Unifying Learning to Rank and Domain Adaptation: Enabling Cross-Task Document Scoring *

Mianwei Zhou, Kevin Chen-Chuan Chang
Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA
zhou18@illinois.edu, kcchang@illinois.edu

ABSTRACT

For document scoring, although *learning to rank* and *domain adaptation* are treated as two different problems in previous works, we discover that they actually share the same challenge of adapting keyword contribution across different queries or domains. In this paper, we propose to study the *cross-task document scoring* problem, where a task refers to a query to rank or a domain to adapt to, as the first attempt to unify these two problems. Existing solutions for learning to rank and domain adaptation either leave the heavy burden of adapting keyword contribution to feature designers, or are difficult to be generalized. To resolve such limitations, we abstract the *keyword scoring principle*, pointing out that the contribution of a keyword essentially depends on, first, its importance to a task and, second, its importance to the document. For determining these two aspects of keyword importance, we further propose the concept of *feature decoupling*, suggesting using two types of easy-to-design features: *meta-features* and *intra-features*. Towards learning a scorer based on the decoupled features, we require that our framework fulfill *inferred sparsity* to eliminate the interference of noisy keywords, and employ *distant supervision* to tackle the lack of keyword labels. We propose the *Tree-structured Boltzmann Machine* (T-RBM), a novel two-stage Markov Network, as our solution. Experiments on three different applications confirm the effectiveness of T-RBM, which achieves significant improvement compared with four state-of-the-art baseline methods.

Categories and Subject Descriptors

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

Keywords

Learning to Rank; Domain Adaptation; Feature Decoupling; Tree-Structured Restricted Boltzmann Machine

*This material is based upon work partially supported by NSF Grant IIS 1018723, the Advanced Digital Science Center of UIUC and the Multimodal Information Access and Synthesis Center at UIUC. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reject the views of the funding agencies.

KDD'14, August 24–27, 2014, New York, NY, USA.

ACM 978-1-4503-2956-9/14/08 ...$15.00.
http://dx.doi.org/10.1145/2623330.2623739.

1. INTRODUCTION

With much information in the world represented by unstructured text, many applications study how to develop document *scorers*, which can analyze the content of a document and determine its relevance for some information need, *e.g.*, text categorization [8], document retrieval [5]. Since manually crafting scorers is difficult, people are interested in how to automatically *learn* such scorers from labeled documents.

Scorer learning techniques have been actively studied for years – unlike early applications which usually deal with only one topic (*e.g.*, judging if a document is about "finance" [8]), with the prevalence of the Web serving the whole world of users and connecting numerous sources of data, most applications nowadays must handle various user needs, *i.e.*, *queries*, and diverse sources, *i.e.*, *domains*.

First, in terms of *queries*, many applications aim at ranking documents for queries that represent users' information needs, *i.e.*, realizing *learning to rank*. Unlike traditional IR which deals with short queries, learning to rank applications nowadays have to handle more sophisticated queries:

Application 1: *Verbose-Query-based Retrieval* [11, 2], which addresses verbose queries that consist of one or more long sentences. It is more challenging than traditional IR, because long queries usually contain extraneous terms which might hinder retrieval of relevant documents.

Application 2: *Entity-Centric Document Filtering* [22], which studies, given an entity (*e.g.*, a person) characterized by an identification page (*e.g.*, her Wikipedia page) as a query, how to identify documents relevant to the entity. Since an identification page is usually long and noisy, the solution has to identify keywords that represent the characteristics of the entity (*e.g.*, "microsoft" for entity "Bill Gates") for better retrieval accuracy.

Second, in terms of *domains*, traditional scorer learning techniques work well only when the training and testing documents use similar distributions of keywords, *i.e.*, documents are from the same domain. Due to the expensive cost of labeling, we expect to learn a document scorer that could be adapted across different domains, *i.e.*, realizing *domain adaptation*, for example:

Application 3: *Cross-Domain Sentiment Analysis* [4], which studies how to adapt a sentiment scorer across different domains of reviews. It is challenging because people tend to use different sentiment keywords in different domains, *e.g.*, "boring" in book reviews, and "leaking" in kitchen appliance reviews instead.

