Personalized News Recommendation with Context Trees

Florent Garcin Christos Dimitrakakis Boi Faltings

Artificial Intelligence Lab
Ecole Polytechnique Fédérale de Lausanne
Switzerland
firstname.lastname@epfl.ch

ABSTRACT

The proliferation of online news creates a need for filtering interesting articles. Compared to other products, however, recommending news has specific challenges: news preferences are subject to trends, users do not want to see multiple articles with similar content, and frequently we have insufficient information to profile the reader.

In this paper, we introduce a class of news recommendation systems based on context trees. They can provide high-quality news recommendations to anonymous visitors based on present browsing behaviour. Using an unbiased testing methodology, we show that they make accurate and novel recommendations, and that they are sufficiently flexible for the challenges of news recommendation.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering*

Keywords

online recommender system, personalization, news

1. INTRODUCTION

The first recommender systems were originally designed for news forums. Since then, they have been used with considerable success for products such as books and movies, but have found surprisingly little application in recommending news articles, due to the unique challenges of the area.

When users are identifiable as regular visitors to a news website, techniques from product recommendation can be adapted [8, 11]. However, most websites operated by individual newspapers do not have a strong base of electronic subscribers. Visitors to these websites are casual users, often accessing them through a search engine, and little is known about them except what can be gathered through an ephemeral browsing history.

RecSys'13, October 12–16, 2013, Hong Kong, China.

http://dx.doi.org/10.1145/2507157.2507166.

Figure 1: A story with dynamic recommendations on the right side, and manually-generated recommendations on the bottom left (red-dashed areas).

The main page of a news website is already a set of recommended articles, which simultaneously addresses the needs of many users (Fig. 1). More specific recommendations are sometimes available to readers of individual articles. There are two shortcomings to this strategy: first, recommendations are usually edited manually, and second, they only consider the last article read. Our goal is to construct recommendations automatically and use the complete browsing history as a basis for giving personalized recommendations.

In principle, common recommender techniques such as collaborative filtering could be applied to such a task, and have been adapted to temporal sequences [28, 24, 19]. However, they face several challenges specific to news. First, news are rapidly evolving: new stories and topics appear and disappear quickly, and old news are no longer interesting. Second, recommendations should provide added value, and not just consist of the most popular stories that the reader would have already seen on the front page.

To address these issues, we propose a class of *online* recommendation algorithms based on *Context Trees (CT)*, which provide recommendations and are updated fully incrementally. A CT defines a *partition tree* organised in a hierarchy

of increasingly precise partitions of a space of contexts. We consider as context the sequence of articles, the sequence of topics, or the distribution of topics. Each node in the tree is called *context* and corresponds to a set of sequences or topic distributions within a partition. The main idea is to give context-dependent recommendations, with contexts becoming progressively finer-grained deeper in the tree.

To make actual recommendations, we associate a set of prediction models, called *experts*, with each context. Their predictions are combined to make recommendations. We tailor our expert models to specifically take into account the idiosyncrasies of news. In particular, our expert models take into account the *popularity* and *freshness* of news items.

Using an unbiased testing methodology, emulating the process involved in implementing a system on a real website, we show that CT recommendations have state-of-the-art performance both with respect to prediction accuracy and to recommendation novelty, which is crucial for news articles since users want to read stories they do not know.

2. RELATED WORK

In general, there are two classes of recommender systems: collaborative filtering [25], which use similar users' preferences to make recommendations, and content-based systems [16], which use content similarity of news items.

The Grouplens project is the earliest example of collaborative filtering for news recommendation, applied to newsgroups [21]. News aggregation systems such as Google News [8] also implement such algorithms. Google News uses the probabilistic latent semantic indexing and MinHash for clustering news items, and item covisitation for recommendation. Their system builds a graph where the nodes are the stories and the edges represent the number of covisitations. Each of the approaches generates a score for a given news, aggregated into a single score using a linear combination.

Content-based recommendation is more common for news personalization [5, 1, 11]. NewsWeeder [12] is probably the first content-based approach for recommendations, but applied to newsgroups. NewsDude [5] and more recently YourNews [1] implemented a content-based system.

It is possible to combine the two types in a hybrid system [7, 14, 13]. For example, Liu et al. [14] extend the Google News study by looking at the user click behaviour in order to create accurate user profiles. They propose a Bayesian model to recommend news based on the user's interests and the news trend of a group of users. They combine this approach with the one by Das et al. [8] to generate personalized recommendations. Li et al. [13] introduce an algorithm based on a contextual bandit which learns to recommend by selecting news stories to serve users based on contextual information about the users and stories. At the same time, the algorithm adapts its selection strategy based on user-click feedback to maximize the total user clicks.

We focus on a class of recommender systems based on context trees. Usually, these trees are used to estimate Variable-order Markov Models (VMM). VMMs have been originally applied to lossless data compression, in which a long sequence of symbols is represented as a set of contexts and statistics about symbols are combined into a predictive model [22]. VMMs have many other applications [2].

Closely related, variable-order hidden Markov models [26], hidden Markov models [17] and Markov models [18, 23, 9] have been extensively studied for the related problem of click prediction. These models suffer from high state complexity. Although techniques [27] exist to decrease this complexity, multiple models have to be maintained, making these approaches not scalable and not suitable for online learning.

