Towards a Context-Sensitive Online Newspaper

Jeremy Jancsary jeremy.jancsary@ofai.at
Friedrich Neubarth friedrich.neubarth@ofai.at
Stephanie Schreitter stephanie.schreitter@ofai.at
Austrian Research Institute for Artificial Intelligence

Harald Trost harald.trost@meduniwien.ac.at
Section for Artificial Intelligence, Medical University of Vienna

ABSTRACT

We give a detailed account of our experiences in implementing a personalized online newspaper that draws—among other hints—on the context of the user. At the algorithmic core of our framework lies a machine learning model that incorporates numerous features of the eligible articles and the user's current situation. Some of the most important design decisions, however, concern the presentation of suggestions, the collection of explicit and implicit feedback, as well as diversity of the recommendations. We present numerical results obtained during the pilot phase of the project that address a number of these concerns and end with a discussion of open questions and future directions.

Author Keywords
Online Newspaper, User Profiling, Context-Awareness

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: Miscellaneous; H.3.4 Information Storage and Retrieval: Systems and Software—*User profiles and alert services*

INTRODUCTION

In the MAGNIFICENT project, we aim at the design and development of a personalized online newspaper that adapts to the preferences and the current context of a user. Our partner in this endeavor is the popular Austrian online newspaper DERSTANDARD.AT, which has been operating since February 1995 and was the first German-language newspaper to appear on the web. A major asset of the project is that we have access to a real-life user base amounting to 600,000 registered profiles. The approach we pursue is centered around individual users. We do not draw on collaborative filtering or behavior of the crowds, but rather seek to gain a better understanding of individual reading preferences. Towards this end, we investigate the relevant parameters of stories and the contribution of contextual information.

We know from studies concerned with printed media that the propositional contents of an article only account for about 40% of a story's satisfaction rating [11]. The remaining factors can be as diverse as readability concerns, writing style, the type of a story, visual complexity, or simply the use of proper photography. Moreover, factors that are only partially dependent on the articles themselves seem to play a major role. For instance, many readers seem to have a strong preference for articles regarding football during weekends. Presumably, this is related to the fact that most games of the Austrian football league happen on weekends. Such tendencies can only be captured by recognizing the crucial influence of the *context* at a given point in time. Indeed, capitalizing on user context has been recognized as one of the most promising directions in enhancing the state-of-the-art in recommender systems [1].

In earlier work [8], we presented a study regarding the contribution of a variety of characteristics of articles—and to some extent the user context—to user preference. However, the study was based on clickstream logs and comment postings that had been collected by DERSTANDARD.AT prior to implementation of MAGNIFICENT. Since then, the project has progressed well and a first pilot phase has been initiated. Over 140 real-life users of DERSTANDARD.AT were given access to an early version of the personalized newspaper that integrates seamlessly into the regular website. This first run with real-life users is designed to answer two key questions: First, can feedback be obtained from users implicitly by monitoring their behavior? Second, assuming that we are able to recommend those articles that most closely match a user's preference in the current situation, what degree of diversity should be injected in a list of k recommendations?

OVERVIEW

Our approach towards recommendation is centered around individual users, each of whom is associated with a separate *preference model*. This model determines the importance of salient characteristics of articles and the user context, which we refer to as *features*, to a particular user. The task of the system is then to *learn* a preference model that expresses its confidence in the preference of a user for a certain article given the current context. Recommendations are obtained by analyzing eligible articles in real-time and inspecting their confidence scores. The model is updated whenever we receive feedback about previous recommendations, either implicitly through monitoring of a user's behavior, or explicitly through ratings. Obviously, the features play a major role in our approach, since they determine the capability of the system to capture those aspects that are really important to a user. In the introduction, we already mentioned several characteristics of articles that we consider important. Hence, let us now discuss the role of *context*. We will give a few examples to illustrate the intricacy of the hypothesis space that governs user preference in news articles.

On the Role of Context

The most trivial example that illustrates the importance of context in the domain of news articles is *recency*. It is only rarely that a user will be interested in articles about events that happened years ago. As another example, observe that frequent readers of newspapers develop a routine that determines the *order* in which they flick through the different sections of the paper. Clearly, if at a given point in time, we want to generate suggestions that are in line with such progressions, there is a dependency on the *previously* read articles. This is part of the context of a user. Alternatively, one might want to generate a sequence of articles that are connected through a common theme [13]. Again, to generate suggestions for the next article to read, this requires knowledge of the previous articles. As yet another example, statistics of DERSTANDARD.AT show that people read different kinds of articles in the morning than in the evening. We shall now attempt to generalize these considerations into definitions of *context*, as far as MAGNIFICENT is concerned.

Attempt at a conceptual definition of context

From a conceptual point of view, we define the context of a user in terms of previous actions she has taken, or in terms of the current point in time. More generally, a user's context comprises any information that is not invariably defined through the articles that might be presented to the user.

Attempt at a computational definition of context

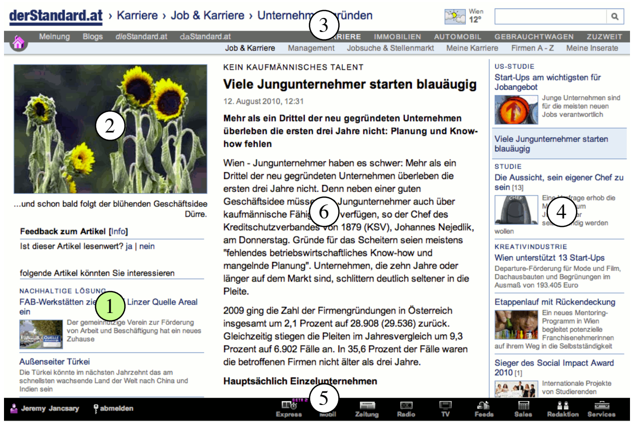
From a computational point of view, we consider the context of a user as those *features* which are not invariably defined in terms of particular articles. In other words, the context comprises those features that depend on the current point in time and hence cannot be pre-computed.

