SIGIR 2010

Geneva, Switzerland
July 19-23, 2010

Feature Generation and Selection for Information Retrieval

Workshop of the 33rd Annual International
ACM SIGIR Conference
on Research and Development
in Information Retrieval

Workshop organizers:
Evgeniy Gabrilovich (Yahoo! Research)
Alex Smola (Australian National University and Yahoo! Research)
Naftali Tishby (Hebrew University of Jerusalem)

Association for Computing Machinery

Contents

Organizing Committee

Organizers	Evgeniy Gabrilovich, Yahoo! Research
	Alexander Smola, Yahoo! Research and Australian National University
	Naftali Tishby, Hebrew University Jerusalem
Program Committee	Francis Bach, INRIA, USA
	Misha Bilenko, Microsoft Research, USA
	David Blei, Princeton, USA
	Karsten Borgwardt, Max Planck Institute, Germany
	Wray Buntine, NICTA, Australia
	Raman Chandrasekar, Microsoft Research, USA
	Kevyn Collins-Thompson, Microsoft Research, USA
	Silviu Cucerzan, Microsoft Research
	Brian Davison, Lehigh University, USA
	Gideon Dror, Academic College of Tel-Aviv-Yaffo, Israel
	Arkady Epshteyn, Google, USA
	Wai Lam, CUHK, Hong Kong
	Tie-Yan Liu, Microsoft Research Asia, China
	Shaul Markovitch, Technion, Israel
	Donald Metzler, Yahoo Research, USA
	Daichi Mochihashi, NTT, Japan
	Patrick Pantel, Yahoo, USA
	Filip Radlinski, Microsoft Research, United Kingdom
	Rajat Raina, Facebook, USA
	Pradeep Ravikumar, University of Texas at Austin, USA
	Mehran Sahami, Stanford, USA
	Le Song, CMU, USA
	Krysta Svore, Microsoft Research, USA
	Volker Tresp, Siemens, Germany
	Eric Xing, CMU, USA
	Kai Yu, NEC, USA
	ChengXiang Zhai, UIUC, USA
	Jerry Zhu, University of Wisconsin

Sponsors

We gratefully acknowledge contributions from the Pascal 2 Network of Excellence towards the travel expenses of the keynote speakers.

Webpage

`http://alex.smola.org/workshops/sigir2010`

Preface

Feature extraction, generation, and selection for textual documents is a rather hotly contested subject. Opinions range from the advice to perform *no* feature selection whatsoever to highly sophisticated nonparametric Bayesian models which generate latent representations of documents. This perspective is muddled further by the fact that quite often in documents 'words are the best features', yet a naive use of words does not lead to state of the art document processing tools. Matters are even more complex once the existence of curated corpora such as Wikipedia, ontologies, and lists are taken into account.

There is much opportunity in designing algorithms based on information retrieval, statistics, machine learning, data mining, and systems design to address this need for representation and extraction. Some algorithms, while quite attractive at a smaller scale, may fall short due to scalability issues (e.g. graphical models with state spaces that exceed the available RAM of a computer) unless approximations are made. At the same time, purely frequency based algorithms may not suffice to capture document content.

Hence much research remains to be done in designing simple, scalable and effective methods for representing and describing documents and the knowledge contained in them. This was the motivation for organizing a workshop on feature extraction and generation in information retrieval. We have attempted to bring together a broad range of (opposing) viewpoints such that a fertile discussion may ensue.

Evgeniy Gabrilovich, Alexander J. Smola, and Naftali Tishby
Santa Clara, July 7, 2010

Schedule

Friday, July 23, 2010

8:30–8:35	**Welcome and Opening Remarks** Evgeniy Gabrilovich
8:35–9:30	**KEYNOTE — The DDI Approach to Features: Don't Do It** Ken Church
9:30–10:00	**Online Feature Selection for Information Retrieval** Niranjan Balasubramanian, Giridhar Kumaran, and Vitor Carvalho
10:00–10:30	**Hubness in the Context of Feature Selection and Generation** Alexandros Nanopoulos
10:30–11:00	**Coffee Break**
11:00–11:30	**Information Retrieval and Machine Learning at the Long Tail** Neel Sundaresan
11:30–12:00	**Greedy RankRLS: a Linear Time Algorithm for Learning Sparse Ranking Models** Tapio Pahikkala, Antti Airola, Pekka Naula, and Tapio Salakoski
12:00–12:30	**Extensions to Self-Taught Hashing: Kernelisation and Supervision** Dell Zhang, Jun Wang, Deng Cai, and Jinsong Lu
12:30-14:00	**Lunch**
14:00-15:00	**KEYNOTE — Hierarchical Bayesian Models of Language and Text** Yeh Whye Teh
15:00-15:30	**Coffee Break**
15:30-15:50	**Feature Selection for Document Ranking using Best First Search and Coordinate Ascent** Van Dang and W. Bruce Croft
15:50-16:10	**Enhancing Query-oriented Summarization based on Sentence Wikification** Yajie Miao and Chunping Li
16:10-16:30	**A Compression Framework for Generating User Profiles** Xiaoxiao Shi, Kevin Chang, Vijay K. Narayanan, Vanja Josifovski, and Alex J. Smola
16:30-16:50	**An SVM Based Approach to Feature Selection for Topical Relevance Judgement of Feeds** Yongwook Shin and Jonghun Park
16:50-17:20	**Representing and Exploiting Searcher Interaction Data** Eugene Agichtein

KEYNOTE — The DDI Approach to Features: Don't Do It

Kenneth Church Kenneth.Church@jhu.edu
HUMAN LANGUAGE TECHNOLOGY CENTER OF EXCELLENCE
JOHNS HOPKINS UNIVERSITY, BALTIMORE, MD

25 years ago, Mercer, a member of the IBM speech group, tossed the bomb shell, "There is no data like more data!" A few years later, his boss, Jelinek, added, "Whenever I fire a linguist our system performance improves." Jelinek later moved to Hopkins and hired Brill who concluded, "It never pays to think until you've run out of data," and therefore we should "fire everyone and spend the money on data." I will refer to this period as the DDI (don't do it) approach to features (and everything else).

A decade later, the web came along and life was good for the DDI approach. But the community is beginning to discover that although the web is large, it may not be that large. The web is also making it clear that Information Retrieval is not a simple game of Solitaire with a user up against the house (the library), but modern web search is a much more interesting multiplayer game with authors, users, advertisers and spammers all working together (as well as cross purposes). Different features are more appropriate for different perspectives. Features like TF, IDF and Page Rank shed light on authors' perspectives, but users are more interested in readers' perspectives.

Recently, we have been working on classification of audio documents with few (if any) resources (dictionaries/corpora). It is standard practice to connect black boxes (ASR speech recognition, IR, NLP) in series. Although this approach tends to multiply errors, GALE has demonstrated that it can be effective, especially for high resource languages (English, Chinese and Arabic). We have been exploring an alternative for languages with few resources (most languages) that applies classification directly to the speech without taking a dependency on ASR. This approach takes us full circle back to the DDI approach to features.

Biography

Ken is currently the Chief Scientist of the Human Language Technology Center of Excellence at the Johns Hopkins University. Ken is also VP-elect of the Association for Computational Linguistics (ACL), and President of SIGDAT (special interest group that organizes EMNLP conferences). Before moving to Hopkins, Ken was a researcher at Microsoft Research in Redmond, and before that he was the head of a data mining department in AT&T Labs-Research (formally AT&T Bell Labs). He received his BS, Masters and PhD from MIT in computer science in 1978, 1980 and 1983, respectively. He enjoys working with very large corpora such as the Associated Press newswire back in the 1990s when that was considered large (1 million words per week) and larger datasets such as telephone call detail (1-10 billion records per month) and web logs (even larger). He has worked on many topics in computational linguistics including: web search, language modeling, text analysis, spelling correction, word-sense disambiguation, terminology, translation, lexicography, compression, speech (recognition and synthesis), OCR, as well as applications that go well beyond computational linguistics such as cloud computing, revenue assurance and virtual integration (using screen scraping and web crawling to integrate systems that traditionally don't talk together as well as they could such as billing and customer care).

Online Feature Selection for Information Retrieval

Niranjan Balasubramanian, Giridhar Kumaran and Vitor Carvalho niranjan@cs.umass.edu
University of Massachusetts, Amherst, MA
Microsoft Corporation, Redmond, WA

In this talk, we describe an approach for automatic query reduction on the Web, and discuss its connections to online feature selection for information retrieval.

Long web queries form an important segment in Web search. However, most web search engines perform poorly for long queries. We propose automatic query reduction as an approach for improving the effectiveness of long web queries. Specifically, we focus on a relative effectiveness estimation technique that enables automatic selection of reduced representations of the original long query.

This approach is related to feature selection in two ways. First, the effectiveness estimation technique that we develop can be viewed as an online feature selection technique. Web search engines use features that can be associated with each query term (or a set of query terms). These features are then combined in a machine learning framework to score documents. In this setting, selecting the best reduced version of a long query can be viewed as selecting the best subset of features during ranking. Second, the effectiveness estimation technique can also be used to directly select between ranking algorithms trained on multiple feature subsets. We will present some early results in this direction.

While traditional feature selection techniques that aim to reduce redundancy in features and select the best subset of features that work well for all future queries, the techniques we propose aim to identify features that work the best for each individual query. Given the typically large query-level variance in retrieval performance, we believe that such query-dependent selection of features can further help to improve the performance of learning algorithms.

This is joint work with Giridhar Kumaran and Vitor Carvalho.

Hubness in the Context of Feature Selection and Generation (Extended Abstract)

Miloš Radovanović
Department of Mathematics and Informatics
University of Novi Sad
Serbia
radacha@dmi.uns.ac.rs

Alexandros Nanopoulos
Institute of Computer Science
University of Hildesheim
Germany
nanopoulos@ismll.de

Mirjana Ivanović
Department of Mathematics and Informatics
University of Novi Sad
Serbia
mira@dmi.uns.ac.rs

Hubness is a property of vector-space data expressed by the tendency of some points (hubs) to be included in unexpectedly many k-nearest neighbor (k-NN) lists of other points in a data set, according to commonly used similarity/distance measures. Alternatively, hubness can be viewed as increased skewness of the distribution of node in-degrees in the k-NN digraph obtained from a data set. Hubness has been observed in several communities (e.g., audio retrieval), where it was described as a problematic situation. The true causes of hubness have, for the most part, eluded researchers, which is somewhat surprising given that the phenomenon represents a fundamental characteristic of vector-space data.

In recent work, we have shown that hubness is actually an inherent property of data distributions in multidimensional space, caused by high intrinsic dimensionality of data. Hubness can therefore be viewed as a notable novel aspect of the "curse of dimensionality." We have explored the implications of hubness on various tasks, including distance-based methods for machine learning [1], and vector space models for information retrieval [2], with the common conclusion that hubness is an important concern when dealing with data that is intrinsically high-dimensional.

We have concentrated our research efforts so far on explaining the origins of hubness and its effects on different tasks, assuming a given set of features defined for a particular data set. Although our works briefly consider the interaction of hubness with dimensionality reduction, the implications of hubness on feature selection, and especially generation, are still open research questions.

Besides hubness, we will center our current discussion on the notion of the *cluster assumption* (CA) which, assuming that data contains labels, roughly states that two points in the same cluster should in most cases be of the same class. This assumption is one of the pillars of semi-supervised ML methods, and is also known in IR circles as the *cluster hypothesis*, which is formulated in an analogous manner using the notion of relevance. The cluster assumption, i.e., the degree of its violation, effectively represents the degree to which the featural representation of the data fails to correspond with some notion of "ground truth" about the data given, e.g., by class labels. A high degree of CA violation indicates that models will be difficult to build from the data, and may suggest a reconsideration of the featural representation.

For a given data set and distance measure, let $N_k(\mathbf{x})$ denote the number of times point $\mathbf{x}$ occurs in the k-NN lists of other points in the data set. We express hubness using the skewness of the distribution of $N_k(\mathbf{x})$, as its standardized third moment, denoted S_{N_k}. Also, let $BN_k(\mathbf{x})$ be the number of times $\mathbf{x}$ occurs in the k-NN lists of other points, where the labels of $\mathbf{x}$ and the points in question do not match (making $BN_k(\mathbf{x})$ a measure of "badness" of $\mathbf{x}$). The normalized sum of $BN_k(\mathbf{x})$ for a given data set, BN_k ratio, represents one way to express the degree of violation of the CA.

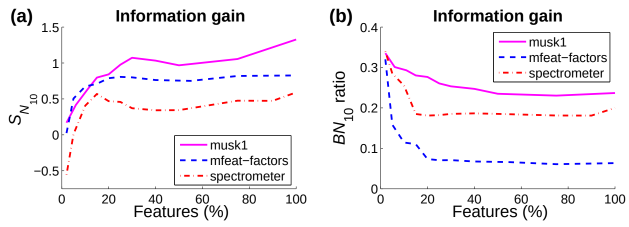


Figure 1: Skewness and "badness" ratio.

Figure 1 illustrates how S_{N_k} and BN_k ratio change when features are selected using the classical information gain method, on three data sets from the UCI repository also used in [1]. Regarding Fig. 1(a), looking from right to left, skewness of N_k stays relatively constant until a small percentage of the original number of features is left, when it abruptly drops. This is the point where the intrinsic dimensionality is reached, with further selection incurring loss of information. This loss is also visible in Fig. 1(b), where at similar points there is an increase in BN_k ratio, suggesting that the reduced representation ceases to reflect the information provided by labels very well.

Observing the two charts in the opposite direction, from left to right, offers a glimpse into the benefits and drawbacks of feature *generation*. Adding features that bring new information to the data representation will ultimately increase S_{N_k} and produce hubs. Furthermore, for the chosen examples, the reduction of BN_k ratio "flattens out" fairly quickly, limiting the usefulness of adding new features in the sense of being able to express the "ground truth." Depending on the application, instead of BN_k ratio some other criterion could have been used in Fig. 1(b), like classifier error rate, producing similarly shaped curves. While the majority of research in feature selection/generation has focused on optimizing criteria reminiscent to those in Fig. 1(b), little attention has been paid to the fact that in intrinsically high-dimensional data hubness will result in an uneven distribution of the cluster assumption violation (in our case, hubs will generally attract more label mismatches with neighboring points), and with it an uneven distribution of responsibility for classification or retrieval error among data points. We believe that investigating the interaction between hubness and different notions and analogues of CA violation can result in important new insights relevant to the tasks of feature selection and generation.

REFERENCES

[1] M. Radovanović, A. Nanopoulos, and M. Ivanović. Hubs in space: Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research*. Forthcoming.

[2] M. Radovanović, A. Nanopoulos, and M. Ivanović. On the existence of obstinate results in vector space models. In *Proc. SIGIR*, 2010.

Information Retrieval and Machine Learning at the Long Tail

Neel Sundaresan nsundaresan@ebay.com
eBay Research Labs, San Jose, CA

In this presentation we discuss how algorithms in IR and ML are adapted to build systems in the context of eBay Marketplace. The long tail nature of ”few-of-a-kind” items create unique challenges for building catalogable products. The lack of catalogable products, in turn, create unique challenges building search applications, recommender systems and managing trust. However, the participatory nature of the users combined with availability of large amount of data also provides unique opportunities for incorporating user input into building simple scalable algorithms.

Greedy RankRLS: a Linear Time Algorithm for Learning Sparse Ranking Models

Tapio Pahikkala
University of Turku and Turku Centre for Computer Science (TUCS), Turku, Finland
firstname.lastname@utu.fi

Antti Airola
University of Turku and Turku Centre for Computer Science (TUCS), Turku, Finland
firstname.lastname@utu.fi

Pekka Naula
University of Turku
Turku, Finland
firstname.lastname@utu.fi

Tapio Salakoski
University of Turku and Turku Centre for Computer Science (TUCS), Turku, Finland
firstname.lastname@utu.fi

ABSTRACT

Ranking is a central problem in information retrieval. Much work has been done in the recent years to automate the development of ranking models by means of supervised machine learning. Feature selection aims to provide sparse models which are computationally efficient to evaluate, and have good ranking performance. We propose integrating the feature selection as part of the training process for the ranking algorithm, by means of a wrapper method which performs greedy forward selection, using leave-query-out cross-validation estimate of performance as the selection criterion. We introduce a linear time training algorithm we call greedy RankRLS, which combines the aforementioned procedure, together with regularized risk minimization based on pairwise least-squares loss. The training complexity of the method is $O(kmn)$, where k is the number of features to be selected, m is the number of training examples, and n is the overall number of features. Experiments on the LETOR benchmark data set demonstrate that the approach works in practice.

Keywords

feature selection, learning to rank, ranking, RankRLS, regularized least-squares, variable selection

1. INTRODUCTION

Learning to rank for information retrieval has been a topic of intense research during the recent years. The possible benefits of automatically inducing ranking models from data, compared to purely handcrafted systems, include reduced manual labor, increased ranking performance, and adaptivity to individual user preferences. A number of supervised machine learning methods have been proposed, and successfully applied for this task. These include both pairwise approaches such as RankSVM [8], RankNet [2], and RankRLS [16, 17], as well approaches which optimize multivariate loss functions defined over queries, also known as the listwise approach [3, 4, 25].

In this article we consider the task of feature selection for learning to rank, specifically concentrating on the task of document retrieval. The task is to recognize an informative subset of the features, such that a machine learning method trained on the subset achieves good ranking performance on unseen future data. Perhaps the most fundamental advantage of this approach is that it leads to sparse models, as only a limited subset of features is used for prediction. Since applications such as web-search engines are typically constrained by strict real-time response demands, being able to restrict the number of features that need to be calculated can be quite useful. Further, feature selection can also provide feedback on the quality of different features, which can be very useful when developing and testing new ones.

Feature selection methods are typically divided into three categories [6]. In the filter approach the features are selected as a pre-processing step before applying a learning algorithm, wrapper methods select features through interaction with a learning algorithm, and embedded methods perform the selection as part of the learning process itself. Feature selection for ranking is not as of yet a well studied area, but a number of approaches have been introduced during the past few years. Geng et al. [5] proposed a filter method for selecting such features which produce good rankings, while at the same time aiming to minimize the redundancy in the set of selected features. The work was further improved upon by Yu et al. [24]. Metzler [12] and Pan et al. [19] considered feature selection for ranking with Markov random fields and boosted trees, respectively.

Since the final goal of the feature selection process is to produce a sparse ranking model with good performance, we argue that the most natural selection criterion is the so-called wrapper approach [6, 10, 11]. We select such features, which result in maximal ranking performance for the supervised learning method that we are actually using to learn our model. A standard approach for estimating the generalization performance of a model trained on a subset of features is to use cross-validation, as proposed by [10] for wrapper based feature selection. More specifically, we propose using leave-query-out (LQO) cross-validation. This allows one to make maximal use of training data, and guarantees that data points related to the same query are never split between the training and test folds. Further, to make the search over the power set of features feasible, we propose to use greedy forward selection, where on each iteration the feature whose addition yields best cross-validation performance is selected.

In this article we propose the greedy RankRLS algorithm, that is able to efficiently perform the aforementioned selection procedure. The algorithm is equivalent to a wrapper method that for each tested feature set, and each round of cross-validation would train the RankRLS method [16,17], which minimizes the pairwise regularized least-squares loss. It can also be considered as an embedded method, since the proposed training algorithm for greedy RankRLS training is far more efficient than the straightforward approach of using RankRLS as a black-box method within the selection process would be. Previously, RankRLS has been shown to produce good ranking results in document retrieval, and in general achieve ranking performance similar to that of RankSVM [16,17].

To achieve computational efficiency for the wrapper method, we combine the training algorithm with matrix algebra based shortcuts. These are made possible by the fact that RankRLS has a closed form solution, which can be fully expressed in terms of matrix operations. Firstly, the developed shortcuts allow efficient update of the solution when new features are added, without having to recompute the solution from scratch. We have previously proposed similar shortcuts for the greedy RLS algorithm [13,14], which allows one to train sparse regressors and classifiers in linear time. Second, based on the results in [1,15] we derive a formula for the exact LQO estimate that is more efficient than the one previously proposed in [16], and combine it with the update operation for feature addition. The resulting complexity of greedy RankRLS training is $O(kmn)$, where k is the number of features to be selected, m is the number of training examples, and n is the overall number of features. The memory complexity of the method is $O(mn)$. We are not aware of as efficient greedy forward selection methods with cross-validation based selection criterion for other state-of-the-art learning to rank methods.

2. SETTING

We start by introducing some notation. Let $\mathbb{R}^m$ and $\mathbb{R}^{n\times m}$, where $n, m \in \mathbb{N}$, denote the sets of real valued column vectors and $n \times m$-matrices, respectively. To denote real valued matrices and vectors we use bold capital letters and bold lower case letters, respectively. Moreover, index sets are denoted with calligraphic capital letters. By denoting $\mathbf{M}_i$, $\mathbf{M}_{:,j}$, and $\mathbf{M}_{i,j}$, we refer to the ith row, jth column, and i,jth entry of the matrix $\mathbf{M} \in \mathbb{R}^{n\times m}$, respectively. Similarly, for index sets $\mathcal{R} \subseteq \{1,\ldots,n\}$ and $\mathcal{L} \subseteq \{1,\ldots,m\}$, we denote the submatrices of $\mathbf{M}$ having their rows indexed by $\mathcal{R}$, the columns by $\mathcal{L}$, and the rows by $\mathcal{R}$ and columns by $\mathcal{L}$ as $\mathbf{M}_{\mathcal{R}}$, $\mathbf{M}_{:,\mathcal{L}}$, and $\mathbf{M}_{\mathcal{R},\mathcal{L}}$, respectively. We use an analogous notation also for column vectors, that is, $\mathbf{v}_i$ refers to the ith entry of the vector $\mathbf{v}$.

We assume, that we are given a training set in a form of data matrix $\mathbf{X} \in \mathbb{R}^{n\times m}$ and a label vector $\mathbf{y} \in \mathbb{R}^m$. The rows of the data matrix are indexed by the n features and the columns by the m training examples. Thus, by $\mathbf{X}_{:,i}$ we denote the column vector containing the features of the ith example, sliced from the data matrix and by $\mathbf{y}_i$ we denote its corresponding real value label. Let $\mathcal{I} = \{1\ldots m\}$ denote the index set for the training set. The index set is divided into a number of disjoint queries, where $Q = \{\mathcal{Q}^{(1)},\ldots,\mathcal{Q}^{(|Q|)}\}$ is the set of queries, and $\mathcal{Q}^{(i)} \subset \mathcal{I}$, $\bigcup_{i=1}^{|Q|}\mathcal{Q}^{(i)} = \mathcal{I}$ and $\mathcal{Q}^{(i)} \cap \mathcal{Q}^{(j)} = \emptyset$, if $i \neq j$. Each example represents a query-document pair. The features are a joint feature representation for the query the example is associated with, and a candidate document, and the label denotes how relevant the document is with respect to the query. The ranking of the documents associated with a query can be obtained by sorting them according to the values of their labels.

