Galle, and B. Faltings. Focal: A personalized mobile news reader. RecSys'14, pages 369–370. ACM, 2014.

[5] J. A. Gulla, A. D. Fidjestøl, X. Su, and H. Castejon. Implicit user profiling in news recommender systems. In *WEBIST (1)*, pages 185–192, 2014.

[6] I. Ilievski and S. Roy. Personalized news recommendation based on implicit feedback. In *Proceedings of the 2013 International News Recommender Systems Workshop and Challenge*, NRS '13, pages 10–15, New York, NY, USA, 2013. ACM.

[7] D. Lagun and M. Lalmas. Understanding user attention and engagement in online news reading. WSDM '16, pages 113–122, New York, NY, USA, 2016. ACM.

[8] A. Said, J. Lin, A. Bellogín, and A. de Vries. A month in the life of a production news recommender system. In *Proc. Workshop on Living Labs for IR Evaluation*, pages 7–10. ACM, 2013.

[9] M. Tavakolifard, J. A. Gulla, K. C. Almeroth, J. E. Ingvaldesn, G. Nygreen, and E. Berg. Tailored news in the palm of your hand: A multi-perspective transparent approach to news recommendation. WWW'13 Companion, pages 305–308, 2013.