Few works [28, 24, 19] apply such Markov models to recommender systems. Zimdars et al. [28] describe a sequential model with a fixed history. Predictions are made by learning a forest of decision trees, one for each item. When the number of items is big, this approach does not scale. Shani et al. [24] consider a finite mixture of Markov models with fixed weights. They need to maintain a reward function in order to solve a Markov decision process for generating recommendations. As future work, they suggest the use of a context-specific mixture of weights to improve prediction accuracy. In this work, we follow such an approach. Rendle et al. [19] combine matrix factorization and a Markov chain model for baskets recommendation. The idea of factoring Markov chains is interesting and could be complementary to our approach. Their limitation is that they consider only first-order Markov chains. A bigger order is not tractable because the states are baskets which contain many items.

3. PRELIMINARIES

Because of the sequential nature of news reading, it is intuitive to model news browsing as a k-order Markov process [24]. The user's state can be summarised by the last k items visited, and predictions can be based only on this information. Unfortunately, it is not clear how to select the order k. A *variable-order Markov model* (VMM) alleviates this problem by using a context-dependent order. In fact, VMM is a special type of context-tree model [2].

There are two key ideas behind a CT recommender system. First, it creates a hierarchy of contexts, arranged in a tree such that a child node completely contains the context of its parents. In this work, a context can be the set of sequences of news items, sequence of topics, or a set of topic distributions. As new articles are added, more contexts are created. Contexts corresponding to old articles are removed as soon as they disappear from the current article pool.

The second key idea is to assign a local prediction model to each context, called an expert. For instance, a particular expert gives predictions only for users who have read a particular sequence of stories, or users who have read an article that was sufficiently close to a particular topic distribution.

In the following, we first introduce the notion of context tree. Then, we describe various prediction models, how to associate them with the context tree and combine them in order to make recommendations.

3.1 Sequence Context Tree

When a user browses a news website, we track the sequence of articles read.

Definition 1. A *sequence* $\mathbf{s} = \langle n_1, \ldots, n_l \rangle$ is an ordered list of articles $n_i \in \mathcal{N}$ read by a user, and we denote $\mathbf{s}_t$ the sequence of articles read until time t. We write the set of all sequences by $\mathcal{S}$.

Note that a sequence can also be a sequence of topics of articles.

A context tree is built based on these sequences and their corresponding suffixes.

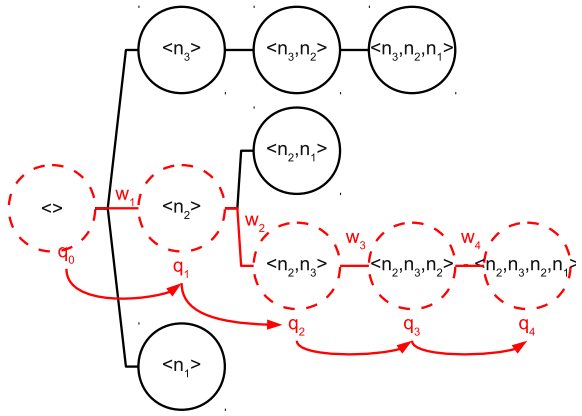


Figure 2: VMM context tree for the sequence $\mathbf{s} = \langle n_1, n_2, n_3, n_2 \rangle$. Nodes in red-dashed are active experts $\mu \in A(\mathbf{s})$.

Definition 2. A k-length sequence ξ of is a *suffix* of a l-length sequence $\mathbf{s}$, if $k \leq l$, and the last elements of $\mathbf{s}$ are equal to ξ, and we write $\xi \prec \mathbf{s}$ when ξ is a suffix of $\mathbf{s}$.

For instance, one suffix ξ of the sequence $\mathbf{s} = \langle n_1, n_2, n_3, n_4 \rangle$ is given by $\xi = \langle n_3, n_4 \rangle$.

If two sequences have similar context, the next article a user wants to read should also be similar.

Definition 3. A *context* $S = \{\mathbf{s} \in \mathcal{S} : \xi \prec \mathbf{s}\}$, $S \subset \mathcal{S}$ is the set of all possible sequences $\mathcal{S}$ ending with the suffix ξ.

We can now give a formal definition of a context tree.

Definition 4. A *context tree* $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ and edges $\mathcal{E}$ is a partition tree over the contexts $\mathcal{S}$. It has the following properties: (a) *The set of contexts at a given depth forms a partition*: If $\mathcal{V}_k$ are the nodes at depth k of the tree, then $S_i \cap S_j = \emptyset \ \forall i, j \in \mathcal{V}_k$, while $\bigcup_{i \in \mathcal{V}_k} S_i = \mathcal{S}$ (b) *Successive refinement*: If node i is the parent of j then $S_j \subset S_i$.

Thus, each node $i \in \mathcal{V}$ in the context tree corresponds to a context S_i. Initially the context tree $\mathcal{T}$ only contains a root node with context $S_0 = \mathcal{S}$. Every time a new article n_t is read, the active leaf node is split in a number of subsets, which then become nodes in the tree. This construction results in a variable-order Markov model, illustrated in Fig. 2.

The main difference between news articles and products is that articles continuously appear and disappear, and the system thus maintains a current article pool that is always changing. The model for recommendation changes along with the article pool, using a dynamically evolving context tree. As new articles are added, new branches are created corresponding to sequences or topic distributions. At the same time, nodes corresponding to old articles are removed as soon as they disappear from the current pool.