In feature selection, the task is to select a subset $\mathcal{S} \subset \{1\ldots n\}$, $|\mathcal{S}| = k$, of the n available features, such that the resulting predictor is sparse, but still produces a good ranking performance on new data. The number of selected features k may be decided in advance, or selected against a validation set, according to application specific criteria. We consider linear predictors of type

$$f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x}_{\mathcal{S}}, \tag{1}$$

where $\mathbf{w}$ is the k-dimensional vector representation of the learned predictor and $\mathbf{x}_{\mathcal{S}}$ can be considered as a mapping of the data point x into k-dimensional feature space. Note that the vector $\mathbf{w}$ only contains entries corresponding to the features indexed by $\mathcal{S}$. The rest of the features of the data points are not used in prediction phase. The computational complexity of making predictions with (1) and the space complexity of the predictor are both $O(k)$, provided that the feature vector representation $\mathbf{x}_{\mathcal{S}}$ for the data point x is given.

The pairwise ranking error for a learned predictor f can be defined as

$$\frac{1}{|Q|}\sum_{\mathcal{Q}^{(i)}\in Q}\frac{1}{N^{(i)}}\sum_{j,k\in\mathcal{Q},\mathbf{y}_j<\mathbf{y}_k} H(f(\mathbf{X}_{:,j}) - f(\mathbf{X}_{:,k})), \tag{2}$$

where H is the Heaviside step function defined as

$$H(a) = \begin{cases} 1, & \text{if } a > 0 \\ 1/2, & \text{if } a = 0 \\ 0, & \text{if } a < 0 \end{cases}$$

and $N^{(i)}$ is the number of pairs in the i:th query, for which $\mathbf{y}_j < \mathbf{y}_k$ holds true. In this definition, the error is normalized so that each considered query has the same importance, regardless of size. Since (2) is non-convex, successful pairwise approaches for learning to rank typically minimize convex approximations instead.

The RankRLS algorithm [16,17] is based on regularized risk minimization, where a least-squares based approximation of (2) is minimized, together with a quadratic regularizer. We approximate (2) with the pairwise least-squares loss, where step function H is replaced with the pairwise squared loss

$$l(i,j) = (\mathbf{y}_i - \mathbf{y}_j - f(\mathbf{X}_{:,i}) + f(\mathbf{X}_{:,j}))^2.$$

Compared to the discrete pairwise loss, this loss also enforces the magnitudes of the prediction differences. For the purpose of simplifying the derivation and implementation of the learning algorithm, we modify the normalizers, and also include tied predictions within the same query in the loss. The linear RankRLS solution is found by solving

$$\underset{\mathbf{w}\in\mathbb{R}^{|\mathcal{S}|}}{\operatorname{argmin}}\left\{\sum_{\mathcal{Q}\in Q}\frac{1}{2|\mathcal{Q}|}\sum_{i,j\in\mathcal{Q}}l(i,j)+\lambda\|\mathbf{w}\|^2\right\} \tag{3}$$

where the first term is the empirical risk measuring how well the model determined by $\mathbf{w}$ fits to the training data, and the second term called regularizer penalizes complex models. The regularization parameter $\lambda > 0$ controls the tradeoff between these terms.

Let

$$\mathbf{L}_l = \mathbf{I} - \frac{1}{l}\mathbf{1}\mathbf{1}^{\mathrm{T}}$$

be the $l \times l$-centering matrix with $l \in \mathbb{N}$. The matrix $\mathbf{L}$ is an idempotent matrix and multiplying it with a vector removes the mean of the vector entries from all elements of the vector. Moreover, the following equality can be shown

$$\frac{1}{2l}\sum_{i,j=1}^{l}(c_i - c_j) = \mathbf{c}^{\mathrm{T}}\mathbf{L}_l\mathbf{c}\,,$$

where c_i are the entries of any vector $\mathbf{c}$. Without loss of generality, we can assume that the training data is ordered according to the queries, so that first come the examples belonging to the first query, next to the second, etc. Now, let us consider the following quasi-diagonal matrix:

$$\mathbf{L} = \begin{pmatrix} \mathbf{L}_{l_1} & & \\ & \ddots & \\ & & \mathbf{L}_{l_{|Q|}} \end{pmatrix},$$

where $l_i = |\mathcal{Q}_i|$ for $i \in \{1,\ldots,|Q|\}$. The matrix $\mathbf{L}$ is again symmetric and idempotent, and can be interpreted as a query-wise centering matrix, that removes the mean from the prediction errors for each query [18]. It is also a normalized version of the Laplacian matrix encoding the structure of the preference graph induced by the queries, which has been used in previous derivations of RankRLS [16,17].

Now, (3) can be re-written in matrix notation as

$$\underset{\mathbf{w}\in\mathbb{R}^{|\mathcal{S}|}}{\operatorname{argmin}}\left\{((\mathbf{w}^{\mathrm{T}}\mathbf{X}_{\mathcal{S}})^{\mathrm{T}}-\mathbf{y})^{\mathrm{T}}\mathbf{L}((\mathbf{w}^{\mathrm{T}}\mathbf{X}_{\mathcal{S}})^{\mathrm{T}}-\mathbf{y})+\lambda\mathbf{w}^{\mathrm{T}}\mathbf{w}\right\}. \tag{4}$$

Analogously to the results in [16,17], a minimizer of (4) is

$$\mathbf{w} = (\mathbf{X}_{\mathcal{S}}\mathbf{L}(\mathbf{X}_{\mathcal{S}})^{\mathrm{T}}+\lambda\mathbf{I})^{-1}\mathbf{X}_{\mathcal{S}}\mathbf{L}\mathbf{y}.$$

Due to the symmetry and idempotence of $\mathbf{L}$, this can be re-written as

$$\mathbf{w} = (\widehat{\mathbf{X}}_{\mathcal{S}}(\widehat{\mathbf{X}}_{\mathcal{S}})^{\mathrm{T}}+\lambda\mathbf{I})^{-1}\widehat{\mathbf{X}}_{\mathcal{S}}\widehat{\mathbf{y}}, \tag{5}$$

where $\widehat{\mathbf{X}}_{\mathcal{S}} = \mathbf{X}_{\mathcal{S}}\mathbf{L}$ and $\widehat{\mathbf{y}} = \mathbf{L}\mathbf{y}$. Using the sparse decomposition of $\mathbf{L}$, the first multiplication can be performed in $O(m|\mathcal{S}|)$, and the second in $O(m)$ time. We note (see e.g. [7]) that equivalently, the RankRLS solution can be obtained from

$$\mathbf{w} = \widehat{\mathbf{X}}_{\mathcal{S}}((\widehat{\mathbf{X}}_{\mathcal{S}})^{\mathrm{T}}\widehat{\mathbf{X}}_{\mathcal{S}}+\lambda\mathbf{I})^{-1}\widehat{\mathbf{y}}. \tag{6}$$

The overall complexity of solving (5) is $O(m|\mathcal{S}|^2+|\mathcal{S}|^3)$, and that of solving (6) $O(m^2|\mathcal{S}|+m^3)$. We note that these are solutions to the ordinary regularized least squares (RLS) problem. Thus, by the query-wise centering of the data matrix the RankRLS problem can be mapped to that of solving the ordinary RLS problem.

Finally, we consider the issue of cross-validation. As discussed before, we aim to perform LQO cross-validation, where in turn each query is left out of the training set, and used for testing. Let $\mathcal{Q}$ be the index set of a query, and let $\overline{\mathcal{Q}} = \mathcal{I}\setminus\mathcal{Q}$ be the complement of this index set.

Let us consider the centered data matrix from which the rows corresponding to the $\mathcal{Q}$ have been removed, $\widehat{\mathbf{X}}_{\mathcal{S}\overline{\mathcal{Q}}}$. Due to the quasi-diagonal structure of $\mathbf{L}$, the submatrices $\mathbf{L}_{\mathcal{Q}\overline{\mathcal{Q}}}$ have only zero entries. Therefore, we have

$$\mathbf{X}_{\mathcal{S}\overline{\mathcal{Q}}}\mathbf{L}_{\overline{\mathcal{Q}}\overline{\mathcal{Q}}} = \widehat{\mathbf{X}}_{\mathcal{S}\overline{\mathcal{Q}}} - \mathbf{X}_{\mathcal{S}\mathcal{Q}}\mathbf{L}_{\mathcal{Q}\overline{\mathcal{Q}}} = \widehat{\mathbf{X}}_{\mathcal{S}\overline{\mathcal{Q}}}.$$

The significance of this result is that when removing a query from the training set, we can recover the centered representation of the remaining data simply by slicing $\widehat{\mathbf{X}}$. The centering operation does not need to be re-calculated. The feature representation for the test query is also centered, if we recover it from the centered training data matrix. Since using centered data does not affect the relative ordering or relative differences in prediction values, as long as ranking based performance measures are used, this makes no difference.

These results considerably simplify the development of efficient cross-validation methods for RankRLS. As long as folds are defined along query lines, the task of performing cross-validation with RankRLS is identical to that of performing cross-validation with RLS using centered training data. In problem settings where $\mathbf{L}$ does not have quasi-diagonal structure, such as when learning from a single global ranking, such results do not exist, making development of cross-validation shortcuts more challenging.

3. ALGORITHM DESCRIPTION

Here, we present the computational short-cuts that enable the efficient feature subset search strategy for RankRLS with LQO error as a heuristic. First, we recall an approach for computing the hold-out error for the RLS algorithm. By hold-out, we indicate the method that is used to estimate the performance of the learning algorithm by holding a part of the given data set as a test set and training a learner with the rest of the data. Our hold-out formulation assumes that the whole data set is used to train a RLS predictor and the hold-out set is then "unlearned" afterwards. The formulation can then be used, for example, to perform a N-fold CV by holding out a different part of the data set at a time and averaging the results. In this paper, we use it specifically for LQO-CV, that is, each query is held out from the training set at a time, and for a particular query, the corresponding hold-out set consists of all the training examples associated with the query.

Now, let us define

$$\mathbf{G} = ((\widehat{\mathbf{X}}_{\mathcal{S}})^{\mathrm{T}}\widehat{\mathbf{X}}_{\mathcal{S}}+\lambda\mathbf{I})^{-1}$$

and

$$\mathbf{a} = \mathbf{G}\widehat{\mathbf{y}}.$$

The following theorem can be straightforwardly inferred from the results presented by [1,15].

Theorem 3.1. *The predictions for the data points indexed by $\mathcal{Q}$ made by a RLS predictor trained using the features indexed by $\mathcal{S}$ and with the whole training set except the examples indexed by $\mathcal{Q}$ can be obtained from*

$$\widehat{\mathbf{y}}_{\mathcal{Q}} - (\mathbf{G}_{\mathcal{Q}\mathcal{Q}})^{-1}\mathbf{a}_{\mathcal{Q}}.$$

According to the above theorem, the result of LQO-CV with squared error as a performance measure can be obtained from

$$\sum_{\mathcal{Q}\in Q}(\mathbf{p}_\mathcal{Q})^\mathrm{T}\mathbf{p}_\mathcal{Q}, \tag{7}$$

where

$$\mathbf{p}_\mathcal{Q} = (\mathbf{G}_{\mathcal{Q}\mathcal{Q}})^{-1}\mathbf{a}_\mathcal{Q}.$$

It is quite straightforward to show that the vector of hold-out errors is centered query-wise, that is, $\mathbf{p} = \mathbf{L}\mathbf{p}$, because we use $\widehat{\mathbf{X}}$ and $\widehat{\mathbf{y}}$ in place of $\mathbf{X}$ and $\mathbf{y}$. Therefore, the sum of squared hold-out errors (7) is, in fact, the sum of squared query-wise centered hold-out errors. As shown earlier, this corresponds to the sum of pairwise squared losses, calculated for each query separately.

Algorithm 1: Greedy RankRLS

```
   Input: $\widehat{\mathbf{X}} \in \mathbb{R}^{n\times m}$, $\widehat{\mathbf{y}} \in \mathbb{R}^m$, $k$, $\lambda$
   Output: $\mathcal{S}$, $\mathbf{w}$
1  $\mathbf{a} \leftarrow \lambda^{-1}\widehat{\mathbf{y}}$;
2  $\mathbf{C} \leftarrow \lambda^{-1}\widehat{\mathbf{X}}^\mathrm{T}$;
3  $\mathbf{U} \leftarrow \widehat{\mathbf{X}}^\mathrm{T}$;
4  $\mathbf{p} \leftarrow \widehat{\mathbf{y}}$;
5  $\mathcal{S} \leftarrow \emptyset$;
6  while $|\mathcal{S}| < k$ do
7      $e \leftarrow \infty$;
8      $b \leftarrow 0$;
9      foreach $i \in \{1,\dots,n\} \setminus \mathcal{S}$ do
10         $c \leftarrow (1 + \widehat{\mathbf{X}}_i\mathbf{C}_{:,i})^{-1}$;
11         $d \leftarrow c{\mathbf{C}^\mathrm{T}}_i\widehat{\mathbf{y}}$;
12         $e_i \leftarrow 0$;
13         foreach $\mathcal{Q} \in Q$ do
14             $\gamma \leftarrow (-c^{-1} + {\mathbf{C}^\mathrm{T}}_{i,\mathcal{Q}}\mathbf{U}_{\mathcal{Q},i})^{-1}$;
15             $\tilde{\mathbf{p}}_\mathcal{Q} \leftarrow \mathbf{p}_\mathcal{Q} - d\mathbf{U}_{\mathcal{Q},i} - \gamma\mathbf{U}_{\mathcal{Q},i}({\mathbf{U}^\mathrm{T}}_{i,\mathcal{Q}}(\mathbf{a}_\mathcal{Q} - d\mathbf{C}_{\mathcal{Q},i}))$;
16             $e_i \leftarrow e_i + (\tilde{\mathbf{p}}_\mathcal{Q})^\mathrm{T}\tilde{\mathbf{p}}_\mathcal{Q}$;
17         if $e_i < e$ then
18             $e \leftarrow e_i$;
19             $b \leftarrow i$;
20     $c \leftarrow (1 + \widehat{\mathbf{X}}_b\mathbf{C}_{:,b})^{-1}$;
21     $d \leftarrow c{\mathbf{C}^\mathrm{T}}_b\widehat{\mathbf{y}}$;
22     $\mathbf{t} \leftarrow c\widehat{\mathbf{X}}_b\mathbf{C}$;
23     foreach $\mathcal{Q} \in Q$ do
24         $\gamma \leftarrow (-c^{-1} + {\mathbf{C}^\mathrm{T}}_{b,\mathcal{Q}}\mathbf{U}_{\mathcal{Q},b})^{-1}$;
25         $\mathbf{p}_\mathcal{Q} \leftarrow \mathbf{p}_\mathcal{Q} - d\mathbf{U}_{\mathcal{Q},b} - \gamma\mathbf{U}_{\mathcal{Q},b}({\mathbf{U}^\mathrm{T}}_{b,\mathcal{Q}}(\mathbf{a}_\mathcal{Q} - d\mathbf{C}_{\mathcal{Q},b}))$;
26         $\mathbf{U}_\mathcal{Q} \leftarrow \mathbf{U}_\mathcal{Q} - \mathbf{U}_{\mathcal{Q},b}\mathbf{t} - \gamma\mathbf{U}_{\mathcal{Q},b}({\mathbf{U}^\mathrm{T}}_{b,\mathcal{Q}}(\mathbf{C}_\mathcal{Q} - \mathbf{C}_{\mathcal{Q},b}\mathbf{t}))$;
27     $\mathbf{a} \leftarrow \mathbf{a} - d\mathbf{C}_{:,b}$;
28     $\mathbf{C} \leftarrow \mathbf{C} - \mathbf{C}_{:,b}\mathbf{t}$;
29     $\mathcal{S} \leftarrow \mathcal{S} \cup \{b\}$;
30 $\mathbf{w} \leftarrow \widehat{\mathbf{X}}_\mathcal{S}\mathbf{a}$;
```

Next, we go through the actual feature selection algorithm whose pseudo code is presented in Algorithm 1. Let us first define the following quasi-diagonal matrix:

$$\mathbf{Q} = \begin{pmatrix} (\mathbf{G}_{\mathcal{Q}_1,\mathcal{Q}_1})^{-1} & & \\ & \ddots & \\ & & (\mathbf{G}_{\mathcal{Q}_{|Q|},\mathcal{Q}_{|Q|}})^{-1} \end{pmatrix}.$$

In order to take advantage of the computational short-cuts, the feature selection algorithm maintains the current set of selected features $\mathcal{S} \subseteq \{1,\dots,n\}$, the vectors $\mathbf{a},\mathbf{p} \in \mathbb{R}^m$, and the matrices $\mathbf{C},\mathbf{U} \in \mathbb{R}^{m\times n}$ whose values are defined as

$$\begin{aligned} \mathbf{a} &= \mathbf{G}\widehat{\mathbf{y}}, \\ \mathbf{C} &= \mathbf{G}\widehat{\mathbf{X}}^\mathrm{T}, \\ \mathbf{U} &= \mathbf{Q}\mathbf{G}\widehat{\mathbf{X}}^\mathrm{T}, \\ \mathbf{p} &= \mathbf{Q}\mathbf{G}\widehat{\mathbf{y}}. \end{aligned}$$

In the initialization phase of the greedy RankRLS algorithm the set of selected features is empty, and hence the values of $\mathbf{a}$, $\mathbf{C}$, $\mathbf{U}$, and $\mathbf{p}$ are initialized to $\lambda^{-1}\widehat{\mathbf{y}}$, $\lambda^{-1}\widehat{\mathbf{X}}^\mathrm{T}$, $\widehat{\mathbf{X}}^\mathrm{T}$, and $\widehat{\mathbf{y}}$, respectively. The computational complexity of the initialization phase is dominated by the $O(mn)$ time required for storing the matrices $\mathbf{C}$, $\mathbf{U}$, and $\widehat{\mathbf{X}}$ in memory. Thus, the initialization phase is no more complex than one pass through the training data.

Let us now consider the computation of the LQO performance for the modified feature set $\mathcal{S} \cup \{i\}$, where i is the index of the feature to be added. Recall that the hold-out prediction for the examples that are associated with query $\mathcal{Q}$ can be computed from $\mathbf{p}_\mathcal{Q} = (\mathbf{G}_{\mathcal{Q},\mathcal{Q}})^{-1}\mathbf{a}_\mathcal{Q}$, where $\mathbf{a} = \mathbf{G}\widehat{\mathbf{y}}$. However, since a new feature is temporarily added into the set of selected features, we must use the matrix

$$\widetilde{\mathbf{G}} = ((\widehat{\mathbf{X}}_\mathcal{S})^\mathrm{T}\widehat{\mathbf{X}}_\mathcal{S} + (\widehat{\mathbf{X}}_i)^\mathrm{T}\widehat{\mathbf{X}}_i + \lambda\mathbf{I})^{-1}$$

in place of $\mathbf{G}$. Due to the well-known Sherman-Morrison-Woodbury (SMW) formula, the matrix $\widetilde{\mathbf{G}}$ can be rewritten as

$$\begin{aligned} \widetilde{\mathbf{G}} &= \mathbf{G} - \mathbf{G}(\widehat{\mathbf{X}}_i)^\mathrm{T}(1 + \widehat{\mathbf{X}}_i\mathbf{G}(\widehat{\mathbf{X}}_i)^\mathrm{T})^{-1}\widehat{\mathbf{X}}_i\mathbf{G} \\ &= \mathbf{G} - c\mathbf{C}_{:,i}{\mathbf{C}^\mathrm{T}}_i, \end{aligned}$$

where

$$c = (1 + \widehat{\mathbf{X}}_i\mathbf{C}_{:,i})^{-1}.$$

Accordingly, the updated vector of dual variables $\tilde{\mathbf{a}}$ can be written as

$$\begin{aligned} \tilde{\mathbf{a}} &= \widetilde{\mathbf{G}}\widehat{\mathbf{y}} \\ &= (\mathbf{G} - c\mathbf{C}_{:,i}{\mathbf{C}^\mathrm{T}}_i)\widehat{\mathbf{y}} \\ &= \mathbf{a} - d\mathbf{C}_{:,i}, \end{aligned}$$

where

$$d = c{\mathbf{C}^\mathrm{T}}_i\widehat{\mathbf{y}}.$$

Now, concerning $(\widetilde{\mathbf{G}}_{\mathcal{Q},\mathcal{Q}})^{-1}$, we have

$$\begin{aligned} (\widetilde{\mathbf{G}}_{\mathcal{Q},\mathcal{Q}})^{-1} &= ((\mathbf{G} - c\mathbf{C}_{:,i}{\mathbf{C}^\mathrm{T}}_i)_{\mathcal{Q},\mathcal{Q}})^{-1} \\ &= (\mathbf{G}_{\mathcal{Q},\mathcal{Q}} - c\mathbf{C}_{\mathcal{Q},i}{\mathbf{C}^\mathrm{T}}_{i,\mathcal{Q}})^{-1} \\ &= (\mathbf{G}_{\mathcal{Q},\mathcal{Q}})^{-1} \\ &\quad -\gamma(\mathbf{G}_{\mathcal{Q},\mathcal{Q}})^{-1}\mathbf{C}_{\mathcal{Q},i}{\mathbf{C}^\mathrm{T}}_{i,\mathcal{Q}}(\mathbf{G}_{\mathcal{Q},\mathcal{Q}})^{-1} \\ &= (\mathbf{G}_{\mathcal{Q},\mathcal{Q}})^{-1} - \gamma\mathbf{U}_{\mathcal{Q},i}{\mathbf{U}^\mathrm{T}}_{i,\mathcal{Q}}, \end{aligned}$$

where

$$\gamma = (-c^{-1} + {\mathbf{C}^\mathrm{T}}_{i,\mathcal{Q}}\mathbf{U}_{\mathcal{Q},i})^{-1}$$

and the equality between the second and third rows are again due to the SMW formula. Finally, we can compute the hold-out predictions $\tilde{\mathbf{p}}_\mathcal{Q}$ for the updated feature set as

$$\begin{aligned} \tilde{\mathbf{p}}_\mathcal{Q} &= (\widetilde{\mathbf{G}}_{\mathcal{Q},\mathcal{Q}})^{-1}\tilde{\mathbf{a}}_\mathcal{Q} \\ &= (\mathbf{G}_{\mathcal{Q},\mathcal{Q}})^{-1}\tilde{\mathbf{a}}_\mathcal{Q} - \gamma\mathbf{U}_{\mathcal{Q},i}({\mathbf{U}^\mathrm{T}}_{i,\mathcal{Q}}\tilde{\mathbf{a}}_\mathcal{Q}) \\ &= \mathbf{p}_\mathcal{Q} - d\mathbf{U}_{\mathcal{Q},i} - \gamma\mathbf{U}_{\mathcal{Q},i}({\mathbf{U}^\mathrm{T}}_{i,\mathcal{Q}}(\mathbf{a}_\mathcal{Q} - d\mathbf{C}_{\mathcal{Q},i})). \end{aligned}$$

The calculation of the last row requires only $O(|\mathcal{Q}|)$ time provided that we have all the required caches available. Since the sizes of the query index sets sum up to m, the overall complexity of LQO-CV for the updated feature set becomes $O(m)$. Further, as the greedy forward selection approach tests each of the order of n unselected features before the best of them is added into the set of selected features, the complexity of the selection step is $O(mn)$.