USER INTERFACE AND INTERACTION DESIGN

Before we dive into technical details, we will discuss the way recommendations are actually presented to a user.

What Is a Personalized Online Newspaper?

There are several conceivable ways of integrating recommendation technology into an online newspaper. For instance, one might opt for a dynamically generated front page. One argument in support of such a strategy is the fact that users have limited patience for locating information in deeply nested website hierarchies [10]. On the other hand, one might argue that the front page should be used to highlight news items of general interest. In a real-life online newspaper, the choice of articles on the front page is then perhaps



KEIN KAUFMÄNNISCHES TALENT

Viele Jungunternehmer starten blauäugig

12. August 2010, 12:31

Mehr als ein Drittel der neu gegründeten Unternehmen überleben die ersten drei Jahre nicht: Planung und Know-how fehlen

Wien - Jungunternehmer haben es schwer: Mehr als ein Drittel der neu gegründeten Unternehmen überleben die ersten drei Jahre nicht. Denn neben einer guten Geschäftsidee müsse Jungunternehmer auch über kaufmännische Fähig verfügen, so der Chef des Kreditschutzverbandes von 1879 (KSV), Johannes Nejedlik, am Donnerstag. Gründe für das Scheitern seien meistens "fehlendes betriebswirtschaftliches Know-how und mangelnde Planung". Unternehmen, die zehn Jahre oder länger auf dem Markt sind, schlittern deutlich seltener in die Pleite.

2009 ging die Zahl der Firmengründungen in Österreich insgesamt um 2,1 Prozent auf 28.908 (29.536) zurück. Gleichzeitig stiegen die Pleiten im Jahresvergleich um 9,3 Prozent auf 6.902 Fälle an. In 35,6 Prozent der Fälle waren die betroffenen Firmen nicht älter als drei Jahre.

Hauptsächlich Einzelunternehmen

...und schon bald folgt der blühenden Geschäftsidee Dürre.

Feedback zum Artikel [Info]

Ist dieser Artikel lesenwert? ja | nein

folgende Artikel könnten Sie interessieren

NACHHALTIGE LÖSUNG
FAB-Werkstätten zie Linzer Quelle Areal ein
Der gemeinnützige Verein zur Förderung von Arbeit und Beschäftigung hat ein neues Zuhause

Außenseiter Türkei
Die Türkei könnte im nächsten Jahrzehnt das am schnellsten wachsende Land der Welt nach China und Indien sein

US-STUDIE
Start-Ups am wichtigsten für Jobangebot
Junge Unternehmen sind für die meisten neuen Jobs verantwortlich

Viele Jungunternehmer starten blauäugig

STUDIE
Die Aussicht, sein eigener Chef zu sein [13]
Eine frage erhob die M um J er se dig werden wollen

KREATIVINDUSTRIE
Wien unterstützt 13 Start-Ups
Departure-Förderung für Mode und Film, Dachausbauten und Begrünungen im Ausmaß von 193.405 Euro

Etappenlauf mit Rückendeckung
Ein neues Mentoring-Programm in Wien begleitet potenzielle Franchisenehmerinnen auf ihrem Weg in die Selbstständigkeit

Sieger des Social Impact Award 2010 [1]
Internationale Projekte von Studierenden

Figure 1. Position of the Magnificent pane (1) within the website

best left to the editorial staff. A different idea we played with initially was to implement a completely separate application that follows the *next button* metaphor, whereby the user is only suggested a single article at any given time. The article would then be displayed immediately—without any explicit choice by the user—and could simply be *skipped* if undesired. Such an interaction model is frequently found in music recommendation systems.[1] However, in discussions with DERSTANDARD.AT, it was found that a completely separate interface was unlikely to attract significant attention by the readers of the website.

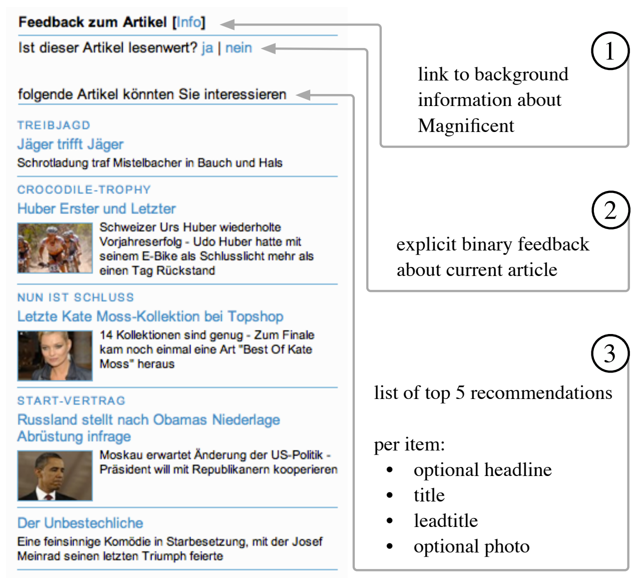
Enhancing the Regular Article View

In light of the above considerations, and technical restrictions on the placement of dynamic contents, the decision was made to enhance the regular *article view* by a dynamically generated list of current recommendations. This is illustrated in Figure 1. The dynamic content generated by our recommendation system (1) is displayed to the left of the main body of an article (6), just below optional photographs or illustrations (2). The view also comprises control elements such as a navigation bar (3), a tool bar (5) and a separate pane that lists related articles (4). Except for the MAGNIFICENT pane (1), the elements are unchanged from the usual presentation of articles on DERSTANDARD.AT.

