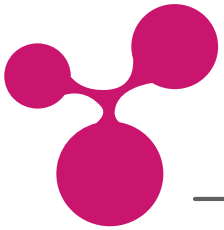
Technische Universität Dresden
Medienzentrum

Prof. Dr. Thomas Köhler
Jun.-Prof. Dr. Nina Kahnwald
(Hrsg.)



GeNeMe '13

Gemeinschaften in Neuen Medien

an der
Technischen Universität Dresden
mit Unterstützung der

BPS Bildungsportal Sachsen GmbH
Campus M21
Communardo Software GmbH
Dresden International University
eScience – Forschungsnetzwerk Sachsen
Gesellschaft der Freunde und Förderer der TU Dresden e.V.
Gesellschaft für Informatik e.V.
Gesellschaft für Medien in der Wissenschaft e.V.
IBM Deutschland
itsax – pludoni GmbH
Kontext E GmbH
Learnical GbR
Medienzentrum, TU Dresden
ObjectFab GmbH
Transinsight GmbH
T-Systems Multimedia Solutions GmbH
Universität Siegen

am 07. und 08. Oktober 2013 in Dresden

www.geneme.de
info@geneme.de

E.3 Recommending in an Enterprise Social Media Stream without Explicit User Feedback

Torsten Lunze[1], Philipp Katz[2], Dirk Röhrborn[3], Alexander Schill[2]
[1]Communote GmbH
[2]Dresden University of Technology, Chair for Computer Networks
[3]Communardo Software GmbH

Abstract
Social Media Streams allow users to share user-generated content as well as aggregate different streams into one single stream. Additional Enterprise Social Media Streams organize the stream messages into projects with different usage patterns compared to public collaboration platforms such as Twitter. The aggregated stream helps the user to access the information in one single place but also leads to an information overload. Here, a recommendation engine can help to distinguish between relevant and irrelevant information for the users.

In previous work we showed how features inferred from messages can predict relevant information and can be used to learn a user model. In this paper we show how this approach can be used in a productive enterprise social media stream application without using explicit user feedback. We develop a time binned evaluation measure which suits the scenario to steadily recommend messages of the stream. Finally, we evaluate our algorithm in different variations and show that it helps to identify relevant messages.

1 Introduction

A social media stream contains messages of different sources with a steady flow of new incoming messages. Users are interacting with each other and with the message, e.g. message can be liked, commented, favored and organized into topics or discussions. In enterprise social media streams topics will be assigned with permissions defining what a user may read or not. The stream is steadily filled with new information and the urge arises to provide the user with the possibility to easily distinguish between interesting and uninteresting messages. This is a differentiation from typical recommender systems where specific items or messages are picked and recommended to the user.

A stream recommender can be defined as follows: A stream recommender or stream recommender system (SRS) computes a relevance score for each message of the stream at the moment when the message occurs in the stream using information that has been obtained during the past stream interaction. Therefore a SRS must compute a relevance score for each user who has access to the item.

Following such scenarios, a SRS must fulfill the following constraints: First, it must be able to compute the relevance score of an incoming message in near real-time. Second, it must be able to find interesting messages based on the past experience of the user. Third, it must be able to handle changing interest. Fourth, it must be able to learn without explicit user feedback.

In [Lunz13] we presented an algorithm that fulfills the first three constraints. In this paper we will focus on the fourth point to learn without explicit user feedback and evaluate it within the scenario described in Section 3. Then, in Section 4 related work is given. In Section 5 we define different learning strategies for a stream based recommender that will not use any explicit feedback or ratings. The strategies are then evaluated in Section 6 by using a time binned evaluation measure. Finally in Section 7, a conclusion is drawn.

2 Scenario

In an Enterprise Social Media Stream Application employees work together in different projects and share their information within those projects. As more projects and more employees are active within such an application, the amount of messages increases and the application should provide employees a possibility to filter for relevant messages. Besides any specific recommendation algorithm, only messages the employee did not yet interact with need to be considered for recommendation. Messages the employee liked, answered or read do not need to be included in the recommendation since the employee already knows the message.

Typically, most of the messages in a stream are getting irrelevant as time passes. Especially in an enterprise scenario messages should be read within the same day or a couple of days. Therefore sorting messages by a computed relevance score without time decay will not be useful at all. A simple solution is to only show the Top N messages per day or week sorted by a relevance score. This also has the advantage that this sort order is easily understandable by the end user.

