Fair and Balanced: Learning to Present News Stories

Amr Ahmed*[1], Choon Hui Teo*[1], S.V. N. Vishwanathan[2], Alex Smola[1]
* Co-first authors.
[1]Yahoo! Research, Santa Clara, CA 95053, USA
[2]Purdue University, West Lafayette, IN 47907, USA
{amahmed,choonhui,smola}@yahoo-inc.com, vishy@stat.purdue.edu

ABSTRACT

Relevance, diversity and personalization are key issues when presenting content which is apt to pique a user's interest. This is particularly true when presenting an engaging set of news stories. In this paper we propose an efficient algorithm for selecting a small subset of relevant articles from a streaming news corpus. It offers three key pieces of improvement over past work: 1) It is based on a detailed model of a user's viewing behavior which does not require explicit feedback. 2) We use the notion of submodularity to estimate the propensity of interacting with content. This improves over the classical context independent relevance ranking algorithms. Unlike existing methods, we learn the submodular function from the data. 3) We present an efficient online algorithm which can be adapted for personalization, story adaptation, and factorization models. Experiments show that our system yields a significant improvement over a retrieval system deployed in production.

Categories and Subject Descriptors

I.2.6 [**Computing Methodologies**]: Artificial Intelligence-Learning

General Terms

Algorithms, Experimentation

Keywords

Submodularity, Personalization, Online Learning, Graphical Models

1. INTRODUCTION

As global events unfold (e.g. the debt crisis, the BP oil spill, or the Japan earthquake) a large number of newspapers, blogs and other online news sources report developments. Selecting a personalized subset of articles from this large, ever-changing corpus of news articles is vital for dealing with information overload. In this context, the goal is to provide a comprehensive yet personalized subset of articles which cover all relevant aspects of the events of any given day. Moreover, the selection algorithm we develop should be able to adapt to the users' tastes. For instance, a user may have specific interest in news that relates to a certain topic (e.g. politics, sports) or might favor a specific news source (e.g. Fox news, New York Times). Building such a system entails addressing the following challenges:

Relevance: Clearly we want to ensure that we retrieve the articles that have the highest quality score, i.e. which are obtained from high quality sources, attract a large number of clicks, that are relevant for a given user, and which satisfy other commercial motives (e.g. traffic shaping).

Coverage: We also would like to ensure that information is not only relevant but also complete in the sense that it covers all aspects of the event, e.g. in terms of content, time and with regard to the sources of information.

Key Events: When displaying news with respect to a timeline it is desirable to detect the *key events* in the thread. This means that we generally want to get the article which describes the outbreak of an earthquake rather than the summary recap written one month later (unless the latter is extremely well written and the article occurred a long time ago). In other words, we want summaries which are attached to key events but whose temporal attachment to events decreases with the age of the event.

Recency: Users are generally more interested in the present rather than the past. This means that when displaying content we should focus preferably on recent events more than events that occurred a long time in the past (unless they are very important). This is also relevant when invoking the system in an online fashion. In this case we would like to obtain novel articles as part of an ongoing news stream.

Personalization: Given that users' preferences may vary widely we need to adjust any relevance scores by the source, topical, and page quality interests of a user. In particular, we want to learn which results are preferred by which user, and which pages on a site are of most interest to the visitors of that site.

Storyline Adjustment: A complementary aspect to personalization is the ability to adjust to coverage of stories. That is, we want to be able to learn from user interactions

WSDM'12 February 8–12, 2012, Seattle, Washington USA

which aspects of a story are relevant. For instance, some users might be more interested in the political aspects of the debt crisis while the others might be more interested in its economic implications.

Deduplication: While implicit in the above discussion, deduplication is absolutely necessary since duplicates create a rather unpleasant user experience. It is desirable to deal with this problem in an *integrated* fashion.

Of course, the above desiderata are not just limited to presenting news stories. More generally, one can consider the scenario of a user interacting with a source of information. This may be a search results page where users where users would like to obtain relevant and diverse answers. Likewise it might be a set of articles related to a security on a finance page, or we might consider aggregating feeds from various social networks. Selecting a personalized subset of documents from such corpora remains an important problem.

Contributions

We argue in this paper that all the above aspects of the problem can be addressed by a model that

1. takes implicit user feedback into account,
2. uses a submodular rather than modular objective for estimating item relevance, and
3. uses a composite model for personalization.

Specifically, we build a detailed and adaptive user interaction model that accurately captures the way a user traverses a list of items. Towards this end we use a graphical model to describe both the click and viewing behavior of a user in a fully adaptive fashion through a latent variable model, thereby extending the work of [5].

Secondly, we propose an information gain score for *estimating* the click propensity of a user. This is a departure from two lines of research: the classical relevance ranking approach, which views relevance as an aggregate of the relevance of individual objects [17] and a combinatorial optimization driven approach, which tries to approximate the problem of selecting a relevant and diverse subset by crafting a suitable submodular objective function, possibly subject to side constraints [9, 1]. That is, we first and foremost attempt to model the problem accurately rather than approximating it by a mathematically simpler object.

Finally, we discuss effective optimization algorithms and parametrizations of user interest and article relevance which are well suited to composite weights (e.g. for personalization, storyline adjustment, categorical preferences, etc.). They arise naturally from the information-gain objective used to characterize user interest. A side-effect of the proposed model is that it provides effective cold-start customization for new stories and new users alike. Moreover, this yields an algorithm which is capable of selecting from a parametric set of submodular objectives. This is different from prior work [1, 9, 15] who posit the existence of a particular submodular gain function which is supposed to describe user interest accurately. Instead, our framework can *learn* the degree of submodularity for each attribute individually.

We show that our system improves the quality of estimates considerably. In aggregate, we find that our system shows a lift of over 20% over the currently deployed system in Yahoo!. Furthermore, it outperforms a number of recent related work including Turning down the Noise (TDN) algorithm of [9], the Session Utility Model (SUM) of [7], and the Cascade Click Model (CCM) of [6] significantly.

2. MODEL DESCRIPTION

In what follows, we assume that we have access to an algorithm which aggregates articles into storylines, such as described by [2] or a low-level search engine to find articles relevant for a query. This yields sets of articles which are largely coherent in accordance. Given that, we describe our model by tackling each of the following aspects: building a detailed user interaction model by using implicit user feedback (Section 2.1), modeling article relevance (Section 2.2), personalization by using a composite set of weights (Section 2.3), online optimization of the model parameters (Section 2.4) and finally article recommendation (Section 2.5).

2.1 Sequential View Click Model

Our Sequential View Click Model (SVCM) for user interaction draws on the view-click model of [5]. That is, we assume that a user traverses a list of results (URLs) from top to bottom in *sequential* order. At any given point a user can perform one of four possible actions:

1. follow a link and leave,
2. follow a link and return to examine the remainder of the results afterwards,
3. skip forward to the next result,
4. abandon the task.