3.2 Topic Distribution Context Tree

Because of the large number of news items relative to topics, a context tree on topics might make better predictions. In particular, stories that have not been read by anyone can be recommended thanks to topic similarity. In this type of tree, each context represents a subset of the possible *topic distributions* of the last read article. The structure of the tree is slightly different and is modelled via a k-d tree.

A k-d tree is a binary tree that iteratively partitions a k-dimensional space $\mathcal{S}$ into smaller sets [4]. The i-th node corresponds to a hyper-rectangle $S_i \subset \mathcal{S}$ and has two children j, j' such that $S_j \cup S_{j'} = S_i$ and $S_j \cap S_{j'} = \emptyset$. In particular, the two children are always defined via a hyperplane splitting S_i in half, through the center of S_i, and which is perpendicular to one principal axis. In practice, we simply associate each node to one of the k axes based on the depth such that we cycle through all possible axes: $a = depth \bmod k$. The set $\mathcal{S}$ is $[0, 1]^k$, the set of k-dimensional multinomial distributions on the possible topics.

In analogy to the sequence CT, a context is a hyper-rectangle S_i and a suffix is the center θ of S_i in a topic distribution CT.

For instance, consider a node i with center $\theta \in S_i$ and associated axis a. Its two children correspond to two sets of topic distributions: Its left child j contains the distributions $\theta' \in S_i$ with $\theta'_a < \theta_a$, while its right child j' is the set on the other side of the hyperplane: $S_{j'} = \{\theta' \in S_i : \theta'_a \geq \theta_a\}$. When the system observes a new topic distribution θ, the distribution is added to the tree, and possibly the tree expands.

3.3 Experts

We assign a local prediction model called *expert* to each context (node) in the tree. More formally,

Definition 5. An *expert* μ_i is a function associated with a specific context S_i that computes an estimated probability of the next article n_{t+1} given that context, i.e. $\mathbb{P}_i(n_{t+1} \mid \mathbf{s}_t)$.

The user's browsing history $\mathbf{s}_t$ is matched to the context tree and identifies a path of nodes (see Fig. 2). All experts associated with these nodes are called *active* and are responsible for the recommendation.

Definition 6. The set of *active* experts $\mathcal{A}(\mathbf{s}_t) = \{\mu_i : \xi_i \prec \mathbf{s}_t\}$ is the set of experts μ_i associated to contexts $S_i = \{\mathbf{s} : \xi_i \prec \mathbf{s}_t\}$ such that ξ_i are suffix of $\mathbf{s}_t$.

3.4 Combining Experts into Predictions

The active experts $\mathcal{A}(\mathbf{s}_t)$ are combined by marginalizing to obtain a mixture of probabilities of all active experts:

$$\mathbb{P}(n_{t+1} = x \mid \mathbf{s}_t) = \sum_{i \in \mathcal{A}(\mathbf{s}_t)} u_i(\mathbf{s}_t)\, \mathbb{P}_i(n_{t+1} = x \mid \mathbf{s}_t), \qquad (1)$$

with $u_i(\mathbf{s}_t) = \mathbb{P}(i|\mathbf{s}_t)$ being the probability of the i-th expert relevant for this context. These probabilities are derived as follows.

With each node i in the context tree we associate a weight $w_i \in [0, 1]$ that represents the usefulness of the corresponding expert. Given a path in the context tree, we consider experts in the order of the most specific to the most general context, i.e. along the path from the most specific node to the root. In this process, with probability equal to the weight w_i we stop at a node without considering the more general experts. Thus, we take into account the relative usefulness of the experts.

Letting w_j be the probability of stopping at j given that we have not stopped yet, we thus obtain the probability u_i that the i-th expert is considered as $u_i(\mathbf{s}_t) = w_i \prod_{j : S_j \subset S_i} (1 - w_j)$ if $\mathbf{s}_t \in S_i$ and 0 otherwise.

The calculation of the total probability can be made efficiently via the recursion $q_k = w_k \mathbb{P}_k(n_{t+1} = x|\mathbf{s}_t) + (1 -$

$w_k)q_{k-1}$, where q_k is the combined prediction of the first k experts. In Figure 2, the prediction of the root expert for the next item x is q_0, while q_4 is the complete prediction by the model for this sequence.

The weights are updated by taking into account the success of a recommendation. When a user reads a new article x, we update the weights of the active experts corresponding to the suffixes ending before x according to the probability $q_k(x)$ of predicting x sequentially via Bayes' theorem [10]:

$$w'_k = \frac{w_k \mathbb{P}_k(n_{t+1} = x \mid \mathbf{s}_t)}{q_k(x)}. \quad (2)$$

No other weights are updated[1]. Finally, we also update the local models of the active experts (see Section 3.5).

3.5 Expert Models

Recommending news articles depends on multiple factors: the popularity of the news item, the freshness of the story, the sequence of news items or topics that the user has seen so far. Thus, each expert is decomposed into a set of *local models*, each modelling one of these properties. The first model ignores the temporal dynamics of the process. The second model assumes that users are mainly looking at popular items, and the last model that they are interested in fresh items (i.e. breaking news).

3.5.1 Standard model

A naïve approach for estimating the multinomial probability distribution over the news items is to use a Dirichlet-multinomial prior for each expert μ_i. The probability of reading a particular news item x depends only on the number of times α_x it has been read when the expert is active.

$$\mathbb{P}_i^{std}(n_{t+1} = x|\mathbf{s}_t) = \frac{\alpha_x + \alpha_0}{\sum_{j\in\mathcal{N}}(\alpha_j + \alpha_0)}, \quad (3)$$

where α_0 is the initial count of the Dirichlet prior.