What is still left in our consideration is the phase in which the caches are updated after an new feature is added into the set of selected features. The vectors $\mathbf{a}$ and $\mathbf{p}$ are updated in the same way as they were temporarily updated in the LQO-CV computations. The update processes of the matrices $\mathbf{U}$ and $\mathbf{C}$ are analogous to those of the vectors $\mathbf{p}$ and $\mathbf{a}$ except that the matrix $\mathbf{C}$ is used in place of the vector $\mathbf{a}$ and the vector

$$\mathbf{t} = c\widehat{\mathbf{X}}_b\mathbf{C},$$

where b is the index of the selected feature, is used in place of the constant d. The computational time required for updating $\mathbf{U}$ and $\mathbf{C}$ is $O(mn)$, that is, updating the caches after the selection step is not more complex than the selection step itself.

Putting everything together, the overall computational complexity of greedy RankRLS is $O(kmn)$, where k is the number of features the algorithm selects until it stops. This is because the algorithm performs k iterations during which it adds one new feature to the set of selected features and each iteration requires $O(mn)$ time as shown above. The space complexity of the algorithm is $O(mn)$ which is dominated by keeping the matrices $\mathbf{C}$, $\mathbf{U}$, and $\widehat{\mathbf{X}}$ in memory.

4. EXPERIMENTS

We perform experiments on the publicly available LETOR benchmark data set (version 4.0) for learning to rank for information retrieval [1] [21]. We run experiments on two data sets, MQ2007 and MQ2008. MQ2007 consists of 69623 examples divided into 1700 queries, and MQ2800 contains 15211 examples divided into 800 queries. In both data sets the examples have the same 46 high-level features.

We follow the experimental setup proposed by the authors of LETOR. All results are averages from 5-fold cross-validation, where on each round 3 folds are used for training, 1 for parameter selection and 1 for testing. We use the exact splits provided in the data sets. Mean Average Precision (MAP) is used when selecting parameters. In addition to average precision, we measure Normalized Discountive Cumulative Gain (NDCG) when calculating test performance. In the results we present MAP, P@10, mean NDCG, and NDCG@10 values.

We compare greedy RankRLS to RankRLS and RankSVM, which are trained on all the features. For greedy RankRLS, we choose via grid search both the number of selected features and value of regularization parameter, such that lead to best MAP performance on the validation fold. For normal RankRLS, which is trained on all the features, only the value of the regularization parameter needs to be tuned. RankRLS and greedy RankRLS are implemented as part of the RLScore open source machine learning framework [2]. The RankSVM results are taken directly from the baselines section of the LETOR distribution website. The experimental setup for the RankSVM runs, as described by LETOR authors, is the same as outlined here, the used implementation was the SVM^{rank} of Joachims[3] [9]. We also plot performance curves as a function of the number of selected features on the validation sets, and examine the feature sets selected on different folds.

Tables 1 and 2 contains the selected features on the MQ2007 and MQ2008 data sets, respectively. Where more than 10 features was selected, we present only the first 10. On two of the folds of MQ2007, the optimal number of features are 11 and 12, on three of the folds almost all of the features are chosen. On MQ2008 relatively few features were chosen on all of the folds, on two of the folds the best validation performance was reached with only one feature. There are differences in the feature sets selected in the different rounds of cross-validation, but one thing remains constant. On both data sets, and on each cross-validation round, the feature selected first is feature number 39, "LMIR.DIR of whole document". The feature is a language model based feature which corresponds to a posteriori estimate of the likelihood of the query given the whole document, where a Dirichlet prior over the documents is used [26]. Based on our results this feature seems to be very useful for ranking, since as it turns out models using only it can in some cases be competitive with models trained on all the features.

In Figures 1, 2, 3, and 4 are the average MAP and mean NDCG performances over the validation folds, plotted for different regularization parameter values. We note that the results are quite unstable, suggesting that reliable selection of the regularization parameter and number of selected features remains a challenging problem. On MQ2007 the performance increases with the number of selected features. This is why in three of the folds the selection strategy used in our study lead to selecting almost all of them. However, on MQ2008 best validation performances are reached with relatively few features, after which the performance decreases. On MQ2007 close to optimal validation results can be reached already with around 15 features. This suggests that perhaps a multi-objective criterion should be used in parameter selection, which in addition to favoring high validation performance would also penalize models that use too many features.

In Tables 3, 4, 5, and 6, are the test results for MQ2007, and in Tables 7, 8, 9, and 10 are the test results for MQ2008. Overall, the results for greedy RankRLS, RankRLS and RankSVM are very close to each other, even on fold-by-fold basis. The results further verify the earlier results in [16,17], which suggest that RankRLS and RankSVM optimization often lead to very similar results. Further, the results show that at least on this data, sparse models learned using greedy forward selection are competitive with models learned using all the features.

5. DISCUSSION AND FUTURE WORK

The greedy RankRLS implementation presented in this paper is computationally feasible when dealing with data sets, such as LETOR, in which the overall number of available features is not very large. However, the situation is different if the data points are represented, for example, as

[1] `http://research.microsoft.com/en-us/um/beijing/projects/letor/`

[2] `http://www.tucs.fi/rlscore`

[3] `http://www.cs.cornell.edu/People/tj/svm_light/svm_rank.html`

Table 1: Selected features on MQ2007.

Model	fold1	fold2	fold3	fold4	fold5
λ	2^8	2^6	2^9	2^8	2^7
k	11	40	46	44	12
selected 1	39	39	39	39	39
selected 2	19	32	27	28	25
selected 3	25	19	23	45	19
selected 4	23	26	19	23	43
selected 5	32	23	13	43	23
selected 6	16	16	18	33	29
selected 7	43	5	42	13	22
selected 8	22	33	33	18	18
selected 9	5	18	16	22	5
selected 10	33	3	5	15	16

Table 2: Selected features on MQ2008

Model	fold1	fold2	fold3	fold4	fold5
λ	2^0	2^{10}	2^3	2^6	2^0
k	1	4	7	4	1
selected 1	39	39	39	39	39
selected 2		23	29	29	
selected 3		37	25	25	
selected 4		32	23	23	
selected 5			46		
selected 6			37		
selected 7			19		

raw text documents, and the words or their composites occurring in the documents form the set of available features. In this case, there is the drawback that $m \times n$-dimensional dense matrices has to be maintained in memory, while the data are stored in a sparse matrix of the same size having only a few nonzero entries. This is, because each document has nonzero values only for a small subset of features. In addition to the feasibility problems with memory, the $O(mn)$ time required per iteration may be too expensive in practise. Fortunately, it is possible to design such variations of greedy RankRLS that are better suited for this type of data.

First, we can avoid storing the dense $m \times n$-matrices by spending more computational resources. This is possible with a variation whose time complexity is $O(k^2mn)$. As an additional modification, we can reduce the time spent in each iteration by selecting the new feature from a random subset of the available features, resulting to a time complexity $O(k^2m\kappa)$, where κ is the size of random subsets. This type of idea is used, for example, for selecting the basis vectors for Gaussian process regressors by [22]. Finally, we can take advantage of the sparsity of the data matrix in reducing the time complexity down to $O(k^2\overline{m}\kappa)$, where $\overline{m}$ is the average number of training examples for which the features have nonzero values, if we use so-called back-fitting variation of our algorithm instead of performing pre-fitting as our current implementation does. For descriptions of the terms back-fitting and pre-fitting, we refer to [23]. The detailed

Table 3: Map results on MQ2007

Fold	GRankRLS	RankRLS	RankSVM
1	0.4859	0.4912	0.4894
2	0.4571	0.4573	0.4573
3	0.4655	0.4655	0.4676
4	0.4423	0.4425	0.4401
5	0.4709	0.4687	0.4680
avg	0.4643	0.4650	0.4645

Table 4: P@10 results on MQ2007

Fold	GRankRLS	RankRLS	RankSVM
1	0.3958	0.3997	0.4036
2	0.3858	0.3855	0.3932
3	0.3684	0.3684	0.3699
4	0.3670	0.3673	0.3652
5	0.3808	0.3811	0.3847
avg	0.3796	0.3804	0.3833

Table 5: MeanNDCG results on MQ2007

Fold	GRankRLS	RankRLS	RankSVM
1	0.5228	0.5281	0.5278
2	0.4840	0.4841	0.4810
3	0.5056	0.5056	0.5042
4	0.4757	0.4754	0.4699
5	0.5033	0.5003	0.5003
avg	0.4983	0.4987	0.4966

Table 6: NDCG@10 results on MQ2007

Fold	GRankRLS	RankRLS	RankSVM
1	0.4735	0.4784	0.4818
2	0.4247	0.4246	0.4266
3	0.4466	0.4466	0.4461
4	0.4221	0.4221	0.4163
5	0.4487	0.4460	0.4485
avg	0.4431	0.4435	0.4439

Table 7: Map results on MQ2008

Fold	GRankRLS	RankRLS	RankSVM
1	0.4311	0.4524	0.4502
2	0.4239	0.4300	0.4213
3	0.4582	0.4542	0.4529
4	0.5283	0.5225	0.5284
5	0.5183	0.5006	0.4950
avg	0.4720	0.4719	0.4696

Table 8: P@10 results on MQ2008

Fold	GRankRLS	RankRLS	RankSVM
1	0.2333	0.2391	0.2423
2	0.2178	0.2217	0.2229
3	0.2363	0.2325	0.2357
4	0.2975	0.2949	0.2981
5	0.2484	0.2503	0.2465
avg	0.2467	0.2477	0.2491

Table 9: MeanNDCG results on MQ2008

Fold	GRankRLS	RankRLS	RankSVM
1	0.4454	0.4633	0.4577
2	0.4186	0.4269	0.4296
3	0.4787	0.4741	0.4686
4	0.5403	0.5407	0.5442
5	0.5369	0.5138	0.5159
avg	0.4840	0.4838	0.4832

Table 10: NDCG@10 results on MQ2008

Fold	GRankRLS	RankRLS	RankSVM
1	0.1920	0.2145	0.2117
2	0.1585	0.1669	0.1738
3	0.2558	0.2489	0.2494
4	0.2940	0.2874	0.2892
5	0.2254	0.2165	0.2155
avg	0.2251	0.2268	0.2279

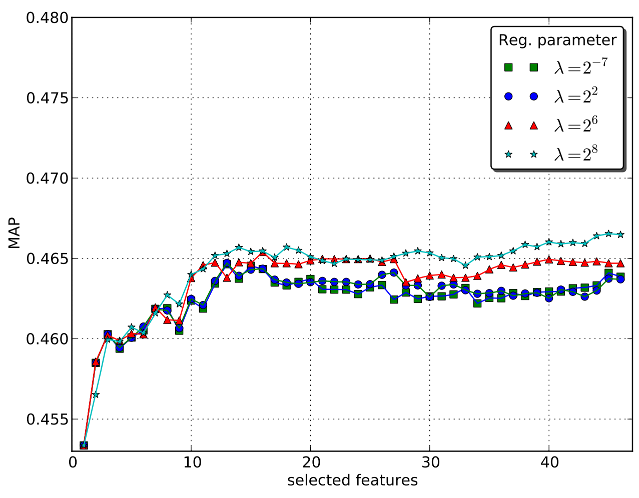


Figure 1: Average MAP on validation sets for MQ2007.

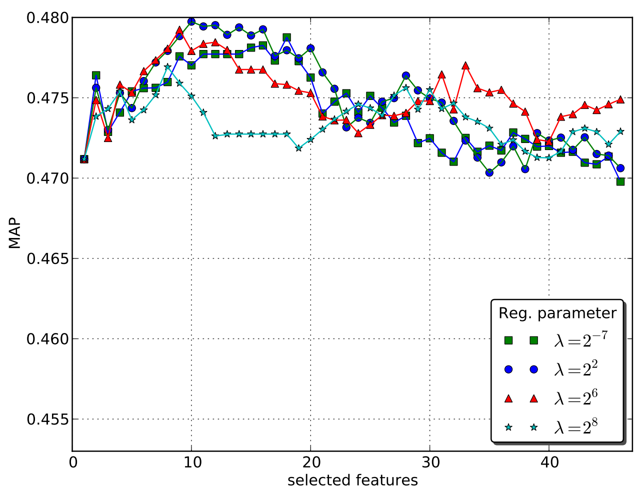


Figure 3: Average MAP on validation sets for MQ2008.

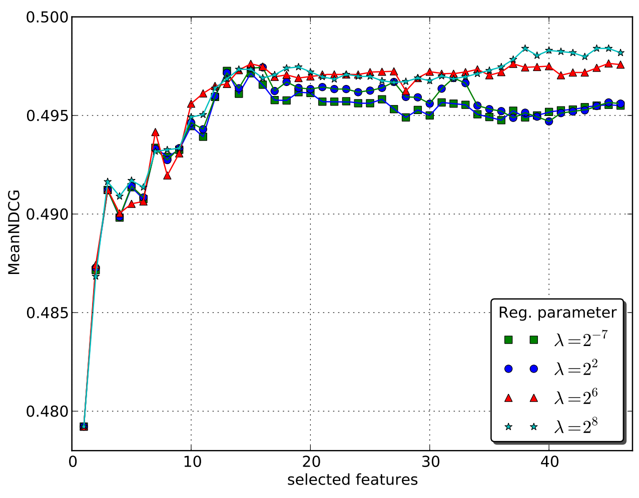


Figure 2: Average Mean NDCG on validation sets for MQ2007.

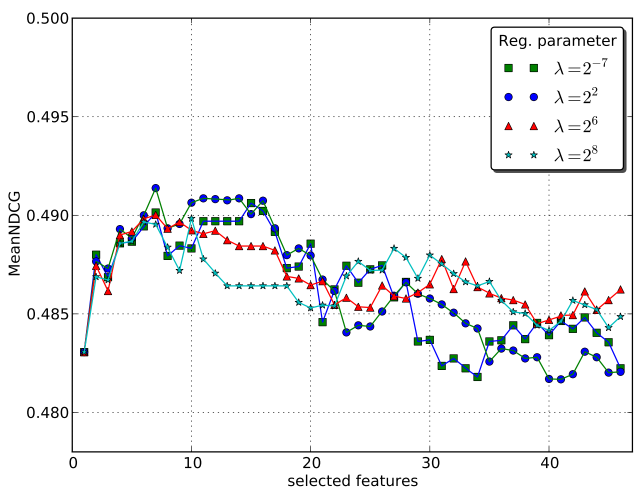


Figure 4: Average Mean NDCG on validation sets for MQ2008.

descriptions of these variations are left for future work.

The greedy forward selection approach can sometimes suffer from the so-called nesting effect, meaning that the best subset of size k, for example, may not necessarily cover the features included in the best subset of size $k-1$. Floating search methods (see e.g. [13, 20, 27]), which are able to discard features selected in previous iterations, have been proposed as a means to deal with this issue. Replacing the greedy search strategy with a floating search would be a fairly straightforward extension to the presented algorithm.

6. CONCLUSION

To conclude, we propose a computationally efficient method for learning sparse predictors for ranking tasks. The method uses on greedy forward selection as a search strategy and leave-query-out cross-validation as a selection criterion. The computational complexity of the method is linear in the number of training examples, in the overall number of features, and in the number of features to be selected. Thus, the method is computationally highly efficient despite the fact that the method optimizes a pairwise ranking loss function and uses a complex cross-validation criterion. Empirical evaluation with the LETOR benchmark data set demonstrates the soundness of the proposed approach.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments. This work has been supported by the Academy of Finland.

7. REFERENCES

[1] S. An, W. Liu, and S. Venkatesh. Fast cross-validation algorithms for least squares support vector machine and kernel ridge regression. *Pattern Recognition*, 40(8):2154–2162, 2007.

[2] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In L. D. Raedt and S. Wrobel, editors, *Proceedings of the 22nd*

international conference on Machine learning (ICML 2005), pages 89–96. ACM, 2005.

[3] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In Z. Ghahramani, editor, *Proceedings of the 24th international conference on Machine learning (ICML 2007)*, pages 129–136. ACM, 2007.

[4] O. Chapelle, Q. Le, and A. Smola. Large margin optimization of ranking measures. In *NIPS Workshop: Machine Learning for Web Search*, 2007.

[5] X. Geng, T.-Y. Liu, T. Qin, and H. Li. Feature selection for ranking. In C. L. Clarke, N. Fuhr, N. Kando, W. Kraaij, and A. P. de Vries, editors, *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR 2007)*, pages 407–414. ACM, 2007.

[6] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[7] H. V. Henderson and S. R. Searle. On deriving the inverse of a sum of matrices. *SIAM Review*, 23(1):53–60, 1981.

[8] T. Joachims. Optimizing search engines using clickthrough data. In D. Hand, D. Keim, and R. Ng, editors, *Proceedings of the 8th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2002)*, pages 133–142. ACM, 2002.

[9] T. Joachims. Training linear SVMs in linear time. In T. Eliassi-Rad, L. H. Ungar, M. Craven, and D. Gunopulos, editors, *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD 2006)*, pages 217–226. ACM, 2006.

[10] G. H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In W. W. Cohen and H. Hirsch, editors, *Proceedings of the Eleventh International Conference on Machine Learning (ICML 1994)*, pages 121–129, San Fransisco, CA, 1994. Morgan Kaufmann Publishers.

[11] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[12] D. A. Metzler. Automatic feature selection in the markov random field model for information retrieval. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management (CIKM'07)*, pages 253–262. ACM, 2007.

[13] T. Pahikkala, A. Airola, and T. Salakoski. Feature selection for regularized least-squares: New computational short-cuts and fast algorithmic implementations. In *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2010. To appear.

[14] T. Pahikkala, A. Airola, and T. Salakoski. Linear time feature selection for regularized least-squares, 2010. Preprint http://arxiv.org/abs/1003.3570.

[15] T. Pahikkala, J. Boberg, and T. Salakoski. Fast n-fold cross-validation for regularized least-squares. In T. Honkela, T. Raiko, J. Kortela, and H. Valpola, editors, *Proceedings of the Ninth Scandinavian Conference on Artificial Intelligence (SCAI 2006)*, pages 83–90. Otamedia, 2006.

[16] T. Pahikkala, E. Tsivtsivadze, A. Airola, J. Boberg, and J. Järvinen. An efficient algorithm for learning to rank from preference graphs. *Machine Learning*, 75(1):129–165, 2009.

[17] T. Pahikkala, E. Tsivtsivadze, A. Airola, J. Boberg, and T. Salakoski. Learning to rank with pairwise regularized least-squares. In T. Joachims, H. Li, T.-Y. Liu, and C. Zhai, editors, *SIGIR 2007 Workshop on Learning to Rank for Information Retrieval*, pages 27–33, 2007.

[18] T. Pahikkala, W. Waegeman, A. Airola, T. Salakoski, and B. De Baets. Conditional ranking on relational data. In *ECML PKDD '10: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, 2010. To appear.

[19] F. Pan, T. Converse, D. Ahn, F. Salvetti, and G. Donato. Feature selection for ranking using boosted trees. In D. W.-L. Cheung, I.-Y. Song, W. W. Chu, X. Hu, and J. J. Lin, editors, *Proceeding of the 18th ACM conference on Information and knowledge management (CIKM'09)*, pages 2025–2028. ACM, 2009.

[20] P. Pudil, J. Novovičová, and J. Kittler. Floating search methods in feature selection. *Pattern Recognition Letters*, 15(11):1119–1125, 1994.

[21] T. Qin, T.-Y. Liu, J. Xu, and H. Li. LETOR: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval*, 2010.

[22] A. J. Smola and P. Bartlett. Sparse greedy gaussian process regression. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 619–625. MIT Press, 2001.

[23] P. Vincent and Y. Bengio. Kernel matching pursuit. *Machine Learning*, 48(1–3):165–187, 2002.

[24] H. Yu, J. Oh, and W.-S. Han. Efficient feature weighting methods for ranking. In D. W.-L. Cheung, I.-Y. Song, W. W. Chu, X. Hu, and J. J. Lin, editors, *Proceeding of the 18th ACM conference on Information and knowledge management (CIKM'09)*, pages 1157–1166. ACM, 2009.

[25] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In W. Kraaij, A. P. de Vries, C. L. A. Clarke, N. Fuhr, and N. Kando, editors, *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR 2007)*, pages 271–278. ACM, 2007.

[26] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In W. B. Croft, D. J. Harper, D. H. Kraft, and J. Zobel, editors, *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR 2001)*, pages 334–342. ACM, 2001.

[27] T. Zhang. Adaptive forward-backward greedy algorithm for sparse learning with linear models. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1921–1928. MIT Press, 2009.

Extensions to Self-Taught Hashing: Kernelisation and Supervision

Dell Zhang
DCSIS
Birkbeck, University of London
Malet Street
London WC1E 7HX, UK
dell.z@ieee.org

Jun Wang
Dept of Computer Science
University College London
Gower Street
London WC1E 6BT, UK
jun.wang@cs.ucl.ac.uk

Deng Cai
State Key Lab of CAD&CG
Zhejiang University
100 Zijinggang Road
Hangzhou 310058, China
dengcai@cad.zju.edu.cn

Jinsong Lu
DEMS
Birkbeck, University of London
Malet Street
London WC1E 7HX, UK
jingsong.lu@gmail.com

ABSTRACT

The ability of fast similarity search at large scale is of great importance to many Information Retrieval (IR) applications. A promising way to accelerate similarity search is semantic hashing which designs compact binary codes for a large number of documents so that semantically similar documents are mapped to similar codes (within a short Hamming distance). Since each bit in the binary code for a document can be regarded as a binary feature of it, semantic hashing is essentially a process of generating a few most informative binary features to represent the documents. Recently, we have proposed a novel Self-Taught Hashing (STH) approach to semantic hashing (that is going to be published in SIGIR-2010): we first find the optimal l-bit binary codes for all documents in the given corpus via unsupervised learning, and then train l classifiers via supervised learning to predict the l-bit code for any query document unseen before. In this paper, we present two further extensions to our STH technique: one is kernelisation (i.e., employing nonlinear kernels to achieve nonlinear hashing), and the other is supervision (i.e., exploiting the category label information to enhance the effectiveness of hashing). The advantages of these extensions have been shown through experiments on synthetic datasets and real-world datasets respectively.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*data mining*; H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; I.2.6 [**Artificial Intelligence**]: Learning; I.5.2 [**Pattern Recognition**]: Design Methodology—*classifier design and evaluation*

General Terms

Algorithms, Experimentation, Performance

Keywords

Similarity Search, Semantic Hashing, Laplacian Eigenmap, Support Vector Machine, Feature Generation.

1. INTRODUCTION

The problem of *similarity search* (aka *nearest neighbour search*) is: given a query document, find its most similar documents from a very large document collection (corpus). It is of great importance to many Information Retrieval (IR) [26] applications, such as near-duplicate detection [18], plagiarism analysis [35], collaborative filtering [23], caching [28], and content-based multimedia retrieval [25].

Recently, with the rapid evolution of the Internet and the increased amounts of data to be processed, how to conduct fast similarity search at large scale has become an urgent research issue. A promising way to accelerate similarity search is *semantic hashing* [29] which designs compact binary codes for a large number of documents so that semantically similar documents are mapped to similar codes (within a short Hamming distance). It is extremely fast to perform similarity search over such binary codes, because we can simply return all the documents that are hashed into a tight Hamming ball centred around the binary code of the query document [34].

Since each bit in the binary code for a document can be regarded as a binary feature of it, semantic hashing is essentially a process of generating a few most informative binary features to represent the documents.