Unfortunately, by using click data we have no way of knowing whether the user continued examining the list of results or whether he simply abandoned inspection, *unless* we observe further clicks by the user. In other words, we are dealing with a model in which some of the variables are latent. This can be addressed by a proper probabilistic description.

The above conditions of state transition provide us with a rather nice grammar which restricts which observations are admissible. In particular, they imply that the user will abandon examining the presented results at some point in time (this could be anywhere between the first and the last position) and after he does so, there is no chance of any further interaction with the result list. Note that this model makes a rather drastic approximation, namely that users always examine results *in order* rather than in some other (possibly nondeterministic) scheme. While not quite true in practice, we found by inspecting click logs that the approximation is fairly accurate.[1] Our model extends [5] in a number of ways:

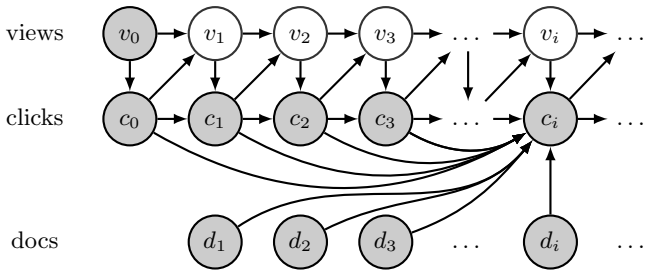
- We allow for different transition and baseline click probabilities per position rather than assuming that these coefficients are identical.
- We assume that the click propensity depends on
  - the number of previous clicks, and
  - the set of previously displayed articles.
- We allow for the user to potentially not examine the displayed results at all.

This additional set of parameters is acceptable since it is minuscule relative to the total set of parameters used in

[1]We are unable to provide specific figures since this is commercially sensitive data.

modeling user behavior, hence we can assume that the estimates are statistically reliable. In order to make matters more specific we now describe the probabilistic model associated with the above assumptions. We use $v^t := (v_1, \ldots, v_t)$ as a shorthand to denote vectors of length t consisting of the first t elements in an array.

In the following we denote by $v_i \in \{0, 1\}$ a variable which indicates whether a user examined result i and by $c_i \in \{0, 1\}$ whether he clicked on result i. We use standard directed graphical models syntax, that is, arrows indicate statistical dependencies from parent to child. Shaded vertices are considered observed, blank ones are considered latent. Extending [5] we obtain the following model. Note that we omitted some edges not pertaining to vertex c_i for readability. Furthermore, we define $v_0 = 1$ and $c_0 = 0$ as auxiliary constants to keep the notation simple.



Note that here c_i is determined by a multitude of random variables, in particular *all* previously shown articles d^{i-1} and the current article d_i. The equivalent joint probability distribution of $p(v, c|d)$ is given by

$$p(v, c|d) = \prod_{i=1}^{n} \Big[p(v_i|v_{i-1}, c_{i-1}) p(c_i|v_i, c^{i-1}, d^i) \Big] \quad (1)$$

We use a logistic transfer function to model the binary conditional probabilities, and defer specifying the functional form of the scores until Section 2.2 because it requires us to introduce submodular gains first.

Examination probability: We distinguish two cases: the examination probability whenever a user clicked on a previous item and the examination probability when the previous item was not clicked.

$$p(v_i = 1|v_{i-1} = 0) = 0 \quad (2)$$

$$p(v_i = 1|v_{i-1} = 1, c_{i-1} = 0) = \frac{1}{1 + e^{-\alpha_i}} \quad (3)$$

$$p(v_i = 1|v_{i-1} = 1, c_{i-1} = 1) = \frac{1}{1 + e^{-\beta_i}} \quad (4)$$

In other words, provided that the user examined the previous link, we allow for a logistic dependence which differs according to whether the user returns after having clicked a link or whether he is already on the page. Choosing different coefficients α_i, β_i makes sense since the propensity to click on a result varies in accordance to the location on the results page (e.g. whether a user will be required to scroll).

Click probability: The key aim of our user interaction model is to obtain good estimate of the click probability. As described in the graphical model above, it depends on $c_i|v_i, c^{i-1}, d^i$. In this context we stratify by the value of v_i by setting

$$p(c_i = 1|v_i = 0) = 0,$$

that is, if a user did not examine a result then he will not click on it. Furthermore, we assume that the click probability for a given article is characterized by 1) the *number* of previous clicks, as denoted by $|c^{i-1}|$ and 2) by the relevance of the current article d_i given the previously displayed articles d^{i-1} where the particular order of the latter is irrelevant. These simplifying assumptions lead to the following functional form:

$$p(c_i = 1|v_i = 1, c^{i-1}, d^i) = \frac{1}{1 + e^{-f(|c^{i-1}|, d_i, d^{i-1})}}$$

The specific functional form of f will be described as part of the relevance model next.

2.2 Relevance Model

A key component in our model is a score to capture notion of information gain based on a set of articles presented to the user. The intuition behind our approach is that two articles which might be equally relevant might attract considerably different number of clicks depending on which of them is displayed above the other. We model this effect via a submodular coverage function. There is plenty of evidence that submodular functions capture the effects of diversity fairly accurately both in terms of user satisfaction and in terms of their mathematical properties (see e.g. [9] and references therein).

Here we present key results about submodular functions necessary for the development of our relevance models, for more details, please see Appendix A. In a nutshell, submodularity is characterized by its diminishing returns property. That is, for a set S, a subset $A \subseteq S$, elements $x, y \in S$, and a submodular function $f : \{0, 1\}^S \to \mathbb{R}$ we have

$$f(A \cup \{x\}) - f(A) \geq f(A \cup \{x, y\}) - f(A \cup \{y\}) \quad (5)$$

and the improvement decreases as we add more elements to the set A. Moreover, the set of submodular functions forms a convex cone. One of the main reasons for the popularity of submodular functions is the fact that constrained submodular maximization can be carried out efficiently through a greedy procedure, as described in the celebrated paper of [16] which can be accelerated even further by a lazy evaluation procedure as described in [14].

Consider two sets of articles: a set S of "source" articles to be covered and a set D of articles chosen to represent the content described in S. Given those sets we may define a coverage score via

$$\rho(S, D) := \sum_{s \in S} \sum_{j} [s]_j \rho_j(D), \quad (6)$$

where $[s]_j$ is a real value describing the extent to which feature j is present in s, and $\rho_j(D)$ a monotonically non-decreasing submodular function capturing how well the feature j of article collection D covers the same feature in article s. We assume that S is generated, e.g. by a set of search results or a storyline clustering module (e.g. articles about the debt crisis).