The dynamic of news items is more complex. A news item provides new content and therefore has been seen by few users. News is subject to trends and frequent variations of preferences. We improve this simple model by augmenting it with models for *popular* or *fresh* news items.

3.5.2 Popularity model

A news item $x \in \mathcal{P}$ is in the set of popular items $\mathcal{P}$ when it has been read at least once among the last $|\mathcal{P}|$ read news items. We compute the probability of a news item x given that x is *popular* as:

$$\mathbb{P}_i^{pop}(n_{t+1} = x|\mathbf{s}_t) = \frac{c_x + \alpha_0}{\sum_{j\in\mathcal{N}}(c_j + \alpha_0)}, \quad (4)$$

where c_x is the total number of clicks received for news item x. Note that c_x is not equal to α_x (Eq. 3). α_x is the number of clicks for news item x when the expert is *active*, while c_x is the number of clicks received by news item x in *total* whether the expert is active or not.

The number of popular items $|\mathcal{P}|$ is important because it is unique for each news website. When $|\mathcal{P}|$ is small, the expert considers only the most recent read news. It is important to tune this parameter appropriately.

[1] $w_0 = 1$ since we must always stop at the root node.

3.5.3 Freshness model

A news item $x \in \mathcal{F}$ is in the set of fresh items $\mathcal{F}$ when it has not been read by anyone but is among the next $|\mathcal{F}|$ news items to be published on the website, i.e. a breaking news. We compute the probability of news item x given that x is *fresh* as:

$$\mathbb{P}_i^{fresh}(n_{t+1} = x|\mathbf{s}_t) = \begin{cases} \frac{1}{|\mathcal{F}|+1}, & \text{if } x \in \mathcal{F} \\ \frac{1}{(|\mathcal{F}|+1)(|\mathcal{N}|-|\mathcal{F}|)}, & \text{if } x \notin \mathcal{F}. \end{cases} \quad (5)$$

The number of fresh items $|\mathcal{F}|$ influences the prediction made by this expert, and is unique for each news website.

3.5.4 Mixing the expert models

We combine the three expert models using this mixture:

$$\mathbb{P}_i(n_{t+1} = x|\mathbf{s}_t) = \sum_{\tau\in\{std,pop,fresh\}} \mathbb{P}_i^{\tau}(n_{t+1} = x|\mathbf{s}_t)p_i^{\tau}. \quad (6)$$

There are two ways to compute the probabilities p_i^{τ}: either by using a Dirichlet prior that ignores the expert prediction or by a Bayesian update to calculate the posterior probability of each expert according to their accuracy.

For the first approach, the probability of the next news item being *popular* is:

$$\begin{aligned} p_i^{pop} = \mathbb{P}_i(n_{t+1} \in \mathcal{P}) &= \frac{\alpha_{pop} + \alpha_0}{(\alpha_{pop} + \alpha_0) + (\alpha_{notpop} + \alpha_0)} \\ &= \frac{\alpha_{pop} + \alpha_0}{2\alpha_0 + \sum_j \alpha_j}, \end{aligned} \quad (7)$$

where $\sum_j \alpha_j$ represents the number of times the expert μ_i has been active, α_{pop} and α_{notpop} the number of read news items which were respectively popular and not popular when the expert μ_i was active.

Similarly, the probability of the next news item being *fresh* is given by:

$$p_i^{fresh} = \mathbb{P}_i(n_{t+1} \in \mathcal{F}) = \frac{\alpha_{fresh} + \alpha_0}{2\alpha_0 + \sum_j \alpha_j}, \quad (8)$$

where α_{fresh} is the number of read news items which were fresh when the expert μ_i was active.

Noting that $\mathcal{P} \cap \mathcal{F} = \emptyset$, the probability of the next news item being neither popular nor fresh is:

$$p_i^{std} = \mathbb{P}_i(n_{t+1} \notin \mathcal{P} \cup \mathcal{F}) = 1 - \mathbb{P}_i(n_{t+1} \in \mathcal{P}) - \mathbb{P}_i(n_{t+1} \in \mathcal{F}). \quad (9)$$

It might happen that by using the Dirichlet priors, predictions are mainly made by only one expert model. To overcome this issue, we compute the probabilities p_i^{τ}, $\tau \in \{std, pop, fresh\}$ via a Bayesian update, which adapts them based on the performance of each expert model:

$$p_i^{\tau} \leftarrow \frac{\mathbb{P}_i^{\tau}(n_{t+1} = x|\mathbf{s}_t)p_i^{\tau}}{\mathbb{P}_i(n_{t+1} = x|\mathbf{s}_t)}. \quad (10)$$

4. CONTEXT-TREE RECOMMENDERS

We describe here the general algorithm to generate recommendations for the class of context-tree recommender systems. This algorithm can be applied to domains other than news in which timeliness and concept drift are of concern. We then focus on the news domain and describe in more details three VMM-based recommender systems and one based on the k-d context tree.

