International Journal of Engineering, Science and Mathematics
Vol. 6 Issue 7, November 2017,
ISSN: 2320-0294 Impact Factor: 6.238
Journal Homepage: http://www.ijesm.co.in, Email: ijesmj@gmail.com Double-Blind Peer Reviewed Refereed Open Access International Journal - Included in the International Serial Directories Indexed & Listed at: Ulrich's Periodicals Directory ©, U.S.A., Open J-Gage as well as in Cabell's Directories of Publishing Opportunities, U.S.A

EVALUATION OF T.REX EXTRACTS SEVERAL SIGNALS FROM THE REAL-TIME WEB TO PREDICT USER INTEREST

L K Sravanthi Potti[1], Dr. Satheesh Kumar Nagineni[2]
Department of Computer Science and Engineering
[1,2]OPJS University, Churu, Rajasthan

Abstract: The real-time Web can provide timely information on important events, and mining it online with stream processing allows yielding maximum benefit from it. In this chapter we proposed an application to online news recommendation. Our goal is to help the user keep up with the torrent of news by providing personalized and timely suggestions. We described T.REX, an online recommender that combines stream and graph mining to harness the information available on Twitter. T.REX creates personalized entity-based user models from the information in the tweet streams coming from the user and his social circle, and further tracks entity popularity in Twitter and news streams. T.REX constantly updates the models in order to continuously provide fresh suggestions.We framed our problem as a click-prediction task and we tested our system on real-world data from Twitter and Yahoo! news. We combined the signals from the two streams by using a learning-to-rank approach with training data extracted from Yahoo! toolbar logs. T.REX is able to predict with good accuracy the news articles clicked by the users and rank them higher on average. Our results show that the real-time Web is a useful resource to build powerful predictors of user interest.

1. Introduction

Real-time Web is an umbrella term that encompasses several social micro-blogging services. It can be modeled as a graph of nodes exchanging streams of updates via publish-subscribe channels. As such it crosses the boundaries between graphs and streams as defined in the taxonomy of Famous examples include Facebook's news feed and Twitter.

In this chapter we tackle the problem of mining the real-time Web to suggest articles from the news stream. We propose T.REX, a new methodology for generating personalized recommendations of news articles by leveraging the information in users' Twitter persona. We use a mix of signals to model relevance of news articles for users: the content of the tweet stream of the users, the profile of their social circles, and recent topic popularity in news and Twitter streams.

We validate our approach on a real-world dataset from Yahoo! news and one month of tweets that we use to build user profiles. We model the task as click prediction and learn a personalized ranking function from click-through data. Our results show that a mix of various signals from the real-time Web is an effective indicator of user interest in news.

2. Problem definition and model

Our goal is to harness the information present in tweets posted by users and by their social circles in order to make relevant and timely recommendation of news articles. We proceed by introducing our notation and define formally the problem that we consider in this work. For quickreference, our notation is also summarized in Table

Table 1: Table of symbols.

Definition(News stream)**.**LetN={n_0; n_1,n2, , , }be an unbounded streamof news arriving from a set of news sources, where news article n_i is published at time (n_i).

Symbol	Definition
$\mathcal{N} = \{n_0, n_1, \ldots\}$	Stream of news
n_i	i-th news article
$\mathcal{T} = \{t_0, t_1, \ldots\}$	Stream of tweets
t_i	i-th tweet
$\mathcal{U} = \{u_0, u_1, \ldots\}$	Set of users
$\mathcal{Z} = \{z_0, z_1, \ldots\}$	Set of entities
$\tau(n_i)$	Timestamp of the i-th news article n_i
$\tau(t_i)$	Timestamp of the i-th tweet t_i
$c(n_i)$	Timestamp of the click on n_i
S	Social network matrix
S^*	Social-influence network matrix
A	User $\times$ tweet authorship matrix
T	Tweet $\times$ entity relatedness matrix
N	Entity $\times$ news relatedness matrix
$M = T \cdot N$	Tweet $\times$ news relatedness matrix
$\Gamma = A \cdot M$	User $\times$ news content relatedness
$\Sigma = S^* \cdot A \cdot M$	User $\times$ news social relatedness
Z	Entity popularity
$\Pi = Z \cdot N$	News popularity
$R_\tau(u, n)$	Ranking score of news n for user u at time τ

Definition (Tweet stream)**.** LetT={t_0; t_1, , , }be an unbounded streamof tweets arriving from the set of Twitter users, where tweet t_i is published at time (t_i).