Next we need to define the properties of ρ. For this we introduce the notion of a generating function to deal with a large family of submodular functions abstractly.

Definition 1 *A monotonic, concave and nonnegative function $\sigma : [0, \infty) \to [0, 1]$ is a cover generator.*

Lemma 2 *Given a domain X and coefficients $c_x \geq 0$ for all $x \in X$, and $A \subseteq X$, the function $\rho(A) := \sigma(\sum_{x \in A} c_x)$ is submodular whenever σ is a cover generator.*

PROOF. Clearly ρ is monotonic with respect to addition to A. The diminishing returns property follows immediately from concavity in σ and the fact that $c_x \geq 0$ for all $x \in X$. Hence ρ is submodular. □

Our definition covers a number of popular submodular coverage scores as special case:

$$\sigma(z) = \begin{cases} 0 & \text{if } z = 0 \\ 1 & \text{if } z > 0 \end{cases} \qquad \text{(set cover)} \tag{7}$$

Here we simply assume that once we discussed a subject even once it is considered covered.

$$\sigma(z) = 1 - e^{-\theta z} \text{ for } \theta > 0 \qquad \text{(probabilistic cover)} \tag{8}$$

This equation can be seen, e.g. from [9], where one chooses the probability of not finding what is required, i.e. $1 - \prod_{x \in A}(1 - \pi_x)$ to be the coverage score, where π_x is the probability of the features in x. By setting $c_x := -\log(1 - \pi_x)$ and $\theta = 1$ we obtain this case.

$$\sigma(z) = \log(\theta z + 1) \text{ for } \theta > 0 \qquad \text{(logarithmic cover)} \tag{9}$$

Recall that modular function is a special case of submodular function and that submodular functions form a convex cone. Hence, $a \sum_{d \in D} [d]_j + b\rho_j(D)$ is submodular for $a, b \geq 0$. By plugging this generalization into (6), it leads to a powerful parametrization of $\rho(S, D)$ as follows:

$$\rho(S, D|a, b) := \sum_{s \in S} \sum_j [s]_j \Big(a_j \sum_{d \in D} [d]_j + b_j \rho_j(D) \Big). \tag{10}$$

The significance of the expansion is that it combines both modular and submodular scores into a unified formula and allows us to learn the weights of the modularity a_j and submodularity b_j of a each feature j. Covering a modular feature always increases the coverage score, but covering a submodular feature has a diminishing return effect.

Given this definition of coverage, we now convert it into a relevance score to be used in the model of Section 2.1 by defining:

$$\begin{aligned} & f(|c^{i-1}|, d_i, d^{i-1}) \qquad (11) \\ :=& \rho(S, d^i|a, b) - \rho(S, d^{i-1}|a, b) + \gamma_{|c^{i-1}|} + \delta_i \\ :=& \sum_{s \in S} \sum_j [s]_j \left(a_j \sum_{d \in d^i} [d]_j + b_j \Big(\rho_j(d^i) - \rho_j(d^{i-1}) \Big) \right) \\ & + \gamma_{|c^{i-1}|} + \delta_i \end{aligned}$$

Here $\gamma_{|c^{i-1}|}$ is a correction coefficient which captures the effect of having visited $|c^{i-1}|$ links previously. δ_i captures the position specific effect, i.e. how clickable different positions in the result set are. Furthermore note that for $b = 0$ we are back to a modular relevance ranking model where each article is considered regardless of previously shown content. In other words, we recover the vector space model as a special case [18]. Moreover, setting $a = 0$, we recover the submodular score in [9] as another special case.

2.3 Composite Weights

Let $\Psi = (\alpha, \beta, a, b, \gamma, \delta)$ denotes the parameter set of our model. This parameter set is shared across all stories and users. A significant advantage of our parametrization is that the expected negative log-likelihood of the model is convex in the Ψ. This allows us to perform the following personalization extensions rather easily:

User Personalization: Different users have different preferences. It is therefore desirable that we learn from such interactions and personalize the results. We resort to an additive model along the lines of [20], i.e. we assume that the parameters Ψ are given by $\Psi_0 + \Psi_u$ where the latter are user-specific terms and the former are common parameters which ensure that we obtain a generally relevant set of articles. In other words, the latter pair allows us to personalize results to a user's preferences whereas the former pair ensures good coldstart behavior.

Storyline Adjustment: It is unlikely that all stories would be characterized by the same set of attributes. Hence, it is equally desirable to learn how to adjust the summaries to the stories at hand. Based on the amount of interaction with users we hope to improve the estimates after recording sufficient amounts of user feedback. Our model is flexible, for instance, one can have a weight for each storyline (Ψ_s) and also group stories based on their category and learn corresponding category weights (Ψ_c). We can address all of this by an additive decomposition. In conjunction with user personalization this leads to

$$\Psi = \Psi_0 + \Psi_u + \Psi_s + \Psi_c. \tag{12}$$

Our experiments show that personalization and storyline adjustment can significantly improve retrieval quality.

2.4 Online Learning

Our goal of learning is to find a weigh vector Ψ^* that fits our proposed probabilistic click models (1) on an observed data set of presumably independent examples. We formulate this learning process as a (regularized) maximum likelihood estimation problem which can be written as the following convex minimization problem:

$$\Psi^* = \lambda\Omega(\Psi) + \operatorname*{argmin}_{\Psi} \sum_{(c,d)} -\log p(c|\Psi, d). \tag{13}$$

Here, $-\log p(c|\Psi, d)$ is the marginal log-likelihood of the observed click behavior i.e., (1)). $\Omega(\Psi)$ is a regularizer defined as:

$$\Omega(\Psi) = \|\Psi_0\|_2^2 + \sum_u \|\Psi_u\|_2^2 + \sum_s \|\Psi_s\|_2^2 + \sum_c \|\Psi_c\|_2^2 . \tag{14}$$

The regularizer helps prevent overfitting and the regularization constant λ determines the extent to which we prefer smooth solutions to solutions which maximize the likelihood.