4.1 General Algorithm

Algorithm 1 presents a sketch of the CT recommender algorithm. For simplicity, we split our system in two procedures: *learn* and *recommend*. Both are executed for each read article x of a user with browsing history $\mathbf{s}$ in an online algorithm such as [20, 15], without any further offline computation. The candidate pool $\mathcal{C}$ is always changing and contains the popular $\mathcal{P}$ and fresh $\mathcal{F}$ stories. The system estimates the probability of each candidate and recommends the news items with the highest probability. In order to estimate the probability of a candidate item, the system *1)* selects the active experts $\mathcal{A}(\mathbf{s})$ which correspond to a path in the context tree from the most general to the most specific context, *2)* propagates q from the root down to the leaf, i.e the most specific context. q at the leaf expert is the estimated probability of the recommender system for the candidate item x, i.e. $\mathbb{P}(n_{t+1} = x|\mathbf{s}_t)$ (see Eq. 1).

Algorithm 1 CT recommender system

procedure LEARN($x, \mathbf{s}$, context set Ξ)
  $q \leftarrow 0$ and $t \leftarrow |\mathcal{A}(\mathbf{s})|$
// loop from most general expert μ_0 to most specific expert μ_t
  for $i \leftarrow 0, t$ **do**
    $p_i \leftarrow \mathbb{P}_i(n_{t+1} = x|\mathbf{s})$ // ith expert prediction
    $q \leftarrow w_i p_i + (1 - w_i)q$ // combined prediction
    $w_i \leftarrow \frac{w_i p_i}{q}$ // weight update
    update $p_i^{std}, p_i^{pop}, p_i^{fresh}$ (Eq. 7-9 or Eq. 10)
  if $x \circ \mathbf{s} \notin \Xi$ **then** // is the context in the tree?
    $\Xi = \Xi \bigcup \{x \circ \mathbf{s}\}$ // add a new leaf.
end procedure
procedure RECOMMEND($\mathbf{s}$)
  for all candidate $n \in \mathcal{C}$ **do**
    $q^{(n)} \leftarrow 0$ and $t \leftarrow |\mathcal{A}(\mathbf{s})|$
// loop from most general expert μ_0 to most specific expert μ_t
    for $i \leftarrow 0, t$ **do**
      $q^{(n)} \leftarrow w_i \mathbb{P}_i(n|\mathbf{s}) + (1 - w_i)q^{(n)}$
  $\mathcal{R} \leftarrow$ sort all $n \in \mathcal{C}$ by $q^{(n)}$ in descending order
  return first k elements of $\mathcal{R}$
end procedure

4.2 Recommender Systems

We consider three VMM variants of recommender systems, and one based on the k-d context tree.

4.2.1 VMM Recsys

The standard VMM recommender builds a context tree on the sequences of news items. That is $\mathbf{s}_t = \langle n_1, \ldots, n_t \rangle$ is a sequence of news items, and each active expert predicts n_{t+1}, the next news item.

4.2.2 Content-based VMM (CVMM) Recsys

In order to build a CT on topic sequences, we find a set of topics for each story, and assign the most probable topic to the news item. We then perform predictions on *topics*.

More precisely, we use the Latent Dirichlet Allocation (LDA) [6] as a probabilistic topic model. After concatenating the title, summary and content of the news item together, we tokenize the words and remove stopwords. We then apply LDA to all the news stories in the dataset, and obtain a topic distribution vector $\theta^{(n)}$ for each news item n.

We now define a context tree as follows. Let z_t be the most probable topic of the t-th news item. Then the *topic sequence* is $\mathbf{s}_t = \langle z_1, ..., z_t \rangle$ and ξ is a suffix of topic sequences. The context tree generates a topic probability distribution $\mathbb{P}(z_{t+1} = j|\mathbf{s}_t)$, while the LDA model provides us with a topic distribution $\mathbb{P}(z = j|n)$ for each news item n. These are then combined into the following score:

$$score(n \mid \mathbf{s}_t) = \max_j \{\mathbb{P}(z_{t+1} = j \mid \mathbf{s}_t) \cdot \mathbb{P}(z = j \mid n)\}. \quad (11)$$

The system recommends the articles with the highest scores.

4.2.3 Hybrid VMM (HVMM) Recsys

We combine the standard VMM with the content-based system into a hybrid version. The context tree is built on *topics*, similarly to the CVMM system, but the experts make predictions about *news items*, like the VMM system.

HVMM system builds a tree in the space of *topic sequences*. Each suffix ξ of size k is a sequence of most probable topics $\langle z_1, z_2, ..., z_k \rangle$. However, all predictive probabilities (Eq. 1 and later) are defined on the space of news items.

4.2.4 k-d Context-Tree (k-CT) Recsys

The CVMM and HVMM structures make predictions on the basis of the sequence of most probable topics. Instead, we consider a model that takes advantage of the complete topic distribution of the last news item. We use a k-d tree to build a context model in the space of *topic distributions*.

4.2.5 Baselines

In addition, we have the following baselines:

Z-k is a fixed k-order Markov chain recommender similar to the ones by Zimdars et al. [28].

MinHash is the minhash system used in Google News [8].

MostPopular recommends a set of stories with the highest number of clicks among the last read news items.

5. EVALUATION AND COMPARISON

We investigate whether the class of CT recommender systems has an advantage over standard methods and if so, what is the best combination of partition and expert model.

We measure performance both with respect to accuracy and novelty of recommendations. Novelty is essential because it exposes the reader to relevant news items that she would not have seen by herself. Obvious but accurate recommendations of most-popular items are of little use.

We evaluate our systems on two datasets described below. We examine on the first dataset the sensitivity of the CT models to hyperparameters. The second dataset is used to perform an unbiased comparison between the different models. We select the optimal hyperparameters on the first dataset, and then measure the performance on the second dataset. This methodology [3] mirrors the approach that would be followed by a practitioner who wants to implement a recommender system on a newspaper website.