Problem (News recommendation problem)**.** Given a stream of newsN, aset of users U = {u_0; u_1, , , } and their stream of tweets T , find the top-k most relevant news for user u 2 U at time .

We aim at exploiting the tweet and news streams to identify newsof general interest, but also at exploiting a Twitter-based user profile to provide personalized recommendations. That is, for any user u 2 U at any given time , the problem requires to rank the stream of past news in N according to a user-dependent relevance criteria. We also aim at incorporating time recency into our model, so that our recommendations favor the most recently published news articles.

We now proceed to model the factors that affect the relevance of news for a given user. We first model the social-network aspect. In our case, the social component is induced by the Twitter following relationship. We define S to be the social network adjacency matrix, were S(i; j) is equal to 1 divided by the number of users followed by user u_i if u_i follows u_j, and 0 otherwise.

We also adopt a functional ranking (Baeza-Yates et al.,2006) that spreads the interests of a user among its neighbors recursively. By limiting the maximum hop distance d, we define the social influence in a network as follows.

3. Learning algorithm

The next step is to estimate the parameters of the relevance model that we developed in the previous section. The parameters consist of the coefficients, , used to adjust the relative weight of the components of the ranking function R . Another parameter is the damping factor used in the computation of social influence, but we empirically observed that it does not influence the results very much, so we used a default value of σ= 0:85 common in PageRank. We consider only direct neighbors by setting the maximum hop distance d = 1. This choice is for practical reasons, since crawling Twitter is rate limited and thus very slow.

Our approach is to learn the parameters of the model by using train-ing data obtained from action logs. In particular, we use click-log data from Yahoo! toolbar as a source of user feedback, The Yahoo! toolbar anonymously collects clicks of several millions of users on the Web, in-cluding clicks on news. Our working hypothesis is that a high-quality news recommender ranks high the news that will be clicked by users. We actually formulate our recommendation task according to the learning-to-rank framework (Joachims, 2002).

Consider a user u 2 U and a news article n 2 N with publication timestamp (n) , where is the current time. We say that the news article n should be ranked in the i-th position of the recommendation list for user u, if it will be the i-th article to be clicked in the future by user u.

This formulation allows to define precedence constraints on the rank-ing function R . Let c(n) be the time at which the news n is clicked, and let c(n) = +∞ if n is never clicked. At time , for two news n_i, n_j with

$t(n_i)$and $t(n_j) \leq t$, we obtain the following constraint:

Ift≤$c(n_i) < c(n_j)$ then $R(u; n_i) > R(u; n_j)$:

The click stream from Yahoo! toolbar identifies a large number of constraints according to Equation (5.1) that the optimal ranking function must satisfy. As for the learning-to-rank problem Joachims (2002), finding the optimal ranking function is an NP-hard problem. Additionally, considering that some of the constraints could be contradictory, a feasible solution may not exist. As usual, the learning problem is translated to a Ranking-SVM optimization problem.

4. Constraint selection

The formulation of Problem 4 includes a potentially huge number of constraints. Every click occurring after time on a news, generates a new constraints involving n and every other non-clicked news published before time. Such a large number of constraints also includes relation-ships on "stale" news articles, e.g., a non-clicked news article published weeks or months before . Clicks are the signals driving the learning process. So it is important to select pairs of clicked news articles that eliminate biases such as the one caused by stale news articles. Clearly, the more constraints are taken into consideration during the learning process, the more robust is the final model. On the other hand, increasing the number of constraints affects the complexity of the minimization algorithm. We propose the following strategy in order to select only the most interesting constraints and thus simplify the optimization problem.

