Learning to Rank Search Results for Time-Sensitive Queries

Nattiya Kanhabua
L3S Research Center
Leibniz Universität Hannover
Hannover, Germany
kanhabua@L3S.de

Kjetil Nørvåg
Dept. of Computer Science
Norwegian University of Science and Technology
Trondheim, Norway
noervaag@idi.ntnu.no

ABSTRACT

Retrieval effectiveness of temporal queries can be improved by taking into account the time dimension. Existing temporal ranking models follow one of two main approaches: 1) a mixture model linearly combining textual similarity and temporal similarity, and 2) a probabilistic model generating a query from the textual and temporal part of document independently. In this paper, we propose a novel time-aware ranking model based on learning-to-rank techniques. We employ two classes of features for learning a ranking model, *entity-based* and *temporal* features, which are derived from annotation data. Entity-based features are aimed at capturing the *semantic similarity* between a query and a document, whereas temporal features measure the *temporal similarity*. Through extensive experiments we show that our ranking model significantly improves the retrieval effectiveness over existing time-aware ranking models.

Categories and Subject Descriptors H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval

General Terms Algorithms, Experimentation

Keywords temporal queries, time-aware ranking models

1. INTRODUCTION

Searching in temporal collections such as news and web archives is not straightforward, because relevant documents are dependent on time. More precisely, documents are about events that happened at a particular time period, and also accesses to the contents are time-sensitive, i.e., time is part of information needs as represented by *temporal queries* (e.g., Illinois earthquake 1968 or Iraq 2001). As shown previously by analyzing real-world query logs, 1.5% of queries are explicitly provided with temporal criteria [19], i.e., containing temporal expressions, while about 7% of web queries have temporal intent implicitly provided [18].

As shown in previous work, taking the time dimension into account in ranking can significantly improve the retrieval effectiveness of temporal queries [2, 7, 5, 14, 16, 18]. The existing time-aware ranking models follows one of two main approaches: 1) a mixture model linearly combining textual similarity and temporal similarity, and 2) a probabilistic model generating a query from the textual and the temporal parts of documents independently.

In this paper, we present a new approach for the above task: *a time-aware ranking model based on learning-to-rank techniques*. A fundamental issue in learning to rank, is the selection of features to be used in learning a ranking model. In this paper, we employ two classes of features that are derived from annotated documents: *entity-based* and *temporal* features. Thus, the main contributions of this paper are: 1) a new time-aware ranking model learned using two classes of features, 2) identification of appropriate features to be used, and 3) extensive experiments for evaluating the proposed time-aware ranking model and the features, using the New York Times Annotated Corpus in combination with temporal queries and relevance assessments from [2].

The organization of the rest of the paper is as follows. In Section 2, we give an overview of related work. In Section 3, we outline the models for documents, annotated documents, and temporal queries, and present the ranking model. In Section 4, we propose two classes of features for learning a time-aware ranking model. In Section 5, we evaluate the proposed ranking model by comparing with existing time-aware ranking methods. Finally, in Section 6, we conclude the paper.

2. RELATED WORK

A number of ranking models exploiting temporal information have been proposed, including [2, 7, 16, 18]. In [16], Li and Croft incorporated time into language models, called time-based language models, by assigning a document prior using an exponential decay function of a document creation date. They focused on recency queries, where the more recent documents obtain higher probabilities of relevance. In [7], Diaz and Jones also used document creation dates to measure the distribution of retrieved documents and create the temporal profile of a query. They showed that the temporal profile together with the contents of retrieved documents can improve average precision for the query by using a set of different features for discriminating between temporal profiles. Berberich et al. [2] integrated temporal expressions into query-likelihood language modeling, which considers uncertainty inherent to temporal expressions in a query and documents, i.e., temporal expressions can refer to the same time interval even if they are not exactly equal. Metzler et al. [18] considered implicit temporal information needs. They proposed mining query logs and analyze query frequencies over time in order to identify strongly time-related queries. Moreover, they presented a ranking concerning implicit temporal needs, and the experimental results showed the improvement of the retrieval effectiveness of temporal queries for web search.