5.1 Datasets

We collected data from the websites of two daily Swiss-French newspapers called *Tribune de Genève* (TDG) and *24 Heures* (24H)[2]. Their websites contain news stories ranging

[2]www.tdg.ch and www.24heures.ch

	News stories	Visits	Clicks
TDG	10'400	600'256	1'069'131
24H	8'613	249'099	509'978

Table 1: Datasets after filtering.

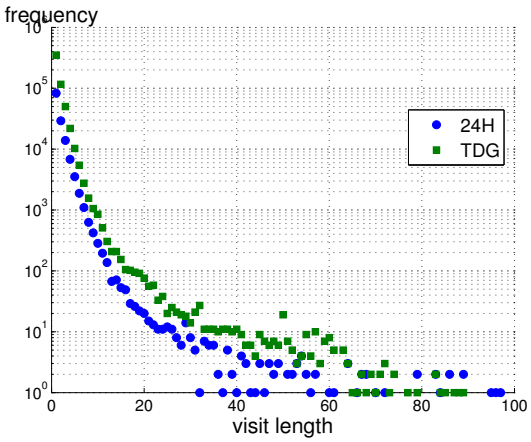


Figure 3: Distribution of the length of visits.

from local news, national and international events, sports to culture and entertainment.

The datasets span from Nov. 2008 until May 2009. They contain all the news stories displayed, and all the visits by anonymous users within the time period. Note that a new visit is created every time a user browses the website, even if she browsed the website before. The raw data has a lot of noise due to, for instance, crawling bots from search engines or browsing on mobile devices with unreliable internet connections. Table 1 shows the dataset statistics after filtering out this noise, and Figure 3 illustrates the distribution of visit length for each dataset.

5.2 Evaluation Metrics

We evaluate how good the systems are in predicting the future news a user is going to read. Specifically, we consider sequences of news items $\mathbf{s} = \langle n_1, n_2, ..., n_l \rangle$, $n_i \in \mathcal{N}, \mathbf{s} \in \mathcal{S}$ read by anonymous users. The sequences and the news items in each sequence are sorted by increasing order of visit time. When an anonymous user starts to read a news item n_1, the system generates 5 recommendations. As soon as the user reads another news item n_2, the system updates its model with the past observations n_1 and n_2, and generates a new set of recommendations. Hence the training set and the testing set are split based on the current time: at time t, the training set contains all news items accessed before t, and the testing set has items accessed after t.

We consider three metrics. The first is the **Success@5** (s@5). For a given sequence $\mathbf{s} = \langle n_1, n_2, ..., n_t, ..., n_l \rangle$, a current news item n_t in this sequence, and a set of recommended news items $\mathcal{R}$, s@5 is equal to 1 if $n_{t+1} \in \mathcal{R}$, 0 otherwise.

The second metric is **personalized s@5**, where we remove the popular items $\mathcal{R}_{\mathcal{T}}$ from the set $\mathcal{R}$, to get a reduced set $\mathcal{R}_P = \mathcal{R} \setminus \mathcal{R}_T$. This metric is important because it filters out the bias due the fact that data is collected from websites which recommend the most popular items by default.

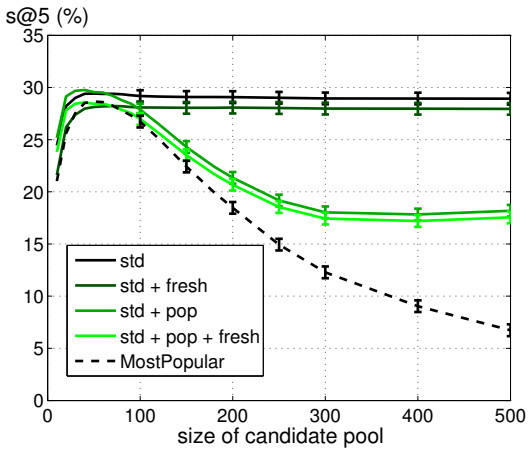


Figure 4: VMM recommender system: different mixtures of experts (Bayesian update, $|\mathcal{F}| = 10$).

The final metric is **novelty**, defined by the ratio of unseen and recommended items over the recommended items: $novelty = |\mathcal{R} \cap \mathcal{F}|/|\mathcal{R}|$. This metric is essential because users want to read about something they do not already know.