5. Experimental evaluation

5.1 Dataset

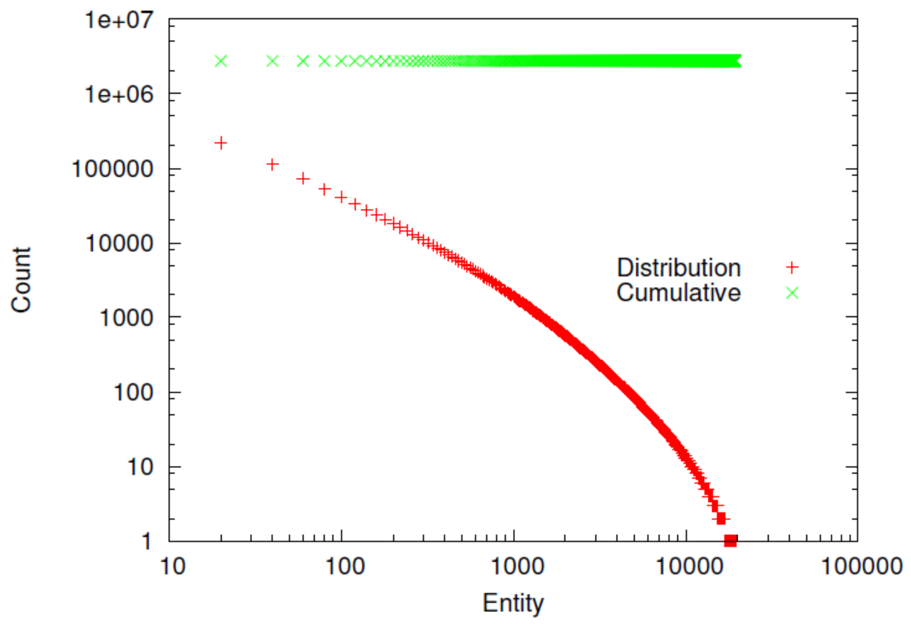

Figure 1: Distribution of entities in Twitter.

To build our recommendation system we need the following sources of information: Twitter stream, news stream, the social network of users,

International Journal of Engineering, Science and Mathematics
Vol. 6 Issue 7, November 2017,
ISSN: 2320-0294 Impact Factor: 6.238
Journal Homepage: http://www.ijesm.co.in, **Email: ijesmj@gmail.com** Double-Blind Peer Reviewed Refereed Open Access International Journal - Included in the International Serial Directories Indexed & Listed at: Ulrich's Periodicals Directory ©, U.S.A., Open J-Gage as well as in Cabell's Directories of Publishing Opportunities, U.S.A

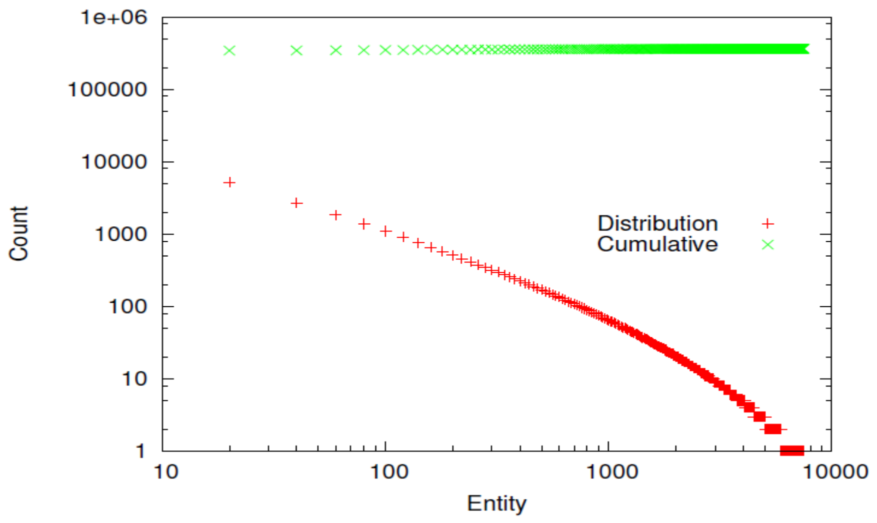

Figure 2: Distribution of entities in news.