In Figure 1: Filter and sort messages by relevance score it is shown how a selector between different views on a stream can be integrated into the Enterprise Social

Media Stream Application Communote[1]. The standard view shows all messages of the stream but it can be switched to only show messages the employee did not interact with, and sort it by the relevance score per day separately.

Figure 1: Filter and sort messages by relevance score

3 Related Work

There are several categories of recommender systems: [Burk07] and [Ricc11] distinguish between collaborative, content-based, demographic, knowledge-based, community-based and hybrid recommenders. Relevant for this work are recommenders that can be applied on social media streams, such as news recommenders. The most of those recommenders are using some form of content-based, collaborative or community-based methods for recommendation, such as [Diaz12]. In [Lops11] it is mentioned, that the content-based in contrast to the collaborative recommenders have their advantages in user independence, transparency and in the ability to handle the new item problem. The new item problem is crucial for stream recommenders for ranking a new item fast. Collaborative methods have their advantage, once enough relations between users and items exist and can be exploited. There are methods for collaborative recommenders that are using stereotypes or clusters [Wan11] for the new item problem. For new items with a new context it is not sufficient to infer a rank with those methods as long as only little interaction has been observed for the new items.

In [LiWZ2011] content-based news recommender are distinguished into term weighting and concept weighting recommenders. The term weighting recommender uses mainly information filtering methods and applies them to learned user models. In contrast, concept weighting recommenders use ontologies to discover term similarities. News recommendation is used for example in [Das07] and [LiWL2011] and only a

1 http://www.communote.com

few recommenders such as [Guy10] or [Lunz09] focus on news recommendation in enterprises. [Guy11] uses activity streams for recommendation. [Lunz09] uses a first approach of a stream recommender using a content based recommender but it lacks necessary performance and quality. An overall system architecture for stream based recommender is given in [Katz11].

4 Learning Strategies

In [Lunz13] a basic stream based algorithm is presented that extracts features from messages. Based on the features, a subset of messages can be recommended to users. We showed there a new message is likely to be relevant for a user, if the user participates in the discussion associated with this message or is notified within such a discussion. Also, if the new message contains a mention for the user, the message is highly relevant for the user. In this way a high relevance score can be determined for a subset of messages. For the other set of messages - that are not part of an existing discussion or part of a discussion the user is not involved in - no trivial features can be used to determine a relevance. As an result of the algorithm a relevance score in the range of [0..1] is computed per user and message.

Therefore we trained a term based user model in [Lunz13] using the terms of the messages where a positive relevance has been determined based on the trivial features. This user model is then used to compute a relevance score for new messages per user if none of the trivial features applies. The question arises which features to use in which way for learning. So we extended the algorithm of [Lunz13] with different learning and ranking strategies:

1. **Standard:** If the user is the author of a message, is mentioned within the message or the message is part of a discussion where the user is author or mentioned, the terms of the messages are integrated into the user model.
2. **Learn from Direct Parents**: It extends No. 1 by using the direct parent message, if there is one, to be integrated into the user model.
3. **Learn from All Parents**: It extends No. 2 by using all parent messages recursively, if there is one, to be integrated into the user model.
4. **No Discussion Learning**: It limits No 1. by only integrating messages into the user model, the user is author of or is mentioned within.
5. **Half score on Non Participation**: This uses the learning strategy of No 1. If the message to rank is part of a discussion the user is not an author of or not mentioned, the relevance score is multiplied by 0.5. The reason of this strategy is that, if the user did not participate or was not involved within the discussion it is likely to be irrelevant.

5 Evaluation

Dataset

For evaluation we used Communote which is used within a real world company for over five years. During December 2012 to April 2013, ten users submitted a total of 30,000 ratings. A rating refers to a message that its user either marked as relevant or irrelevant. As implementation for the stream-based algorithm we used the open source framework SPEKTRUM2.

Evaluation Measures

To match the scenario defined in Section 3 we only considered ratings that refer to messages

- the user is not author of,
- the user is not mentioned within and
- the user did not reply to the message (if the user replied to the message, he must have read the referring message).

In all these cases it is highly likely that the user will already have read the message and it is not necessary to recommend it and hence it would blur the evaluation. Therefore only the messages the user did not interact with are used in the further evaluation. This reduces the number of ratings to 2,600. We used all those ratings for evaluation; no ratings at all have been used for training. Hence the user model is created and maintained by the features extracted as described before.