In addition to the work above, there is also work that has fo-

CIKM'12, October 29-November 2, 2012, Maui, HI, USA.

cused on recency ranking [4, 8, 9, 10]. That work is different to our work in term of a search scenario, since in our case a user can issue time as part of a query, so-called *temporal criteria*. There has also been work on analyzing queries over time, e.g., Kulkarni et al. [15] studied how users' information needs change over time, and Shokouhi [21] employed a time series analysis method for detecting seasonal queries. For an entity-ranking task, Demartini et al. [6] analyzed news history (i.e., past related articles) for identifying relevant entities in current news articles.

3. PRELIMINARIES

In this section, we outline the models for documents, annotated documents and temporal queries, and then present a model for ranking documents.

3.1 Annotated Document Model

Our document collection is composed of unstructured text documents: $C = \{d_1, \ldots, d_n\}$. A document d is represented as a bag-of-words or an unordered list of terms: $d_i = \{w_1, \ldots, w_k\}$, where its publication date is denoted *PubTime*(d_i). Since the two classes of features are extracted from annotation data of documents, we present a model of annotated documents as follows.

For each document d_i, its associated annotated document $\hat{d}_i$ is composed of 3 parts. First, $\hat{d}_i$ contains a set of named entities $\{e_1, \ldots, e_n\}$, where a named entity can be a person, location, or organization. The second part is a set of temporal expressions or event dates $\{t_1, \ldots, t_m\}$. Finally, $\hat{d}_i$ contains a set of sentences $\{s_1, \ldots, s_z\}$, where each sentence s_y consists of tokens (terms), the part-of-speech and position information of each token.

A key aspect is that we distinguish between two temporal dimensions associated with a document d_i: 1) publication time (i.e., when a document was published), and 2) content time (i.e., what time a document refers to). The content time of a document (denoted *ContentTime*(d_i)) or temporal expressions mentioned in d_i will be automatically extracted using a time and event recognition algorithm. The algorithm extracts temporal expressions mentioned in a document and normalizes them to dates so they can be anchored on a timeline. As explained in [1], temporal expressions can be explicit, implicit or relative. Examples of explicit temporal expressions are May 25, 2012 or June 17, 2011 that can be mapped directly to dates months, or years on the Gregorian calendar. An implicit temporal expression is an imprecise time point or interval, e.g., Independence Day 2011 that can be mapped to July 04, 2011. Examples of relative temporal expressions are yesterday, last week or one month ago.

3.2 Temporal Query Model

A temporal query q_j is composed of two parts: keywords q_{text}, and temporal expressions q_{time}. Recall that a temporal expression can be explicitly provided as a part of temporal query, or implicitly provided. An example of the first type is Illinois earthquake 1968 where the user is interested in documents about Illinois earthquake in 1968. Queries of the second type can be implicitly associated with particular time especially queries related to major real-world events, or seasonal queries[21]. Examples of a real-world event query and a seasonal query are Thailand tsunami associated with the year 2004 and U.S. presidential election, which can be associated with the years 2000, 2004, and 2008. When q_{time} is not given explicitly by the user, it has to be determined by the system [14]. In this paper, we assume that q_{time} is *explicitly* provided.

3.3 Ranking Model

A ranking model $h(d, q)$ is obtained by training a set of labeled query/document pairs using a learning algorithm. A learned ranking model is essentially a weighted coefficient w_i of a feature x_i. An unseen document/query pair (d', q') will be ranked according to a weighted sum of feature scores:

$$score(d', q') = \sum_{i=1}^{N} w_i \times x_i^{q'}$$

where N is the number of features. Many existing learning algorithms have been proposed, and can be categorized into three approaches: pointwise, pairwise, and listwise approaches. For a more detailed description of each approach, please refer to [17]. In this work, we employ different learning-to-rank algorithms, such as, RankSVM [11], SVM^{MAP} [22], and three stochastic gradient descent algorithms: SGD-SVM [23], PegasosSVM [20], and PA-Perceptron [3].

4. FEATURES

In this section, we present the two classes of features (temporal and entity-based) that are used for learning a time-aware ranking model.

4.1 Temporal Features

Temporal features represent the *temporal* similarity between a query and a document. In this work, we employ five different methods for measuring temporal similarity: LMT and LMTU [2], TS and TSU [14], and FuzzySet [12].