The key challenge in solving the problem (13) lies in the facts that $p(c|\Psi, d)$ is the marginal of (1) with respect to v and that v's after the last clicked position are unobserved (without the use of more advanced tracking mechanism for user browsing behavior). Thus we resort to an Expectation Maximization (EM) algorithm and use an variational upper bound on the negative log likelihood on observed data as

follows:

$$\begin{aligned}-\log p(c) &\leq -\log p(c) + D(q(v)\|p(v|c)) \\ &= \mathbf{E}_{v\sim q(v)}\left[-\log p(c) + \log q(v) - \log p(v|c)\right] \\ &= \mathbf{E}_{v\sim q(v)}\left[-\log p(c,v)\right] - H(q(v)).\end{aligned}$$

Here $q(v)$ is a distribution over the latent variables, $H(q)$ is the entropy of the distribution q, and $D(p\|q)$ is the Kullback-Leibler divergence between p and q. Putting everything together, we have the following optimization problem:

$$\min_{q(v),\Psi} \lambda\Omega(\Psi) + \sum_{(c,d)} \mathbf{E}_{v\sim q(v)}\left[-\log p(c,v|d,\Psi)\right] - H(q(v)). \tag{15}$$

The standard EM algorithm that runs in batch mode and require all data to be available at hand is not appropriate in our case as a successful and efficient deployment of our system must process data in near real-time as we observe them in a streaming fashion. To meet the system requirement, we employ an online EM [4] that performs an approximate yet complete EM step on a example-by-example basis. We describe these two steps in more details below.

E-step: Expected Log-Likelihood

In the E-step, we need to compute $q(v)$, however, we note that the bound on the marginal log-likelihood is tight when $q = p(v|c,d)$, which is the minimizer of Eq (15). In this Section we show how to compute $p(v|c,d)$, i.e. the probability distribution over views, given the observed clicks c and the presented set of articles d.

While the graphical model in Section 2.1 suggests that dynamic programming might be needed, computing $p(v|c,d)$ is, in fact, considerably simpler as we may exploit the fact that v_i is a monotonically decreasing sequence, that is, it is fully specified by counting how many views there are. In other words, since $p(v_i = 1|v_{i-1} = 0) = 0$ and $p(v_i = 0|v_{i-1} = 0) = 1$, the only admissible values for v are those which consists of a sequence of 1s followed by a sequence of zeros and as such v is completely defined by counting the number of 1s in v. Hence we denote by $\bar{v} := |v|$ the view count. Next denote by π_l the unnormalized likelihood scores for the user viewing l results.

$$\begin{aligned}\pi_l :=& p(\bar{v} = l, c, d) \qquad (16)\\ =& \prod_{i=1}^{l}\left[p(v_i = 1|v_{i-1} = 1, c_{i-1})p(c_i|v_i = 1, c^{i-1}, d^i)\right]\cdot \\ & p(v_{l+1} = 0|v_l = 1, c_l)\cdot \begin{cases}1 & \text{if } c_j = 0 \text{ for } j > l\\ 0 & \text{otherwise}\end{cases}\end{aligned}$$

It is easy to see that (16) can be computed for all values of l in linear time. This yields

$$q_l := p(\bar{v} = l|c,d) = \frac{\pi_l}{\sum_i \pi_i} \text{ and } r_l := p(\bar{v} \geq l|c,d) = \sum_{i=l}^{n} q_i \tag{17}$$

Here q_l, r_l are the coefficients of the variational distribution to be used in the M-step as described below.

M-step: Stochastic Gradient Descent

In the M-step we optimize over Ψ. First, we plug $q(v)$ derived in Section 2.4 and specified in (17) into the negative log-likelihood of the click and view probability in (15) (ignoring the last term $H(q(v))$ as it does not depend on Ψ). We have that $\mathbf{E}_{\bar{v}\sim q}\left[\log p(v,c|d,\Psi)\right]$ is given by:

$$\begin{aligned}&\sum_{i=1}^{n} \mathbf{E}_{\bar{v}\sim q}\left[\log p(v_i|v_{i-1}, c_{i-1}, \Psi)\right] + \mathbf{E}_{\bar{v}\sim q}\left[\log p(c_i|v_i, c^{i-1}, d^i, \Psi)\right]\\ =&\sum_{i=1}^{n} r_i \log p(v_i = 1|v_{i-1} = 1, c_{i-1}, \Psi) +\\ &\sum_{i=1}^{n} q_{i-1} \log p(v_i = 1|v_{i-1} = 0, c_{i-1}, \Psi) +\\ &\sum_{i=1}^{n} q_i \log p(c_i|v_i = 1, c^{i-1}, d^i, \Psi). \qquad (18)\end{aligned}$$

The remaining terms all vanish from the above expansion. For instance, $\log p(v_i = 0|v_{i-1} = 0) = 0$. The regularized expected log-likelihood is now amenable to convex optimization with respect to the parameters Ψ. Since our goal is to learn the weight vector Ψ online, we use the SVMSGD2 algorithm described in [3] to update Ψ. The update procedure at example $t+1$ is as follows:

$$g := \frac{\partial}{\partial\Psi}\mathbf{E}_{\bar{v}\sim q}\left[\log p(v,c|d,\Psi^t)\right] \tag{19}$$

$$\Psi^{t+1} := \Psi^t - \frac{1}{\lambda(t+t_0)}g \tag{20}$$

$$\Psi^{t+1} := \Psi^{t+1} - \frac{\texttt{skip}}{t+t_0}\Psi^{t+1} \quad [\text{if } (\texttt{skip} \bmod t) = 0] \tag{21}$$

where t_0 and `skip` are predefined hyperparameters. To maintain submodularity, we project (i.e. truncate) the a, b part of Ψ so that they remain nonnegative. The average regret (with respect to the best parameters learned in hindsight if we observe all the clicks at once, i.e. as in batch settings) vanishes at rate $O(t^{-\frac{1}{2}})$ for the above projected gradient algorithm, where t is the number of examples seen [22].

Online Model Selection

The hyper-parameters in our model that we need to specify are the scale *theta* in the cover function (8), regularization constant λ, and SGD parameters t_0 and `skip`. In batch setting one can select these hyper-parameters using cross-validation, however this is not feasible in our online setting. We adopt an online approach to tune these hyper-parameters as learning progresses by learning a set of K candidate models, each of which corresponds to a specific value of these hyper-parameters. After we receive an example, we ask each model to first predict which documents the user clicked on, and we keep track of the accuracy of each model. The best model is determined periodically based on its accumulated accuracy. Recommendation of articles (see Section 2.5) will be made using the best model until the next model selection. We note that the user's interaction is used to update all candidate models and this setup is similar is spirit to the mixture of expert framework, where each model is treated as an expert. Experimentally, we found that this strategy works well. In our experiments we start with $K = 48$ and we then shrink K to 5 after observing the first 10000 sessions for efficiency.