In our paper that is going to appear in the Proceedings of SIGIR-2010 [41], we have proposed a novel Self-Taught Hashing (STH) approach to semantic hashing: we first find the optimal l-bit binary codes for all documents in the given corpus via unsupervised learning, and then train l classifiers via supervised learning to predict the l-bit code for any query document unseen before. STH using binarised Laplacian Eigenmap (LapEig) [3] and linear Support Vector Machine (SVM) [22, 31] significantly outperforms state-of-the-art techniques in our experiments.

In this paper that has been accepted by the SIGIR-2010 Workshop on Feature Generation and Selection for Information Retrieval (FGSIR), we present two further extensions to our STH technique: one is kernelisation (i.e., employing nonlinear kernels to achieve nonlinear hashing), and the other is supervision (i.e., exploiting the category label information to enhance the effectiveness of hashing). The advantages of these extensions have been shown through our experiments on synthetic datasets and real-world datasets respectively.

The rest of this paper is organised as follows. In Section 2, we describe the related work. In Section 3, we review our recently proposed STH technique. In Section 4, we present two further extensions to STH, kernelisation and supervision, in details. In Section 5, we make conclusions.

2. RELATED WORK

There has been extensive research on fast similarity search due to its central importance in many applications. For a low-dimensional feature space, similarity search can be carried out efficiently with pre-built space-partitioning index structures (such as KD-tree) or data-partitioning index structures (such as R-tree) [8]. However, when the dimensionality of feature space is high (say > 10), similarity search aiming to return exact results cannot be done better than the naive method — a linear scan of the entire collection [37]. In the IR domain, documents are typically represented as feature vectors in a space of more than thousands of dimensions [26]. Nevertheless, if the complete exactness of results is not really necessary, similarity search in a high-dimensional space can be dramatically speeded up by using hash-based methods which are purposefully designed to approximately answer queries in virtually constant time [34].

Such hash-based methods for fast similarity search can be considered as a means for embedding high-dimensional feature vectors to a low-dimensional Hamming space (the set of all 2^l binary strings of length l), while retaining as much as possible the semantic similarity structure of data. Unlike standard dimensionality reduction techniques such as Latent Semantic Indexing (LSI) [5, 10] and Locality-Preserving Indexing (LPI) [17, 16], hashing techniques map feature vectors to *binary* codes, which is key to extremely fast similarity search (see Section 1). One possible way to get binary codes for text documents is to binarise the real-valued low-dimensional vectors (obtained from dimensionality reduction techniques like LSI) via thresholding [29]. An improvement on binarised-LSI that directly optimises a Hamming distance based objective function, namely Laplacian Co-Hashing (LCH), has been proposed recently [40].

The most well-known hashing technique that preserves similarity information is probably Locality-Sensitive Hashing (LSH) [1]. LSH simply employs random linear projections (followed by random thresholding) to map data points close in a Euclidean space to similar codes. It is theoretically guaranteed that as the code length increases, the Hamming distance between two codes will asymptotically approach the Euclidean distance between their corresponding data points. However, since the design of hash functions for LSH is *data-oblivious*, LSH may lead to quite inefficient (long) codes in practice [29, 38].

Several recently proposed hashing techniques attempt to overcome this problem by finding good *data-aware* hash functions through machine learning. In [29], the authors proposed to use stacked Restricted Boltzmann Machine (RBM) [19, 20], and showed that it was indeed able to generate compact binary codes to accelerate document retrieval. Researchers have also tried the boosting approach to Similarity Sensitive Coding (SSC) [32] and Forgiving Hashing (FgH) [2] — they first train AdaBoost [30] classifiers with similar pairs of items as positive examples (and also non-similar pairs of items as negative examples in SSC), and then take the output of all (decision stump) weak learners on a given document as its binary code. In [36], both stacked-RBM and boosting-SSC were found to work significantly better and faster than LSH when applied to a database containing tens of millions of images. In [38], a new technique called Spectral Hashing (SpH) was proposed. It has demonstrated significant improvements over LSH, stacked-RBM and boosting-SSC in terms of the number of bits required to find good similar items. However, in order to obtain the binary codes for query documents that are unseen before, SpH has to assume that the data are uniformly distributed in a hyper-rectangle, which is apparently very restrictive. In contrast, our proposed Self-Taught Hashing (STH) approach can work with any data distribution so it is much more flexible (see Section 3).

Table 1 gives a brief summary of the above mentioned typical techniques for accelerating similarity search.

3. A REVIEW OF STH

In our paper that is going to appear in the Proceedings of SIGIR-2010 [41], we have proposed a novel Self-Taught Hashing (STH) approach to semantic hashing. As illustrated in Figure 1, STH is a *general* learning framework that consists of two distinct stages: we first find the optimal l-bit binary codes for all documents in the given corpus via unsupervised learning, and then train l classifiers via supervised learning to predict the l-bit code for any query document unseen before. We call the approach "self-taught" because the hash function is learnt from the data that are auto-labelled by itself in the previous stage. We explain these two stages in details as follows.

3.1 Stage 1: Unsupervised Learning of Binary Codes

Given a collection of n documents which are represented as m-dimensional vectors $\{\mathbf{x}_i\}_{i=1}^n \subset \mathbb{R}^m$. Let X denote the $m \times n$ term-document matrix: $[\mathbf{x}_1, \ldots, \mathbf{x}_n]$. Suppose that the desired length of code is l bits. We use $\mathbf{y}_i \in \{-1, +1\}^l$ to represent the binary code for document vector $\mathbf{x}_i$, where the p-th element of $\mathbf{y}_i$, i.e., $y_i^{(p)}$, is $+1$ if the p-th bit of code is on, or -1 otherwise. Let Y denote the $n \times l$ matrix whose i-th row is the code for the i-th document, i.e., $[\mathbf{y}_1, \ldots, \mathbf{y}_n]^T$.

A "good" semantic hashing should be ***similarity preserving*** to ensure effectiveness. That is to say, semantically similar documents should be mapped to similar codes within a short Hamming distance.

Unlike the existing approaches (such as SpH [38]) that aim to preserve the *global* similarity structure of all document pairs, we focus on the *local* similarity structure, i.e.,

Table 1: **Typical techniques for accelerating similarity search.**

low-dimensional space	exact similarity search	data-aware	KD-tree, R-tree
high-dimensional space	approximate similarity search	data-oblivious	LSH
		data-aware	LSI, LCH, RBM, SSC, FgH, SpH, STH

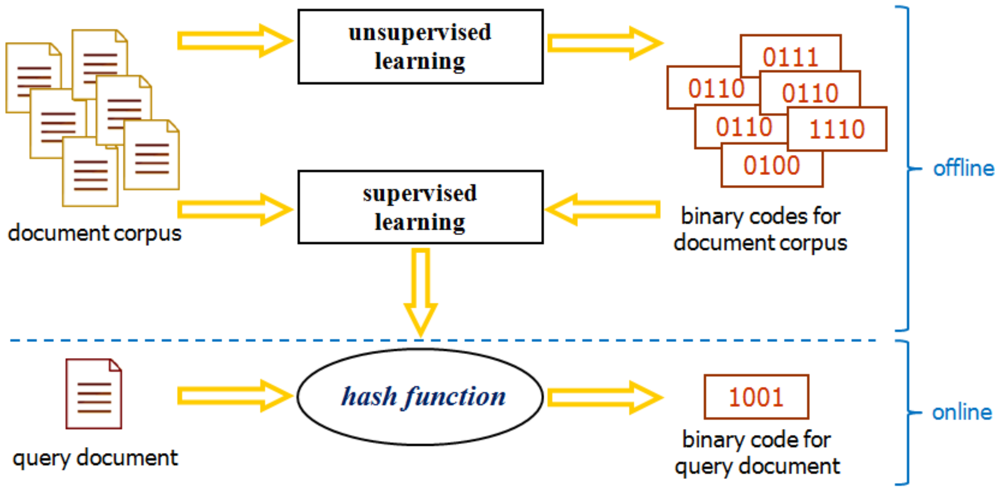


Figure 1: Our recently proposed Self Taught Hashing (STH) technique for accelerating similarity search [41].

k-nearest-neighbourhood, for each document. Since IR applications usually put emphasis on a small number of most similar documents for a given query document [26], preserving the global similarity structure is not only unnecessary but also likely to be sub-optimal for our problem. Therefore, we construct the $n \times n$ *local* similarity matrix W for the given corpus using the *cosine similarity* [26] or the *heat kernel* [3] as follows:

$$W_{ij} = \begin{cases} \left(\frac{\mathbf{x}_i^T}{\|\mathbf{x}_i\|}\right) \cdot \left(\frac{\mathbf{x}_j}{\|\mathbf{x}_j\|}\right) & \text{if } \mathbf{x}_i \in N_k(\mathbf{x}_j) \text{ or vice versa} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$W_{ij} = \begin{cases} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right) & \text{if } \mathbf{x}_i \in N_k(\mathbf{x}_j) \text{ or vice versa} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $N_k(\mathbf{x})$ represents the set of k-nearest-neighbours of document $\mathbf{x}$. In other words, W is the adjacency matrix of the k-nearest-neighbours graph for the given corpus [3]. A by-product of focusing on such a local similarity structure instead of the global one is that W becomes a sparse matrix. This not only leads to much lower storage overhead, but also brings a significant reduction to the computational complexity of subsequent operations. Furthermore, we introduce a diagonal $n \times n$ matrix D whose entries are given by $D_{ii} = \sum_{j=1}^{n} W_{ij}$. The matrix D provides a natural measure of document importance: the bigger the value of D_{ii} is, the more "important" is the document $\mathbf{x}_i$ as its neighbours are strongly connected to it [3].

The Hamming distance between two binary codes $\mathbf{y}_i$ and $\mathbf{y}_j$ (corresponding to documents $\mathbf{x}_i$ and $\mathbf{x}_j$) is given by the number of bits that are different between them, which can be calculated as $\frac{1}{4}\|\mathbf{y}_i - \mathbf{y}_j\|^2$. To meet the *similarity preserving* criterion, we seek to minimise the weighted average Hamming distance (as in SpH [38])

$$\frac{1}{4}\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij}\|\mathbf{y}_i - \mathbf{y}_j\|^2 \tag{3}$$

because it incurs a heavy penalty if two similar documents are mapped far apart. After some simple mathematical transformation, the above objective function can be rewritten in matrix form as $\frac{1}{4}\operatorname{Tr}(Y^T L Y)$, where $L = D - W$ is the *graph Laplacian* [7], and $\operatorname{Tr}(\cdot)$ means the matrix trace.

We found the above objective function (3) actually proportional to that of a well-known manifold learning algorithm, Laplacian Eigenmap (LapEig) [3], except that LapEig does not have the constraint $\mathbf{y}_i \in \{-1, +1\}^l$. So, if we relax this discreteness condition but just keep the *similarity preserving* requirement, we can get the optimal l-dimensional real-valued vector $\tilde{\mathbf{y}}_i$ to represent each document $\mathbf{x}_i$ by solving the following LapEig problem:

$$\begin{aligned} \arg\min_{\tilde{Y}} \quad & \operatorname{Tr}(\tilde{Y}^T L \tilde{Y}) \\ \text{subject to} \quad & \tilde{Y}^T D \tilde{Y} = I \\ & \tilde{Y}^T D \mathbf{1} = \mathbf{0} \end{aligned} \tag{4}$$

where $\operatorname{Tr}(\tilde{Y}^T L \tilde{Y})$ gives the real relaxation of the weighted average Hamming distance $\operatorname{Tr}(Y^T L Y)$, and the two constraints prevent the collapse into a subspace of dimension less than l. The solution of this optimisation problem is given by $\tilde{Y} = [\mathbf{v}_1, \ldots, \mathbf{v}_l]$ whose columns are the l eigenvectors corresponding to the smallest eigenvalues of the following generalised eigenvalue problem (except the trivial eigenvalue 0):

$$L\mathbf{v} = \lambda D\mathbf{v} \tag{5}$$

We now convert the above l-dimensional real-valued vectors $\tilde{\mathbf{y}}_1, \ldots, \tilde{\mathbf{y}}_n$ into binary codes via thresholding: if the p-th element of $\tilde{\mathbf{y}}_i$ is larger than the specified threshold, $y_i^{(p)} = +1$ (i.e., the p-th bit of the i-th code is on); otherwise, $y_i^{(p)} = -1$ (i.e., the p-th bit of the i-th code is off).

A "good" semantic hashing should also be ***entropy maximising*** to ensure efficiency, as pointed out by [2]. According to the *information theory* [33]: the maximal entropy of a source alphabet is attained by having a uniform probability distribution. If the entropy of codes over the corpus is small, it means that documents are mapped to only a small number of codes (hash bins), thereby rendering the hash table inefficient. To meet this *entropy maximising* criterion, we set the threshold for binarising $\tilde{y}_1^{(p)}, \ldots, \tilde{y}_n^{(p)}$ to be the *median* value of $\mathbf{v}_p$. In this way, the p-th bit will be on for half of the corpus and off for the other half. Furthermore, as the eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_l$ given by LapEig are orthogonal to each other, different bits $y^{(1)}, \ldots, y^{(l)}$ in the generated binary codes will be largely uncorrelated. Therefore this thresholding method gives each distinct binary code roughly equal probability of occurring in the document collection, thus achieves the best utilisation of the hash table.

3.2 Stage 2: Supervised Learning of Hash Function

Mapping all documents in the given corpus to binary codes does not completely solve the problem of semantic hashing, because we also need to know how to obtain the binary codes for query documents, i.e., new documents that are unseen before. This problem, called *out-of-sample extension* in manifold learning, is often addressed using the Nystrom method [4, 11]. However, calculating the Nystrom extension of a new document is as computationally expensive as an exhaustive similarity search over the corpus (that may contain millions of documents), which makes it impractical for semantic hashing. In LPI [17, 16], LapEig [3] is extended to deal with new samples by approximating a linear function to the embedding of LapEig. However, the computational complexity of LPI is very high because its learning algorithm involves eigen-decompositions of two large dense matrices. It is infeasible to apply LPI if the given training corpus is large. In SpH [38], new samples are handled by utilising the latest results on the convergence of graph Laplacian eigenvectors to the Laplace-Beltrami eigenfunctions of manifolds. It can achieve both fast learning and fast prediction, but it relies on a very restrictive assumption that the data are uniformly distributed in a hyper-rectangle.

Overcoming the limitations of the above techniques [4, 11, 17, 16, 38], we propose a novel method to compute the binary codes for query documents by considering it as a supervised learning problem: we think of each bit $y_i^{(p)} \in \{+1, -1\}$ in the binary code for document $\mathbf{x}_i$ as a binary class label (class-"on" or class-"off") for that document, and train a binary classifier $y^{(p)} = f^{(p)}(\mathbf{x})$ on the given corpus that has already been "labelled" by the above binarised-LapEig method, then we can use the learned binary classifiers $f^{(1)}, \ldots, f^{(l)}$ to predict the l-bit binary code $y^{(1)}, \ldots, y^{(l)}$ for any query document $\mathbf{x}$. As mentioned in the previous section, different bits $y^{(1)}, \ldots, y^{(l)}$ in the generated binary codes are uncorrelated. Hence there is no redundancy among the binary classifiers $f^{(1)}, \ldots, f^{(l)}$, and they can also be trained independently.

In STH, we choose to use the Support Vector Machine (SVM) [22, 31] algorithm to train these binary classifiers. SVM in its simplest form, linear SVM $f(\mathbf{x}) = \text{sgn}(\mathbf{w}^T\mathbf{x})$ consistently provides state-of-the-art performance for text classification tasks [12, 21, 39]. Given the documents $\mathbf{x}_1, \ldots, \mathbf{x}_n$ together with their *self-taught* binary labels for the p-th bit $y_1^{(p)}, \ldots, y_n^{(p)}$, the corresponding linear SVM can be trained by solving the following quadratic optimisation problem

$$\begin{aligned} \arg\min_{\mathbf{w},\xi_i} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{n}\sum_{i=1}^{n}\xi_i \qquad (6) \\ \text{subject to} \quad & y_i^{(p)}\mathbf{w}^T\mathbf{x}_i \geq 1 - \xi_i, \quad i = 1, \ldots, n \\ & \xi_i \geq 0, \quad i = 1, \ldots, n \end{aligned}$$

Thus, the *predicting* process of STH for a given query document is simply to classify the query document using those l learned classifiers and then assemble the output l binary labels into an l-bit binary code.

4. EXTENSIONS TO STH

4.1 Kernelisation

A prominent advantage of using SVM classifiers in the second stage of STH is that we can easily achieve nonlinear hashing if necessary by rewriting the quadratic optimisation problem (6) into its dual form

$$\begin{aligned} \arg\min_{\alpha} \quad & \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} y_i^{(p)}y_j^{(p)}\alpha_i\alpha_j\mathbf{x}_i^T\mathbf{x}_j \qquad (7) \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n \\ & \sum_{i=1}^{n}\alpha_i y_i^{(p)} = 0 \qquad (8) \end{aligned}$$

and replacing the inner product between $\mathbf{x}_i$ and $\mathbf{x}_j$ by a nonlinear kernel [31] such as the Gaussian kernel:

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right) \qquad (9)$$

Then the p-th bit (i.e., binary feature) of the binary code for a query document $\mathbf{x}$ would be given by

$$f(\mathbf{x}) = \text{sgn}\left(\sum_{i=1}^{n}\alpha_i y_i^{(p)} K(\mathbf{x}, \mathbf{x}_i)\right) \qquad (10)$$

which is a nonlinear mapping.

Here we would like to demonstrate the power of nonlinear hashing through experiments on a couple of synthetic datasets. In the following STH experiments, the parameter $k = 25$ when constructing the k-nearest-neighbours graph for LapEig, and the nonlinear SVM implementation is from LIBSVM [6] with the default parameter values except that the parameter σ for Gaussian kernel (9) is set to a small value 0.1 to facilitate the generation of flexible nonlinear hashing.

Let's first consider a simple "pie" dataset where the data points are uniformly distributed in a 2D unit circle. There are 1000 data points for training and another 1000 for testing. Suppose that we want to use 16-bit binary codes to encode all the documents, the hash function would need to encompass $l = 16$ mappings each of which corresponds to a bit (i.e., binary feature). Each such mapping, either linear or nonlinear, is essentially a *cut* of the dataset into two separate partitions: one with the corresponding bit on (1) and the other with the corresponding bit off (0). According to the *entropy maximising* criterion that we have explained in Section 3.1, the ideal cuts should be both balanced and orthogonal. If we choose to use only linear hashing, then it is impossible to keep more than 2 cuts balanced and meanwhile orthogonal. This is because each cut would be just one straight-line dividing the dataset, here the pie, into two partitions, as illustrated in Figure 2. If a straight-line cut is required to be balanced, then it should divide the pie into equal halves, so it must pass the centre point of the pie. Hence, l balanced linear cuts can only divide the pie into $2l = 2 \times 16 = 32$ sectors, which would lead to very inefficient (long) codes. This simple thought experiment reveals the severe limitation of all linear hashing techniques (including LSH, SSC, etc.). In contrast, SpH and STH (employing a nonlinear SVM with Gaussian Kernel) can achieve nonlinear hashing with cuts that are both balanced and orthogonal, as shown in Figure 3(a) and Figure 3(b) where the black colour represents bit 1 and the grey colour represents bit 0. Therefore, using 16-bit binary codes, SpH and STH would be able to effectively divide the pie into $2^l = 2^{16} = 65536$ small pieces. That is the reason why they can generate compact codes for large datasets. Furthermore, the nonlinear cuts of SpH turn out to be just straight-stripes, while those generated by STH are much more flexible. The superiority of STH to SpH becomes more apparent when we examine their 16-bit hash functions on the "two-moon" dataset, as shown in Figure 4(a) and Figure 4(b). Even though the dataset is in a very irregular shape, SpH keeps using straight-stripe cuts as the hash function, which results from its restrictive assumption that the data are uniformly distributed in a hyper-rectangle. STH, however, adapts its nonlinear cuts to the real distribution of the data, therefore provides better hash functions.

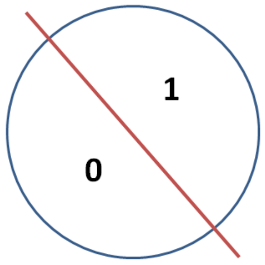


Figure 2: Linear hash functions as straight-line cuts of the dataset.

4.2 Supervision

The k-Nearest-Neighbours (kNN) algorithm [27] is a classic non-parametric machine learning method widely used for data classification. Its efficiency and scalability depend on the speed of similarity search over the large collection of labelled objects, i.e., training examples. By enabling fast similarity search at large scale, semantic hashing techniques like STH makes it feasible to exploit "the unreasonable effectiveness of data" [14] to accomplish traditionally difficult tasks. For example, researchers recently achieved great success in scene completion and scene recognition using millions of images on the Web as training data [15, 36].

The standard STH technique [41] is unsupervised. Although a supervised learning algorithm such as SVM is employed in the second stage of STH, it uses only the *pseudo*-labels input from the previous stage, but still consider the document collection unlabelled.

It is actually easy to extend our STH technique to incorporate the class label information available in the training set of documents for kNN classification. In the first stage of STH — unsupervised learning of binary codes (see Section 3.1) — we make use of the class label information in the construction of k-nearest-neighbour graph for LapEig: a training document $\mathbf{x}$'s k-nearest-neighbourhood $N_k(\mathbf{x})$ would only contain k documents in the same class as $\mathbf{x}$ that are most similar to $\mathbf{x}$. Let **STHs** denote such a supervised version of STH to distinguish it from the standard unsupervised version of STH.

One may ask why it is meaningful to use the kNN algorithm on top of STH that employs SVMs but not use the more powerful SVM algorithm to make classifications directly. The answer is that the KNN algorithm still has its advantages over SVMs in some aspects. As a non-parametric learning algorithm, kNN can yield increasingly complex decision boundaries given more and more training data. More importantly, the learning and prediction time of any multi-class SVM classification mechanism would grow at least linearly in the number of classes, whereas the kNN algorithm has no dependence on the number of classes. For example, if there are 1000 classes, the multi-class SVM approach may need 1000 binary SVM classifiers using the one-vs-rest ensemble scheme, but the STH plus kNN approach using 16-bit binary codes would only require 16 binary SVM classifiers.

We now empirically evaluate supervised version of STH (e.g., STHs), and compare its performance with vanilla STH as well as binarised-LSI [29], LCH [40], and SpH [38] that represents the state of the art (see Section 2). In the following STH and STHs experiments, the parameter $k = 25$ when constructing the k-nearest-neighbours graph for LapEig, and the linear SVM implementation is from LIBLINEAR [13] with the default parameter values.

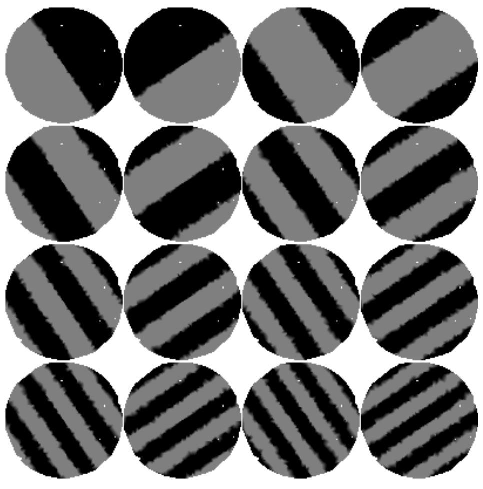
We have conducted experiments on three publicly available real-world text datasets: Reuters21578[1], 20Newsgroups[2] and TDT2[3].