The time-aware ranking methods we study differ from each other in two main aspects: 1) whether or not time uncertainty is concerned, and 2) whether the publication time or the content time of a document is used in ranking. LMT ignores time uncertainty and it exploits the content time of d. LMT can be calculated as:

$$P(t_q|t_d)_{LMT} = \begin{cases} 0 & \text{if } t_q \neq t_d, \\ 1 & \text{if } t_q = t_d. \end{cases}$$

where $t_d \in$ *ContentTime*(d), and the score will be equal to 1 iff a temporal expression t_d is exactly equal to t_q. LMTU concerns time uncertainty by assuming equal likelihood for any time interval t'_q that t_q can refer to, that is, $t_q = \{t'_q | t'_q \in t_q\}$. The simplified calculation of $P(t_q|t_d)$ for LMTU is given as:

$$P(t_q|t_d)_{LMTU} = \frac{|t_q \cap t_d|}{|t_q| \cdot |t_d|}$$

where $t_d \in$ *ContentTime*(d). The detailed computation of $|t_q \cap t_d|$, $|t_q|$ and $|t_d|$ is described in [2].

TS ignores time uncertainty. $P(t_q|t_d)_{TS}$ can be computed similar to $P(t_q|t_d)_{LMT}$, but t_d corresponds to the publication time of d instead of the content time as computed for LMT. TSU exploits the publication time of d as done for TS, but it also takes time-uncertainty into account. $P(t_q|t_d)_{TSU}$ is defined using an exponential decay function:

$$P(t_q|t_d)_{TSU} = DecayRate^{\lambda \cdot \frac{|t_q - t_d|}{\mu}}$$

$$|t_q - t_d| = \frac{|tb_l^q - tb_l^d| + |tb_u^q - tb_u^d| + |te_l^q - te_l^d| + |te_u^q - te_u^d|}{4}$$

where $t_d =$ *PubTime*(d), $DecayRate$ and λ are constant, $0 < DecayRate < 1$ and $\lambda > 0$, and μ is a unit of time distance. The main idea is to give a score that decreases proportional to the time distance between t_q and t_d. The less time distance, the more temporally similar they are.

FuzzySet measures temporal similarity using a fuzzy membership function and it exploits the publication time of d for determining temporal similarity. $P(t_q|t_d)_{FuzzySet}$ is given as:

$$P(t_q|t_d)_{FuzzySet} = \begin{cases} 0 & \text{if } t_d < a_1, \\ f_1(t_d) & \text{if } t_d \geq a_1 \wedge t_d \leq a_2, \\ 1 & \text{if } t_d > a_2 \wedge t_d \leq a_3, \\ f_2(t_d) & \text{if } t_d > a_3 \wedge t_d \leq a_4, \\ 0 & \text{if } t_d > a_4. \end{cases}$$

where $t_d = \textit{PubTime}(d)$. $f_1(t_d)$ is $\left(\frac{a_1 - t_d}{a_1 - a_2}\right)^n$ if $a_1 \neq a_2$, or 1 if $a_1 = a_2$. $f_2(t_d)$ is $\left(\frac{a_4 - t_d}{a_4 - a_3}\right)^m$ if $a_3 \neq a_4$, or 1 if $a_3 = a_4$. The parameters a_1, a_4, n, m are determined empirically.

4.2 Entity-based Features

In addition to the temporal features presented above, we also use ten entity-based features aimed at measuring the similarity between a query and a document. The intuition is that a traditional term-matching method that use only statistics, e.g., TFIDF, ignores the semantic role of a query term. For example, consider the temporal query Iraq 2001. A statistics-based model will rank a document having many occurrences of the terms Iraq or 2001 *higher than* a document with less frequency of the same terms without taking into account *a semantic relationship* between query terms, which can be determined by, e.g., a term distance in a sentence.