The Magnificent Pane

Figure 2 shows a magnified view of the MAGNIFICENT pane that allows for a closer look. At the very top of the pane (1), there is a link to background information about our project. This article explains in some detail the purpose of the project, how to use the new functionality, and what to expect from the system. Underneath this link, the user is presented with the opportunity of providing explicit feedback regarding the current article (2). We restrict the possible user input to binary feedback in response to the question "Is this article worth reading?". Two comments are in order here: First, we chose binary feedback over a more sophisticated scheme in order to avoid arbitrariness of the ratings. Among other considerations, this is motivated by the experience of YouTube, who

[1]One example is the last.fm player at `http://www.last.fm`.



Feedback zum Artikel [Info]

Ist dieser Artikel lesenwert? ja | nein

folgende Artikel könnten Sie interessieren

TREIBJAGD
Jäger trifft Jäger
Schrotladung traf Mistelbacher in Bauch und Hals

CROCODILE-TROPHY
Huber Erster und Letzter
Schweizer Urs Huber wiederholte Vorjahreserfolg - Udo Huber hatte mit seinem E-Bike als Schlusslicht mehr als einen Tag Rückstand

NUN IST SCHLUSS
Letzte Kate Moss-Kollektion bei Topshop
14 Kollektionen sind genug - Zum Finale kam noch einmal eine Art "Best Of Kate Moss" heraus

START-VERTRAG
Russland stellt nach Obamas Niederlage Abrüstung infrage
Moskau erwartet Änderung der US-Politik - Präsident will mit Republikanern kooperieren

Der Unbestechliche
Eine feinsinnige Komödie in Starbesetzung, mit der Josef Meinrad seinen letzten Triumph feierte

1
link to background information about Magnificent
2
explicit binary feedback about current article
3
list of top 5 recommendations
per item:
• optional headline
• title
• leadtitle
• optional photo

Figure 2. Contents of the Magnificent pane

switched to binary feedback since more fine-grained ratings by users were found to be of questionable accuracy (discussed at the ACM RecSys 2010 Industry Panel, [6]). Second, the question that invites the feedback must be expected to have significant influence. We put considerable effort into the phrasing of the question, which hopefully dampens the tendency of users to actually express their feelings about opinions or events *described* in the article. For instance, just because a reader strongly disagrees with the opinion expressed in a commentary, this does not mean that the article is not interesting to the user—quite on the contrary. Finally, the main real estate of the pane is consumed by a list of the current top 5 recommendations. We feel that 5 is a reasonable number, since it enables comprehension of the list on first glance, while allowing for a certain degree of diversity.

Interaction Model

The user can navigate freely within the website and has a number of options in order to do so. In particular, one need not choose from the list of recommendations. A positive aspect of this noncommittal way of interacting with the system is that the user can always navigate to particular sections of the newspaper on her own account if the selection of recommended articles is found to be too narrow or not to exhibit sufficient diversity. In this way, the user can bring those articles to the attention of the system that had previously been classified as undesirable. To some degree, this is an inherent protection against the tendency of the preference model to suggest a particular kind of article as it gains a more mature view of the preference of the user. Of course, sufficient diversity can also be ensured through technical means—we discuss one such strategy later on.

On the negative side, the cognitive load our interface imposes on the user is significant. For instance, we were slightly worried about establishing a clear distinction between the Magnificent pane (Figure 1, element 1) and the list of related articles (Figure 1, element 4) in the perception of the user. So far, our strategy in this matter has been to explicitly highlight the difference in the background information: the list of related articles contains items that are connected to the currently viewed article via a common theme and is manually edited by editorial staff; the Magnificent pane, on the other hand, contains automatically generated suggestions that need not be related to the currently viewed article, but rather match the general preference of a user.

TECHNICAL REALIZATION

We will next briefly describe the requirements towards our software stack and the resulting architecture.

Requirements

Our recommender system is realized in the setting of a real-life online newspaper that needs to serve a substantial number of page requests. Hence, several technical requirements towards the software stack emerged.

Loose coupling: The existing backend software of the site must be loosely coupled to the implementation of MAGNIFICENT. All communication shall occur through clearly defined interfaces.

Response times: Long response times of the recommender system must not affect page delivery times of the online newspaper adversely.

Dependability: The functionality of the recommender system must be implemented such that it does not impact the dependability of the website. In particular, failure of the recommender system must not have consequences for regular operation of the online newspaper.

Maintenance: The recommender system is expected to store and back up any data it collects autonomously and in a format that is amenable to data recovery.

Data retention: Any information collected about users must be stored in an anonymized fashion such that it is impossible to infer the identity of a user from the records.

All of these requirements had a major impact on the design of the system architecture.