2.5 Recommendation

An important aspect to be addressed is the issue of display optimization. That is, we are not learning the user and rel-

evance model for its own sake. Instead, this is done in order to increase the amount of engagement a user displays with respect to the website. It follows from our model (1) that the amount of interaction is maximized whenever the score functions $f(|c^{i-1}|, d_i, d^{i-1})$ are as large as possible. While it is difficult to find an optimal strategy, we may at least strive to achieve two goals:

- Find a set of articles such that $\sum_{i=1}^{m} f(|c^{i-1}|, d_i, d^{i-1})$ is maximized. This goal is identical to maximizing a submodular objective as can be seen easily from the fact that f is given by the difference between two submodular functions and from the fact that the sum telescopes. Hence the greedy procedure at least guarantees that we will obtain a near optimal solution of this problem, as stated in Theorem 4.
- Choose articles with the largest contributions in f at the beginning of the list. Due to the diminishing returns property of submodular functions we are guaranteed that the value of the score function is monotonically decreasing, the more articles have already been selected. Combining this with a greedy selection procedure guarantees that the values $f(|c^{i-1}|, d_i, d^{i-1})$ are monotonically decreasing in i.

We use the second strategy in selecting and ordering articles in our experiments.

3. RELATED WORK

Probably closest to the present work is the Turning Down the Noise (TDN) algorithm of [9]. It uses a submodular score function to estimate the relevance of a retrieved set. This allows the use of a greedy algorithm for obtaining a summary of the current set of events. To deal with user interaction data it requires users to respond explicitly with a preference rating for articles. The latter is then used to ensure that the articles preferred by a user are retrieved in the order of preference — it treats each stage of the greedy procedure as a classification problem where the preferred articles are to be selected over the ones a user dislikes. Optimization is carried out by means of an exponentiated gradient (EG) algorithm.

While theoretically elegant, the TDN algorithm suffers from a number of drawbacks. It requires explicit feedback from the users, that is, a user has to indicate if she is interested/not interested/indifferent to a presented article. This is unrealistic in a real-world setting. Instead, we need systems which are capable of exploiting the implicit feedback inherent in the interaction of a user with an application. In particular, we would like to decouple the user model from the relevance score to some extent. This is desirable since results can be presented in different modalities (desktop vs. mobile application, push vs. pull), yet we would like to use all data to improve our estimates.

Secondly, while the notion of submodularity is desirable in general for proving approximation guarantees for otherwise NP hard problems, it somewhat obscures the (more generally applicable) property of diminishing returns in the context of relevance ranking. More to the point, while several simple problems can be modelled quite conveniently as submodular maximization problems, this is not necessarily the case for general user interaction models. For instance, to deal with the relatively simple addition of position bias in ranking one needs to add matroid constraints to the optimization problem. This, in turn, leads to considerably weaker approximation guarantees than the greedy approach of [16], thus negating some benefits of a pure submodular maximization strategy. In summary, we consider the fixation to submodular objectives rather harmful, yet the notion of diminishing returns in user-modeling is valuable.

Thirdly, since the EG algorithm converges rather rapidly to a sparse solution placing emphasis on a small number of features, it is prone to focusing on a very small number of terms [10], thus potentially leading to the 'filter bubble' of many recommendation algorithms (also see our experiments in Section 4). We conjecture that this effect did not manifest itself in [9] due to the short length of the feedback sessions (only up to 5 interactions per user). Moreover, exponentiated gradient algorithms are not quite so easily amenable to composite weights and bilinear recommendation algorithms, such as those by [11].

The present paper addresses all these concerns by learning the submodular score, via a dedicated user model, and by employing a stochastic gradient descent procedure. The key differences between TDN and our algorithm can be summarized as follows:

	optimization	objective	usermodel
TDN	EG	submod.	explicit
This paper	SGD	submod. + mod.	sequence

Another relevant work is the Cascade Click Model (CCM) originally proposed by [6] to explain the effect of position bias in web search results. This model assumes that users examine the documents in sequential order and click on one document and then abandon the session.

Finally, [7] use the metaphor of user satisfaction in their Session Utility Model (SUM) to capture the effect of diversity and satisfaction in a session. This leads to a (slightly nonstandard) objective function which characterizes the stopping probability of a user. While exciting, we found it not to be an ideal fit to our datastream. More to the point, for news-search drill down clicklogs we observed that often $\Pr(c > n+1 | c \geq n+1) > \Pr(c > n | c \geq n)$ holds. Here c denotes the number of clicks and $n \in \mathbb{N}$. In other words, the user is often *more* likely to click at least one more time given that he already clicked once than clicking at least once. This is contrary to what the model of [7] describes. This insight is matched in our experiments.

4. EXPERIMENTS

We perform extensive experimental evaluation to compare our method with existing algorithms, and to answer the following natural questions:

- What is the effect of the user interaction model on the overall performance of the system?
- Does the coverage function (10) which combines modular and submodular coverage scores work better than the submodular only score proposed in [9]?
- How does changing the optimizer impact the overall performance of the system?
- What is the effect of the composite weights on the performance of the model, especially in the context of personalization.
- Does our system personalize to a user and therefore lead to improved user satisfaction?

In order to answer these questions we collected data from logs of user interaction with news content. The user provides implicit feedback by clicking or skipping over the displayed articles, and this is used to adjust our model. The goal is to learn to present articles so as to maximize the positions of clicked articles (i.e. places more relevant and diverse articles at the top).

As our performance metrics we use the Precision at One (Prec@1) and Precision at Full Click (Prec@FC). Prec@1 measures the fraction of times a user clicks on the first displayed article. On the other hand, if k denotes the number of clicks in a session then Prec@FC measures the fraction of the top k articles that were clicked. For instance, if the user clicked on positions 1, 3, and 5 then the Prec@FC is 3/5. All results are relative to the performance of the deployed system which is computed as follows: $100 \times \frac{a-b}{b}$ where a is the performance of our system and b is the performance of the deployed system.

We perform online evaluation. At every iteration, the algorithm receives a set of articles from a given story out of which it selects 10 articles and presents them to the user. Feedback is provided by the user in the form of clicks. This is used to compute the performance metrics and to update the model. The data set used in the experiments consists of user sessions collected over a period of 10 weeks between May and July, 2011. A random subset of around 140K sessions with user feedback was chosen for reporting results.

4.1 Features

We use the following features which adequately capture the essence of news stories.

Entities and Concepts: We used Yahoo!'s *Spectrum* named entity detection and resolution tools to determine key and relevant tokens in the article [21]. Particular importance was given to terms in headlines and abstracts of an article and terms were weighted according to their term frequency. This ensures that we retrieve articles that are relevant in terms of their content with respect to the ground set of articles.

Recency: Another set of attributes is the distribution of time stamps of articles. To be independent of the absolute time of the event we use the most recent article in the source collection of articles S as reference. The count distribution is then aggregated in a histogram. This approach automatically selects the key events of a set of articles. After all, we expect that news intensity should correlate with the relevance of an event. If we were to use this in the context of financial news we could simply reweight events with a measure of volatility in the price of the security under consideration. A secondary effect is that due to the modular part of the relevance score we are able to learn the relative relevance of recent articles with respect to older articles.