5.2. Test set

Evaluating a recommendation strategy is a complex task. The ideal evaluation method is to deploy a live system and gather click-through statistics. While such a deployment gives the best accuracy, it is also very expensive and not always feasible.

User studies are a commonly used alternative evaluation method. However, getting judgements from human experts does not scale well to a large number of recommendations. Furthermore, these judgement are often biased because of the small sample sizes. Finally, user studies cannot be automated, and they are impractical if more than one strategy or many parameters need to be tested.

For these reasons, we propose an automated method to evaluate our recommendation algorithm. The proposed evaluation method exploits the available click data collected by the Yahoo! toolbar, and it is similar in spirit with the learning process. Given a stream of news and a stream of tweets for the user, we identify an event that a user clicks at a news article. Assume that such a click occurs at time = c(n). Suppose the user just logged in the system at time. Then the recommendation strategy should rank the news article n as high as possible.

To create a test instance, we collect all news articles that have been published within the time interval [;], and select a subset as de-scribed by Equation (5.4): we pick the news with largest scores according to the components of content, social, popularity, and number of clicks un-til time . In total we generate a pool of k = 1,000 candidate news. All these news articles are ranked using our ranking function R, and we then examine what is the ranking of the article n in the list.

We use the last 20%, in chronological order, of the Yahoo! toolbar log to test the T.REX and T.REX+ algorithms, as well as all the baselines.

6. Results

We report MRR, precision and coverage results. The two variants of our system, T.REX and T.REX+, have the best results overall.

T.REX+ has the highest MRR of all the alternatives. This result means that our model has a good overall performance across the dataset. CONTENT has also a very high MRR. Unfortunately, the coverage achieved by the CONTENT strategy is very low. This issue is mainly caused by the sparsity of the user profiles. It is well know that most of Twitter users belong to the “silent majority,” and do not tweet very much.The SOCIAL strategy is affected by the same problem, albeit to a much lesser extent. The reason for this difference is that SOCIAL draws from a large social neighborhood of user profiles, instead of just one. So it has more chances to provide a recommendation. The quality of the recommendation is however quite low, probably because the social-based profile only is not able to catch the specific user interests.

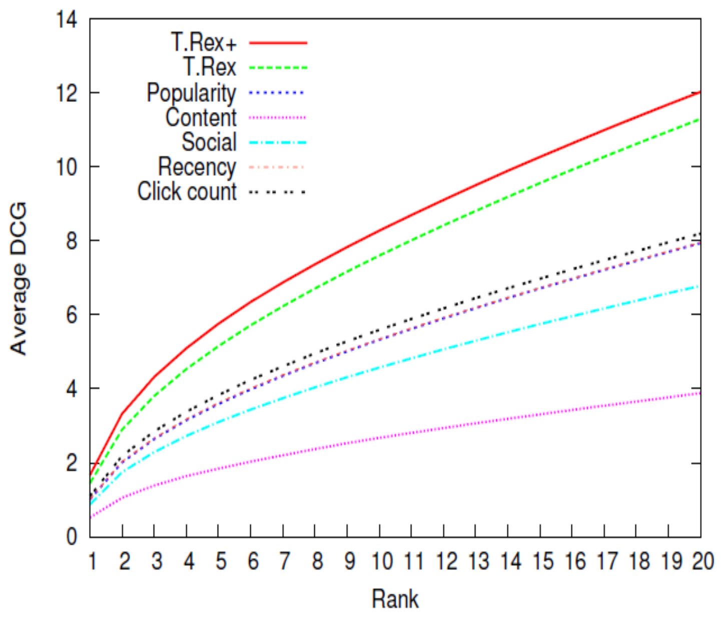

Figure 4: Average discounted cumulated gain on related entities.

