Proceedings of Machine Learning Research 109:36–45, 2019 ORSUM 2019

Personalized Push Notifications for News Recommendation

Babak Loni BABAK.LONI@PERSGROEP.NL
DPG Media, Amsterdam, the Netherlands

Anne Schuth ANNE.SCHUTH@PERSGROEP.NL
DPG Media, Amsterdam, the Netherlands

Lucas de Haas LUCAS.DE.HAAS@PERSGROEP.NL
DPG Media, Amsterdam, the Netherlands

Jeroen Jansze JEROEN.JANSZE@PERSGROEP.NL
DPG Media, Amsterdam, the Netherlands

Vasco Visser VASCO.VISSER@PERSGROEP.NL
DPG Media, Amsterdam, the Netherlands

Marlies van der Wees MARLIES.VAN.DER.WEES@PERSGROEP.NL
DPG Media, Amsterdam, the Netherlands

Abstract

Push notifications on mobile devices are an important way for users to stay up to date with news. Push notifications can also be a major source of annoyance for users: being interrupted at the wrong time for something you do not care about is frustrating. It is crucial to ensure the right push is sent to the right user at the right moment.

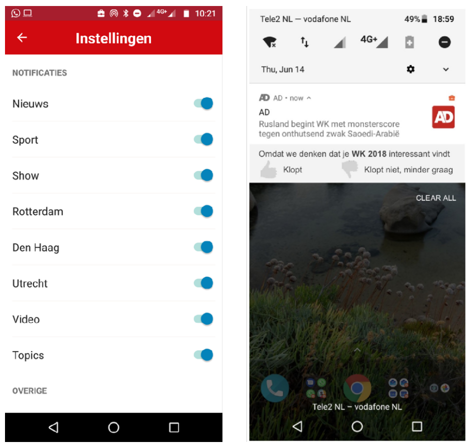
In this paper we address this problem of personalized push notifications. We introduce our streaming push personalization pipeline, describe how we personalize pushes, discuss challenges, and end with open questions.

1. Introduction

The abundance of online news services makes recommender systems convenient techniques to generate personalized recommendations and help users to discover relevant articles. In such systems, recommendations are typically created by learning to generate a personalized ranked-list for each user. The recommendations are then displayed as a list of items on the web or mobile apps.

Despite the presence of many studies where recommendations are generated as a ranked list of items (Hopfgartner et al., 2016), to our knowledge, there are no studies that propose to serve personalized news recommendation with push notifications. A push notification is a fast and effective way to inform users about the latest and most important news via an in-app message on a mobile device or via a browser message.

In this work we propose a distributed system to create personalized push notifications in the context of a major European news publisher. In Section 2 We first motivate why it is important to personalize push notifications. We then describe our Personalized Push Notification system and its architecture in Section 3. In Section 4 we discuss open questions


10:21
Instellingen

NOTIFICATIES
Nieuws
Sport
Show
Rotterdam
Den Haag
Utrecht
Video
Topics
OVERIGE

Tele2 NL — vodafone NL 49% 18:59
Thu, Jun 14
AD • now
AD
Rusland begint WK met monsterscore tegen onthutsend zwak Saoedi-Arabië
Omdat we denken dat je **WK 2018** interessant vindt
Klopt
Klopt niet, minder graag
CLEAR ALL
Tele2 NL – vodafone NL

Figure 1: Category subscription (left) and a the design of a push notification (right) of the AD news app with a *recommendation reason* and *feedback buttons*.

and challenges of making personalized pushes. We briefly describe some related work in Section 5 and finally we draw some conclusion in Section 6.

2. News Consumption via Push Notifications

In this section we describe the current model of sending push notifications, as used at DPG Media,[1] discuss the challenges and limitations of the current system and motivate the importance of using a personalized push notification system.