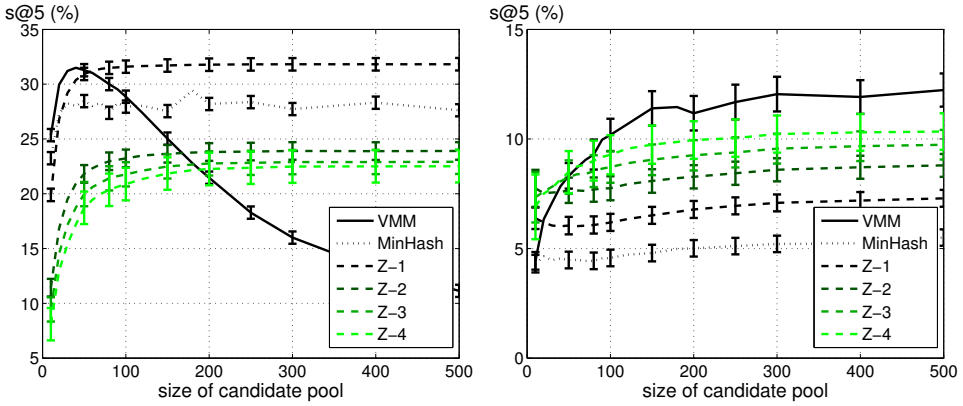
5.3 Sensitivity Evaluation

For all systems, we use a prior $\alpha_0 = 1/|\mathcal{N}|$ for the Dirichlet models, and the initial weights for the experts as $w_k = 2^{-k}$, where k is the depth of the node. We evaluated experimentally the optimal number of topics in the range of 30 to 500, and found that 50 topics bring the best accuracy. We varied the size of candidate pool: number of popular items $|\mathcal{P}|$ from 10 to 500, and fresh items $|\mathcal{F}|$ from 10 to 100. When the candidate set is small, the experts consider only the most recent read stories. We report averages over all recommendations with confidence intervals at 95%. We omit figures for the TDG dataset because we witnessed the same behaviours.

Although naïve, the approach of recommending the most popular stories is actually used very often on newspaper websites. This strategy does not pay off when the size of candidate pool increases. "Good" recommendations are drowned in popular items. This can also be seen by the fact that mixture of expert models integrating the popularity model are very sensitive to the number of popular items while others are more robust (e.g. Fig. 4 for VMM recsys).

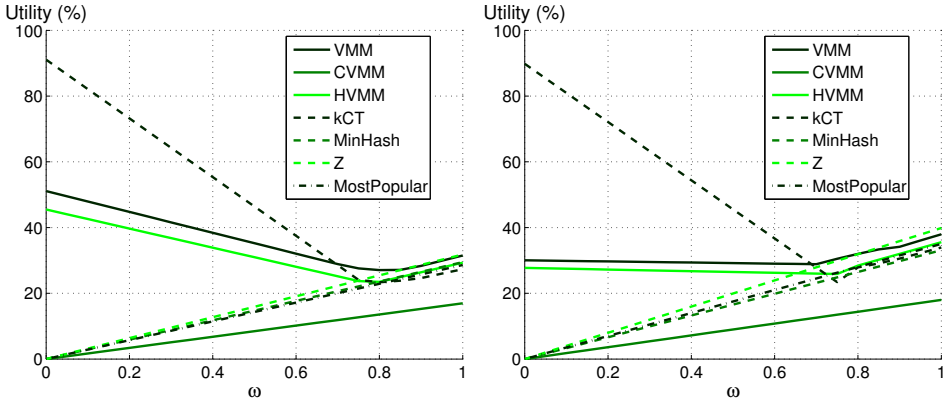
We noticed that, when using the Dirichlet priors to update the mixture probabilities, the prediction was mostly made by the popularity model, resulting in the same behaviour as the most-popular recommender system as the size of candidate pool increases. However, as the Bayesian update (Eq. 10) adapts the probabilities based on the performance of each expert model, it is more robust when we increase the candidate set. We also observed that as the number of fresh items increased, CT models were getting slightly better.

When we look at the general accuracy of CT recommender systems (Fig. 5(a)), their performance is close to the existing techniques. However when we consider only personalized items (Fig. 5(b)), CT recommender systems outperform current techniques, showing that the order of the model is important. Indeed, we observed that the weights of the experts are well distributed over the space even for long sequences. If the sequence is not important, the weights of the experts



(a) Non-personalized news items (b) Personalized news items

Figure 5: Accuracy for personalized and non-personalized news items.



(a) Tuning: 24H dataset (b) Testing: TDG dataset with optimal parameters from 24H dataset.

Figure 6: Expected performance curves: accuracy and novelty trade-off.

would have been 0 except for the root expert, resulting in a performance similar to Z-1.

We have seen that recommending popular news is easy and relatively accurate. However, a recommender system with a high accuracy on an offline evaluation does not imply that in practice it will give useful recommendations to the users. In the context of news, novelty plays a crucial role, and the users expect both personalized and novel recommendations. In the next section, we study this trade-off.

5.4 Comparison

In practice, we may be interested in some particular mix of accuracy and novelty. We formalize this by defining a utility function: $U(\omega|D, A) = \omega * s@5 + (1 - \omega) * novelty$, where ω specifies the trade-off between accuracy and novelty, D is the dataset (24H or TDG) and A is an assignment of parameters. For CT systems, the parameters are the number of popular $|\mathcal{P}|$ and fresh $|\mathcal{F}|$ items, whether the probabilities are computed via a Bayesian update or not, the mixture of experts (standard, popularity and/or freshness). It might be the case that different parameters are optimal for different utilities. The same holds for the parameters of the other methods we compare against.

To perform the comparison, we simulate the process of a designer who is going to tune each system on a small dataset (24H), before deploying the recommender online (on the TDG dataset). For any value of ω, we find the best parameters for the 24H dataset, and then measure the performance on the other dataset. This gives the Expected Performance Curve (EPC) [3], which provides an unbiased evaluation of the performance obtained by different methods.

Figure 6 illustrates the EPCs for 24H and TDG datasets. Fig. 6(a) shows the optimal utility $U(\omega|D, A^*(\omega, D))$ with $A^*(\omega, D) = \arg\max_A U(\omega|D, A)$ for $D =$ 24H dataset. Figure 6(b) shows the corresponding utility $U(\omega|D', A^*(\omega, D))$ achieved on the test dataset $D' =$ TDG using the parameters found for the tuning dataset. First, we observe that all methods are robust in that they have similar performance in the testing dataset. Second, we observe that the purely content-based method (CVMM) performs poorly both with respect to novelty and accuracy. The hybrid approach (HVMM) is significantly better. Third, the approaches that disregard the content (VMM and Z) perform similarly in terms of accuracy, but only the VMM has a reasonable novelty. Finally, the k-d tree approach (kCT) has a much higher novelty than anything else. Thus, if one were to select a method based

on performance on the tuning set, one should choose kCT for smaller values of ω and VMM for larger values.