Entity-based features are computed for each entity e_j in an annotated document $\hat{d}_i$, and the proposed features includes *querySim*, *title*, *titleSim*, *senPos*, *senLen*, *cntSenSubj*, *cntEvent*, *cntEventSubj*, *timeDist*, and *tagSim* [13]. The first feature *querySim* is the term similarity score between q_j and an entity e_j in $\hat{d}_i$. Here, we use Jaccard coefficient for measuring term similarity. Feature *title* indicates whether e_j is in the title of d_i. Feature *titleSim* is the term similarity score between e_j and the title. Feature *senPos* gives a normalized score of the position of the 1^{st} sentence where e_j occurs in d_i, while the feature *senLen* gives a normalized score of the length of the 1^{st} sentence of e_j. Feature *cntSenSubj* is a normalized score of the number of sentences where e_j is a subject. Feature *cntEvent* is a normalized score of the number of event sentences (or sentences annotated with temporal expressions) of e_j, while the feature *cntEventSubj* a normalized score of the number of event sentences that e_j is a subject. Feature *timeDist* is a normalized distance score of e_j and a temporal expression within a sentence. Feature *tagSim* is the term similarity score between e_j and an entity tagged in d_i. Note that the last feature is only applicable for a document collection provided with tags (e.g., the New York Times Annotated Corpus).

These features, except *querySim*, can be computed off-line because they are query-independent. In order to represent d_i by a feature vector, we have to select a representative entity e_j in $\hat{d}_i$, by choosing the e_j that is the most similar to a query q_j, or e_j that maximizes *querySim*. As a rule, a typical feature that is used for learning to rank is a retrieved score of a traditional ranking function [17]. We also employ a retrieved score $\textit{retScore}(q_j, d_i)$ as one of the features for learning a ranking model. This score must be normalized (to have a value between 0 and 1) by dividing by $\max_{d_i \in \mathcal{D}_q} \textit{retScore}(q_j, d_i)$ where $\mathcal{D}_q$ is a set of retrieved documents. The detailed computation of the entity-based features can be found in [13].

5. EXPERIMENTS

In this section, we evaluate different time-aware ranking models based on learning-to-rank algorithms. We first describe the experimental setting. Then, we show the experimental results as well as perform a feature analysis.

5.1 Experimental Setting

We used the New York Times Annotated Corpus (containing over 1.8 million news articles published between January 1987 and June 2007) as a temporal document collection, and the 40 temporal queries and crowdsourced relevance assessments from [2]. We employed a series of language processing tools for annotating documents, including OpenNLP (for tokenization, sentence splitting and part-of-speech tagging, and shallow parsing), the SuperSense tagger (for named entity recognition) and TARSQI Toolkit (for annotating documents with TimeML). The result of this is for each document: 1) entity information, e.g., all of persons, locations and organizations, 2) temporal expressions, e.g., all of event dates, and 3) sentence information, e.g., all sentences, entities and event dates occurs in each sentence, as well as position information. For temporal features, an exponential decay rate $DecayRate = 0.5$, and $\lambda_2 = 0.5$ are used. The *fuzzySet* parameters are $n = 2$, $m = 2$, $a_1 = a2 - (0.25 \times (a3 - a2))$, and $a4 = a3 + (0.50 \times (a3 - a2))$. The smoothing parameter λ_1 is varied, and only the results of those performed best will be reported.

For learning a time-aware ranking model, we employed different learning-to-rank algorithms, where default parameters of each learner were used. We performed five-fold cross validation by randomly partitioning 40 temporal queries into five folds (8 queries per fold): F1, F2, F3, F4, and F5. For each fold, four other folds (4*8=32 queries) are used for training a ranking model. In order to evaluate ranking models, the Apache Lucene search engine version 2.9.3 was employed. We have five competitive baselines. The first baseline is Lucene's default similarity function (a variant of TFIDF). The four other baselines are proposed in [2]: LMT-IN, LMT-EX, LMTU-IN, and LMTU-EX, where suffixes IN and EX refer to *inclusive* and *exclusive* mode respectively (whether query's temporal expressions are also included as a part of query keywords q_{text} or are excluded). The baseline TFIDF treats query's temporal expressions as a part of q_{text}, i.e., the inclusive mode. The retrieval effectiveness of time-aware ranking is measured by the precision at 1, 5 and 10 documents (P@1, P@5 and P@10 respectively), Mean Reciprocal Rank (MRR), and Mean Average Precision (MAP). The average performance over the five folds is used to measure the overall performance of each ranking model. For all experiments, we measured statistical significance using a t-test with $p < 0.05$. In the tables of results, bold face is used to indicate statistically significant difference from the respective baselines.

5.2 Experimental Results

The ranking performance of the baselines and learned ranking models are displayed in Table 1. The results among the baselines are similar to those reported in [2]. In general, the exclusive mode performed better than the inclusive mode for both LMT and LMTU, and LMTU-EX gained the best performance over the other baselines.