Currently, push notifications at DPG Media are not personalized. Instead, users of most mobile apps of DPG Media can subscribe to one or more predefined news categories[2] in which they are interested in. A news editor then decides whether a newly published article should be pushed to all users who have subscribed to its category. The push-worthiness of the article is thus determined by humans, based on factors such as urgency and importance of the events described in the article. Figure 1 illustrates the category subscription and a sample push notification from Algemeen Dagblad (AD), the app of one of the most popular newspapers in Europe, published by DPG Media.

The existing, non-personalized, setup suffers from several issues. First and foremost, not every article that is pushed within a specific category is relevant to all the users who subscribed to that category. For instance, within the *sports* category, news editors can decide to push articles about *soccer*, *cycling*, *darts*, *formula 1* etc., however a user subscribed to the *sports* category will typically not have an interest in all of these subcategories.

Second, the push-worthiness of a news article is subjective and is not necessarily equal for all users. For example, whether or not a piece of news about *formula 1 championship* is important and should be pushed is subject to the opinion of the news editor and might not necessarily be relevant to the user.

1. DPG Media is a major publisher of newspapers in the Netherlands, Belgium, and Denmark.
2. Common categories are *general news*, *sports*, or *show*.

Finally, since users can only subscribe to a limited number of broad categories, news editors have no means of reaching smaller audiences than those subscribed to the broad categories. As a result, our editors only push very few articles, based on their expected relevance to a large audience. Niche articles, however, will never be pushed to anyone since there is no way for editors to reach a niche audience.

With personalized push notifications, we aim to address the above shortcomings. We want to bring niche articles to only those people who want to read them, while simultaneously filtering out articles that are irrelevant for a given user. Our first instantiation of niches is hyper locality: articles that are specific to a very small region are typically only interesting to a small number of users.

3. Personalized Push Notification

We address the challenge of news personalization with personalized push notifications. Our goal is to find the most relevant news for each user while still bringing diverse news, hence keeping the *filter bubble* (Pariser, 2011) to a minimum. More specifically, personalized push notifications should meet the following criteria:

- **Personalized**: Push notifications should be personalized, that is, the interests and the preferences of the users should be taken into account.
- **Explainable**: The reason why a push notification was sent should be explainable to justify the recommendations and establish a ”sense of forgiveness” when users do not find the recommendations relevant to them (Van Barneveld and Van Setten, 2004).
- **Include important news**: Regardless of the degree of personalization, push notifications should still inform users about breaking news and important updates.
- **Diversity and opposing opinions**: To make sure the personalized recommendations do not create filter bubbles for users, the system should make sure that recommendations are diverse and are not necessarily supporting the opinion that users often read. In particular for a news product, as opposed to a product that is merely entertainment, it is important to maintain objectivity.
- **(Hyper-)local news**: In particular, (*hyper-*)*local news* can only ever be pushed when push is personalized. (Hyper-)local news is only relevant to very few users, but to these users, it is typically highly relevant: users care about what happens around the corner. This type of news is not pushed in non-personalized systems: categories that users can subscribe to will never be fine-grained enough. More importantly, in a non-personalized system, human editors will never be able to identify all the small regions an article is relevant to.
- **Anonymous users**: The majority of the readers are users without an account. The system should be able to learn the preferences of such users.
- **Address the cold-start problem**: An inherent challenge in news recommendation is the cold-start problem: news is most relevant shortly after it is published. At that moment, there is no or a very limited number of interactions on the newly published articles and it is therefore not possible to effectively exploit algorithms such

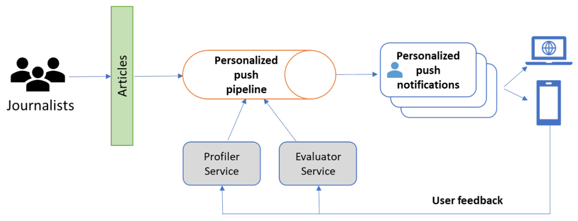


Figure 2: The high level architecture of the personalized push notification pipeline. The personalized push pipeline is described in more detail in Figure 3.