In the first evaluation the Precision, Recall and the F-Score[3], as well as the RMSE[4] were computed using the computed relevance scores and ratings. To compute the RMSE, a relevant rating is used with a score of 1, otherwise 0. To compute the other measures, a decision had to be made, if the computed relevance score is either relevant or not relevant. Different thresholds had been used to determine the threshold which leads to the highest F-Score. The RMSE itself is independent from the decision, and it gives an indication of the overall error.

The problem of this evaluation measure is that it does not reflect the number of messages returned. It also gives no indication if the relevant message found distribute evenly over time or are concentrated within a specific time range. Therefore we used

2 http://spektrumprojekt.de

3 F-Score: Combines the Precision and Recall into a single measure. The F-1-Score weights Precision and Recall equal.

4 RMSE: Root Mean Squared Error

the Precision@ and the Average Precision as measures. The Precision@ is defined as follows, whereby k is number of elements to be considered and p the number of found relevant elements in the returned dataset:

$$P@_k = \frac{p}{k}$$

The Average Precision also considered the ordering of the elements, if a relevant element is ranked higher in the list of returned elements it will lead to a higher Average Precision. With R(i) returning 1 if the element at position i in the returned set of elements is relevant or returning 0 if it is irrelevant, then the Average Precision is defined as:

$$AP@_k = \frac{\sum_{i=0}^{k} R(i) \times P@_i}{k}$$

Instead of computing the overall values, we divide the messages into time bins of days and weeks. The Precision@ and the Average Precision can then be computed first per user per time bin and then aggregated by user and time bin. We use the number of ratings per user per time bin as weights to balance the precision based on the actual number of ratings used to determine the Precision@ of a time bin. With U as the set of all users, T the set of all time bins, φ_u the number of ratings for a user u, $\omega_{(u,t)}$

the number of ratings for user in time bin t and P@(u,t) as the precision @ for user in time bin t, the Time-Binned Precision can be defined as follows:

$$Precision = \frac{1}{\sum_{u \in U} \varphi_u)} \sum_{u \in U} \left(\frac{\varphi_u}{\sum_{t \in T} \omega_{u,t}} \sum_{t \in T} \omega_{u,t} \times P@(u,t) \right)$$

Since $\varphi_u = \sum_{t \in T} \omega_{u,t}$ and with the definition of this can be reduced to:

$$Precision = \frac{1}{\sum_{u \in U} \varphi_u)} \sum_{u \in U} \sum_{t \in T} p_t$$

Similar the Time-Binned Average Precision can then be defined as:

$$AveragePrecision = \frac{1}{\sum_{u \in U} \varphi_u} \sum_{u \in U} \sum_{t \in T} \sum_{i=0}^{k} R(i) \times \frac{p_{t,i}}{i}$$

For example the Time-Binned Precision and Average Precision for the Top 25 per day follows this order:

1. For each day the Top-25 messages with the highest computed relevance score per user is determined.
2. Of those Top-25 messages per user: The messages with ratings in the evaluation dataset are determined.
3. The Precision and Average Precision for that time bin is computed on the Top-25 messages with existing ratings in the evaluation dataset. Each time bin per user is weighted by the number of ratings available in that time bin. This is necessary if the ratings are sparse and not available for all possible messages per user. This way, users with only a few ratings will have less impact as users with many ratings.
4. For each time bin, the Precision (and Average Precision) is weighted by the number of ratings in the evaluation dataset; that is the sum of all ratings per user.

Evaluation Results

In Figure 2: RMSE and F-1 Score for the different Learning Strategies the RMSE and F-1 Score are shown for the different Learning Strategies for non-interacted messages only. It is compared to a Random Ranker which uses random rank per message and user. We cannot use a collaborative algorithm for comparison since it is not suitable for a stream-based recommender. The F-1 score is slightly higher for the Learning Strategy 4 and 5 in contrast to the random ranker. The RMSE is significantly lower and therefore better for all Learning Strategy compared to the random ranker. The lowest RMSE and highest F-1 Score are reached for the Learning Strategy 5.

The time binned Precision and Average Precision for top-configurations Top-25 per Day and Top-50 per Week are shown in Figure 3: Time-Binned Precision and Average Precision for Top 25 Messages per Day and Figure 4: Time-Binned Precision and Average Precision for Top 50 Messages per Week, respectively. Here it is clearly shown that for all those top-configurations the Learning Strategies are better than the Random Ranker. The best values are reached with Learning Strategy 5, which shows the usefulness of the idea to lower the relevance score of a message which is part of a discussion the user did not participate in.