The Reuters21578 corpus is a collection of documents that appeared on Reuters newswire in 1987. It contains 21578 documents in 135 categories. In our experiments, those documents appearing in more than one category were discarded, and only the largest 10 categories were kept, thus leaving us with 7285 documents in total. We use the ModeApte split here which gives 5228 (72%) documents for training and 2057 (28%) documents for testing.

The 20Newsgroups corpus was collected and originally used for document categorisation by Lang [24]. We use the popular 'bydate' version which contains 18846 documents, evenly distributed across 20 categories. The time-based split leads to 11314 (60%) documents for training and 7532 (40%) documents for testing.

The TDT2 (NIST Topic Detection and Tracking) corpus consists of data collected during the first half of 1998 and taken from 6 sources, including 2 newswires (APW, NYT), 2 radio programs (VOA, PRI) and 2 television programs

[1]http://www.daviddlewis.com/resources/testcollections/reuters21578/
[2]http://people.csail.mit.edu/jrennie/20Newsgroups/
[3]http://www.nist.gov/speech/tests/tdt/tdt98/index.htm


(a) SpH

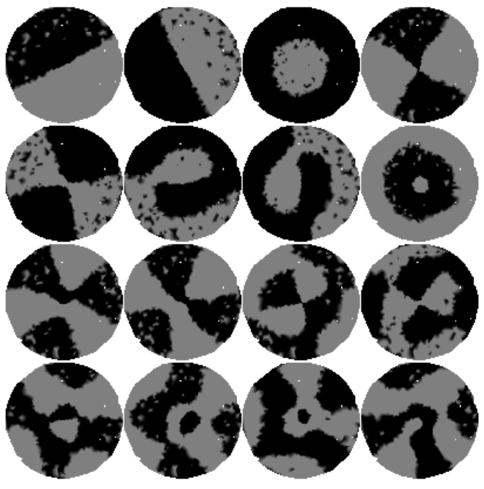
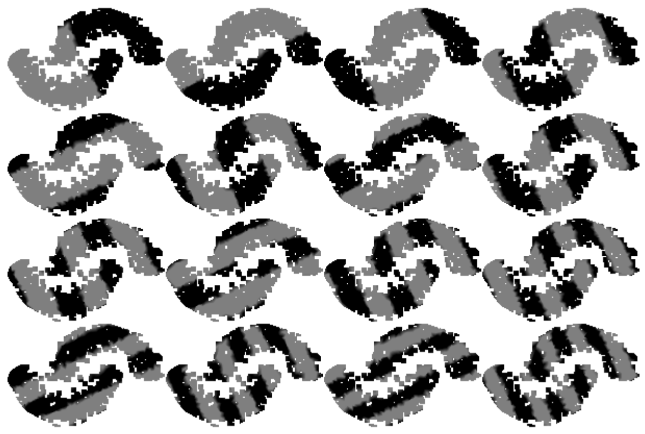

(b) STH

Figure 3: The 16-bit hash function for the pie dataset.


(a) SpH

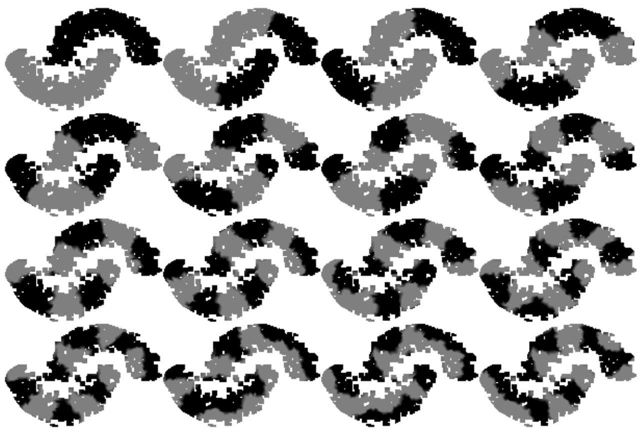

(b) STH

Figure 4: The 16-bit hash function for the two-moon dataset.

(CNN, ABC). It consists of 11201 on-topic documents which are classified into 96 semantic categories. In our experiments, those documents appearing in more than one category were discarded, and only the largest 30 categories were kept, thus leaving us with 9394 documents in total. We randomly selected 5597 (60%) documents for training and 3797 (40%) documents for testing. The averaged performance based on 10 such random selections is reported in this paper.

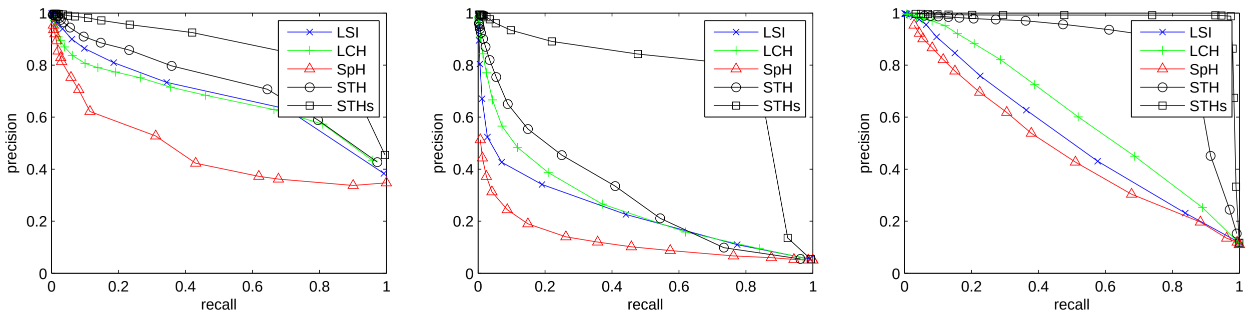
All the above datasets have been pre-processed by stopword removal, Porter stemming, and TF-IDF weighting [26].

Given a dataset, we use each document in the test set as a query to retrieve documents in the training set within a fixed Hamming ball of radius 1, and then measure the performance of a semantic hashing technique while varying the code length from 4-bit to 64-bit. Figure 5 shows the averaged *precision-recall curve* [26] for retrieving same-class documents. Figure 6 shows the accuracy for kNN classification using semantic hashing based similarity search, i.e., each test document is classified by the majority label in the Hamming ball centred around it. Here the number of nearest neighbours for kNN classification is actually not fixed at k, but depends on how many training documents are contained in the specific Hamming ball. It is clear that on all datasets and under both evaluation methodologies, STHs outperforms STH (and other semantic hashing techniques). Using 16-bit codes and Hamming ball radius 1, the performance improvements are all statistically significant (P value < 0.01) according to one-sided micro sign test (s-test) [39].

5. CONCLUSIONS

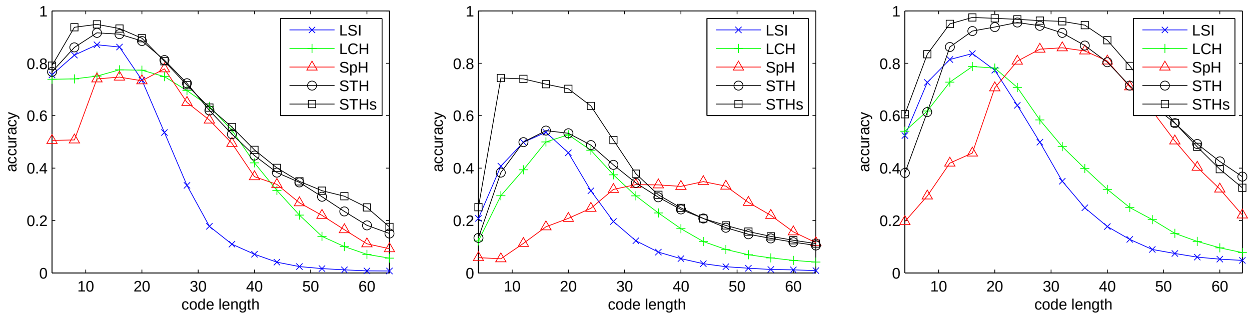
Our recently proposed STH technique for fast similarity search can be essentially considered as a process of generating a few most informative binary features to represent the documents.

The main contribution of this paper is to present two further extensions to STH, kernelisation and supervision, and moreover show their advantages through experiments on synthetic datasets and real-world datasets respectively. Although both extensions are just minor modifications, they



(a) Reuters21578 (b) 20Newsgroups (c) TDT2

Figure 5: The averaged precision-recall curve for retrieving same-class documents.



(a) Reuters21578 (b) 20Newsgroups (c) TDT2

Figure 6: The accuracy for kNN classification using semantic hashing based similarity search.

do work very well in practice to enhance the power or performance of STH.

The future work in our mind includes implementing the STH technique in the MapReduce [9] framework and apply it to large-scale content-based multimedia retrieval [25].

6. ACKNOWLEDGEMENTS

We are grateful to Dr Xi Chen (Alberta) for his valuable discussion and the London Mathematical Society (LMS) for their support of this work (SC7-09/10-6). We would also like to thank the anonymous reviewers for their helpful comments.

7. REFERENCES

[1] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 459–468, Berkeley, CA, USA, 2006.

[2] S. Baluja and M. Covell. Learning to hash: Forgiving hash functions and applications. *Data Mining and Knowledge Discovery (DMKD)*, 17(3):402–430, 2008.

[3] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[4] S. Belongie, C. Fowlkes, F. Chung, and J. Malik. Spectral partitioning with indefinite kernels using the nystrom extension. In *Proceedings of the 7th European Conference on Computer Vision (ECCV)*, pages 531–542, Copenhagen, Denmark, 2002.

[5] M. W. Berry, S. T. Dumais, and G. W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):573–595, 1995.

[6] C.-C. Chang and C.-J. Lin. *LIBSVM: a Library for Support Vector Machines*, 2001.

[7] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 2nd edition, 2001.

[9] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *Proceedings of the 6th Symposium on Operating System Design and Implementation (OSDI)*, pages 137–150, San Francisco, CA, USA, 2004.

[10] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science (JASIS)*, 41(6):391–407, 1990.

[11] P. Drineas and M. W. Mahoney. On the nystrom method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research (JMLR)*, 6:2153–2175, 2005.

[12] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the 7th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 148–155, Bethesda, MD, 1998.

[13] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[14] A. Y. Halevy, P. Norvig, and F. Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.

[15] J. Hays and A. A. Efros. Scene completion using millions of photographs. *ACM Transactions on Graphics (TOG)*, 26(3):4, 2007.

[16] X. He, D. Cai, H. Liu, and W.-Y. Ma. Locality preserving indexing for document representation. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 96–103, Sheffield, UK, 2004.

[17] X. He and P. Niyogi. Locality preserving projections. In *Advances in Neural Information Processing Systems (NIPS)*, volume 16, pages 153–160, Vancouver and Whistler, Canada, 2003.

[18] M. R. Henzinger. Finding near-duplicate web pages: A large-scale evaluation of algorithms. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 284–291, Seattle, WA, USA, 2006.

[19] G. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[20] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, July 2006.

[21] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning (ECML)*, pages 137–142, Chemnitz, Germany, 1998.

[22] T. Joachims. *Learning to Classify Text using Support Vector Machines.* Kluwer, 2002.

[23] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 426–434, Las Vegas, NV, USA, 2008.

[24] K. Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning (ICML)*, pages 331–339, Tahoe City, CA, 1995.

[25] M. S. Lew, N. Sebe, C. Djeraba, and R. Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 2(1):1–19, 2006.

[26] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval.* Cambridge University Press, 2008.

[27] T. Mitchell. *Machine Learning.* McGraw Hill, international edition, 1997.

[28] S. Pandey, A. Broder, F. Chierichetti, V. Josifovski, R. Kumar, and S. Vassilvitskii. Nearest-neighbor caching for content-match applications. In *Proceedings of the 18th International Conference on World Wide Web (WWW)*, pages 441–450, Madrid, Spain, 2009.

[29] R. Salakhutdinov and G. Hinton. Semantic hashing. *International Journal of Approximate Reasoning (IJAR)*, 50(7):969–978, 2009.

[30] R. E. Schapire. The boosting approach to machine learning: An overview. In *Nonlinear Estimation and Classification.* Springer, 2003.

[31] B. Scholkopf and A. J. Smola. *Learning with Kernels.* MIT Press, Cambridge, MA, 2002.

[32] G. Shakhnarovich, P. A. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *Proceedings of the 9th IEEE International Conference on Computer Vision (ICCV)*, pages 750–759, Nice, France, 2003.

[33] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[34] B. Stein. Principles of hash-based text retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 527–534, Amsterdam, The Netherlands, 2007.

[35] B. Stein, S. M. zu Eissen, and M. Potthast. Strategies for retrieving plagiarized documents. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 825–826, Amsterdam, The Netherlands, 2007.

[36] A. B. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, Anchorage, AK, USA, 2008.

[37] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of 24th International Conference on Very Large Data Bases (VLDB)*, pages 194–205, New York City, USA, 1998.

[38] Y. Weiss, A. B. Torralba, and R. Fergus. Spectral hashing. In *Advances in Neural Information Processing Systems (NIPS)*, volume 21, pages 1753–1760, Vancouver, Canada, 2008.

[39] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 42–49, Berkeley, CA, 1999.

[40] D. Zhang, J. Wang, D. Cai, and J. Lu. Laplacian co-hashing of terms and documents. In *Proceedings of the 32nd European Conference on IR Research (ECIR)*, pages 577–580, Milton Keynes, UK, 2010.

[41] D. Zhang, J. Wang, D. Cai, and J. Lu. Self-taught hashing for fast similarity search. In *Proceedings of the 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, Geneva, Switzerland, 2010.

KEYNOTE — Hierarchical Bayesian Models of Language and Text

Yee Whye Teh ywteh@gatsby.ucl.ac.uk
Gatsby Computational Neuroscience Unit
University College of London, UK

In this talk I will present a new approach to modelling sequence data called the sequence memoizer. As opposed to most other sequence models, our model does not make any Markovian assumptions. Instead, we use a hierarchical Bayesian approach which enforces sharing of statistical strength across the different parts of the model. To make computations with the model efficient, and to better model the power-law statistics often observed in sequence data, we use a Bayesian nonparametric prior called the Pitman-Yor process as building blocks in the hierarchical model. We show state-of-the-art results on language modelling and text compression.

This is joint work with Frank Wood, Jan Gasthaus, Cedric Archambeau and Lancelot James.

Biography

Yee Whye Teh is a Lecturer (equivalent to an assistant professor in US system) at the Gatsby Computational Neuroscience Unit, UCL. He is interested in machine learning and Bayesian statistics. His current focus is on developing Bayesian nonparametric methodologies for unsupervised learning, computational linguistics, and genetics. Prior to his appointment he was Lee Kuan Yew Postdoctoral Fellow at the National University of Singapore and a postdoctoral fellow at University of California at Berkeley. He obtained his Ph.D. in Computer Science at the University of Toronto in 2003. He is programme co-chair of AISTATS 2010.

Feature Selection for Document Ranking using Best First Search and Coordinate Ascent

Van Dang and W. Bruce Croft
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
{vdang, croft}@cs.umass.edu

ABSTRACT

Feature selection is an important problem in machine learning since it helps reduce the number of features a learner has to examine and reduce errors from irrelevant features. Even though feature selection is well studied in the area of classification, this is not the case for ranking algorithms. In this paper, we propose a feature selection technique for ranking based on the wrapper approach used in classification. Our method uses the best first search strategy incrementally to partition the feature set into subsets. Features in each subset are then combined into a single feature using coordinate ascent in such a way that it maximizes any defined retrieval measure on a training set. Our experiments with many state-of-the-art ranking algorithms, namely RankNet, RankBoost, AdaRank and Coordinate Ascent, have shown that the proposed method can reduce the original set of features to a much more compact set while at least retaining the ranking effectiveness regardless of the ranking method in use.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Selection process

General Terms

Algorithms, Measurement, Performance, Experimentation.

Keywords

Learning to rank, feature selection

1. INTRODUCTION

An effective ranking framework is certainly the core component of any information retrieval (IR) system. Many ranking models have been proposed, the most popular of which are *BM25* [13] and the language modeling framework [6]. These models make use of a small number of features such as term frequency, inverse document frequency and document length. They have the advantage of being fast and produce reasonably good resutls. When more features become available, however, incorporating them into these models is usually difficult since it requires a significant change in the underlying model. For example, the BM25 model was modified to include PageRank as a prior [5] or to incorporate term proximity information [3].

Supervised learning to rank algorithms [10, 8, 1, 14, 7] can help overcome that limitation. They treat query-document pairs as objects to rank, each of which is represented using any set of features. Existing work has shown, by incorporating many features, they produce better results than classical models mentioned above [14, 7].

In the area of machine learning, feature selection is the task of selecting a subset of features to be considered by the learner. This is important since learning with too many features is wasteful and even worse, learning from the wrong features will make the resulting learner less effective. In the classification problem, feature selection approaches can be divided into three categories: the *filter* approach which selects features based on some criterion that is independent of the metric being optimized, the *wrapper* approach which picks features that, with the learning technique in use, produces the best result with respect to the metric being considered, and the *embedding* approach which embeds the feature selection procedure into the learning process.

Even though feature selection for classification is well studied, there has been less research on this topic for ranking. The most recent technique that we are aware of is the work by Geng et al. [9] which follows the *filter* approach. Instead, we prefer the *wrapper* method since the selection of features is based on the effective metric that will be optimized by the learning procedure. Our approach uses *best first search* to come up with subsets of features and uses *coordinate ascent* to learn the weights for those features. Once a best subset is obtained, a new feature is defined based on the learned combination of features in this set and these features are removed from the feature pool. The process is repeated until all features are considered. The set of new features will be used to train the ranker instead of the original features. Our experiments with four well-known ranking algorithms – RankNet [1], RankBoost [8], AdaRank [14] and Coordinate Ascent [7] – show that the set of new features, while being much more compact, is at least as effective as the original set in terms of *NDCG@5*.

2. PROPOSED METHOD

Our proposed method is a simple modification of the standard *wrapper* approach that can be found in [11].

Table 1: A Best-First-Search procedure. $R = 5$ is used in our experiments.

```
$P = \emptyset$
$best = null$
Randomly pick a node $v$
Train $\mathcal{A}$ on $\mathcal{T}$ using $v$ to maximize $\Lambda(\mathcal{T}, \mathcal{A}_v)$
Add $v$ to $P$
while $|P| > 0$
    $v \leftarrow \arg\max_{u \in P} \Lambda(\mathcal{T}, \mathcal{A}_u)$
    Remove $v$ from $P$
    if $\Lambda(\mathcal{T}, \mathcal{A}_v) > \Lambda(\mathcal{T}, \mathcal{A}_{best})$ then $best \leftarrow v$
    if $best$ did not change in the last $R$ rounds
        then STOP and RETURN $best$
    for each of $v$'s neighbors $u$
        Train $\mathcal{A}$ on $\mathcal{T}$ using $u$ to maximize $\Lambda(\mathcal{T}, \mathcal{A}_u)$
        add $u$ to $P$
    end for
end while
RETURN $best$
```

2.1 Notation

Let $\mathcal{F} = \{f_1, f_2, ..., f_n\}$ be the set of original features. Let $\mathcal{T} = \{r_1, r_2, ..., r_m\}$ be the set of training samples where r_i is the list of documents for the query q_i. Let d_i^j be the j-th document in the list r_i. Each d_i^j is represented as a feature vector $\{f_1^{(i)(j)}, f_2^{(i)(j)}, ..., f_n^{(i)(j)}\}$

Let $\mathcal{A}_{\mathcal{F}}$ be any ranking algorithm that utilizes the set of features $\mathcal{F}$. To rank a list of documents r_i using $\mathcal{A}_{\mathcal{F}}$, we reorder all d_i^j based on $\mathcal{A}_{\mathcal{F}}(d_i^j)$, which is the score the ranker assigns to this document.

Let Λ be the metric that the learner tries to optimize. It can be any effectiveness metric such as *NDCG* or *MAP*. We define $\Lambda(\mathcal{T}, \mathcal{A}_{\mathcal{F}})$ to be the metric score that the ranking algorithm $\mathcal{A}$ using the set of features $\mathcal{F}$ achieves on the dataset $\mathcal{T}$. Note that the goal of the training process is to learn $\mathcal{A}$ such that it maximizes $\Lambda(\mathcal{T}, \mathcal{A}_{\mathcal{F}})$.

2.2 Method

The goal of our technique is to partition $\mathcal{F}$ in a greedy way into k non-overlapping subsets $\{\mathcal{F}_1, \mathcal{F}_2, ..., \mathcal{F}_k\}$ – each of which has size of at most s – together with a set of learned rankers $\{\mathcal{A}_{\mathcal{F}_1}, \mathcal{A}_{\mathcal{F}_2}, ..., \mathcal{A}_{\mathcal{F}_k}\}$ where each $\mathcal{A}_{\mathcal{F}_t}$ is trained to maximize $\Lambda(\mathcal{T}, \mathcal{A}_{\mathcal{F}_t})$ using the set of features $\mathcal{F}_t$.

We do that in a slightly different way than the method described in [11]. We first put all the features into a pool. We build an undirected graph where each node represents a subset of features in that pool that has size at most s. An edge is created for any pairs of nodes where one of them can be obtained from the other by adding *exactly one* feature. Then we apply the best first search procedure described in Table 1 to come up with $\mathcal{F}_1$ and $\mathcal{A}_{\mathcal{F}_1}$ ($R = 5$ is used in all of our experiments). All features in $\mathcal{F}_1$ are then removed from the pool. We rebuild the graph from the remaining features and repeat the same procedure until the pool is empty.

Once the feature selection procedure is done, we define a new feature f'_t corresponding to each $\mathcal{F}_t$ such that $f'^{(i)(j)}_t = \mathcal{A}_{\mathcal{F}_t}(d_i^j)$. Therefore, each original feature vector $\{f_1^{(i)(j)}, f_2^{(i)(j)}, ..., f_n^{(i)(j)}\}$ becomes $\{f'^{(i)(j)}_1, f'^{(i)(j)}_2, ..., f'^{(i)(j)}_k\}$ where $k < n$. The learning process then proceeds with this new feature vector.

3. RANKING METHODS

To evaluate our feature selection technique, we test it with four popular ranking algorithms: RankNet [1], RankBoost [8], AdaRank [14] and Coordinate Ascent [7].

3.1 RankNet

RankNet [1] is a probabilistic pair-wise ranking framework based on neural networks. For every pair of correctly ranked documents, each document is propagated through the net separately. The difference between the two outputs are mapped to a probability by the logistic function. The cross entropy loss is then computed from that probability and the true label for that pair. Next, all weights in the network are updated using the error back propagation and the gradient descent method.

3.2 RankBoost

RankBoost [8] is a pair-wise boosting technique for ranking. Training proceeds in rounds. It starts with all document pairs being assigned with an equal weight. At each round, the learner selects the weak ranker that achieves the smallest pair-wise loss on the training data with respect to the current weight distribution. Pairs that are correctly ranked have their weight decreased and those that are incorrectly ranked have their weight increased so that the learner will focus more on the hard samples in the next round. The final model is essentially a linear combination of weak rankers. Weak rankers theoretically can be of any type but they are most commonly chosen as a binary function with a single feature and a threshold. This function assigns a score of 1 to a document of which feature value exceeds the threshold and 0 otherwise.