Source: Clearly news sources are highly relevant for customizing results to the tastes of individual users. Consequently we want to ensure that the identity of a source is used as a feature. Quite interestingly, its use in the modular vs. submodular part has different effects: a contribution to the modular part via a_j means that we prefer articles predominantly originating from a particular news source. On the other hand, their use in the submodular part ensures that we faithfully represent the article distribution found in the source set S.

4.2 Results

User Interaction Models

For a fair comparison we use the same set of features and the same optimizer (SGD) for each user model. In other words, everything else is held equal and therefore the difference in performance observed in Figure 1 is solely due to the use of different user models. The click aggregation model used by TDN requires explicit feedback from the user to each displayed item as either: like, dislike or indifferent. To cast click feedback into the TDN setting, we assign a *like* label to all articles clicked by the user, *dislike* label to all articles that appear before the last clicked article but were not clicked, and *indifferent* label to remaining articles. We can see that TDN's performance is sub-optimal here as it cannot handle the implicit feedback adequately[2]. We use enhanced versions of the Session Utility Model (SUM) and Cascade Click Model (CCM) which use the submodular selection strategy. We call them SUM+ and CCM+ in Figure 1.

Even though SUM+ takes into account the sequential nature of the interaction, it is not well suited for our application as it was originally proposed for web search with a different assumption on the way user consume information. CCM+ does not incorporate articles after the first clicked article and hence is not able to perform well. In contrast to all the aforementioned models, our Sequential View Click Model (SVCM) is able to faithfully capture user interaction and therefore comprehensively outperforms the other models including the deployed system.

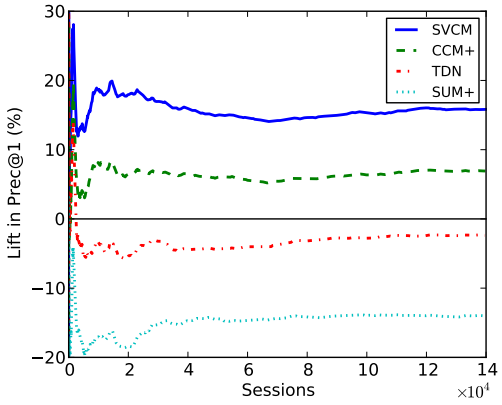
Coverage Scores

We now shift our focus to comparing the performance of the combined coverage score (10) to that of the submodular only scores [9]. The results can be found in Figure 2. For this experiment we used SVCM for the user interaction model and SGD as the optimizer. Clearly, the combination of modular and submodular scores comprehensively outperforms the submodular only score in terms of both Prec@1 and Prec@FC. This is because the model with combined coverage score is more powerful in learning the shape of the submodular function and is able to compensate the potential unnecessary penalization due to submodular score with the gain from the modular score. In other words, this combined score can learn the degree of modularity and submodularity for each feature separately.

Optimization Procedure

Next we study the impact of the optimization procedure on the overall performance. Results for Prec@1 and Prec@FC are shown in Figure 3. Clearly, stochastic gradient descent (SGD) outperforms Exponentiated Gradient (EG) in our setting. To understand this result further we examined the distribution of the final weights obtained by the two methods. Because of the multiplicative nature of EG, once a weight is set to zero (e.g. because it falls below machine precision) it can never recover. Consequently, the final weight vector of EG is sparse e.g. the sparsity of the best EG model

[2]we even obtained worse results if we don't use negative feedback at all, i.e. only positive (for clicked articles) and indifferent feedback for all other articles



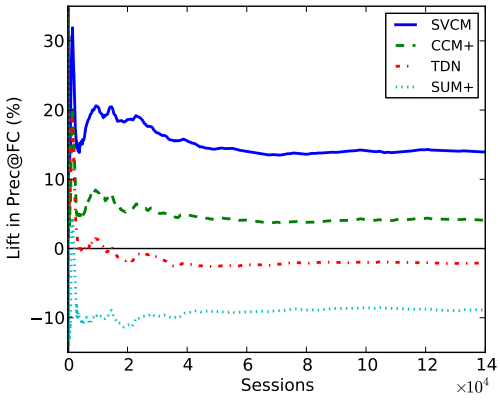
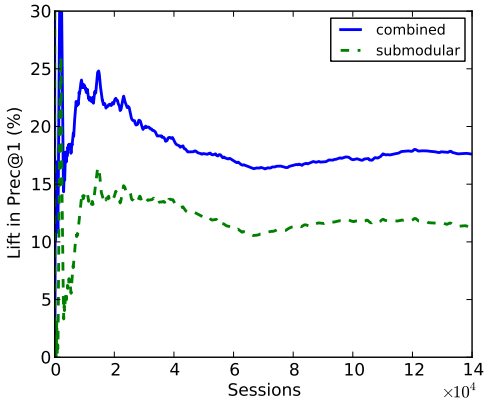


Figure 1: Prec@1 (left) and Prec@FC (right) of various user models relative to the deployed model. SVCM is the Sequential View Click Model, TDN is the model in [9], SUM is the Session Utility Model of [7], CCM is the Cascade Click Model of [6]. All models were trained using SGD (See Figure 3).



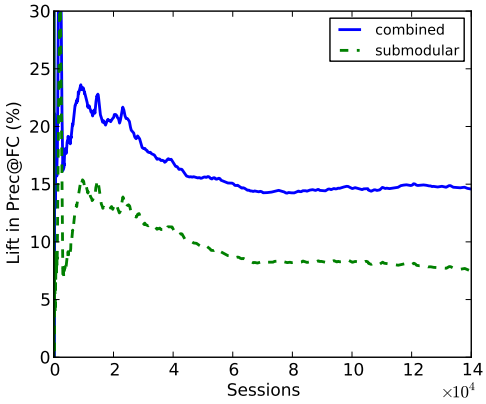
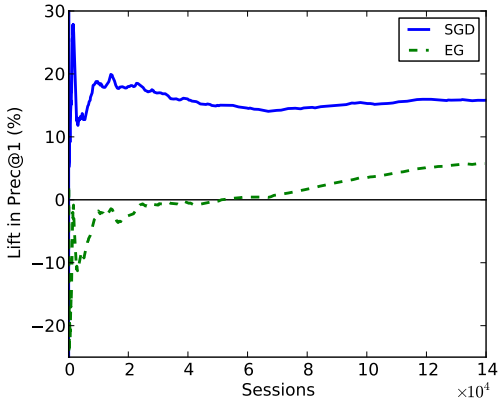


Figure 2: Prec@1 (left) and Prec@FC (right) of the combined (10) and submodular only coverage scores, relative to the deployed model.