We observe that, although learning to rank and domain adaptation seem distinct from each other, both of them have to handle the varying importance of each *keyword* in different queries/domains (in this paper, we do not consider other properties which are orthogonal to the keyword content, *e.g.*, pagerank, although they can be easily incorporated as well). For example, in entity-centric document

filtering, keyword "Microsoft" is important for query "Bill Gates," but not for "Michael Jordan;" in sentiment analysis, as mentioned earlier, different domains would use different keywords to represent the same sentiment. Therefore, as the common challenge for these two types of applications, both of them have to consider how to bridge keywords across different queries or domains.

Problem – Cross-Task Document Scoring. As the *first contribution* of this paper, observing such a common challenge, we propose to study a general *cross-task document scoring* problem, which, as far as we know, is the first attempt to unify learning to rank and domain adaptation. Formally, cross-task document scoring aims at learning a scorer $\mathcal{F}(d,t)$ to predict the relevance of document d for task t, where the notion of "*task*" represents the scoring of documents for one query in one domain. In the training phase, we are given some documents labeled for some tasks t (*e.g.*, some queries) to learn $\mathcal{F}(d,t)$; in the testing phase, as highlighted by "cross-task," $\mathcal{F}(d,t)$ should be capable of handling new tasks t' (*e.g.*, new queries) which do not appear in the labeled data.

Challenge – Learning to Adapt Keyword Contribution. As our *second contribution*, we identify the core challenge of cross-task document scoring as learning to adapt keyword contribution across tasks. Formally, if we use *Contrib*(w,d,t) to denote the contribution of keyword w for document d with respect to task t, the relevance of document d is essentially the accumulation of its keywords' contribution *Contrib*(w,d,t). Therefore, the challenge of learning $\mathcal{F}(d,t)$ lies in how to determine keyword contribution *Contrib*(w,d,t) for tasks that do not appear in the training data, *i.e.*, *learning to adapt keyword contribution*.

Learning to adapt keyword contribution is critical for enabling cross-task scoring; however, traditional learning to rank frameworks (*e.g.*, RankSVM [10]) simply circumvent the problem. To learn a scorer, such frameworks adopt an ensembling idea to combine a set of manually crafted sub-scorers $f_k(d,t)$ as features, *e.g.*, in document retrieval [5, 14], $f_k(d,t)$ could be BM25, language model. These learning to rank frameworks are feasible for traditional document retrieval, as there exist many readily studied scorers that could be used as features; however, for newly proposed applications, such scorers are seldom available, and we need laborious feature engineering in order to adopt these frameworks.

Insight – Keyword Scoring Principle. As our *third contribution*, towards learning to adapt keyword contribution, we propose our key insight:

Keyword Scoring Principle – the importance of keyword w for document d with respect to task t, i.e., Contrib(w,d,t), depends on: 1. the importance of keyword w for task t; 2. the importance of keyword w for document d.

Although such a principle is not abstracted before, it is intuitive and implicitly followed in the previous design of manually crafted scorers. Take BM25 as example: in terms of the first aspect, BM25 assumes that keywords with high inverse document frequency (*i.e.*, high *IDF*) are more important for the query; in terms of the second aspect, BM25 assumes that keywords mentioned a lot in the document (*i.e.*, high *TF*) should have higher contribution.

Abstract – Feature Decoupling. As our *fourth contribution*, to fulfill our goal of learning *Contrib*(w,d,t) based on the principle, we propose the idea of *feature decoupling*, which suggests "decoupling" the original scorer-as-feature design in traditional learning to rank frameworks into two types of more elementary features:

To determine "the importance of keyword w for task t," we design *meta-features*, denoted by $f_k^{(M)}(w,t)$, to represent task-related keyword properties. In learning to rank, some recent works [1, 11, 3, 22] adopt such an idea, *e.g.*, using meta-features such as keyword position to identify important keywords from queries. In domain adaptation, Blitzer *et al.* [4] propose a model to use keyword correlation to bridge keywords from different domains. We can use meta-features to realize the same insight, which will be discussed in details in Section 3.4.

To determine "the importance of keyword w for document d," we propose the concept of *intra-features*, denoted by $f_k^{(I)}(w,d)$, to characterize how keyword w occurs in document d. The motivation is that, besides simply counting keywords as most applications do, we can characterize the keyword occurrence more generally and systematically for higher prediction accuracy. For example, keywords that appear in the title, anchor text or URL usually have larger contribution to the relevance.