Software Architecture

MAGNIFICENT is designed as multi-tiered architecture. Figure 3 shows the control flow of a single page request. When a user wishes to view an article, an HTTP request is first generated by the browser. This request is in turn picked up by the web server of DERSTANDARD.AT, which delivers the HTML contents of the requested article. Embedded in the HTML code of the page is JavaScript code that subsequently triggers an asynchronous request to the backend of DERSTANDARD.AT. The purpose of this request is to fetch the contents of the MAGNIFICENT pane. The backend of DERSTANDARD.AT in turn contacts the MAGNIFICENT system by issuing an XMLRPC request, which is answered in terms of a list of the current top k recommendations. Once the answer is received at the backend of DERSTANDARD.AT,

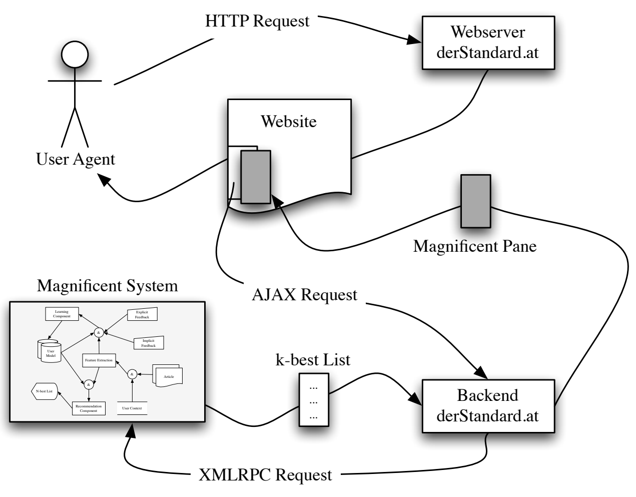


Figure 3. Control flow of a single page view

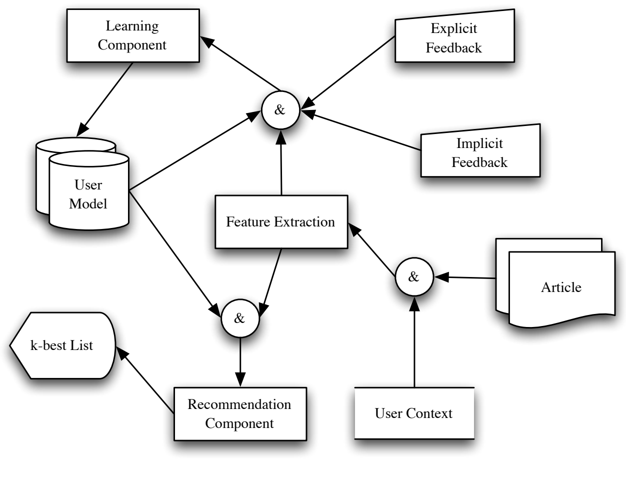


Figure 4. Interaction between main system components

the HTML contents of the MAGNIFICENT pane is prepared and returned to the browser of the user. Finally, the browser injects the contents into the DOM tree of the web page such that the MAGNIFICENT pane is displayed.

Fault tolerance
One big advantage of this architecture is its inherent fault-tolerance. All major components can be geographically separated and only communicate through well-defined interfaces. Since the contents of the MAGNIFICENT pane is fetched asynchronously, slow response times—or even failure—of the MAGNIFICENT system do not impede the regular page delivery. It also enables extremely loose coupling between the implementation of MAGNIFICENT and the existing infrastructure of DERSTANDARD.AT: a single XMLRPC interface is all the two systems share. Figure 3 only shows the control flow for a regular page view, which in turn invokes the *recommendedArticles* method in the XMLRPC interface. However, the interface currently defines methods for two more essential tasks: *addArticle*, which can be used by the backend of DERSTANDARD.AT to notify the MAGNIFICENT system of newly added news articles to analyze and add to the database; and *processFeedback*, which is used to pass on explicit feedback by the user. The three methods are all that is required for the MAGNIFICENT system to maintain its own state of the available articles and the preference models of the users of DERSTANDARD.AT.

Scalability
Another advantage is that the high level of abstraction makes the system amenable to distributed processing of requests. From a computational point of view, the most costly operation is currently the ranking of articles triggered by a request to *recommendedArticles*. Since the recommendations are computed separately for each user and draw on the current context, this ranking cannot be cached but rather must be performed in real-time. However, given the current architecture, nothing prevents us from spreading these requests over a cluster of servers behind the scene. For instance, the requests could be mapped onto the servers depending on the user. Each cluster node would then maintain a complete database of articles to choose from, but only the preference models of a particular subset of users. The whole process can even be made completely transparent to the backend of DERSTANDARD.AT by introducing a central load-balancing node that implement the XMLRPC interface and simply delegates the user-dependent requests to the proper cluster node, while relaying the general requests to all nodes.

CORE OF THE RECOMMENDER SYSTEM

We now turn to the algorithmic core of our recommender system. The main components involved concern computation of the features, learning of preference models, and generation of recommendations. A brief overview of the interplay between these components is given in Figure 4.