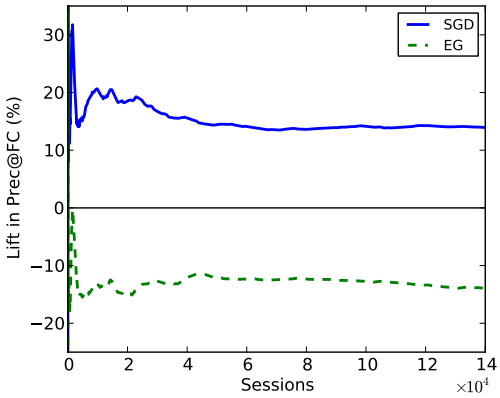
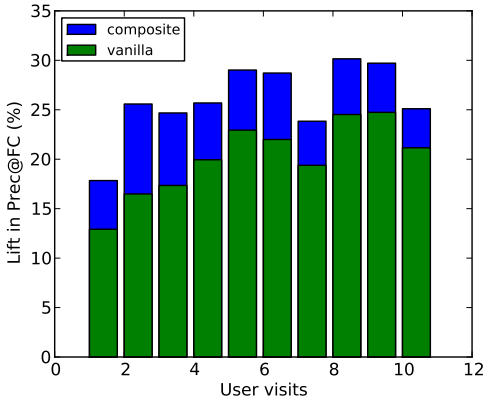


Figure 3: Prec@1 (left) and Prec@FC (right) of stochastic gradient descent (SGD) vs exponentiated gradient (EG) relative to the deployed model.



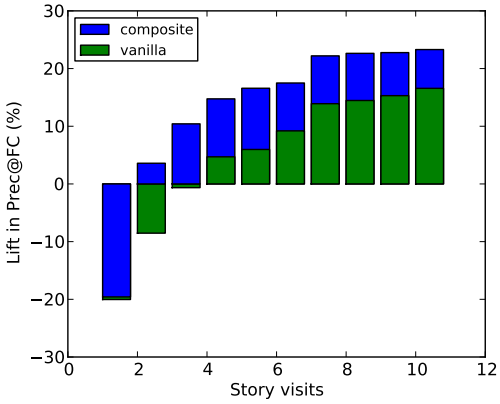


Figure 4: Lift in Prec@FC achieved by SVCM over deployed system vs number of user visits (right) and vs the number of times a story was displayed (left).

is 85% whereas the best SGD model is 0.5%. However, because of the temporal nature of news, aggressively switching off features tends to adversely affect performance.

Composite Weights

Next, in Figure 5 we examine the contribution of each component of the composite weight vector. We restrict this figure to storylines that the user has visited at least twice (i.e. examining the effect of the model in the story following task – when the user returns to examine the development of a given story). We see that more refined composite weights vectors give higher performance lift. We also observed that the contribution of the category weight over the story weight is marginal. There are two reasons for that: 1) the categories of the stories in this experiment are automatically generated using a classifier, thus there are some errors and 2) the assumption that each story belongs to a single category is restrictive since for instance a story about "Occupy Wall Street" belongs to both politics and finance categories. Models like [2] can remedy both of these deficiencies.

In Figure 4 we show how the performance improves as we observe more interactions with the user and as the number of times we display a story (possibly to different users) increases. We show the performance improvement over the deployed system with and without the story and user part of the weights turned on. The numbers in the figure are aggregated over all users (left) and stories (right). Two observations are in order. First, using the composite weighing scheme improves performance as compared to a Vanilla system which does not use composite weights. Second, our system adapts to each user and story preferences and learns to present interesting stories at the top of the list. Our composite scheme is worser than the deployed system the first time a story is displayed. This happens because a new story brings new features which might not have been seen before. This problem can be addressed by using an explore-exploit strategy by first encouraging more exploration about how to present a new story and then exploiting the learned weights to present the story. As can be observed, it only takes 1 to 3 visits before the composite weight shows improvement.

5. SUMMARY

In this paper we proposed a new integrated algorithm for capturing nontrivial interaction effects between articles when presented to a user. We demonstrated that our method is significantly better than both Yahoo's deployed system and recent algorithms which aim to solve the same problem, most notably [9, 7]. Nonetheless, there remains a significant number of problems to be addressed:

- First and foremost we need to combine the user model with an exploration / exploitation algorithm to make it well suited for large scale user interaction problems. Fortunately methods such as Thompson sampling appear to be well suited to the problem.
- While rather more sophisticated than the average user model, the SVCM user interaction model still falls short of the much more detailed representations of user behavior reported by [13]. It is quite natural to attempt to learn the *structure* of such a model directly from users and to supplement this by using parameter fitting from live traffic.
- We only discussed a pull model in the current context. However, it is fairly straightfoward to adapt the relevance score to a push model where the user is provided with additional information whenever a new event occurs. This can be achieved, e.g. by an adjusted submodular information gain measure relative to messages previously sent to the user.
- When showing headlines repeatedly to the same user, it is quite likely that he will experience display fatigue. That is, even if we were to choose the most relevant link, displaying it ten times is unlikely to increase the probability of interaction tenfold. It is common to resort to display rotation to combat this effect. We conjecture that the latter could also be captured quite well through submodularity in a much less heuristic fashion.
- Diminishing returns effects and result interactions are also quite likely in the context of advertising (e.g. three chocolate ads on the same page are probably an undesirable outcome of an auction for the keyword 'sweets'). This implies that the commonly used Generalized Second Price strategy [8] is likely not perfectly adequate for pricing interacting advertisements.

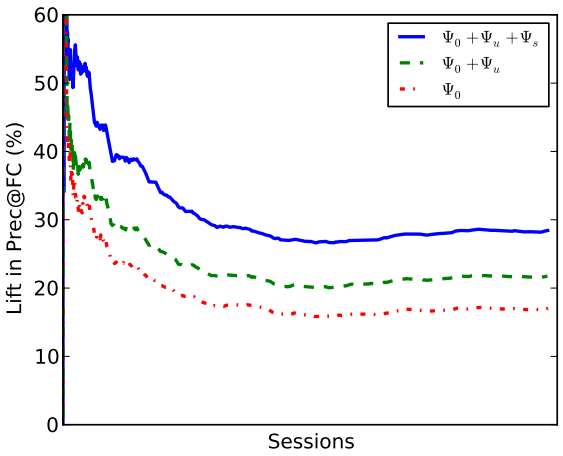


Figure 5: Lift in Prec@FC achieved by SVCM with various combinations of composite weights. The data used here are restricted to the sessions where users have visited a story at least once (we are unable to provide the number of session in the X-axis under this restriction since this is a commercially sensitive data). Here, Ψ_0 denotes the global weights, Ψ_c denotes category specific weights, Ψ_s denotes story specific weights, and Ψ_u denotes user specific weights.