3.3 AdaRank

The idea of AdaRank [14] is similar to that of RankBoost except that it is a list-wise approach. Hence, it directly maximizes any desired IR metric such as *NDCG* and *MAP* whereas RankBoost's objective is to minimize the pair-wise loss.

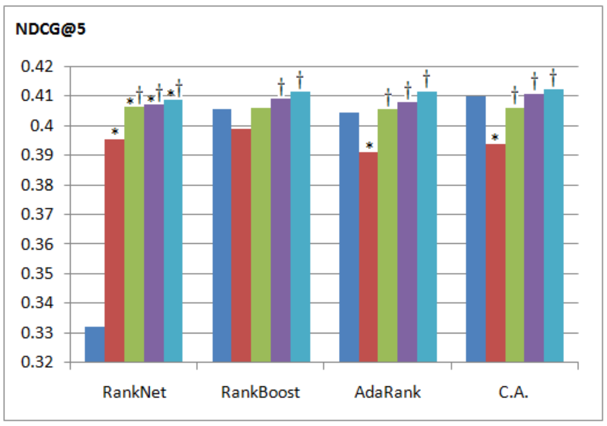
3.4 Coordinate Ascent

Metzler and Croft [7] have proposed a list-wise linear model for information retrieval which uses coordinate ascent to optimize the model's parameters. Coordinate ascent is a well-known technique for unconstrained optimization. It optimizes multivariate objective functions by sequentially doing optimization in one dimension at a time. It cycles through each parameter and optimizes over it while fixing all the others. Note that in the context of this paper, we use the term *Coordinate Ascent* to refer to this particular ranking technique other than the general optimization method.

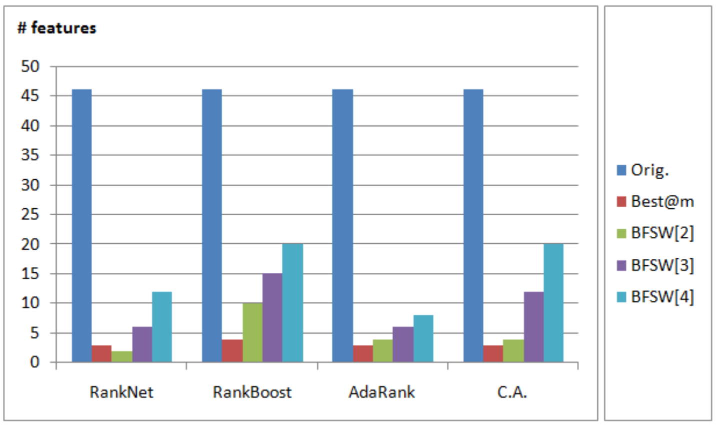
4. EXPERMENTS

4.1 Datasets

We conduct our experiments on the LETOR 4.0 dataset. It was created from the Gov-2 document collection which contains roughly 25 million web pages. Two query sets from Million Query TREC 2007 and 2008 are included in this dataset, referred to as *MQ2007* and *MQ2008* respectively. *MQ2007* contains roughly 1700 queries and *MQ2008* has about 800. Each query-document pair is represented by a 46-feature vector.



(a) NDCG@5



(b) # Features used

Figure 1: Results on the *MQ2007* dataset: (a) Performance of rankers using different set of features and (b) the number of features they use. The first and the second bar in each group correspond to the system using the set of all original features and the set produced by `best@m` respectively. The last three bars are for systems using features generated by our approach with s equals 2, 3, and 4 respectively. * and † indicate significant difference to `orig.` and `best@m` respectively.

4.2 Experimental Design

For our feature selection procedure, we experiment with different values for s (the *size* of the feature subset $\mathcal{F}_i$) varying from 2 to 4. For each value of s, we consider using top-k subsets $\{\mathcal{F}_1, \mathcal{F}_2, ..., \mathcal{F}_k\}$ where $k \in \{1..5\}$ to train the ranker and use a validation set to select k.

Holding the ranker fixed to a particular technique, we compare systems with different s values to the baseline which uses all original features. We will refer to the baseline as `orig.` and systems using features generated by our method as `BFSW`.

We also consider another baseline in which we use each feature from the original set to rank all the list of documents and sort them based on the *NDCG@5* they produce. We then use the top-m of these features to train the ranker. m is also chosen in the range $\{1..5\}$ based on a validation set. The idea is to see whether simply considering only some of the best features from the original set gives better results than using the whole set, and whether our method can do any better than that. This second baseline will be refered to as `best@m`.

All experiments are done using five-fold cross validation. The dataset in each fold is divided into three subsets: the training set, the validation set and the test set. We train all the systems on the training set and select the model that has the best performance on the validation set as the final model which is evaluated on the test set. *NDCG@5* averaged over five folds is used as the performance measure for each system.

4.3 Parameter Settings

We implement our proposed technique as described in section 2.2. Though the ranking algorithm $\mathcal{A}$ used by this procedure can be freely chosen (e.g. to be the same as whatever used in learning process), we fix it to *Coordinate Ascent* for simplicity.

For both *RankNet* and *RankBoost*, we train the ranker for 300 rounds since we observed no performance change after that in our preliminary experiments. In our implementation of *RankNet*, we use the logistic function as its activation function. We set the learning rate to be 0.001 which is halved everytime the cross entropy loss on training data increases as suggested in [1]. For *AdaRank*, we train the ranker until observing a drop in *NDCG@5* between two consecutive rounds that is smaller than a *tolerance* of 0.002. Since *AdaRank* might have the problem of selecting the same feature again and again, we also apply the trick provided in [4]. For *Coordinate Ascent*, we train the ranker with 5 random restarts to avoid local extrema.

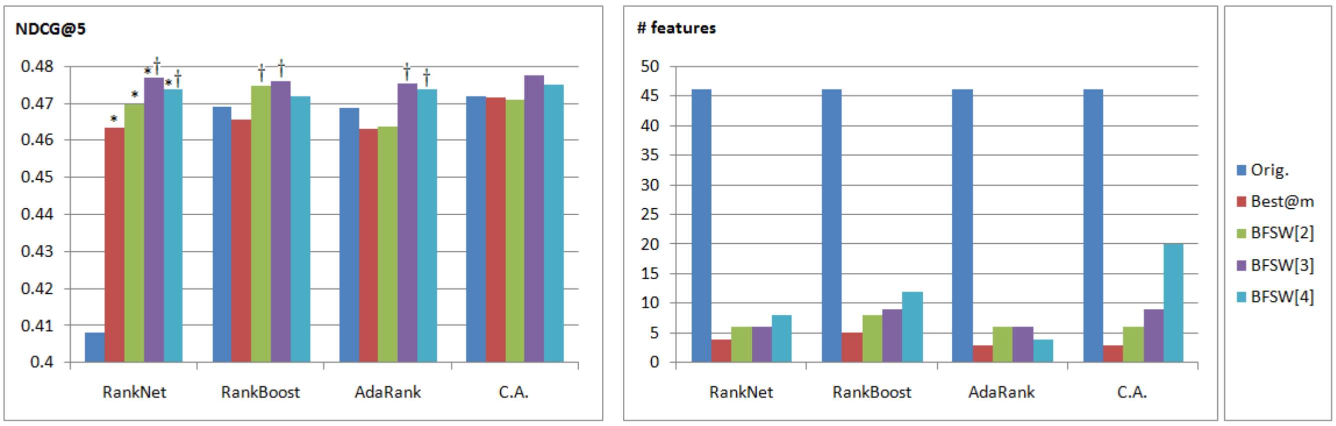
4.4 Results

Fig. 1a demonstrates the results on *MQ2007*. Each group of bars corresponds to one ranking algorithm. Within each group, the first and the second bar show the results obtained using the original set of features and the set of features produced by `best@m` respectively. The last three bars indicates the performance of systems using features generated by our method with s set to 2, 3 and 4 respectively.

It is worth noting from Fig. 1a that with original features, *RankNet* is less effective than other learning to rank algorithms. *RankNet* is based on neural networks which are known to be hard to train. Many important tricks are pointed out in [12]. For this paper, however, we implemented it in a rather straight-forward way. This might be the reason for the bad performance.

The second thing to note from Fig. 1a is that `best@m` is almost always worse than using the original features except for the case of *RankNet*. This suggests that all features are important to some extent and simply using the top-performing features will not help.

Features provided by our selection method with $s \geq 3$, on the other hand, are always more effective than the set of original features and $s = 4$ gives the best *NDCG@5* across learning techniques. Meanwhile, in terms of the number of features the learner needs to consider, Fig. 1b shows that the set generated by our method is much more compact



(a) NDCG@5 **(b) # Features used**

Figure 2: Results on the *MQ2008* dataset: (a) Performance of rankers using different set of features and (b) the number of features they use. The first and second bar in each group correspond to the system using the set of all original features and the set produced by `Best@m` respectively. The last three bars are for systems using features generated by our approach with s equals 2, 3, and 4 respectively. * and † indicate significant difference to `orig.` and `best@m` respectively.

compared to the original set.

We did hypothesis testing using the two-tailed t-test and note that at $p < 0.05$, the improvement our method provide over `orig.` is not significant except in the case of *RankNet*. The performance drop that `best@m` introduces, on the other hand, is significant. Thus, the difference between our method and `best@m` is almost always significant as indicated in Fig. 1a, suggesting that our method is much better than only considering the top-performing features.

One of the goals of any feature selection method is to get the learner to concentrate on important features and neglect those that are not. Our method works by optimizing a small group of features locally then collapsing them into a single features. The learner then only needs to consider a smaller group – compared to the original set – of better features. This is our explanation for the consistency of our results across learning techniques. The results with *RankNet*, in fact, further supports our claim. While its performance using the orginal features is quite bad since we do not apply any tricks, it becomes comparable to other ranking techniques when using feature selection. This indicates that the learning task is easier with fewer features.

Fig. 2 shows results obtained on the *MQ2008* dataset. These reveal similar trends except that the system with $s = 3$ performs better than $s = 4$.

5. CONCLUSIONS

In this paper, we propose a simple wrapper-based method that uses best first search and coordinate ascent to greedily partition a set of features into subsets. Each subset is then replaced by a single new feature. Our results on the LETOR 4.0 dataset have shown that learning from the set of new features, which is much smaller in size, can produce comparable or even better *NDCG@5* performance than learning from the original features.

Future work includes looking into more recent work in feature selection for classification, experimenting with other ranking algorithms such as SVMRank [10] and LambdaRank [2] and comparing our method to other feature selection techniques such as the one proposed in [9].

6. ACKNOWLEDGMENT

This work was supported in part by the Center for Intelligent Information Retrieval and in part by NSF grant #IIS-0711348. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

7. REFERENCES

[1] C.J.C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton and G. Hullender. Learning to rank using gradient descent. In *Proc. of ICML*, pages 89-96, 2005.

[2] C.J.C. Burges, R. Ragno, and Q.V. Le. Learning to rank with nonsmooth cost functions. In *Proc. of NIPS*, 2006.

[3] S. Buttcher, C. L. A. Clarke, and B. Lushman. Term proximity scoring for ad-hoc retrieval on very large text collections. In *Proc. of SIGIR*, pages 621-622, 2006.

[4] M. Cartright, J. Seo and M. Lease. UMass Amherst and UT Austin @ The TREC 2009 Relevance Feedback Track. In *Proc. of TREC*, 2009.

[5] N. Craswell, S. Robertson, H. Zaragoza, and M. Taylor. Relevance weighting for query independent evidence. In *Proc. of SIGIR*, pages 416-423, 2005.

[6] W.B. Croft, D. Metzler, and T. Strohman. *Search Engines: Information Retrieval in Practice.* Addison-Wesley, 2009.

[7] D. Metzler and W.B. Croft. Linear feature-based models for information retrieval. *Information Retrieval*, 10(3): 257-274, 2000.

[8] Y. Freund, R. Iyer, R. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *The Journal of Machine Learning Research*, 4: 933-969, 2003.

[9] X. Geng, T.Y. Liu, T. Qin and H. Li. Feature selection for ranking. In *Proc. of SIGIR*, pages 407-414, 2007.

[10] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of KDD*, pages 133-142, 2002.

[11] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 1997.

[12] Y. LeCun, L. Bottou, G.B. Orr and K.R. Muller. Efficient Backprop. *Neural Networks: Tricks of the trade*, 1998.

[13] S. E. Robertson and D. A. Hull. The TREC-9 filtering track final report. In *TREC*, pages 25-40, 2000.

[14] J. Xu and H. Li. AdaRank: a boosting algorithm for information retrieval. In *Proc. of SIGIR*, pages 391-398, 2007.

Enhancing Query-oriented Summarization based on Sentence Wikification

Yajie Miao, Chunping Li
School of Software
Tsinghua University
Beijing 100084, China

yajiemiao@gmail.com, cli@tsinghua.edu.cn

ABSTRACT

Query-oriented summarization is primarily concerned with synthesizing an informative and well-organized summary from a document collection for a given query. In the existing summarization methods, each individual sentence in the document collection is represented as Bag of Words (BOW). In this paper, we propose a novel framework which improves query-oriented summarization via sentence wikification, i.e., enriching sentence representation with Wikipedia concepts. Furthermore, we exploit semantic relatedness of Wikipedia concepts as a smoothing factor in sentence wikification. The experiments with benchmark dataset show that sentence wikification performs effectively for improving query-oriented summarization, and helps to generate more high-quality summaries.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *Retrieval models*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing – *Text analysis.*

General Terms

Algorithms, Performance, Experimentation.

Keywords

Query-oriented Summarization, Sentence Wikification, Semantic Relatedness.

1. INTRODUCTION

Given a specific query, query-oriented summarization (QS) aims to automatically extract salient information from a document set and assemble them into concise answers in natural language. This task subsumes interesting applications. For instance, though search engines can respond to users' queries by returning lists of web pages, browsing the pages for desired information is either time-consuming or unachievable. In this case, it would be nice if we summarize the points on the target query, which helps users to digest the returned pages easily. A distinct characteristic of QS is that the sentences included in the summary are required to be closely relevant to the query. Therefore, the performance of QS relies highly on accurate measurement of text similarity. Traditionally, QS is based on the BOW approach, in which both the query and sentences are represented with word vectors. This approach suffers from the shortcoming that it merely considers lexical elements (words) in the documents, and ignores semantic relations among sentences.

In this paper, we examine sentences on a different dimension, i.e., Wikipedia concepts. Through sentence wikification, each sentence is mapped to a vector whose elements are Wikipedia concepts. In this feature space, we get the *Concept* similarity which serves to complement the original *Word* similarity derived from BOW. Then, we take *semantic relatedness* of Wikipedia concepts into account. We argue that two different concepts can be taken as matched if they are semantically close to each other. Following this, we propose the *Matrix* similarity, in which semantic relatedness is utilized for smoothing the process of concept matching. Finally the renewed sentence similarity resulting from wikification is employed to benefit query-oriented summarization.

To the best of our knowledge, this study is the first attempt to address the problem of combining sentence wikification with QS. We conduct extensive experimental studies to evaluate the proposed framework on the DUC 2005 dataset. The experiments show that sentence wikification indeed takes effect in boosting the performance of QS. Also, we observe that the Matrix similarity brings more significant improvements than Concept.

The rest of the paper is organized as follows. Section 2 introduces background information on query-oriented summarization. Section 3 presents our proposed framework. In Section 4, we show and discuss the experimental results. Finally, we have conclusion in Section 5.

2. QUERY-ORIENTED SUMMARIZATION

In this section, we introduce two existing solutions to QS. Formally, we denote the given query as q and the collection of documents as D. The goal of QS is to generate a summary which best meets the information needs expressed by q. To do this, a QS system generally takes two steps: first, the sentences in D are ranked with respect to q; second, top sentences are selected until the length of the summary is reached. For convenience, we let S denote all the sentences in D.

A straightforward method [9], i.e., *TFIDF*, is to compute text similarity between the query and sentences, and rank the sentences based on this value. Both the query and the sentences can be represented with TF*IDF vectors. Therefore, the query-sentence similarity is naturally obtained as the cosine value of their TF*IDF vectors. Since this method is quite simple, we do not give elaborations on it.

Graph-based models [1, 2, 3, 5, 8] have been proved to be effective in sentence ranking and summary generating. In these models, a graph is constructed in which each node represents a sentence in S. Each edge measures the similarity between the corresponding pair of sentences. We consider two factors when deciding whether sentence s_i is selected to be included in the summary. First, s_i is relevant to the query q. Second, s_i is similar with other sentences which have high query-sentence similarity. This idea is captured by the model in Figure 1, which mixes query-sentence and sentence-sentence similarity together. In the figure, we denote the similarity of sentence s_i with the query q as $sim(s_i, q)$, and the similarity between sentence s_i and s_j as $sim(s_i, s_j)$. Then, using a computation process based on Random Walk, the saliency score for sentence s_i can be calculated iteratively as follows.

$$Score^{(n+1)}(s_i) = d \cdot \frac{sim(s_i, q)}{\sum_{s_j \in S} sim(s_j, q)} + (1-d) \cdot \sum_{s_j \in S} \frac{sim(s_i, s_j)}{\sum_{s_k \in S} sim(s_j, s_k)} Score^{(n)}(s_j), \quad (1)$$

where $Score^{(n)}(s_i)$ is the score of s_i in n^{th} iteration, d is a combination coefficient for trading off the two parts. We call this model as *Graph* in the following formulations.

3. THE PROPOSED FRAMEWORK

3.1 Sentence Wikification

In the traditional BOW approach, each sentence s_i is mapped to a vector of words, that is,

$$wordvector_i = \{tfidf_{s_i}^{w_1}, tfidf_{s_i}^{w_2}, \cdots, tfidf_{s_i}^{w_N}\}, \quad (2)$$

where $tfidf_{s_i}^{w_j}$ is the TF*IDF value of word w_j in sentence s_i,

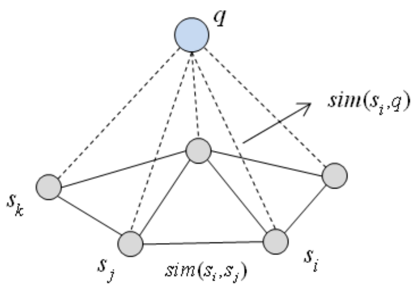


Figure 1. The Graph model.

and N is the total number of words. Then the similarity between two sentences s_i and s_j is measured by the cosine value of $wordvector_i$ and $wordvector_j$. This similarity is shortly named as *Word*, which is denoted as $wordsim(s_i, s_j)$.

Sentence wikification is the practice of representing a sentence with a set of Wikipedia concepts. We take the *exact-match* strategy introduced in [4] as our wikification method. Specifically, to wikify a sentence s_i, we scan this sentence and find Wikipedia concepts that appear explicitly in it. To find high-quality Wikipedia concepts, we also adopt extra operations such as excluding meaningless concepts and merging partial concepts. For instance, for the sentence "How do European Union countries feel about the US opposition to the Kyoto Protocol?", the concepts "Kyoto", "Protocol" and "Kyoto Protocol" (all of them appear in the sentence) should be treated as a single concept "Kyoto Protocol". In addition, the concepts "Position" and "Proto", though in the sentence, obviously cannot act as interpretation of the sentence, and thus should be eliminated.

These searched concepts are used to comprise the *concept vector* for the sentence. Formally, the sentence s_i is associated with a concept vector, that is,

$$conceptvector_i = \{\mathrm{var}_{s_i}^{c_1}, \mathrm{var}_{s_i}^{c_2}, \cdots, \mathrm{var}_{s_i}^{c_W}\}, \quad (3)$$

where $\mathrm{var}_{s_i}^{c_j}$ is a binary variable which indicates whether concept c_j appears in sentence s_i, and W is the total number of Wikipedia concepts appearing in S. Then the Concept similarity between s_i and s_j is the cosine similarity of their concept vectors.

$$conceptsim(s_i, s_j) = \frac{conceptvector_i \cdot conceptvector_j}{|conceptvector_i| \cdot |conceptvector_j|}. \quad (4)$$

3.2 The Matrix Similarity

The Concept similarity is able to represent sentence similarity on the dimension of Wikipedia concepts. However, Concept is too "rigid" because it allows matching of two concepts only when they are identical. In other words, only concepts which are shared by two sentences can contribute to their Concept similarity. As a result, some concept vectors, such as {Kyoto protocol, Emissions trading, Carbon dioxide} and {Global warming, Greenhouse gas, Fossil fuel}, have no Concept similarity, though they are quite close according to human judgment. To solve this problem, we turn to semantic relatedness of Wikipedia concepts, a value indicating the extent to which two Wikipedia concepts are close to each other, e.g., "Kyoto protocol" is closer to "Global warming" than to "Financial crisis". An effective and efficient method for semantic relatedness is Wikipedia Link-based Measure (WLM) [6, 7] which infers semantic relatedness from link structures in Wikipedia. The basic intuition behind WLM is that if two concepts are cited by (or link to) many common concepts, they are much likely to be highly related. A well-developed demo of WLM can be found at http://wdm.cs.waikato.ac.nz:8080/service?task=compare.

With sentence wikification, the set of sentences S are collaboratively represented with a *concept matrix*. We assume that semantic relatedness of each concept pair has been given. The computation of the Matrix similarity takes two steps.

First, with the semantic relatedness values, we create a *relatedness matrix*. The elements of the matrix represent semantic relatedness among concepts. To avoid excessive matching, we set a relatedness confidence, denoted as $confidence$, on the semantic relatedness values. Only two concepts whose semantic relatedness exceeds $confidence$ can have a value in the relatedness matrix. If the semantic relatedness between two concepts c_i and c_j is $SR(c_i, c_j)$, then the (i, j) element in the relatedness matrix is $RM(i, j) = SR(c_i, c_j)$ if $SR(c_i, c_j) > confidence$, and 0 otherwise.

Second, the concept matrix is multiplied by the relatedness matrix (see Figure 2). This matrix multiplication generates a new *relatedness-concept matrix*. Each element of this matrix is

$$r\,\mathrm{var}_{s_i}^{c_j} = \sum_{k=1}^{W} \mathrm{var}_{s_i}^{c_k} \cdot RM(c_k, c_j)\,, \tag{5}$$

where $r\,\mathrm{var}_{s_i}^{c_j}$ is the *relatedness-concept value* of concept c_j in sentence s_i. Equation (5) indicates that the relatedness-concept value of c_j in s_i equals the weighted sum of the values in the original concept vector, and the weighting coefficients are semantic relatedness of concepts.

After these two steps are finished, each sentence s_i is represented with a renewed *relatedness-concept vector*

$$rconceptvector_i = \{r\,\mathrm{var}_{s_i}^{c_1}, r\,\mathrm{var}_{s_i}^{c_2}, \cdots, r\,\mathrm{var}_{s_i}^{c_W}\}\,. \tag{6}$$

Then the Matrix similarity of two sentences is computed as the cosine similarity of their relatedness-concept vectors.

We give an illustration of the steps in Figure 2. An observation is that the relatedness matrix serves as a bridge to associate sentences with their semantically-related concepts. A sentence consequently obtains weights on the concepts that do not appear explicitly in it.