Feature Extraction

The feature extraction component jointly maps a given article and the current context of a user to salient characteristics that we generally deem relevant to the preferences of users. We are not yet concerned with judging the relative importance of these characteristics to a particular user. However, the features that are extracted in this step partially determine the power of our hypothesis space. In previous work [8], we motivated and determined of number of such characteristics. Moreover, we recently conducted a psychological study in order to gain further insights into the factors that really drive user preference in the domain of news articles. We will leave a discussion of these aspects to future work. Here, we just want to give an overview of the features currently implemented in the MAGNIFICENT system and discuss their role in the algorithmic core of our system. Formally, at point t in time, we define the global feature map as

$$\mathbf{f}(\mathbf{a}, \mathbf{c}_t^u) = \sum_i \mathbf{f}_{\text{inv}}^i(\mathbf{a}) + \sum_j \mathbf{f}_{\text{ctx}}^j(\mathbf{a}, \mathbf{c}_t^u), \qquad (1)$$

where we use $\mathbf{a} \in \mathcal{A}$ to denote an arbitrary article, $\mathbf{c}_u^t \in \mathcal{C}$ to denote the current context of user u at time t, and $\mathbf{f}_{\text{inv}}^i$ and $\mathbf{f}_{\text{ctx}}^j$ refer to the map defined by a particular invariant or context-sensitive feature, respectively. Each feature function $\mathbf{f}^i_{\text{inv}} : \mathcal{A} \mapsto \mathbb{R}^n$ and $\mathbf{f}^j_{\text{ctx}} : \mathcal{A} \times \mathcal{C} \mapsto \mathbb{R}^n$ maps its arguments to the global feature space $\mathbb{R}^n$, such that the global feature map $\mathbf{f} : \mathcal{A} \times \mathcal{C} \mapsto \mathbb{R}^n$ is obtained as a sum of these vectors. In practice, each feature function returns one or a few numeric scores and is responsible for mapping these to the proper indices of $\mathbb{R}^n$. The other components of the vector returned by this feature function are then zero.

Invariant features

These are purely defined in terms of an article itself and do not draw on the current context. Hence, they are identical for each user, independent of the current point in time, and can be pre-computed and cached effectively. MAGNIFICENT currently extracts the following invariant features:

Flesch Reading Ease: A single numeric score is computed that determines the readability of an article according to the well-known method of Flesch and Kincaid [4].

Text Likelihood: We apply a unigram language model determined over the whole body of articles of DERSTANDARD.AT in order to compute the likelihood of a given article relative to the whole corpus. The idea is again to obtain a measure of readability.

Affective Dictionary Ulm: We use a particular dictionary [7] in order to compute scores related to affective aspects such as anger and contentedness. These are combined into an overall measure for the amount of affective vocabulary.

Gottschalk-Gleser Score: We draw here on a set of language patterns determined by Gottschalk and Gleser [5] that can be used to determine a measure of the amount of angst or aggressiveness contained in utterances.

Subjectivity: Beforehand, we trained a simple classifier on a hand-selected set of articles that was split into "subjective" and "non-subjective" examples. This classier is used to assign subjectivity scores to new articles.

Category: Maps an article to binary indicators that denote which sections or categories of the newspaper the present article belongs to. We use a subsumption hierarchy of such categories to improve generalization.

Number of Media: The number of media files, such as photographs, illustrations or videos embedded in an article.

Source: Maps to binary indicators for the source of the present article. The source can be the editorial staff or a news agency, to name two examples.

Text Length: The length of an article, measured as the number of tokens in the body of the text.

Through normalization and scaling, we ensure that all numeric scores are of the same magnitude.

Context-sensitive features

These features depend on one or more articles, but also on previous actions of the user or the current point in time. Hence, they have to be computed in real-time.

Current Time: Binary features that encode the day of the week and a real value encoding the current hour. We also formed conjunctive features with the *category* of the present article, as in *day=sunday&category=soccer*.

Category Flow: Binary features that capture the transition of the user between different categories or sections of the newspaper. The goal is to capture typical progressions, such as from entertainment to sports. Again, we use a hierarchy of categories to foster generalization.

Content Flow: TF-IDF weighted cosine similarity between the previous article the user liked and the present article. The idea is to achieve a smooth progression between recommendations.

Preference Flow: Binary features that encode whether the user liked or disliked the previously suggested articles. These features create a "bias" that reflects the current satisfaction of the user and adjusts the "threshold" for an article to receive a positive score.

Cosine Similarity: TF-IDF weighted cosine similarity of the candidate article and regularly adapted term models of those articles the user liked and those she didn't like. One may debate whether this feature actually draws on the context. In our view it does, because it depends on the articles that were *previously* read by the user.

Already Seen: Binary feature that indicates whether the user has already seen the current article. This enables the system to learn that a user does not want to be suggested the same article multiple times.

Learning Component

We maintain a separate *preference model* $\mathbf{w}^u \in \mathbb{R}^n$ for each user u. In essence, this model specifies a single weight or coefficient for each component of the feature map $\mathbf{f}$. This way, the importance of each feature to a particular user is expressed. At a given point in time t, the score or suitability of a particular article $\mathbf{a}$ for a user u is then given in terms of the linear model

$$s(\mathbf{x}^u_t; \mathbf{w}^u_t) = \mathbf{x}^u_t \cdot \mathbf{w}^u_t, \qquad (2)$$

where we use $\mathbf{x}^u_t := \mathbf{f}(\mathbf{a}, \mathbf{c}^u_t)$ for brevity. The central task of the learning component is now to estimate $\mathbf{w}^u$ in an online fashion, such that at any given point in time, those articles obtain the highest score that most closely match a user's preferences in the current context. Information about these preferences is conveyed to the learning component in terms of *explicit* or *implicit* binary feedback by the user. More formally, at time t, we receive a label $y^u_t \in \{-1, +1\}$ that encodes whether a user u liked a certain article $\mathbf{a}$ given the current context $\mathbf{c}^u_t$.