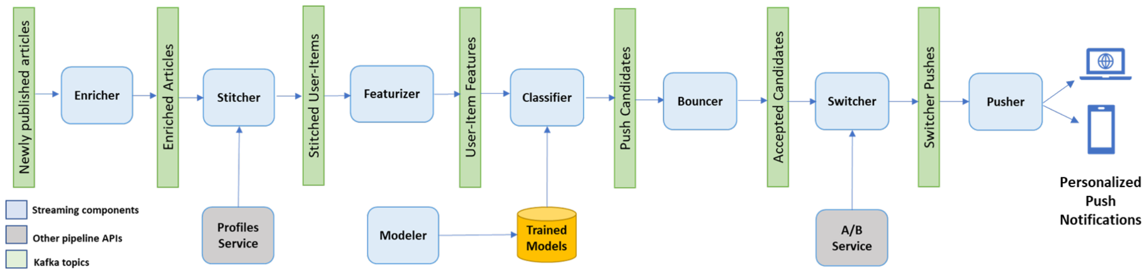


Figure 3: The architecture of the push pipeline. Green blocks are Kafka topics, blue blocks are consumer-producer components. RESTful APIs are depicted in gray.

as collaborative filtering as they heavily rely on user-item interactions. The system should be capable to effectively exploit the content of news articles to tackle the cold-start problem.

3.1. Architecture

We propose a distributed, extensible and scalable architecture consisting of several decoupled components that communicate using an asynchronous messaging system. We also use Redis, a distributed cache system, in several components to make sure the pipelines can effectively handle the large traffic of incoming articles and user interactions. Figure 2 illustrates the high-level architecture of our system. Our system consists of three pipelines, each with a set of independent components:

- **Push pipeline**: this pipeline processes newly published articles and pushes them to the relevant users.
- **Profiler pipeline**: this pipeline processes users' interactions and maintains their profiles.
- **Evaluation pipeline**: this pipeline decides about the models that should be used, maintains experiments and A/B testing and collect users' feedback to tune models.

In the next section, each pipeline is described in more detail.

3.1.1. Push pipeline

As soon as an article is published by a journalist, the article is picked up by the push pipeline. The article is then enriched and stitched to candidate users. The stitched user-item candidates are then classified and the accepted candidates are pushed to the users.

Figure 3 illustrates the architecture of the push pipeline. The pipeline consists of eight independent components that communicate asynchronously in a streaming fashion using Kafka:[3]

- **Enricher**: Given an article in JSON format with fields including title, raw text, and author, the Enricher adds metadata including named entities, sentiment (Pang et al., 2002), and IPTC topics.[4]
- **Stitcher**: Given an enriched article, the Stitcher queries the user profile for all users that might be interested in the article. It produces a tuple consisting of an article and a user profile, i.e., a user-item, to the next Kafka topic.
- **Featurizer**: Given a user-item, the Featurizer extracts features and adds them to the next Kafka topic. Some examples of features are: 1) the degree to which a user is interested in the location of the article, 2) cosine similarity between a content embedding of the article and the user, 3) the amount of interest the user showed in the author of the article. Currently we use a Word2Vec model (Mikolov et al., 2013) to produce embeddings of users and articles from their Bag-of-Words (BoW) representations. A user's BoW is the aggregation of the BoW of the articles that they read in the past.
- **Classifier**: Given the user-item features, the Classifier decides whether the item is relevant for the user. If relevant, the Classifier adds the item to the next Kafka topic.
- **Modeler**: Given a historical dataset of user profiles and items, the Modeler allows for creating classification models for the Classifier.
- **Bouncer**: Given an incoming message from a Kafka topic, the Bouncer decides to add it to the next Kafka topic or not, based on business rules such as a limit on the total number of pushes we send on a day.
- **Switcher**: Given the accepted push candidates, the switcher dispatches the push candidates to different A/B buckets using our A/B testing service. Depending on the configuration of the bucket that a push candidate is assigned to, the switcher decides whether a push candidate should be pushed or not. Bouncer also makes sure that the items that are already pushed to a users are not pushed again.
- **Pusher**: Given user-items, the Pusher formats the actual push notification. Furthermore, it adds a message to the topic for updating the profile and notifying the AB-platform that a push notification is sent out.