3.3 Improving QS

We combine linearly the Concept or Matrix similarity with the basic Word similarity and obtain the final sentence similarity. For instance, when considering Concept, the combined similarity is

$$combinesim(s_i, s_j) = wordsim(s_i, s_j) + \alpha \cdot conceptsim(s_i, s_j)\,, \tag{7}$$

where α is a factor to control the balance between Word and Concept similarity. The combined similarity is substituted into either the TFIDF or the Graph model. Using a similar iterative computation, we can get the saliency score for each sentence. For limit of space, we do not give the updating formula (like Equation (1)) here. Note that we replace both the query-sentence and the sentence-sentence similarity in Graph with this combined similarity.

3.4 Redundancy Checking

After the sentences are scored, some of the top sentences may express similar meaning or convey duplicate information. If they are selected simultaneously, the final summary will be redundant. We adopt such a method to address this redundancy: each candidate sentence, before being added to the final output, is compared with the sentences that are already contained in the summary. Only the candidates, whose similarity with all the sentences in the summary is below a predefined threshold λ, can be added to the summary.

4. EXPERIMENTS

In this section, we conduct experimental studies to test the effectiveness of the proposed framework. Before going to the details, we first describe the dataset and evaluation metrics.

4.1 Dataset and Performance Metrics

Document Understanding Conference (DUC) has set a series of QS tracks and provided benchmark datasets. We use the DUC2005 dataset which consists of 50 queries. Each query corresponds to a collection of 25-50 relevant documents. The task is to generate a summary of 250 words for each query from the associated document collection. For preprocessing, we partition the documents into individual sentences with the Sentence-Detector function of the OpenNLP[1] package. Moreover, stop words are removed from the vocabulary.

For quantitative evaluation, we use the ROUGE toolkit which has been widely adopted by DUC for automatic summarization evaluation. ROUGE measures summary quality by counting overlapping units such as n-gram, word sequence and word pairs between a generated summary and a set of reference summaries. In our experiments, we run ROUGE-1.5.5[2] with the parameter settings consistent with DUC 2005: -n 4 -l 250 -w 1.2 -m -2 4 -u -r 1000 -f A -p 0.5 -t 0, where "-l 250" indicates the evaluated summaries have the length of 250 words. In the results, we report three of the ROUGE metrics: ROUGE-1, ROUGE-L and ROUGE-SU.

4.2 Performance Evaluation

We take the basic TFIDF and Graph models (with Word similarity solely) as baselines. The parameters are set in the following ways. Both the coefficient d in the Graph model and the threshold λ in redundancy checking are empirically set to

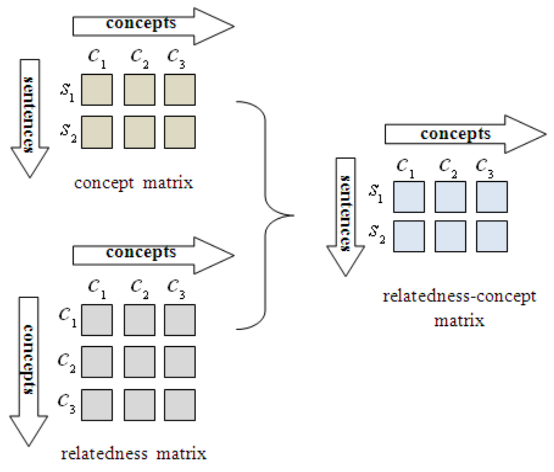


Figure 2. Illustration of the Matrix similarity.

[1] http://opennlp.sourceforge.net/

[2] http://research.microsoft.com/~cyl/download/ROUGE-1.5.5.tgz

0.3. For the controlling factor α , we experiment on a wide range of values and choose the best one in terms of the evaluation metrics. In Graph, when α is fixed to its best value, we continue to tune *confidence* from 0 to 1.0 with 0.1 as the step size. All the iterative algorithms converge when the difference between the scores computed at two successive iterations for any sentences falls below a threshold (10^{-5} in this study).

Table 1. Experimental results.

Models	Similarity	Rouge1	RougeL	RougeSU
TFIDF	Word (Baseline)	0.34974	0.31813	0.11782
	Word + Concept	0.35597	0.32350	0.12190
	Word + Matrix	0.35870	0.32623	0.12076
Graph	Word (Baseline)	0.36648	0.33714	0.12570
	Word + Concept	0.36916	0.33972	0.12664
	Word + Matrix	0.37124	0.34005	0.12780

Table 1 shows the experimental results when different types of similarity are used. We can see that the introduction of the Concept similarity improves QS in each case and on every metric. This proves that sentence wikification is an effective strategy for enhancing the performance of query-oriented summarization. Also, when combining the Word similarity with the Matrix, rather than Concept, similarity, we can obtain even better results. The difference between Concept and Matrix is that in Matrix, we reweight the concept vectors with semantic relatedness. The smoothing effects of semantic relatedness result in sentence similarity which is more consistent with human judgment, and therefore helps to produce more desirable query-oriented summaries.

When the type of sentence similarity is identical, the Graph model consistently outperforms TFIDF. Even the baseline Graph (Word) can perform better than the enhanced TFIDF (Word + Matrix). This observation conforms to previous literatures which show that Graph, considering both query-sentence and sentence-sentence similarity in a unified computation process, has advantages over the simpler TFIDF model.

We also investigate the influence of the relatedness confidence, i.e., *confidence* . Figure 3 shows the ROUGE-1 metric for Word + Matrix when we use the Graph model and *confidence* is set to various values. The best performance is achieved when *confidence* is set to 0.4. This tells us that the concept pairs with small (less-than-0.4) semantic relatedness actually contribute little to the QS task. Meanwhile, when *confidence* exceeds 0.4, we filter the concept pairs too aggressively and lose some valuable information, which causes the performance to decrease.

5. CONCLUSION AND FUTURE WORK

In this paper, we study whether sentence wikification can improve the performance of query-oriented summarization (QS).

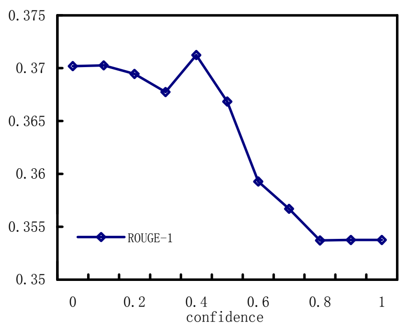


Figure 3. QS performance as *confidence* varies.

In our proposed framework, both queries and sentences in QS are enriched with Wikipedia concepts as additional features. Also, we present the computation of the Matrix similarity, in which semantic relatedness of Wikipedia concepts is considered for smoothing concept matching. Then the combined sentence similarity is employed in the TFIDF or Graph model. From the experiments, we can conclude that sentence wikification improves QS effectively. In addition, the incorporation of semantic relatedness enables us to get better results.

6. ACKNOWLEDGMENTS

This work was supported by National Natural Science Funding of China under Grant No. 90718022 and National 863 Project under Grant No. 2009AA01Z410.

7. REFERENCES

[1] Y. Chali, and S. R. Joty. Improving the Performance of the Random Walk Model for Answering Complex Questions. In *Proceedings of ACL-HLT'08*, pages 9-12, 2008.

[2] Y. Chali, and S. R. Joty. Exploiting Syntactic and Shallow Semantic Kernels to Improve Random Walks for Complex Question Answering. In *Proceedings of ICTAI'08*, pages 123-130, 2008.

[3] G. Erkan, D. R. Radev. LexRank: Graph-based Lexical Centrality as Salience in Text Summarization. In *Journal of Artificial Intelligence Research, Vol. 22, 2004.*

[4] X. Hu, X. Zhang, C. Lu, E. K. Park, and X. Zhou. Exploiting Wikipedia as External Knowledge for Document Clustering. In *Proceedings of SIGKDD'09, pag*es 389-396, 2009.

[5] F. Li, Y. Tang, M. Huang, and X. Zhu. Answering Opinion Questions with Random Walks on Graphs. In *Proceedings of ACL'09*, pages 733-745, 2009.

[6] D. Milne. Computing Semantic Relatedness using Wikipedia Link Structure. In *Proceedings of the 5th New Zealand Computer Science Research Student Conference*, 2007.

[7] D. Milne, and I. H. Witten. An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In *Proceedings of the AAAI Workshop on Wikipedia and Artificial Intelligence (WIKIAI'08)*, 2008.

[8] J. Otterbacher, G. Erkan, and D. R. Radev. Using Random Walks for Question-focused Sentence Retrieval. In *Proceedings of HLT-EMNLP'05*, pages 915–922, 2005.

[9] J. Tang, L. Yao, and D. Chen. Multi-topic based Query-oriented Summarization. In *Proceedings of SDM'09*, pages 1148-1159, 2009.

A Compression Framework for Generating User Profiles

Xiaoxiao Shi
Department of Computer Science
University of Illinois at Chicago
xiaoxiao@cs.uic.com

Kevin Chang, Vijay K. Narayanan
Vanja Josifovski, Alexander J. Smola
Yahoo! Research
Santa Clara, CA 95051, USA
{klchang, vnarayan, vanjaj, smola}@yahoo-inc.com

ABSTRACT

Predicting user preferences is a core task in many online applications from ad targeting to content recommendation. Many prediction methods rely on the being able to represent the user by a profile of features. In this paper we propose a mechanism for generating such profiles by extracting features that summarize their past online behavior. The method relies on finding a compressed representation of the behavior by selecting the dominant features contributing to the Kullback-Leibler divergence between the default distribution over user actions and the user specific properties. We show that the feature selection model of [1] can be extended to a hierarchical encoding of user behavior by means of using an intermediate clustering representation. Preliminary experiments suggest the efficacy of our method.

1. INTRODUCTION

In order to be able to reason about the user, search engines, computational advertising platforms, and recommendation systems require a suitable *representation* of the user. The profile is created based on the user's past behavior and such as to target the user with the suitable content or advertising. The targeting is usually performed by a model that predicts the user's interest. The key challenge in determining the right profile representation is to find predictive features. One alternative for this task is to use a generative methods, such as singular value decomposition of the (user, action) matrix; topical analysis by Latent Dirichlet Allocation [2]; clustering users based on their actions; or Probabilistic Latent Semantic Indexing factorization [4]. These methods describe the user in terms of some other representation (dense vectors, sparse sets of abstract topics, cluster ids). Often this suffices for the purpose of inference, since secondary algorithms do not require *interpretable* features.

In some cases, though, interpretability is important: advertisers *want* to know which users are being targeted by their ad campaigns. For instance some advertising contracts may specify that an ad be shown to males living in California, aged 18-25 years. We aim to obtain more fine-grained yet understandable representations. This problem is not limited to advertising. For instance, social scientists *want* to have understandable representations to support decisions.

We show that is possible to achieve both goals — to obtain interpretable features describing users in a human-understandable fashion *and* to obtain features which are good for predicting a user's actions. Our work builds on that of [1] who suggested a simple algorithm to determine meaningful features of users: choose features whose distribution differs most from the global baseline in terms of their Kullback-Leibler (KL) divergence. This retains an interpretable set of features, since it simply selects a subset of user actions 'pars pro toto'. While this rule was proposed in a slightly ad-hoc fashion, it is rather well founded: choosing the actions which contribute the most to the KL divergence between a given user and the sample average selects terms from the distribution which require the largest number of additional bits to encode. Note that the KL divergence $D(p\|q)$ quantifies the number of additional bits needed by encoding data drawn from p with the code optimal for q.

However, a simple multinomial model as used by [1] is not necessarily ideal when it comes to representing the distribution over the actions of many users. Instead, we may choose more sophisticated formulations such as clustering, topics, or any of the other factorization approaches mentioned above. The simple KL divergence selection heuristic now becomes one of encoding the structure in a sparse fashion and subsequently one of encoding the user with regard to its latent representation. As a running example we use clustering for the latent representation. Hence, in order to represent a user's features we first pick a representation of a user's cluster in terms of sparse features and second we pick the user's actions relative to its cluster. This two-stage approach could be easily extended to many stages by feature extraction in hierarchical clustering. Likewise, in topic models [2] we could represent a user's action in terms of the actions most representative for the topics associated with the user and secondly with the actions which cannot be explained equally well by the topic model (and hence which have large KL divergence). Such multistage models effectively smooth out the actions of a user.

For instance, assume that one set of users would use the term 'car' and another set 'automobile'. While they clearly should belong to the same cluster, thus smoothing over minor differences in behavior, there clearly is additional information to be gained by recording the dichotomy between 'cars' and 'automobiles'.

2. ENCODING FRAMEWORK

Sufficient Statistics

In the following we denote by i users and by x_i their actions. More specifically, x_{ij} encodes action j by user i. It is our goal to extract some features $\phi(x)$ from x such that $\phi(x)$ leads a) to good predictions downstream and b) to understandable representation. First note that whenever $\phi(x)$ is a sufficient statistic of x it follows that

$$p(y|x, \phi(x)) = p(y|\phi(x)). \tag{1}$$

In other words, given a good representation $\phi(x)$ of x we can dispose of x itself. This is particularly desirable since we can assume that x is rather noisy, having been drawn as a sample from the distribution over all actions a particular user might take. Hence, if we find a more compact set of (sufficient) statistics which describe x sans noise, we should be able to obtain equal or better estimation performance after applying $\phi(x)$.

Kullback Leibler Divergence

Recall that the information contained in a stream of samples from some distribution p is given by its entropy, that is by

$$H[p] := \mathbf{E}_{z\sim p}[-\log_2 p(z)] \tag{2}$$

One may show (see e.g. [3]) that the optimal code for encoding z requires $-\log_w p(z)$ bits on average. In this view the Kullback Leibler divergence

$$D(p\|q) := \mathbf{E}_{z\sim p}[\log_2 p(z) - \log_2 q(z)] \tag{3}$$

quantifies the excess number of bits we spend on average by encoding $z \sim p$ with a code which is optimal for $z \sim q$. This provides a natural weight for symbols z via $p(z) \log \frac{p(z)}{q(z)}$, the expected contribution of z to the excess in encoding. In other words, if we want to reduce the excess in encoding q we should focus on setting aside those terms z with the largest contribution to the KL-divergence. This is precisely what the algorithm of [1] amounts to. Symbols z' with small deviations in the encoding contribution are more likely to be due to sampling noise, hence we benefit from omitting them.

Multinomial Distribution Model

The following example clarifies this rather abstract model. Assume that we have a multinomial distribution p over symbols z, parameterized by probabilities θ_z, i.e. $p(z) = \theta_z$. Such a distribution can be estimated, e.g. for a set of users, by the background model over all their actions: denote by m the total number of actions of all users and let $n_z := \sum_{i,j} \{x_{ij} = z\}$ be the number of times any user takes action z. Then we can estimate

$$\theta_z = \frac{n_z}{m} \text{ or } \theta_z = \frac{n_z + \alpha}{m + N\alpha} \tag{4}$$

where the second equation is obtained by smoothing with a Dirichlet prior. Observing actions x_{ij} for user i we define the corresponding quantities $n^i_z := \sum_j \{x_{ij} = z\}$ and $m^i = \sum_z n^i_z$. This yields smoothed estimates for the action distribution for the user via $\theta^i_z = \frac{n^i_z+\alpha'}{m^i+N\alpha'}$ for a Dirichlet smoother α'. When using the background distribution as quantified by θ rather than the user specific distribution θ^i we pay $d(p\|q|z) := \theta^i_z \log \theta^i_z/\theta_z$ additional bits for symbol z. Hence to minimize the amount of inefficiency in coding, we should store θ^i_z for all z with $d(p\|q|z) \geq c$ for some threshold c. It yields understandable features, obviously provided that z amounts to human-understandable actions or tokens.

Mixture of Multinomials Model

A single multinomial distribution is not a very accurate characterization of user actions. One of the simplest modifications is to use a mixture of multinomials rather than a single multinomial. That is, we assume that observations follow the distribution

$$p(x) = \sum_y p(x|y)p(y) \tag{5}$$

where both $p(x|y)$ and $p(y)$ are multinomials (conveniently with an associated Dirichlet smoother). As a result the encoding problem now decomposes into two parts: encoding the *distribution* $p(x|y)$ and encoding the difference between x_i and the estimated cluster y_i. We address this problem by applying the encoding strategy for multinomial distributions twice – once to encode the cluster distribution above the global baseline profile, and another time to encode the user distribution above the cluster profile. That is, for the cluster we select all z which satisfy

$$d(p(\cdot|y_i)\|q|z) \geq c \tag{6}$$

with respect to the common baseline distribution q. Moreover, to encode x_i we select all z which satisfy

$$d(p^i\|p(\cdot|y_i)|z) \geq c \tag{7}$$

Note that (7) is slightly imprecise: since we did not encode $p(\cdot|y_i)$ entirely, it behooves us to compare the smoothed estimate of the user distribution p^i *not* with $p(\cdot|y_i)$ but rather only with the *encoded* subset of tokens from $p(\cdot|y_i)$.

General Framework

Given a model of the form

$$p(x) = \int p(x|\alpha)p(\alpha|\beta)p(\beta|\gamma)\ldots d\alpha d\beta d\gamma \ldots \tag{8}$$

we may resort to a hierarchical encoding by successively selecting z which contribute most to the KL-divergence terms

$$D(p(x|\gamma)\|p(x)) \text{ or } D(p(x|\beta)\|p(x|\gamma)) \text{ or } \ldots D(p^i\|p(x|\alpha)).$$

3. TOKENS FOR USER ACTIONS

We assign tokens to each possible user event that we log. The tokens can be of varying granularity depending on the nature of the end application that uses this user profile. For instance, a user searching for “toyota prius” could be tokenized in a granular manner as “query for toyota prius” or in a coarser manner as “query related to autos”.

We constructed a dataset from the events of a sample of 1 million users at Yahoo!, over the 55 day period from Jan 1, 2010 to Feb 24, 2010. We used events from 6 types of user activities: web page views (pv), views of display ads (adv), clicks on display ads (adc), search queries (sch), clicks on search results (slc) and clicks on search ads (olc). We tag each of these events with categories in a taxonomy of user interests, based on the content of the webpage, ad, or search query, through a combination of editorially labeled dictionaries and automated machine learned categorizers. We represent each event as the token“c<event-type>-<category>”.

We then represent a user's history as a sequence of tokens of actions. For example, if the user had 1) a page view on autos.yahoo.com, followed by 2) a search query for "mutual fund", followed by 3) a click on a result of this search query, the sequence of tokens would be: "cpv-autos", "csch-finance", "cslc-finance". In all, there were a total of 5911 different unique tokens, constructed from the 6 event types and 1205 different categories for these events. We estimated the multinomial user model from this sequence of tokens.

4. EXPERIMENTS

We clustered the users in this dataset and estimated the average profile of users belonging to each cluster, and performed some preliminary experiments using our user profiles for the modeling task of ad conversion modeling.

4.1 Analysis of Clustering

Figure 1 shows the distributions of KL divergence between the user profile and the global profile and that between the user profile and the cluster profile. The distribution of the KL divergence between user and the cluster profiles is distinctively skewed towards lower values compared to the distribution of KL divergence between the user and the global profiles, demonstrating that clustering can effectively encode a large fraction of users. For example, more than 50% of the users have KL divergence in the range $[0, 0.7]$ between the user profile and the associated cluster profile, while less than 25% of the users have KL divergence in this range between the user profile and the global profile. Hence, a cluster profile is a more accurate representation of a user profile than the global profile. Figure 2 shows that clustering can still preserve groups of unique users which have high KL divergence from the global background, and does not wash out the profiles of users with distinctive profiles compared to the global profile.

Figures 1 and 2 show that clustering can effectively group together similar users by their behaviors, and also preserve groups of users with unique profiles compared to the global profile. Another interesting experiment is to understand the relevant features identified by the clustering encoding. Table 1 shows the user features in two different clusters that contribute most to the KL-divergence terms as described in (8). These features are the ones that most distinguish the cluster from the global profile.

It is clear that clustering can group together users with the similar latent interests represented by the tokens. For example, Cluster 1 consists of users with many technology related tokens, while Cluster 2 consists of users with many finance related and mobile phones related tokens.

Table 2 shows the most distinguishing tokens of 2 users selected at random from Cluster 1 and another user from Cluster 2. The highlighted tokens in bold letters show the features in the user profile that are also present in their respective clusters. Comparing the profiles of users 1 and 2, it is clear that while they share events related to technology, they are still different from each other in that user 1 has more events related to finance and sports, while user 2 has events related to beauty, personal care and babies. User 3 on the other hand has a number of events related to finance and cellular phones, and is also further interested in retail products. Thus, we observe that a) clustering effectively groups users with similar events in their profiles b) clustering-based encoding can judiciously capture the most significant tokens among similar users; c) different clusters reveal different significant tokens. These three observations show the potential power of incorporating intermediate clustering representations to encode user profiles.

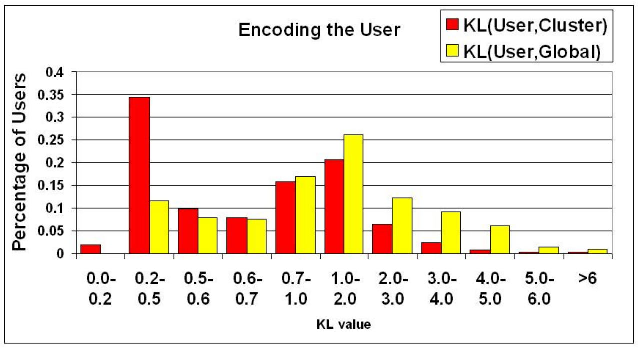


Figure 1: Contribution to the KL divergence between user profile and global profile (yellow) and between user profile and cluster profile (red). Clustering is a better model for encoding the data (most users can be accurately captured by the model small KL divergence.

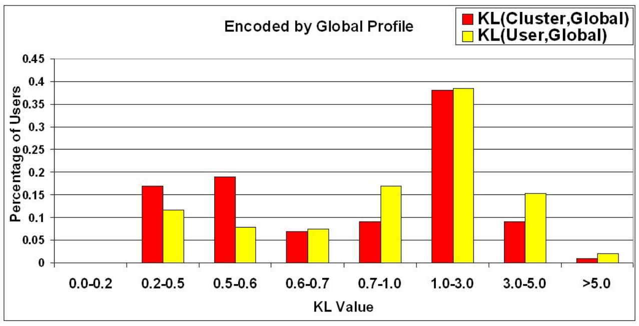


Figure 2: Revealing unique user groups. Clustering can still preserve the groups of unique users with large KL divergence ($1.0 < KL < 3.0$)

4.2 Preliminary conversion modeling results

We constructed a linear model to predict whether or not a user would convert on a specific ad, where we have one model per ad. The baseline model considered raw user events, while the experimental model considered raw user events as well as the user profile, constructed as user's cluster profile, and the 50 most divergent tokens between the user and the global profile, and the user and cluster profile. We evaluated the 191 models by computing the area under the ROC curve (AUC), on a test set that ocurred after the train set in time.

In order to compare the user profile-based models and the baseline across the 155 different ads/models, we tallied the "wins" as the number of ads for which the user profile-based models AUC exceeded the baseline models.

# conv	# ads	% win	profile AUC	base AUC
≥ 10	155	62%	0.57	0.56
$[10, 100]$	112	68%	0.55	0.53
≥ 100	43	47%	0.60	0.60

As an example to aid interpretation of the table, there were 112 ads with between 10 and 100 conversions in the

Table 1: Important events in the cluster specific profile and KL divergence from the baseline. The instances show that user events are well grouped and human understandable. Cluster 1 shows a strong affinity for events related to technology, while Cluster 2 groups events in finance and cellular phone related tokens.