7. Conclusion

Our goal is to help the user keep up with the torrent of news by providing personalized and timely suggestions. We devised T.REX, an online recommender that combines stream and graph mining to harness the information available on Twitter. The proposed system processes all the incoming data online in a streaming fashion and is amenable to parallelization on stream processing engines like S4. This feature allows to provide always fresh recommendations and to cope with the large amount of input data. T.REX extracts several signals from the real-time Web to predict user interest. It is able to harnesses information extracted from user-generated content, social circles and entity popularity in Twitter and news streams. Our results show that the real-time Web is a useful resource to build powerful predictors of user interest.

The research results presented in this thesis open the way to some interesting questions that deserve further investigation. It is reasonable to ask whether all of the useful part of the Web can be modeled as bags, graphs, streams or a mix of these. Furthermore, the Web is highly dy-namic and changes rapidly. Therefore new kinds of data might require new models and new algorithms. On the other hand new DISC systems are emerging and they might provide a good fit for some of the algorithms presented here, especially for graph and iterative computations. Finally, the algorithms presented in Chapter 3 and Chapter 4 can be gen-eralized to contexts different from the Web. Finding the limits of appli-cability of these algorithms is an interesting line of research.

8. Reference

1. AvinashLakshman and Prashant Malik. Cassandra - A Decentralized Structured Storage System. ACM SIGOPS Operating Systems Review, 44(2):35—-40, April 2010. ISSN 01635980. 22
2. Leslie Lamport. The part-time parliament. ACM Transactions on Computer Sys-tems, 16(2):133–169, 1998. 21
3. Silvio Lattanzi, Benjamin Moseley, S. Suri, and Sergei Vassilvitskii. Filtering: A Method for Solving Graph Problems in MapReduce. In ACM, editor, SPAA '11: 23rd Symposium on Parallelism in Algorithms and Architectures, pages 85—-94, 2011. 35, 71
4. Haoyuan Li, Yi Wang, Dong Zhang, Ming Zhang, and Edward Y Chang. PFP: Parallel FP-Growth for Query Recommendation. In RecSys '08: 2nd ACM Con-ference on Recommender Systems, pages 107–114. ACM, October 2008. 34
5. J Lin, D Metzler, T Elsayed, and L Wang. Of Ivory and Smurfs: LoxodontanMapReduce experiments for Web search. In TREC '09: 18th Text REtrieval Con-ference, November 2009. 34
6. Jimmy Lin and Chris Dyer. Data-Intensive Text Processing with MapReduce. Synthesis Lectures on Human Language Technologies, 3(1):1–177, 2010. ISSN 19474040. 34
7. Jimmy Lin and Michael Schatz. Design Patterns for Efficient Graph Algorithms in MapReduce. In MLG '10: 8th Workshop on Mining and Learning with Graphs, pages 78–85. ACM, 2010. 35

8. Chao Liu, Hung-chih Yang, Jinliang Fan, Li-Wei He, and Yi-Min Wang.Dis-tributed Nonnegative Matrix Factorization for Web-Scale Dyadic Data Analy-sis on MapReduce. In WWW '10: 19th International Conference on World Wide Web, pages 681—-690, New York, New York, USA, April 2010. ACM Press. ISBN 9781605587998. 35
9. Zhiyuan Liu, Yuzhou Zhang, Edward Y. Chang, and Maosong Sun. PLDA+: Par-allel Latent Dirichlet Allocation with Data Placement and Pipeline Processing. ACM Transactions on Intelligent Systems and Technology, 2(3):26, April 2011. ISSN 2157-6904. 34
10. ZviLotker, Boaz Patt-Shamir, and Seth Pettie. Improved distributed approxi-mate matching. In SPAA '08: 20th Symposium on Parallelism in Algorithms and Architectures, pages 129–136, 2008. 71