There are now several ways of adjusting the preference model $\mathbf{w}^u$ based on that feedback. In earlier work [8], we experimented with the classic Perceptron [12], the family of Passive-Aggressive online algorithms [2] and the Forgetron [3]. The latter can also incorporate a Mercer kernel in order to overcome the linear decision boundary of (2). Ultimately, we found that a particular instance of the Passive-Aggressive family, which we abbreviate as PA-II for short, gave the best results in our setting. To describe how the corresponding update of the preference model works, we need to introduce one more auxiliary concept. We define the *loss* incurred by our current model $\mathbf{w}_t^u$ on an article given feedback y_t^u as

$$\ell(\mathbf{x}_t^u, y_t^u; \mathbf{w}_t^u) = \begin{cases} 0 & y_t^u(\mathbf{x}_t^u \cdot \mathbf{w}_t^u) \geq 1 \\ 1 - y_t^u(\mathbf{x}_t^u \cdot \mathbf{w}_t^u) & \text{otherwise} \end{cases}.$$

The update performed by PA-II now seeks the smallest change to the model $\mathbf{w}_t^u$ that ensures the current article receives a score of at least $+1$ if the feedback is positive, and at most -1 if negative. To protect against unduly large changes caused by noise, this constraint is not strictly enforced, but constraint violation is penalized via a slack term instead. Formally, the update solves the problem

$$\mathbf{w}_{t+1}^u = \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^n} \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t^u\| + C\xi^2 \text{ s.t. } \ell(\mathbf{x}_t^u, y_t^u; \mathbf{w}_t^u) \leq \xi.$$

By forming the Lagrangian of this constrained optimization problem and deriving the stationary conditions, one obtains a closed-form solution given by

$$\mathbf{w}_{t+1}^u = \mathbf{w}_t^u + \frac{\ell(\mathbf{x}_t^u, y_t^u; \mathbf{w}_t^u) y_t^u \mathbf{x}_t^u}{\|\mathbf{x}_t^u\|^2 + \frac{1}{2C}}. \quad (3)$$

It is easy to see that if we incur a loss of 0, the current model remains unaffected. The hyperparameter C controls the penalization of the constraint violation; larger values of C result in stricter enforcement of the constraint and hence more aggressive updates. Previously [8], we found PA-II to be robust in this respect, and we simply use $C = 100$ in practice.

Recommendation Component

Given the preference model $\mathbf{w}^u$ of a user, which is maintained by the learning component, we can compute the suitability of each article in our database in the current context. Generally, according to this model, a negative score implies that an article does not match a user's preferences, whereas a positive score indicates a suitable article. In principle, one can then choose from a variety of strategies in order to generate k recommendations. We just mention two of these here.

Top-Rated Strategy

Perhaps the most obvious strategy is to simply pick a set of k articles that attains the highest score, that is

$$\vec{\mathbf{a}}_t^u = \operatorname*{argmax}_{\{\mathbf{a}\} \in \mathcal{A}^k} \left\{ \sum_{\mathbf{a}} \mathbf{f}(\mathbf{a}, \mathbf{c}_t^u) \cdot \mathbf{w}_t^u \right\}. \quad (4)$$

In a practical implementation, it is infeasible to evaluate the score of all articles in the database for each request. The first restriction we impose is that articles must not be older than a week.[2] Moreover, we do not consider articles that have already been rated by the user explicitly. The remaining set $\mathcal{E}_t$ is what we call the eligible set of articles at a given point in time t. In our current implementation, we use an approximation whereby a sample of 100 articles is drawn from $\mathcal{E}_t$, which is then ranked according to the score of the articles. The k top scorers in the sample are returned.

[2]Admittedly, this is somewhat restrictive—but it is in line with the intended use of the system.

Greedy Strategy

One potential drawback of the above scheme is that the set of k top scorers can be expected to be rather homogeneous. A possible workaround is to give up the optimality criterion and instead simply demand that each of the k recommended articles should achieve a positive score (that is, be classified as a suitable article). This turns (4) into a constraint satisfaction problem:

$$\vec{\mathbf{a}}_t^u = \operatorname*{argmax}_{\{\mathbf{a}\} \in \mathcal{A}^k} 0 \quad \text{s.t.} \quad \mathbf{f}(\mathbf{a}, \mathbf{c}_t^u) \cdot \mathbf{w}_t^u > 0 \ \text{ for all } \mathbf{a}. \quad (5)$$

In practice, we use a greedy algorithm to solve the problem approximately. We simply draw a single article at a time from $\mathcal{E}_t$ and evaluate its score; we proceed in this manner until a feasible set of cardinality k has been established. If this does not happen within at most 100 evaluations, we fall back to the top-rated strategy and pick those k evaluated articles that attained the highest scores (and thus the lowest constraint violations). One expected property of this scheme is that it results in greater diversity of the k selected articles.

Feedback Elicitation

A crucial ingredient required by the learning component is the feedback y_t^u about an article given the current context.

Explicit Feedback

Such feedback can be provided by the user in an explicit manner by clicking on the respective feedback controls in the Magnificent pane (see Figure 2). One drawback is that this requires some effort by the user. Moreover, we assume that explicit feedback can be subject to distortion if a user is unable to properly reflect on her true preferences.