By comparing the same Learning Strategies with the two different top-configurations, the Top-25 Day configuration is worse compared to the Top-50 per Week. This means that the results per day are not as good as taking a whole week as time bin. This is an indication of a strong variation between the results for each time bin separately. The reasons for this difference are due to research. Besides that it can be clearly stated that our algorithm helps to find relevant messages in our scenario, since it performs better as the Random Ranker in the time binned evaluation measure.

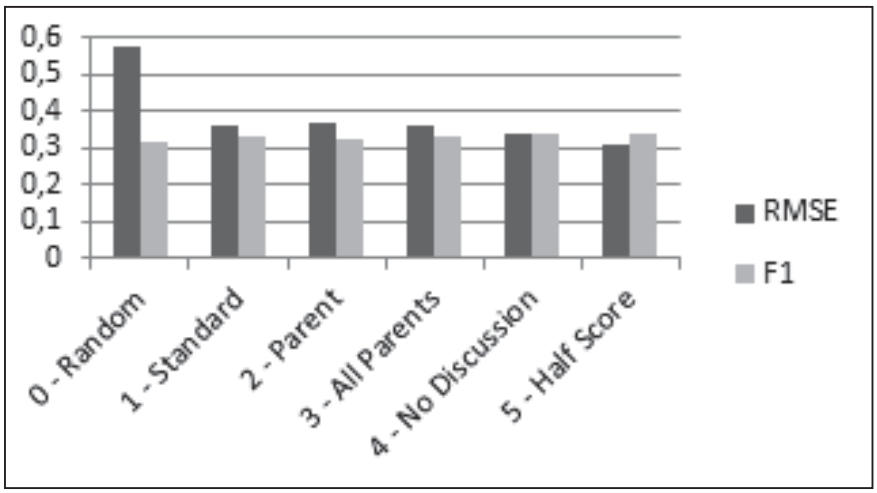


Figure 2: RMSE and F-1 Score for the different Learning Strategies

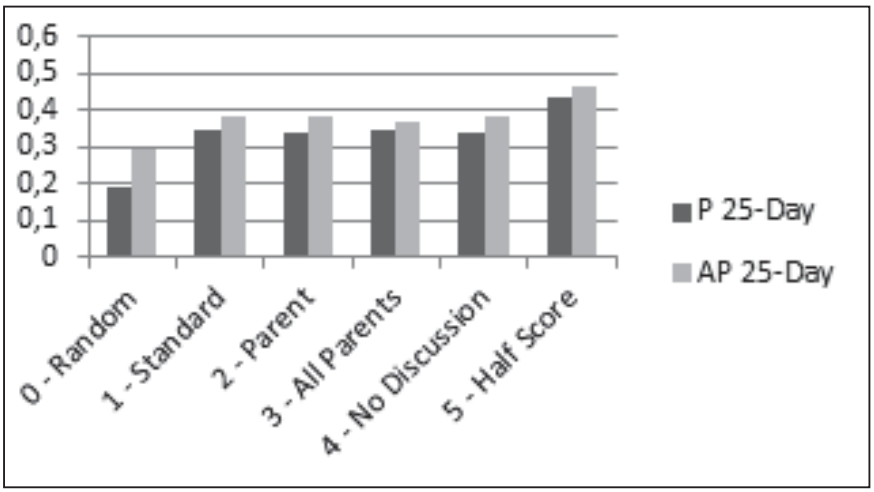


Figure 3: Time-Binned Precision and Average Precision for Top 25 Messages per Day