In summary, we believe that the present paper is but a first step towards a more realistic user model capable of capturing the *interaction* between individual results and we are excited about the new avenues of research listed above.

6. REFERENCES

[1] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In R.A. Baeza-Yates, P. Boldi, B.A. Ribeiro-Neto, and B.B. Cambazoglu, editors, *WSDM*, pages 5–14. ACM, 2009.

[2] A. Ahmed, Q. Ho, C. H. Teo, J. Eisenstein, A. J. Smola and E. P. Xing. Online inference for the Infinite topic-cluster model: storylines from streaming text In *AISTATS*, 2011.

[3] A. Bordes, L. Bottou, and P. Gallinari. SGD-QN: Careful quasi-Newton stochastic gradient descent. *JMLR*, 10:1737–1754, 2009.

[4] O. Cappé and E. Moulines. Online expectation-maximization algorithm for latent data models. *Journal of the Royal Statistics Society: Series B (Statistical Methodology)*, 71, 2009.

[5] O. Chapelle and Y. Zhang. A dynamic bayesian network click model for web search ranking. In J. Quemada, G. León, Y.S. Maarek, and W. Nejdl, editors, *WWW*, pages 1–10. ACM, 2009.

[6] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *WSDM*, pages 87–94, NY, USA, 2008. ACM.

[7] G. Dupret and C. Liao. A model to estimate intrinsic document relevance from the clickthrough logs of a web search engine. In B. D. Davison, T. Suel, N. Craswell, and B. Liu, editors, *WSDM*, pages 181–190. ACM, 2010.

[8] B. Edelman, M. Ostrovsky, and M. Schwarz. Internet advertising and the generalized second price auction: Selling billions of dollars worth of keywords. *American Economic Review*, 97(1):242–259, 2007.

[9] K. El-Arini, G. Veda, D. Shahaf, and C. Guestrin. Turning down the noise in the blogosphere. In J. F. Elder IV, F. Fogelman-Soulié, P. A. Flach, and M. J. Zaki, editors, *KDD*, pages 289–298. ACM, 2009.

[10] J. Kivinen and M. K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–64, 1997.

[11] Y. Koren, R.M. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

[12] A. Krause and C. Guestrin. A note on the budgeted maximization of submodular functions. CMU-CALD 05-103, CMU, 2005.

[13] D. Lagun and E. Agichtein. Viewser: Enabling large-scale remote user studies of web search examination and interaction. In *ACM SIGIR*, 2011.

[14] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J.M. VanBriesen, and N.S. Glance. Cost-effective outbreak detection in networks. In *KDD*, pages 420–429. ACM, 2007.

[15] H. Lin and J. Bilmes. Multi-document summarization via budgeted maximization of submodular functions. In *NAACL*, pages 912–920. ACL, 2010.

[16] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of the approximations for maximizing submodular set functions. *Math. Programming*, 14:265–294, 1978.

[17] M. Richardson, A. Prakash, and E. Brill. Beyond pagerank: machine learning for static ranking. In L. Carr, D. De Roure, A. Iyengar, C.A. Goble, and M. Dahlin, editors, *WWW*, pages 707–715. 2006.

[18] G. Salton. *Automatic Text Processing*. Addison-Wesley, Massachusetts, 1989.

[19] A. Schrijver. *Combinatorial Optimization*, volume 24 of *Algorithms and Combinatorics*. Springer, 2003.

[20] K. Weinberger, A. Dasgupta, J. Attenberg, J. Langford, and A.J. Smola. Feature hashing for large scale multitask learning. In L. Bottou and M. Littman, editors, *ICML*, 2009.

[21] Y. Zhou, L. Nie, O. Rouhani-Kalleh, F. Vasile, and S. Gaffney. Resolving surface forms to wikipedia topics. In *COLING*, pages 1335–1343, 2010.

[22] M. Zinkevich. Online convex programming and generalised infinitesimal gradient ascent. In *ICML*, pages 928–936, 2003.

APPENDIX

A. SUBMODULAR FUNCTIONS

In a nutshell submodularity is characterized by its diminishing returns property. That is, for a set S, subsets $A, B \subseteq S$, and a function $f : \{0,1\}^S \to \mathbb{R}$ we have

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B). \tag{22}$$

An equivalent formulation is that $f(A \cup \{x\}) - f(A) \geq f(A \cup \{x, y\}) - f(A \cup \{y\})$ holds, that is, the improvement decreases as we obtain more instances. Functions for which the comparison holds with *equality* are referred to as *modular*. A particularly useful fact is the following:

Theorem 3 *The set of submodular functions on some domain X forms a convex cone.*

This follows immediately from the fact that (22) remains valid under addition and multiplication with nonnegative numbers. This means that we may add and rescale such functions while the result retains submodular properties. It also makes optimization over the family of such functions easier. One of the main reasons for the popularity of submodular functions is the fact that constrained submodular maximization can be carried out efficiently through a greedy procedure, as described in the celebrated paper of [16].

Theorem 4 *Assume that we are given a monotonically increasing submodular function f defined on a domain X. Denote by $A_k^* := \operatorname{argmax}_{A \subset X \text{ and } |A| \leq k} f(A)$ the optimal set of size bounded by k maximizing f. Then the greedy procedure*

$$A_0 := \emptyset \text{ and } A_{i+1} := A_i \cup \left\{ \operatorname*{argmax}_{x \in X} f(A_i \cup \{x\}) \right\} \tag{23}$$

achieves a solution that satisfies $f(A_k) \geq (1 - e^{-1}) f(A_k^*)$.

Cost-weighted variants of the greedy algorithm exist [12]. Note also that while invoking the algorithm given in (23) appears to require $O(|X|)$ operations per iteration, i.e. a total of $O(k|X|)$ computations, we can accelerate it significantly by lazy evaluation [14]. The idea is as follows:

We know that the gains achievable by adding elements to A can only decrease through the addition of further elements. This is, after all, a submodular function. Hence, after an initial pass which computes

$$\delta_x := f(\{x\}) - f(\emptyset) \tag{24}$$

for all $x \in X$ we can simply exploit the fact that $\delta_x^{\text{old}} \geq \delta_x^{\text{new}} := f(A \cup x) - f(A)$ and select from the list of upper bounds. Each re-evaluation is then cached. This way we need not check every time on instances $x \in X$ that turned out to be useless already before. Empirically [14] this algorithm requires only $\log m$ operations per search once the initial scores are computed thus reducing computation to $O(|X| + k \log |X|)$. This is due to the fact that we are able to rule out a significant number of objects that are simply not promising enough after an initial inspection.