Implicit Feedback

Previous studies (e.g. [9]) have shown that to some degree, the preference of a user is correlated with measurable statistics of her behavior on a website, such as page viewing time or scrolling activity. However, such implicit feedback can be heavily distorted. First, the actual correlation between behavior and preference varies from user to user. Second, measurement of the behavior itself can be distorted and is subject to external influences. For instance, a user may leave open the browser window while picking up a phone call. The page viewing time is then measured incorrectly. Nonetheless, we were interested if such an approach might improve results in our setting. Page viewing time can be determined rather easily in our architecture, so we tried to build a robust model of implicit feedback around that information. Our current procedure works as follows: First, we discard any page view that is shorter than 5 seconds or longer than 5 minutes as an outlier. While somewhat arbitrary, these values were determined upon inspection of clickstream logs and also seem intuitive: 5 seconds are hardly long enough to even get an overview, while viewing times above 5 minutes indicate some external influence. Aside from the strict rules regarding outliers, we maintain statistics of the page viewing times of a user, which we employ as follows: Any value below the first quartile is counted as negative feedback, while any value above the third quartile is considered positive feedback. We discard values in between as uninformative.

Table 1. Basic statistics of the pilot phase

Number of active users:	148
Duration of experiment:	4 weeks
Number of cataloged articles:	31069
Number of distinct articles viewed:	5484
Median page/article viewing time:	47s
Total number of recommendations served:	41649
Total number of positive feedback clicks:	2481
Total number of negative feedback clicks:	331

RESULTS OF THE PILOT PHASE

We next discuss some of the lessons learnt from the first pilot phase of our project. We focus on two particular aspects: First, we wanted to gain insights about the impact of the recommendation strategy in use (top-rated versus greedy). Second, we were interested in the utility of implicit feedback.

Choice of Users

We first determined a set of users of DERSTANDARD.AT that were invited to participate in the pilot phase. The users were chosen from the database of registered users, based on criteria relating to their reading behavior. The criteria were chosen to the effect that users with an excessively large amount of comment postings were avoided. Moreover, we filtered out those users that either seemed to be interested in all sections of the online newspaper, or only interested in a very limited number of topics. The reasoning behind this choice is that we were principally interested in users with a broad but clearly cut range of interests to whom the recommender system could actually provide additional value. Among the qualifying users, we chose based on the number of page impressions in previous months to ensure that the parcipants were going to make frequent use of the system. Overall, 1150 users were invited to participate in the pilot phase on an opt-in basis. Out of these, 148 users actually chose to enable the new functionality. This amounts to a response rate of just above 12%. The original goal of obtaining on the order of 100 active users was thus achieved.

System Setup

We compared four different system configurations, based on the cross-product space of the two recommendation strategies (*top-rated* vs. *greedy*) and the two feedback collection strategies (*explicit only* vs. *explicit and implicit*). The users were assigned to those four systems in a round-robin fashion, such that the number of participants was uniformly spread. All features described in this paper, both invariant and context-sensitive, were enabled in all configurations. The MAGNIFICENT system was run on a single Intel Core 2 Quad machine with four 2.4GHz CPU cores and 8 GB of main memory.

Results

Let us now consider the results obtained after 4 weeks of operation. Table 1 shows some basic statistics of the experiment. In general, it is a difficult task to define meaningful measures that reflect the actual performance of our system, in particular since users can access articles in many ways other than using the recommender system.

Table 2. Precision and recall achieved by the system configurations. The standard deviation over users is indicated using the $\pm$ sign.

	Explicit Feedback		Explicit & Implicit	
	Prec. (%)	Rec. (%)	Prec. (%)	Rec. (%)
Top-R.	2.09$\pm$1.21	5.68$\pm$3.59	2.81$\pm$3.82	7.74$\pm$10.3
Greedy	1.31$\pm$0.75	5.46$\pm$3.42	2.10$\pm$1.14	7.97$\pm$6.60

Precision and Recall

One aspect that we found interesting was to consider the performance of the four competing system configurations in terms of their *precision* and their *recall*. Although our setting deviates slightly from the usual context in which the concepts of precision and recall apply, we found that the following definitions made sense intuitively: We think of *precision* as the fraction of suggested articles that were actually read by a user at some later point in time, out of all articles that were recommended by our system. Analogously, we define *recall* as the fraction of articles that were suggested by the system prior to the user reading them, out of all articles the user read during the four weeks. The results in terms of these two performance figures are shown in Table 2, averaged over all users assigned to the respective system configurations. While the numbers are relatively low, one should keep the following facts in mind: First, the largest share of clicks goes directly to articles that are linked on the front page of the newspaper; in these cases, the recommender system couldn't possibly have recommended the articles prior to the user reading them. Second, the search space of eligible articles is enormous, and so a recommendation that has not been read, or a read article that has not been recommended, does not necessarily imply bad system performance.

In any case, the most interesting aspect here is the *relative* performance of the different system configurations. The first effect that is clearly visible is that the *top-rated* strategy leads to a slightly higher precision. This is to be expected, as the computed recommendations are less diverse. On the other hand, the *greedy* strategy should have increased the recall, which our results do not seem to confirm. The second clear result is that implicit feedback is helpful, regardless of the recommendation strategy. However, one should be careful in drawing general conclusions, since the standard deviation over users is rather high. The results vary a lot depending on the effort a user actually put into *helping* the system, in terms of explicit feedback, or in terms of consistent behavior. For instance, for some users, we can report precision/recall scores as high as 17.2/45.3 percent.