Comparing different ranking models, RankSVM did not gain a significant improvement over the baselines, while PegasosSVM performed worse than the baselines and other learned ranking models. SGD-SVM and PegasosSVM achieved the improvement over the baselines in all measurements. Finally, the listwise ranking SVM^{MAP} performed better than the pairwise models, and also outperformed all the baselines significantly. Using P@1, SVM^{MAP} achieved the improvement over TFIDF and LMTU-EX up to 27.5% and 15% respectively. Using MAP, SVM^{MAP} achieved the improvement over TFIDF and LMTU-EX up to 13.1% and 8.2% respectively.

In order to understand the importance of each feature, we performed feature analysis and the results are shown in Table 2. $\bar{x}_i$ is the average of each feature's values. w_i is a feature's weight obtained from the learning method SVM^{MAP}. The top-5 features with highest weights are *querySim*, *TS*, *FuzzySet*, *retScore* and *senPos*. Entity-based features, i.e., *querySim*, *retScore* and *senPos*, received high weights because they are well represented the importance of query terms within a document. It is interesting that *TS* and *FuzzySet* gained higher weights than other temporal features, although they exploited publication time instead of the content time of a document, whereas *TS* did not consider time uncertainty. More-

Table 1: Effectiveness of different ranking models.

Model	P@1	P@5	P@10	MRR	MAP
TFIDF	.375	.435	.410	.562	.486
LMT-IN	.500	.370	.373	.625	.428
LMT-EX	.425	.395	.385	.588	.447
LMTU-IN	.475	.450	.433	.635	.475
LMTU-EX	.500	.520	.520	.670	.535
RankSVM	.500	.550	.515	.661	.578
SGD-SVM	.575	**.610**	.540	.706	.595
PegasosSVM	.550	**.610**	.543	.690	.595
PA-Perceptron	.500	.455	.433	.630	.496
SVM^{MAP}	**.650**	.605	**.565**	**.753**	**.617**

Table 2: Feature analysis results.

Feature	$\bar{x_i}$	w_i	add_1	add_2	*remove*
retScore	.49	**1.65**	0.00	0.00	-0.25
querySim	.52	**6.29**	**6.45**	**5.50**	**4.55**
title	.05	-0.94	0.77	0.85	0.00
titleSim	.08	-0.77	0.95	0.78	**1.16**
senPos	.74	**1.60**	**1.93**	0.50	**1.05**
senLen	.64	-0.66	**2.78**	**1.80**	**1.88**
cntSenSubj	.02	0.08	0.12	-0.05	0.00
cntEvent	.14	0.23	-0.02	0.79	0.01
cntEventSubj	.02	0.14	0.04	-0.03	0.02
timeDist	.19	0.27	-0.12	0.38	-0.03
tagSim	.18	1.37	1.78	1.06	0.87
LMT	.30	-1.92	1.56	0.92	0.15
LMTU	.83	-0.33	-0.32	0.15	0.16
TS	.25	**2.86**	**4.04**	**4.82**	0.62
TSU	.26	0.95	1.13	1.15	**1.21**
FuzzySet	.29	**2.37**	**4.00**	**4.53**	0.27

over, the results show that *LMT* and *LMTU* received negative weights indicating a negatively correlation with the retrieval effectiveness.

In order to observe the performance of individual features, we conducted 3 additional experiments and measure the improvement in (%)MAP. First, we trained a ranking model with SVM^{MAP} using only *retScore* and selected one additional feature at each time to observe how the selected feature contributes to a ranking model. A baseline in this case is the model trained using *retScore* only with MAP of 0.483. The column add_1 shows the improvement in (%)MAP that each feature could produce on its own compared to the baseline. The top-5 features contributes in MAP for this analysis are *querySim*, *TS*, *FuzzySet*, *senLen*, and *senPos*, while adding *cntEvent*, *timeDist*, or *LMTU* results in the decreased performance.