Note that all our components can be scaled up horizontally with demand.

3. Kafka calls its streams *topics*. Components can *produce* messages to *topics* that other components can then *consume* from. See `https://kafka.apache.org`.
4. See `http://cv.iptc.org/newscodes`.

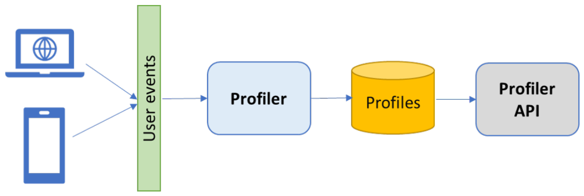


Figure 4: The architechture of the profiler pipeline.

3.1.2. Profiler pipeline

An important aspect of a recommender system is to be able to properly represent a user. In our system, a user is represented by a rich profile that is created based on the users' interaction data and is further enriched with article data and potentially contextual information. We continuously update user profiles while also keeping daily records, i.e., a single representation of each user is created every day. The daily user profiles are later aggregated, with an aggregator job, to have a more complete representation of users. We use Elasticsearch[5] to store user profiles. Daily profiles are also stored in a Redis cache to avoid frequent I/O and to improve the scalability of the pipeline.

Representing users with daily profiles has two advantages. First, the model can be trained with a varying window of interaction data, depending on the intended length of the time window. A benefit of this approach is that we can train and predict with only recent user interactions and therefore the recommendations are adapted with the change of user preferences over time. Recency has been shown to be effective in generating recommendations (Li et al., 2014). Second, the granularity of daily profiles makes it easy to discard old interactions. This is particularly important with the new GDPR regulations [6] since online services are not allowed to keep user interaction histories indefinitely. In our system, this is achieved by deleting old daily profiles.

Figure 4 illustrates the profiler pipeline. User events are collected from different channels, processed and persisted to daily Elasticsearch indices. The profiles are accessible through a RESTful API, which is used by the push pipeline.

3.1.3. Evaluation Pipeline

Our pipeline integrates an ongoing evaluation process that runs multiple experiments with A/B testing and collects feedback on each experiment to adjust models. All push candidates that are created by the push pipeline are assigned to an experiment and a bucket that is determined by our A/B service. The evaluator service evaluates the performance of each bucket based on the collected feedback and provides insights about our experiments.

3.2. Modeling

Our setup allows the integration of any recommendation model to our pipeline. The modular setup of the Featurizer and Classifier makes it possible to run multiple classification models. Our current model combines a content similarity score with a location overlap score. The

5. https://www.elastic.co/
6. https://gdpr-info.eu/

content similarity score is the cosine similarity between the content of the item and the aggregated content read by a user, as stored in his/her user profile.

The location overlap score is a weighted sum of two location ratios: the first measures the ratio of the extracted locations in the article to the number of locations the user has previously read about. The second term measures the ratio of the locations in the article to the number of locations that the user has previously physically visited while reading articles in the AD app. Formally, we calculate the location overlap score as follows:

$$l_{overlap} = w\frac{|l_{user} \cap l_{item}|}{|l_{user}|} + (1-w)\frac{|gl_{user} \cap l_{item}|}{|gl_{user}|}. \tag{1}$$

where l_{user} is the set of article locations previously read by the user, gl_{item} is the set of physical locations of the user while reading articles in the AD app, l_{item} is the set of extracted locations from the article, and w is a weight parameter that controls the contribution of the above ratios on the calculated score. In our current setup, based on an experiment on a small number of users, we found $w = 0.7$ to be an appealing value for the weight parameter. Future work involves further tunning of this parameter based on experiments on larger number of users.