Trends regarding feedback

We were also interested in the behavior of users regarding explicit feedback. In particular, we wanted to know if explicit feedback declines over time, and if there are substantial differences between the different system configurations. The results of our analysis are shown in Figure 5. In general, there is a rather clear trend for the number of explicit feedback clicks to decline over time. This development might be attributed to two different causes, however: First, the necessity of feedback might decrease as the system learns a more

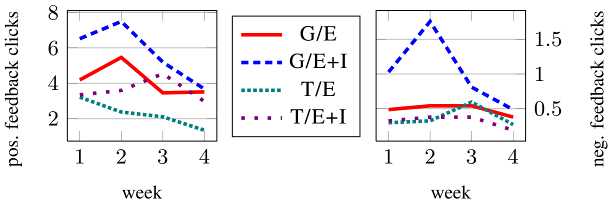


Figure 5. Average number of explicit feedback clicks per user—trend over 4 weeks. Left: positive feedback. Right: negative feedback. Both: G stands for Greedy, T for Top-Rated, E for Explicit Feedback, E+I for Explicit&Implicit Feedback.

accurate model of the preferences of a user. Second, interest in the newly introduced recommender system as a whole might decline over time. Another conclusion of ours is that the *greedy* recommendation strategy seems to trigger more feedback clicks. We attribute this effect to the more diverse nature of the recommendations generated by this strategy.

Page Viewing Times

A final observation that we found rather interesting concerns the distribution of page viewing times. A histogram is shown in Figure 6. It is rather remarkable that viewing times between 0 and 10 seconds seem to be most common. Clearly, an average news article cannot be comprehended in such a short time span. On the other hand, 10 seconds may suffice to gain a first impression of an article, after which the readers seem to move on immediately if the article is not found to be of interest.

DISCUSSION AND OUTLOOK

Based on our experimental findings, it seems that implicit feedback that is collected in terms of page viewing times can effectively improve the performance of our recommender system. We attribute this to the fact that the context-sensitive features we employ require a considerable amount of feedback in order to realize their potential benefit: the parameter space is expanded substantially, and without further guidance as to the choice of those parameters, the capability of modeling more powerful hypotheses is essentially useless. As far as the choice of the recommendation strategy is concerned, such a clear statement cannot be made. In terms of the precision of the resulting recommender system, it seems to be preferable to choose those items that receive the highest score according to the preference model. On the other hand, we found subjectively that this strategy produces rather homogeneous sets of recommendations after only a short amount of time. The greedy recommendation strategy we presented—which relaxes the optimality constraint—does not expose this behavior, but fares slightly worse in terms of precision. As far as recall is concerned, the two strategies are more or less tied.

We look forward to a second pilot phase involving a considerably larger number of users. The server load caused by the 148 users during the first four weeks was minimal, and we assume that 10,000 users can be handled readily without further performance optimizations or additional servers. This will allow us to split the users into even more groups in order to evaluate a larger number of choices, in particular with respect to the selection of features. Furthermore, it should enable us to gain results that are statistically significant.

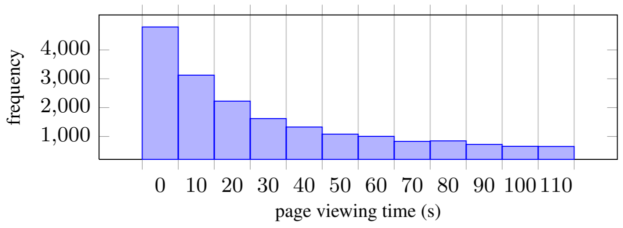


Figure 6. Histogram of page viewing times (over all configurations)

ACKNOWLEDGMENTS

We thank our anonymous reviewers. MAGNIFICENT is supported by FIT-IT grant #819,567 of the FFG. The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry for Transport, Innovation, and Technology.

REFERENCES

1. G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE TKDE*, 17:734–749, 2005.
2. K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *JMLR*, 7:551–585, 2006.
3. O. Dekel, S. Shalev-Shwartz, and Y. Singer. The Forgetron: A kernel-based perceptron on a budget. *SIAM Journal on Computing*, 37:1342–1372, 2008.
4. R. Flesch. A new readability yardstick. *Journal of Applied Psychology*, 32:221–233, 1948.
5. L. A. Gottschalk and G. C. Gleser. *The measurement of psychological states through the content analysis of verbal behavior*. University of California, 1969.
6. I. Guy, A. Jaimes, P. Agulló, P. Moore, P. Nandy, C. Nastar, and H. Schinzel. The RecSys 2010 industry panel. In *4th ACM Conference on Recommender Systems*, 2010.
7. M. Hölzer, N. Scheytt, and H. Kächele. Das "Affektive Diktionär Ulm" als eine Methode der quantitativen Vokabularbestimmung. In *Textanalyse – Anwendungen der computerunterstützten Inhaltsanalyse*, pages 185–212. Westdeutscher Verlag, 1992.
8. J. Jancsary, F. Neubarth, and H. Trost. Towards context-aware personalization and a broad perspective on the semantics of news articles. In *4th ACM Conference on Recommender Systems*, pages 289–292, 2010.
9. T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Transactions on Information Systems*, 25(2), 2007.
10. J. Palmer. Designing for web site usability. *Computer*, 35:102–103, 2002.
11. Readership Institute. Inside satisfaction: What it means, how to increase it. `http://www.readership.org/content/editorial/data/elements_of_satisfaction.pdf`, 2002.
12. F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.
13. D. Shahaf and C. Guestrin. Connecting the dots between news articles. In *16th ACM KDD Conference on Knowledge Discovery and Data Mining*, 2010.