Cluster ID	Important User Behaviors	KL Divergence
Cluster 1	cpv-Technology/Internet Services	0.3883
(Technology Group)	cpv-Technology/Internet Services/Online Community	0.3854
	cpv-Technology	0.3840
	cpv-Technology/Internet Services/Online Community/Email	0.2829
	cpv-Technology/Internet Services/Online Community/Portals	0.2806
	cpv-Technology/Internet Services/Online Community/Photos	0.0122
Cluster 2	cadv-Finance	0.0365
(Finance Group)	cadv-Finance/Credit Services	0.0274
	cadv-Finance/Insurance	0.0145
	cadv-Finance/Insurance/Automobile	0.0119
	cadv-Telecommunications/Cellular and Wireless Services	0.0081
	cadv-Technology/Consumer Electronics/Comms/Mobile/Cellular Telephones	0.0078

Table 2: Profiles of 2 users randomly selected from Cluster 1 and one user from Cluster 2, with their event tokens and frequencies. The boldface tokens are common to the average profile of the relevant cluster.

User Index	Behavior set
User 1	**cadv-Technology/Internet Services/Online Community**:134
Cluster 1	**cpv-Technology/Internet Services/Online Community**:190
	cpv-Technology/Internet Services:192
	cpv-Technology:192
	cpv-Technology/Internet Services/Online Community/Email:190
	cadv-Technology:150
	cadv-Technology/Internet Services/Online Community/Email:132
	cadv-Technology/Internet Services:142
	cadv-Finance/Insurance:52
	cadv-Finance:52
	cpv-Sports/Soccer:21
	cpv-Sports/Auto Racing:32
User 2	**cpv-Technology**:212
Cluster 1	**cpv-Technology/Internet Services/Online Community**:212
	cpv-Technology/Internet Services:212
	cpv-Technology/Internet Services/Online Community/Email:212
	cadv-Consumer Packaged Goods:18
	cadv-Consumer Packaged Goods/Beauty and Personal Care:14
	cadv-Life Stages/Parenting and Children/Baby:10
	cadv-Life Stages:10
User 3	**cadv-Finance/Credit Services**:72
Cluster 2	**cadv-Finance**:190
	cadv-Finance/Investment/Discount Brokerages:44
	cadv-Technology/Consumer Electronics/Communication/Mobile/Cellular Telephones:34
	cadv-Technology/Consumer Electronics/Communication/Mobile:104
	cadv-Small Business and B2B:44
	cadv-Life Stages:32
	cadv-Retail:24

test set used to compute the AUC. User profile models had a greater AUC than the baseline for 68% of these ads, and had an average AUC of 0.55 vs 0.53 for the baseline. We note that the user profile-based models perform better when there is less data. Models with little data tend to be more sensitive to noise, and therefore beneift from cluster-based user profiles that provide a smoother set of features.

5. CONCLUSION

We have proposed a method of extracting features from user profiles, based upon first clustering the users and then encoding the user profile as a combination of the cluster profile and the most distinguishing features between the user and this cluster profile. Experimental results show that the two stage approach of first clustering the user profiles, and then encoding the users with the cluster profile and the difference from this cluster profile, is a more effective method to encode a user profile, as opposed to a single step encoding of the users using the distinguishing features compared to the global profile alone. Preliminary modeling results show that the profiles can improve conversion modeling, especially when there are few positive examples in the data.

6. REFERENCES

[1] M. Shmueli-Scheuer, H. Roitman, D. Carmel, Y. Mass, and D. Konopnicki. Extracting User Profiles from Large Scale Data. In *PMDAC*, 2010.

[2] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. *JMLR*, 3:993–1022, 2003.

[3] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* John Wiley and Sons, NY, 1991.

[4] T. Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1/2):177–196, 2001.

An SVM Based Approach to Feature Selection for Topical Relevance Judgement of Feeds

Yongwook Shin and Jonghun Park
Information Management Lab.
Seoul National University
599 Gwanak-ro, Gwanak-gu, Seoul, Korea
{yongwook, jonghun}@snu.ac.kr

ABSTRACT

Relevance judgement method is an essential part of a feed search engine, which determines candidates for ranking query results. However, the different characteristics of feeds from traditional web pages may make existing relevance judgement approaches proposed for web page retrieval produce unsatisfactory result. Compared to web pages, feed is a structured document in that it contains several data elements, including title and description. In addition, feed is a temporal document since it dynamically publishes information on some specific topics over time. Accordingly, the relevance judgement method for feed retrieval needs to effectively address these unique characteristics of feeds. This paper considers a problem of identifying significant features which are a feature set created from feed data elements, with the aim of improving effectiveness of feed retrieval. We conducted extensive experiments to investigate the problem using support vector machine on real-world data set, and found the significant features that can be used for feed search services.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval; H.3.5 [**Information Systems**]: Online Information Services; H.5.4 [**Information Systems**]: Hypertext/Hypermedia

General Terms

Experimentation, Measurement, Performance

Keywords

Information Retrieval, Search Engine, Feed Search, Feed, Feature Selection, Relevance Judgement, Text Classification

1. INTRODUCTION

The number of web sites publishing feeds is dramatically increasing and it is now common to acquire information by subscribing to feeds. Feed is an XML document published by a web site to facilitate syndication of its contents to subscribers. News and blogs are common sources for feeds, and recently social media and microblogging sites are increasingly delivering information through feeds.

Feed search engine that can help users effectively seek for feeds becomes necessary in order to address the challenges imposed by recent explosion of feeds. Through feed search engine, users can discover feeds for the purpose of subscription in order to keep themselves updated with recent information. Feed search engine takes a query from users and generates ranking candidates which are feeds judged to be relevant to the query. Afterwards, it returns a ranked list of feeds by scoring the candidates based on a specific feed retrieval model. Accordingly, relevance judgement method that detemines the ranking candidates plays an important role in enhancing feed search quality.

Under binary relevance judgement framework, the relevance judgement method classifies feeds as relevant or non-relevant against a user query in order to construct the ranking candidates [11]. In particular, in relevance judgement for feed retrieval using feature-based model, the relevance is defined by using a feature set constructed from a user query and a feature set from a feed. It is necessary that a feature set for a feed is constructed by selecting specific features suited to feed relevance judgement. Unfortunately, existing relevance judgement methods for web page retrieval may not be appropriate for relevance judgement for feed retrieval since they are different in terms of available features.

Specifically, relevance judgement problem for feed retrieval poses interesting challenges due to two different properties of feeds, compared to web pages. Feed is a structured as well as temporal document in a sense that it contains several data elements, including a feed title, a feed description, and multiple entries each of which consists of a title and description dynamically published over time, whereas web pages tend to be rather static. With the structural nature of feed, the problem is to find out which data elements need to be selected to define a feature set for relevance judgement. The temporal charateristic of feed raises a problem of determining how many entries need to be considered for relevance judgement.

To the best of authors' knowledge, there has been no study that attempted to define features used for feed relevance judgement method by considering the unique structural and temporal characteristics of feed. In this paper, we attempt to identify significant features for feed relevance judgement with respect to a user query. The significant features are defined as a feature set constructed from feed data elements that can improve effectiveness of feed retrieval.

Ranking function for blog feed retrieval model was studied previously, taking account of unique properties of feeds. Research in [4] investigated whether it is more effective for feed retrieval to view a feed as a single document or multiple documents composed of entries. However, it did not pay attention to feature selection problem for feed relevance judgement. In addition, temporal characteristics of feed as well as structural elements such as feed title and feed description were not considered in [4].

An enhanced ranking model for blog posts, named PTRank, was proposed to improve search quality beyond the simple keyword matching, utilizing various information available from blog feeds [7]. Yet, it suggested the scoring function to assess the degree of relevance between a query and blog posts, instead of selecting the significant featues for feed relevance judgement. Furthermore, temporal characteristic of feed was not considered by PTRank.

In this paper, we attempt to identify the significant features to judge relevance for feed retrieval through feed classification based on a topic using a topic-labeled feed data set. The rest of the paper is organized as follows. Section 2 defines the problem and presents our proposed approach to identify the significant features. In Section 3, we report experimental results and finally, we give concluding remarks in Section 4.

2. PROPOSED APPROACH

Feed is an XML file that contains partial or full descriptions of web page articles along with links to the original contents and other information. Two popular feed formats are RSS (Really Simple Syndication) and Atom [6]. Figure 1 shows a typical example that illustrates major elements of an Atom feed.

Specifically, the schema of Atom 1.0 includes two core elements, namely <feed> and <entry>. <feed> is the root element of an Atom document and contains feed title and description. <feed> should have at least one <entry> element, and it contains mandatory sub-elements, <title> and <subtitle>. <entry> element includes <title> and <summary> elements. While <title> contains the title of a web page article, <summary> element has a short summary or full body of the article.

Among the various feed elements, we consider four elements in identifying the significant features for feed relevance judgement. We refer to <title> element in <feed> as a feed title, and <subtitle> element in <feed> as a feed description. We also refer to <title> element of <entry> as an entry title and <summary> element of <entry> as an entry description.

This study concentrates on identification of the significant features for feature-based relevance judgement method in feed retrieval. The significant features are defined as a feature set created from data elements available from a feed that can maximize effectiveness of relevance judgement.

Specifically, we use a vector space model for representing the features. The notations used in our problem are presented as follows. Given the set of m feeds, $F = \{f_1, f_2, ..., f_m\}$, and the set of terms, T, let $E_i = \{e_{ij} | j = 1, ..., n\}$ represent the set of entries for $f_i, i = 1, ..., m$, constructed by choosing the most recent n entries from the set of all entries published from f_i. Our vecter space model is based on

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
    <title>Example Feed</title>
    <subtitle>A subtitle.</subtitle>
    <link href="ttp://example.org/feed/" rel="self" />
    <link href="http://example.org/" />
    <updated>2003-12-13T18:30:02Z</updated>
    <entry>
        <title>Atom-Powered Robots Run Amok</title>
        <link href="http://example.org/2003/12/13/atom03" />
        <link rel="alternate" type="text/html"
        href="http://example.org/2003/12/13/atom03.html"/>
        <updated>2003-12-13T18:30:02Z</updated>
        <summary>Some text.</summary>
    </entry>
</feed>
```

Figure 1: An example Atom feed

a bag of terms representation, and we denote the term bags of feed title and feed decription as ft_i and fd_i, respectively. The term bag of title in e_{ij} is represented as et_{ij}, while the term bag of description in e_{ij} is represented as ed_{ij}.

In the following, we use notation B for representing a bag of terms, and $|B|$ for the total number of term occurrences in B. In addition, $tf(t, B)$ represents term frequency (TF) of term $t \in T$ in term bag B.

Let Q denote a set of queries, and $q \in Q$ a term set representation of a query. Given query q and E_i, f_i is modeled as a feature vector represented by $(\sigma(q, ft_i)$, $\sigma(q, fd_i)$, $\sigma(q, et_{i1})$, ..., $\sigma(q, et_{in})$, $\sigma(q, ed_{i1})$, ..., $\sigma(q, ed_{in}))$. $\sigma(q, ft_i)$, $\sigma(q, fd_i)$, $\sigma(q, et_{ij})$, and $\sigma(q, ed_{ij})$ for $i = 1, ..., m$ and $j = 1, ..., n$ are scoring functions defined as normalized TF, and they indicate feature scores specified for feed title, feed description, entry title, and entry description, respectively. We consider normalized TF among other alternatives, since it is reported that normalized TF is effective in many text classfication problems [8]. We call normalized TF as TF in the following.

Under the assumption of normalized TF, $\sigma(q, B)$ for term bag B is defined as:

$$\sigma(q, B) = \begin{cases} \sum_{t \in q} tf(t, B)/|B| & \text{if } q \in B \\ 0 & \text{otherwise} \end{cases}$$

We attempt to find the significant features through examining various alternatives for constructing the feature vector, considering the unique properties of feed mentioned in the previous section. First, feed data elements constituting the significant features are identified by investigating several combinations of structural data elements in feature vector construction. Second, we investigate the number of entries considered for the feature vector by varying n to address the temporal characteristics of feed.

We employ support vector machine (SVM) for our feed classification task, since it is reported that SVM is one of the most popular and effective supervised learning methods for text classification problems [3], [10]. In this paper, SVM treats feed classification problem as the one involving binary class labels, which we refer to as "non-relevant" and "relevant". SVM builds a model using the given labeled tranining data, which predicts whether an instance of feature vector of a feed falls into "non-relevant" or "relevant".

To evaluate effectiveness of relevance judgement, we use precision, recall, and F measure which are commonly used to compare the relative performance with different features [2].

Table 1: Results produced by various feature vector construction methods

		P	R	FM
FT	0	**.977**	.390	.557
FD	0	.948	.432	.594
FT+FD	0	.956	.556	.703
ET	40	.941	.425	.586
	80	.946	.418	.580
	120	.947	.426	.588
	160	.946	.429	.590
	200	.947	.436	.597
ED	40	.936	.513	.663
	80	.943	.517	.668
	120	.944	.522	.672
	160	.947	.523	.674
	200	.944	.521	.671
FT+FD +ET	40	.953	.599	.736
	80	.959	.586	.727
	120	.959	.576	.720
	160	.959	.564	.710
	200	.959	.557	.705
FT+FD +ED	40	.954	**.612**	**.746**
	80	.955	.609	.744
	120	.957	.610	.745
	160	.959	.604	.741
	200	.958	.598	.736
FT+FD +ET+ED	40	.957	.603	.740
	80	.961	.589	.730
	120	.959	.577	.720
	160	.961	.566	.712
	200	.958	.557	.704

Precision, P, is a measure of the usefulness of feeds judged as "relevant" and recall, R, is a meaure of the completeness of a relevance judgement method. F measure, FM, is defined as the harmonic mean of recall and precision. FM provides a single metric for comparison across different experiments.

In what follows, among possible configurations of features, the significant features achieve the best performance on a specific evaluation metric. We use notation $SF(EM)$ for representing the significant features identified from experiments on evaluation metric, EM, where EM can be P, R, or FM.

3. EXPERIMENTS

A topic-labeled feed data set for our experiments was collected from "Bundles from Google", which is provided by Google Reader [5], a Google's web-based feed reading service. Several feeds for a topic are bundled in order to serve feed recommendation to Google reader users in "Bundles from Google". We call "Bundles from Google" as "Google data set". Google data set has various topics and broad types of feeds from public media sites, blogs, podcasts, and social media sites. Our data set consists of 3,050 feeds and the size of the most recent entries for each feed is 200. The total number of topics taken as queries is 435, and for each topic, the number of feeds belonging to the topic varies from 3 to 20. On the average, there are 7 feeds per topic.

Baseline features are selected among the features available from feed under the assumption that relevance judgement method using as many as feed data elements and entries possible is most effective. Similarly to the significant features, baseline features are defined in terms of the feed data elements used for constructing the feature vector and the number of entries. Given F, we select ft_i, fd_i, et_{ij}, and ed_{ij}, $i = 1, ..., m$, $j = 1, ..., n$ as structural data elements for baseline features, and set $n = 200$ for the time span of feed since the largest number of entries per feed available from our data sets is 200.

Table 2: Results summarized by feed data elements

	P	R	FM
FT	**.977**	.396	.557
FD	.948	.432	.594
FT+FD	.956	.556	.703
ET	.947	.436	.597
ED	.947	.523	.674
ET+ED	.952	.505	.660
FT+ET	.961	.535	.687
FT+ED	.959	.584	.726
FT+ET+ED	.960	.558	.706
FD+ET	.952	.543	.690
FD+ED	.950	.599	.733
FD+ET+ED	.956	.570	.711
FT+FD+ET	.959	.599	.736
FT+FD+ED	.959	**.612**	**.746**
FT+FD+ET+ED	.961	.603	.740

For our experiments, we used the libSVM toolkit [1] to train the SVM models. We work with linear SVM models since the existing literature on text classification indicates that the nonlinear versions of SVM gain very little in terms of performance, compared to the linear version [9]. A five fold cross-validation technique is used to evaluate the performance.

Table 1 shows our experiment design and results for TF, and it contains representative results selected from complete results, which include the significant features, the baseline features, feed data elements, and some combinations of them. Table 2 summarizes feed data elements and their combinations from complete results in order to investigate which feed data elements need to be used for constructing a feature vector. In Table 2, precision, recall and F measure of feed data elements and their combinations represent the maximum values produced in our experiments.

In the result tables, FT stands for feed title, FD for feed description, ET for entry title, and ED for entry description. The first column on the left side in the result tables indicates feed data elements and their combinations. We denote combination of feed data elements for the feature vector with "+". In particular, the second column of Table 1 indicates n, where $n = 0$ means that entries are not used for constructing the feature vector. For instance, when feed data element combination is FT+ET and n=40, f_i is modeled as a feature vector represented by $(\sigma(q, ft_i)$, $\sigma(q, et_{i1})$, ..., $\sigma(q, et_{i40}))$, $i = 1, ..., m$ without features for feed description and entry description. The first row on the top side of the result tables represents evaluation metrics.

In Table 1, numbers in gray boxs are precision, recall and F measure results of the baseline features defined earlier, and bold numbers in white boxs correspond to precision of $SF(P)$, recall of $SF(R)$ and F measure of $S(FM)$. For $SF(P)$, significant features were constructed from FT without entry, and for $SF(R)$ and $SF(FM)$, significant features were built from FT+FD+ED with 40 entries. From the experimentation results, we found that the significant features outperform the baseline features as Table 1 shows. We present a detail analysis based on feed data element and n with respect to precision and F measure results in the following.

Table 2 indicates that FT achieves the best precision, and precision values of FD, ET, and ED are almost same. Also,

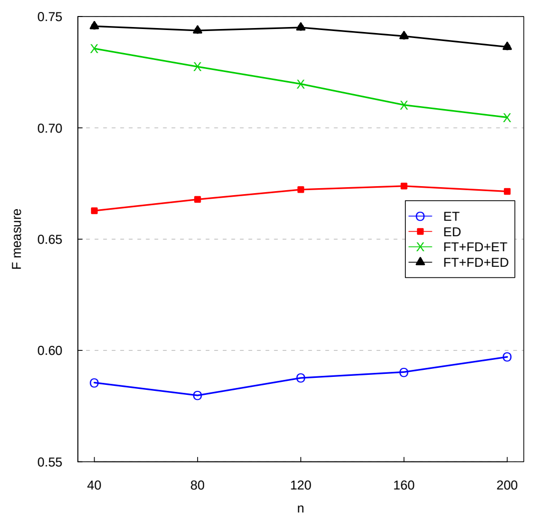


Figure 2: Performance evaluation of feed data elements in various n's

it is remarkable that FT and FD produce higher precision by combining with ET and ED than ET and ED alone. From these observations, we can conclude that FT and FD are more competitive than other feed data elements in terms of the precision performance of feed relevance judgement method.

On the other hand, FT+FD+ED produces the best result for F measure in Table 2. It is interesting to see that FT+FD is more effective than ET and ED for F measure. Furthermore, Table 2 shows that FD and ED with a large number of terms have higher F measure than FT and ET with a small number of terms.

From the fact that FT and FD ahieve low recall and high precision results, it can be concluded that many feeds put their topics into FT and FD, and there is a high probability that terms in FT and FD represent topics of feeds. In addition, FT+FD+ED does not completely outperform FT+FD+ET for F measure, where ED requires high cost for computing features as it has a large number of terms. The difference between F measure of FT+FD+ED and that of FT+FD+ET is only 0.01, suggesting that FT+ FD+ET is more competitive than a combination of all feed data elements, considering the cost for computing feature scores.

Moreover, we examine an effect of n on the performance in Table 1. In general, large n's give better precision results than small n's when entries are used for the feature vector. However, it is interesting to see that precision hardly improves beyond n of 120 as represented in Table 1.

Figure 2 depicts that large n's have better F measure performance than small n's in ET, ED, and ET+ED. On the contrary, small n's yield better performance than large n's for combinations of ET, ED, FT and FD. It turns out that use of FT and FD together with ET and ED is more appropriate to judge relevance of feed than use of ET and ED alone. Accordingly, these results lead us to the conclusion that a large number of entries including old entries are not necessary for constructing the feature vector, but instead that a small number of recent entries are enough to judge relevance of a feed.

Finally, we remark that feed relevance judgement method based on FT+FD+ET and a small number of entries can produce comparable performance to the significant features, $SF(FM)$, while at the same time reducing cost for computing feature scores for entry descriptions.

4. CONCLUSION

In this paper, we investigated the significant features for relevance judgement method in feed retrieval. New feature selection approach for feed relevance judgement was proposed based on the vector space model, considering unique characteristics of feeds.

Extensive experiments were conducted using a data set collected from Google reader. From the experimental results, we found that when a small number of entries are used, feed title, feed description, and entry description become feed data elements for the significant features. In addition, our experimental results show that the relevance judgement method based on feed title, feed descripion and entry title produces similar performance to the identified significant features. It is expected that our research results will contribute to enhancing the retrieval performance of feed search engines.

5. ACKNOWLEDGMENT

This work was supported by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korean government (MOST) (No. R01-2007-000-11167-0).

6. REFERENCES

[1] C. Chang and C. Lin, *LIBSVM : a library for support vector machines*, [Software]. Available: `http://www.csie.ntu.edu.tw/~cjlin/libsvm`. 2001.

[2] W. B. Croft, D. Metzler, and T. Stronhman, *Search engines: information retrieval in practice*, Addison Wesley, 2010.

[3] H. Drucker, D. Wu, and V. Vapnik, "Support vector machines for spam categorization," *IEEE Trans. Neural Networks*, vol. 10(5), pp. 1048-1054, 1999.

[4] J. L. Elsas et al., "Retrieval and feedback models for blog feed search," *Proc. 31st Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR)*, pp. 347-354, 2008.

[5] Google reader, `http://reader.google.com`.

[6] D. Johnson, *RSS and Atom in action: web 2.0 building blocks*, Manning Publications, 2006.

[7] S. Han et al., "Exploring the relationship between keywords and feed elements in blog post search," *World Wide Web*, vol. 12, pp. 381-398, 2009.

[8] Y. H. Li and A. K. Jain, "Classification of text documents," *The Computer Journal*, vol. 41(8), pp. 537-546, 1998.

[9] D. Mladenić et al., "Feature selection using linear classifier weights: interaction with classification models," *Proc. 27nd Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR)*, pp. 234-241, 2004.

[10] L. Zhang, J. Zhu, and T. Yao, "An evaluation of statistical spam filtering techniques," *ACM Trans. Asian Language Information Processing*, vol 3(4), pp. 243-269, 2004.

[11] Z. Zheng et al., "A regression framework for learning ranking functions using relative relevance judgments," *Proc. 30th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR)*, pp. 287-294, 2007.

Representing and Exploiting Searcher Interaction Data

Eugene Agichtein eugene@mathcs.emory.edu
Math & Computer Science Department
Emory University, Atlanta, GA

Understanding the relationship between searcher intent and behavior is crucial for improving web search. Recently, it has been shown that the information contained in fine-grained searcher interactions, such as mouse movements and scrolling, can be useful for predicting user intent, for automatic search evaluation, and for clickthrough prediction. However, representing the searcher behavior and interaction data is a challenging task. This talk will describe our recent experiments on representing and exploiting searcher interaction data for predicting user intent and ad clickthrough behavior.