Given such a decoupled feature design, we have to appropriately "re-couple" $f_k^{(M)}(w,t)$ and $f_k^{(I)}(w,d)$ to determine *Contrib*(w,d,t), and $\mathcal{F}(d,t)$ finally takes the following abstraction:

$$\mathcal{F}(d,t) : \langle f_1^{(M)}(w,t), ...; f_1^{(I)}(w,d), ... \rangle \rightarrow \mathbb{R} \qquad (1)$$

To the best of our knowledge, such a learning framework, which aims at learning a scorer upon two types of elementary features, has not been studied before (previous works [1, 11, 3, 22] which have the concept of meta-features do not model intra-features).

Towards learning keyword contribution based on decoupled features, our framework has to fulfill two requirements:

Requirement 1: Inferred Sparsity. Unlike traditional learning to rank frameworks [5, 14] which do not model the concept of keywords, our scorer should be aware of the potential interference from noisy keywords. For example, both verbose-query-based retrieval and entity-centric document filtering focus on long queries, in which many keywords are unrelated to user intent or target entities; in sentiment analysis, a review usually contains many keywords that are irrelevant to sentiment. Even if we assign such keywords with small contribution, their values, once accumulated, will still severely affect the document score.

Therefore, in order to filter noisy keywords, we require that the keyword contribution function *Contrib*(w,d,t) be *sparse*, which only outputs non-zero values for important keywords. Different from traditional sparse learning [20] which aims at learning sparse feature weightings to enforce feature sparsity, our goal is to *sparsify the inferred value of a function*, *i.e.*, achieving *inferred sparsity* (their difference will be further discussed in Section 4.1).

Requirement 2: Distant Supervision. Toward realizing learning to score keywords, another challenge arises from the lack of the keyword labels. In most applications, we are only provided with document labels, and it is impractical to request manual keyword labels for learning *Contrib*(w,d,t). Therefore, we require that the scorer should fulfill *distant supervision*, by only using document labels to "distantly" guide the adaptation of keyword contribution.

Solution – Tree-Structured Restricted Boltzmann Machine. As our *fifth contribution*, to fulfill these two requirements, we propose a novel *Tree-structured Restricted Boltzmann Machine* (T-RBM) model for the cross-task document scoring problem.

First, to achieve *inferred sparsity*, the model needs a scoring scheme which can eliminate the contribution of noisy keywords. We develop a *two-stage* procedure: in the first stage, we learn a classifier to discretize the importance of keywords into different levels based on their meta-features, and *regularize* the classifier to enforce the elimination of noisy keywords; in the second stage, we determine the contribution of important keywords by their intra-features and set the contribution of unimportant ones to be zero.

Second, to achieve *distant supervision*, we propose to *join* the two stages in one model. Specifically, we take advantage of Markov Network to connect keyword importance and document relevance,

such that we can use document labels to directly supervise the learning of the keyword classifier.

Based on these ideas, we design T-RBM, which, as a variant of Restricted Boltzmann Machine [19] (one type of bipartite Markov Network), models each document as a tree graph with the root node representing a document and the leaf nodes representing its keywords. Free of loop structures, T-RBM could be efficiently trained by exact belief propagation [17].

In the experiments, we performed our evaluation on verbose-query-based retrieval, entity-centric document filtering and cross-domain sentiment analysis in three different datasets. By comparing T-RBM with four state-of-the-art baselines, we observed that our framework not only provides a conceptually unified modeling but also significantly improves the results on different applications.

2. RELATED WORK

In terms of *abstraction*, learning to rank and domain adaptation are separately abstracted and studied in previous works, *e.g.*, document retrieval [5, 1, 11, 3], entity-centric document filtering [22], cross-domain sentiment analysis [4]. Inspired by their works, we identify their common challenge and propose a novel framework to unify these two problems for achieving a more general solution.

In terms of *challenge*, cross-task document scoring boils down to learning to adapt keyword contribution across different tasks.