4. Future Work and Open Questions

Personalizing push notifications breaks some new ground. Below are our biggest challenges.

- **Daily limits**: Imagine the following scenario: we limit the number of push notifications per user per day to 10, and in the morning there is a series of articles that is urgent and relevant to a user. What should we do? Should we deplete the 10 pushes right away, not knowing what the rest of the day might bring us? We currently limit the number of pushes to 1 per 5 minutes, 2 per hour, and 10 per day. But we plan to experiment with these settings and are considering personalizing these as well.

- **Diversity**: As mentioned in Section 3, we aim to recommend diverse news to users. Diversity involves both near duplicate prevention and topic diversification. Since push recommendation is not a ranking problem, diversity becomes an issue as most of the existing literature on diversification focus on diversifying top-N recommendation lists (Hurley and Zhang, 2011). It is not straightforward to get a holistic view on all the sent push notifications in the way we can do this when composing a ranking. Classical diversity algorithms, such as *maximal marginal relevance* – MMR, (Carbonell and Goldstein, 1998) – are therefore not directly applicable. We can, however, take into account the similarity of an article to articles we pushed earlier. Currently, we do this to avoid sending near duplicate pushes. Future plans include exploitation of diversification algorithms that diversify the topics that are covered for each user.

- **Explainability**: In Figure 1 we show a recommendation explanation right below the push notification *"Because we think you are interested in World Cup 2018."* We currently do not support this feature, but we consider providing the most prevalent reason that caused our algorithm to push this specific article to this user, for example using the approach introduced by ter Hoeve et al. (2018).

- **Explicit Feedback**: Once we provided recommendation reasons, we intend to provide users with a means to give feedback on those reasons. In Figure 1 we show feedback buttons that are intended for a user to provide feedback, not on this specific article but on the reason that caused us to push this article. This type of feedback can readily be used to update the profile we maintain for this user.

- **Timing**: Some content is less time sensitive and can be scheduled for the right time for a user. For example, the relevance of articles about parenting depends on a user's life phase. To be able to recommend such articles to users at the right moment, we need a classifier that predicts the 'expiration date' of an item. Currently, we only push articles that are at most 1 hour old.

5. Related Work

The majority of the published articles on news recommendation adapt content-based methods to address common challenges in news recommendation such as cold-start, sparsity, and recency (Karimi et al., 2018). Moreover, news recommender systems should be scalable and fast to be able to serve real-time recommendations to a large number of users in a few milliseconds. The work of Lu et al. (2014) is an example of a scalable news recommender system where the scalability is achieved by adapting MinHash clustering to search similar users within smaller clusters. Doychev et al. (2014) propose a scalable architecture that could address the short-response-time requirement of the Plista news personalization challenge. Their model benefits from pre-computed recommendations that are stored in a Redis cache while the recommendations are updated in the background as frequently as possible.

From the modeling perspective, (Kompan and Bieliková, 2010) and (IJntema et al., 2010) are examples of content-based methods where TF-IDF and ontology-based concepts are used to construct user profiles and find similar articles. The work of Montes-Garcia et al. (2013) is an example where the geographical proximity of users to news articles is exploited as an indication of the relevance of the articles to users.

6. Conclusion

In this paper, we present an overview of our streaming personalized push notification framework for the news domain. We propose an adaptive and scalable system that is capable of integrating different recommender models to build personalized push notifications. In contrast to the mainstream recommender system models where recommendations are typically served as a list, personalized push notifications are arbitrary recommendations that can be sent to users at any time.

Due to the nature of personalized push notifications for news, recommendations should be generated real-time when articles are published. Building such a system requires a proper architecture that can address issues such as scalability, performance, cold-start, timing and daily limits. Moreover, the system should be able to generate diverse, explainable and fair recommendations. Our distributed system is capable of plugging various models and constraints as different components to address the above issues.