6. CONCLUSION

News recommendation is challenging due to the rapid evolution of topics and preferences. We introduced a class of recommender systems based on context trees that accommodate a dynamically changing model. We considered context trees in the space of sequences of news, sequences of topics, and in the space of topic distributions. We defined expert models which consider the popularity and freshness of news items, and examined ways to combine them into a single model. We proposed an incremental algorithm that adapts the models continuously, and is thus better suited to such a dynamic domain as the context tree evolves over time and adapts itself to current trends and reader preferences. Our approach requires only one tree (the context tree), and thus scales very well. Our work is not restricted to the history of logged-in users, but considers a one-time session for recommendation, where users do not log in. Surprisingly, we do not know of any existing research that considers context-tree models for recommender systems.

Each proposed variant has its strengths and weaknesses. To evaluate them, we used the expected performance curve methodology, whereby each method is tuned in a training set according to a parametrized utility metric. In doing so, we showed that if we are interested in accuracy in a static dataset, a context tree that implements a variable-order Markov model is ideal, while novelty is best served with a k-d tree on the space of topics. In addition, we showed that a large order is mainly important when we are not interested in recommending highly popular items. An open question is whether the results we obtained on a static trace will be qualitatively similar in an actual recommender system. In future work, we aim to perform an online test of the system on a real news website.

7. REFERENCES

[1] J. Ahn, P. Brusilovsky, J. Grady, and D. He. Open user profiles for adaptive news systems: help or harm? In *Conf. on WWW*, pages 11–20, 2007.

[2] R. Begleiter, R. El-Yaniv, and G. Yona. On prediction using variable order markov models. *J. of AIR*, pages 385–421, 2004.

[3] S. Bengio, J. Mariethoz, and K. M. The expected performance curve. In *ICML*, pages 9–16, 2005.

[4] J. Bentley. Multidimensional binary search trees used for associative searching. *Com. ACM*, 1975.

[5] D. Billsus and M. Pazzani. A hybrid user model for news story classification. In *Conf. on User Modeling*, pages 99–108, 1999.

[6] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *J. of MLR*, 3:993–1022, 2003.

[7] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12:331–370, November 2002.

[8] A. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *Conf. on WWW*, pages 271–280, 2007.

[9] M. Deshpande and G. Karypis. Selective markov models for predicting web page accesses. *ACM Trans. Internet Technol.*, 4(2):163–184, 2004.

[10] C. Dimitrakakis. Bayesian Variable Order Markov Models. In *Proc. of AIStat*, pages 161–168, 2010.

[11] W. IJntema, F. Goossen, F. Frasincar, and F. Hogenboom. Ontology-based news recommendation. In *W. on Data Semantics*, 2010.

[12] K. Lang. Newsweeder: Learning to filter netnews. In *ICML*, pages 331–339, 1995.

[13] L. Li, W. Chu, J. Langford, and R. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Conf. on WWW*, 2010.

[14] J. Liu, P. Dolan, and E. Pedersen. Personalized news recommendation based on click behavior. In *Proc. of IUI*, pages 31–40, 2010.

[15] N. Liu, M. Zhao, E. Xiang, and Q. Yang. Online evolutionary collaborative filtering. In *Conference on Recommender systems*, pages 95–102, 2010.

[16] P. Lops, M. Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*, pages 73–105. Springer, 2011.

[17] A. Montgomery, S. Li, K. Srinivasan, and J. Liechty. Modeling online browsing and path analysis using clickstream data. *Marketing Sci.*, 23(4):579–595, 2004.

[18] J. Pitkow and P. Pirolli. Mining longest repeating subsequences to predict world wide web surfing. In *Proc. of USITS*, page 13, 1999.

[19] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Conf. on WWW*, pages 811–820, 2010.

[20] S. Rendle and L. Schmidt-Thieme. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *Conference on Recommender systems*, pages 251–258, 2008.

[21] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proc. of CSCW*, pages 175–186, 1994.

[22] J. Rissanen. A universal data compression system. *Trans. Info. Th.*, pages 656–664, 1983.

[23] R. Sarukkai. Link prediction and path analysis using markov chains. *Computer and Telecommunications Networking*, 33(1-6):377–386, 2000.

[24] G. Shani, D. Heckerman, and R. Brafman. An mdp-based recommender system. *J. of MLR*, 6:1265–1295, December 2005.

[25] X. Su and T. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, pages 4:2–4:2, January 2009.

[26] Y. Wang, L. Zhou, J. Feng, J. Wang, and Z. Liu. Mining complex time-series data by learning markovian models. In *Proc. of ICDM*, pages 1136–1140, 2006.

[27] M. Zaki, C. Carothers, and B. Szymanski. Vogue: A variable order hidden markov model with duration based on frequent sequence mining. *Trans. KDD*, 4:1–31, 2010.

[28] A. Zimdars, D. Chickering, and C. Meek. Using temporal data for making recommendations. In *Conf. on UAI*, pages 580–588, 2001.