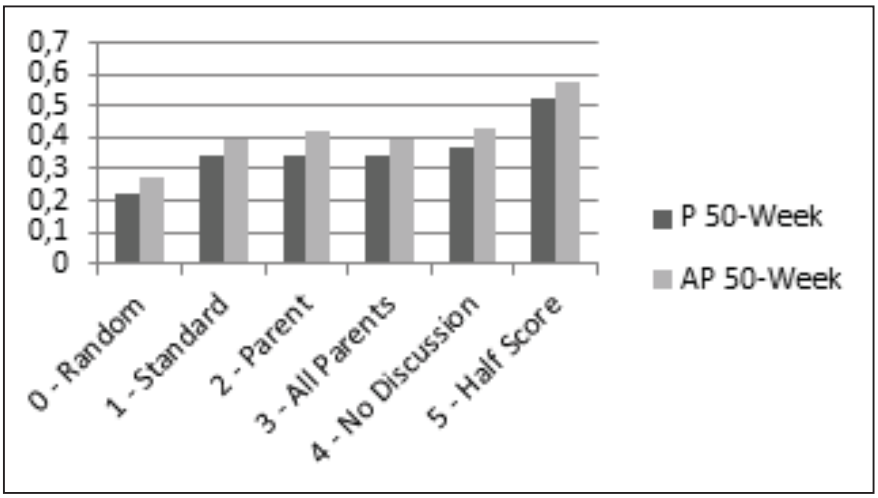


Figure 4: Time-Binned Precision and Average Precision for Top 50 Messages per Week

6 Conclusion

In this paper we described and motivated a scenario for a practical recommendation use case within an Enterprise Social Media Stream. Furthermore, we defined a new evaluation measure that represents this scenario as close as possible. We then put the pieces together on applying our algorithm with different learning strategies to this scenario without using explicit ratings. We showed that those strategies help to filter for the relevant elements in a continuous stream that can be used within a productive Enterprise Social Media Stream Application.

The future work is to research more configurations for different learning strategies to improve the Precision and the Average Precision. Furthermore the approach will be applied within a productive system to obtain direct user feedback about the recommendation results and the integration within the frontend.

Acknowledgements

The results presented in this paper have been developed within the research project SPEKTRUM. This project is funded by the Free State of Saxony and the EU (European Regional Development Fund). We would like to thank all the users at the Communardo Software GmbH participating in creating the dataset for the evaluation.

References

[Burk07] Burke, Robin: The adaptive web, Chapter: Hybrid web recommender systems, pages 377–408, Springer-Verlag, Berlin, Heidelberg, 2007.

[Chan11] B. Chandramouli, J, J, Levandoski, A. Eldawy, M. F. Mokbel: StreamRec: a real-time recommender system, Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, SIGMOD 2011.

[Das07] A. Das, M. Datar, A. Garg, S. Rajaram: Google News Personalization: Scalable Online Collaborative Filtering, Proceedings of the 16th International Conference on World Wide Web, 2007, p. 271.

[Diaz12] E. Diaz-Aviles, L. Drumond, L. Schmidt-Thieme, W. Nejdl: Real-Time Top-N Recommendation in Social Streams, Proceedings of the sixth ACM conference on Recommender systems, 2012, p. 59.

[Guy10] I. Guy, N. Zwerdling, I. Ronen, D. Carmel, E. Uziel: Social Media Recommendation based on People and Tags, Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2010, p. 194.

[Guy11] I. Guy, I. Ronen, A. Raviv: Personalized Activity Streams: Sifting Through the ``River of News'', Proceedings of the fifth ACM Conference on Recommender Systems, 2011, p. 181.

[Katz11] P. Katz, T. Lunze, M. Feldmann, D. Röhrborn, A. Schill: System Architecture for handling the Information Overload in Enterprise Information Aggregation Systems, BIS 2011; Poznan, Poland; 6/2011.

[LiWL11] L. Li, D. Wang, T. Li, D. Knox, B. Padmanabhan: SCENE: A scalable two-stage personalized news recommendation system, Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval, 2011,.

[LiWZ11] L. Li, D. Wang, S. Zhu, T. Li: Personalized News Recommendation: A Review and an Experimental Investigation, Journal of Computer Science and Technology, 2011, p. 754.

[Lops11] P. Lops, M. Gemmis, G. Semeraro: Content-based Recommender Systems: State of the Art and Trends, Recommender Systems Handbook, 2011.

[Lunz09] T. Lunze, M. Feldmann, T. Eixner, S. Canbolat, A. Schill: Aggregation, Filterung und Visualisierung von Nachrichten aus heterogenen Quellen -- Ein System für den unternehmensinternen Einsatz, Proceedings of the GeNeMe'09 Workshop; Dresden, 2009.

[Lunz13] T. Lunze, P. Katz, D. Röhrborn, A. Schill: Stream-based Recommendation for Enterprise Social Media Streams, BIS 2013; Poznan, 2013.

[Ricc11] F. Ricci, L. Rokach and B. Shapira: Introduction to Recommender Systems Handbook, Recommender Systems Handbook, 2011.

[Wan11] Y. Wan, C. Chen: An Effective Cold Start Recommendation Method Using A Web Of Trust, PACIS 2011 Proceedings, 2011.