References

Jaime G Carbonell and Jade Goldstein. The use of mmr and diversity-based reranking for reodering documents and producing summaries. 1998.

Doychin Doychev, Aonghus Lawlor, Rachael Rafter, and Barry Smyth. An analysis of recommender algorithms for online news, 2014.

Frank Hopfgartner, Torben Brodt, Jonas Seiler, Benjamin Kille, Andreas Lommatzsch, Martha Larson, Roberto Turrin, and András Serény. Benchmarking news recommendations: The clef newsreel use case. *SIGIR Forum*, 49(2):129–136, January 2016. ISSN 0163-5840. doi: 10.1145/2888422.2888443. URL http://doi.acm.org/10.1145/2888422.2888443.

Neil Hurley and Mi Zhang. Novelty and diversity in top-n recommendation – analysis and evaluation. *ACM Trans. Internet Technol.*, 10(4):14:1–14:30, March 2011. ISSN 1533-5399. doi: 10.1145/1944339.1944341. URL http://doi.acm.org/10.1145/1944339.1944341.

Wouter IJntema, Frank Goossen, Flavius Frasincar, and Frederik Hogenboom. Ontology-based news recommendation. In *Proceedings of the 2010 EDBT/ICDT Workshops*, EDBT '10, pages 16:1–16:6, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-990-9. doi: 10.1145/1754239.1754257. URL http://doi.acm.org/10.1145/1754239.1754257.

Mozhgan Karimi, Dietmar Jannach, and Michael Jugovac. News recommender systems – survey and roads ahead. *Information Processing and Management*, 54(6):1203 – 1227, 2018. ISSN 0306-4573. doi: https://doi.org/10.1016/j.ipm.2018.04.008. URL http://www.sciencedirect.com/science/article/pii/S030645731730153X.

Michal Kompan and Mária Bieliková. Content-based news recommendation. In Francesco Buccafurri and Giovanni Semeraro, editors, *E-Commerce and Web Technologies*, pages 61–72, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

Lei Li, Li Zheng, Fan Yang, and Tao Li. Modeling and broadening temporal user interest in personalized news recommendation. *Expert Systems with Applications*, 41(7):3168 – 3177, 2014. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2013.11.020. URL http://www.sciencedirect.com/science/article/pii/S0957417413009329.

Meilian Lu, Zhen Qin, Yiming Cao, Zhichao Liu, and Mengxing Wang. Scalable news recommendation using multi-dimensional similarity and jaccard-kmeans clustering. *J. Syst. Softw.*, 95:242–251, September 2014. ISSN 0164-1212. doi: 10.1016/j.jss.2014.04.046. URL http://dx.doi.org/10.1016/j.jss.2014.04.046.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA, 2013. Curran Associates Inc. URL http://dl.acm.org/citation.cfm?id=2999792.2999959.

Alejandro Montes-Garcia, Jose Maria Alvarez-Rodriguez, Jose Emilio Labra-Gayo, and Marcos Martinez-Merino. Towards a journalist-based news recommendation system: The wesomender approach. *Expert Systems with Applications*, 40(17):6735 – 6741, 2013. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2013.06.032. URL http://www.sciencedirect.com/science/article/pii/S0957417413004272.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of EMNLP*, pages 79–86, 2002.

Eli Pariser. *The filter bubble: What the Internet is hiding from you.* Penguin UK, 2011.

Maartje ter Hoeve, Anne Schuth, Daan Odijk, and Maarten de Rijke. Faithfully explaining rankings in a news recommender system. *arXiv preprint arXiv:1805.05447*, 2018.

Jeroen Van Barneveld and Mark Van Setten. *Designing Usable Interfaces for TV Recommender Systems*, pages 259–285. Springer Netherlands, Dordrecht, 2004. ISBN 978-1-4020-2164-0. doi: 10.1007/1-4020-2164-X_10. URL https://doi.org/10.1007/1-4020-2164-X_10.