We then inspected how a single feature contributed to a ranking model when trained using *retScore* and *another feature class*. There are two baselines in this case: 1) the model trained only with *retScore* and all *temporal* features with MAP of 0.537, and 2) the model trained only with *retScore* and all *entity-based* features with MAP of 0.557. The column add_2 shows the improvement that each of entity-based features contributed to the first baseline model, and on the contrary, its shows the improvement that each of *temporal* features contributed to the second model. In summary, the top-2 best entity-based features are *querySim* and *senLen*, and the top-2 best temporal features are *TS* and *FuzzySet*.

Finally, we trained a ranking model using training data that consisted of *all* features except one at each time to see how a ranking model is dependent on that feature. The baseline is the model trained with *all* features, and its performance (MAP) is 0.617. The column *remove* shows the decrease of performance compared to the baseline, which is obtained by removing each feature. The top-5 features that made a significant drop in performance are *querySim*, *senLen*, *TSU*, *titleSim*, and *senPos*.

6. CONCLUSIONS

In this paper, we have proposed a time-aware ranking approach based on learning-to-rank techniques for temporal queries. In order to learn the ranking model, we employed two classes of features derived from annotation data, namely, entity-based and temporal features. Through extensive experiments we have shown that the proposed learning-to-rank model significantly improves the retrieval effectiveness over existing time-aware ranking models.

Acknowledgments

We would like to thank Klaus Berberich for providing the temporal queries and relevance judgments used in this paper.

7. REFERENCES

[1] O. Alonso et al. Clustering and exploring search results using timeline constructions. In *Proceedings of CIKM'2009*, 2009.

[2] K. Berberich et al. A language modeling approach for temporal information needs. In *Proceedings of ECIR'2010*, 2010.

[3] K. Crammer et al. Online passive-aggressive algorithms. *J. Mach. Learn. Res.*, 7:551–585, 2006.

[4] N. Dai, M. Shokouhi, and B. D. Davison. Learning to rank for freshness and relevance. In *Proceeding of SIGIR'2011*, 2011.

[5] W. Dakka, L. Gravano, and P. G. Ipeirotis. Answering general time-sensitive queries. In *Proceeding of CIKM'2008*, 2008.

[6] G. Demartini et al. TAER: time-aware entity retrieval-exploiting the past to find relevant entities in news articles. In *Proceedings of CIKM'2010*, 2010.

[7] F. Diaz and R. Jones. Using temporal profiles of queries for precision prediction. In *Proceedings of SIGIR'2004*, 2004.

[8] A. Dong et al. Time is of the essence: improving recency ranking using twitter data. In *Proceedings of WWW'2010*, 2010.

[9] J. L. Elsas and S. T. Dumais. Leveraging temporal dynamics of document content in relevance ranking. In *Proceedings of WSDM'2010*, 2010.

[10] A. Jatowt, Y. Kawai, and K. Tanaka. Temporal ranking of search engine results. In *Proceedings of WISE'2005*, 2005.

[11] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of KDD'2002*, 2002.

[12] P. J. Kalczynski and A. Chou. Temporal document retrieval model for business news archives. *Inf. Process. Manage.*, 41, 2005.

[13] N. Kanhabua, R. Blanco, and M. Matthews. Ranking related news predictions. In *Proceeding of SIGIR'2011*, 2011.

[14] N. Kanhabua and K. Nørvåg. Determining time of queries for re-ranking search results. In *Proceedings of ECDL'2010*, 2010.

[15] A. Kulkarni et al. Understanding temporal query dynamics. In *Proceedings of WSDM'2011*, 2011.

[16] X. Li and W. B. Croft. Time-based language models. In *Proceedings of CIKM'2003*, 2003.

[17] T.-Y. Liu. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 3(3):225–331, 2009.

[18] D. Metzler et al. Improving search relevance for implicitly temporal queries. In *Proceedings of SIGIR'2009*, 2009.

[19] S. Nunes, C. Ribeiro, and G. David. Use of temporal expressions in web search. In *Proceedings of ECIR'2008*, 2008.

[20] S. Shalev-Shwartz et al. Pegasos: Primal estimated sub-gradient solver for SVM. In *Proceedings of ICML'2007*, 2007.

[21] M. Shokouhi. Detecting seasonal queries by time-series analysis. In *Proceeding of SIGIR'2011*, 2011.

[22] Y. Yue et al. A support vector method for optimizing average precision. In *Proceedings of SIGIR'2007*, 2007.

[23] T. Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of ICML'2004*, 2004.