1. For *learning to rank*, most previous works [5, 14] simply circumvent the challenge, leaving the burden of determining keyword contribution to feature designers. Some of the recent works [1, 11, 3, 22] start to confront the challenge by modeling keyword-level features, *i.e.*, meta-features, to learn keyword contribution; however, none of them explicitly model intra-features, failing to capture different kinds of keyword occurrences in the document.
2. For *domain adaptation*, Blitzer *et al.* [4] propose structural correspondence learning (SCL) to bridge keywords from different domains. The intuition is that, given two domain-specific keywords (*e.g.*, "boring" from book reviews and "leaking" from kitchen appliance reviews), if both of them co-occur a lot with some pivot keywords (*i.e.*, keywords like "bad," "worst" which are commonly used in all domains), these two keywords should share similar sentiment (*e.g.*, both "boring" and "leaking" represent negative sentiment). To realize such an insight, SCL learns a set of pivot predictors, each of which predicts the occurrence of one pivot keyword based on other domains-specific keywords, to relate different domains. Li *et al.* [13] and Pan *et al.* [16] follow the same intuition but use different approaches such as feature alignment [16] and matrix decomposition [13]. Different from these approaches which "hardcode" the logic of keyword adaptation in the model design, our feature decoupling idea allows designers to conveniently incorporate different ways of keyword adaptation by meta-features.

In terms of *technique*, we propose T-RBM, a novel two-stage Markov Network taking the decoupled features as input and fulfilling the requirements of inferred sparsity and distant supervision.

1. With respect to *inferred sparsity*, unlike sparse learning [20] which learns sparse parameters as feature weightings, we aim at sparsifying the inferred value of the keyword contribution function. Existing meta-feature-based solutions have different limitations in fulfilling this requirement. The solutions proposed by Lease *et al.* [11], Bendersky *et al.* [3] and Zhou *et al.* [22] do not fulfill the requirement, making the prediction result vulnerable to noisy keywords. Zhou *et al.* [22] propose another BoostMapping model, which achieves inferred sparsity by clustering keywords based on their meta-features and eliminating noisy clusters; as the limitations, BoostMapping is difficult to solve when we have to model multiple intra-features, and might overfit the training data. We will compare T-RBM with these models in details in Section 4.1.
2. With respect to the requirement of *distant supervision*, similar to T-RBM, Bendersky *et al.* [1] propose to learn a keyword classifier to discover important concepts for retrieval; however, their solution relies on the existence of keyword labels, which is impractical for most applications. Different from their work, taking the advantage of Markov Network, T-RBM manages to train the keyword classifier based on only document labels.
3. With respect to the *model structure*, the most related work to ours is the Markov Random Field model proposed by Lease *et al.* [11]. As the key difference, their solution is a generative model characterizing $P(q, d)$ – the joint probability of observing query q and document d, while our T-RBM model directly models the conditional probability $P(d|q)$. It has been repeatedly confirmed that a discriminative model usually yields better generalization performance compared with a generative model [21]; furthermore, as mentioned earlier, their solution does not fulfill the inferred sparsity requirement.
4. Restricted Boltzmann Machine (RBM), as one type of bipartite Markov Network, is adopted in many applications such as topic modeling [18], deep learning [6], *etc.*With distinct settings and objectives, our proposed T-RBM is different from traditional RBM models in two aspects: first, T-RBM takes a tree structure which can be trained efficiently; second, T-RBM adopts a novel hidden variable regularization technique, which allows the model to control the inferred sparsity of keyword contribution.

3. CROSS-TASK DOCUMENT SCORING

In this section, we formally define the *cross-task document scoring* problem. To tackle the challenge of *learning to adapt keyword contribution*, we propose the insight of *keyword scoring principle* and the concept of *feature decoupling*.

3.1 Problem: Cross-Task Document Scoring

In document scoring, we aim at predicting the relevance of a document d for a particular task t. Formally, task t could be represented as a function $t : d \to \mathbb{R}$, which takes a document d as input, and outputs a score denoting the relevance of d. We define that two tasks t_1 and t_2 are different, if there exists one document d satisfying $t_1(d) \neq t_2(d)$. Therefore, identifying relevant documents for different queries belongs to two different tasks, as one document usually has different relevance for two queries; similarly, the sentiment judgement of a book review is also different from that of a kitchen appliance review.

In contrast with *single-task document scoring* which tackles only one task (*e.g.*, text categorization [8] learns a scorer to predict if a document is about a fixed topic such as "finance"), in a *cross-task document scoring* problem, we are interested in a set of different but related tasks $\mathcal{T}$. For example, in verbose-query-based retrieval and entity-centric document filtering, each $t \in \mathcal{T}$ represents a task of predicting document relevance for one verbose query or one entity; in cross-domain sentiment analysis, each task t is to judge the sentiment of reviews form one particular domain.

Formally, in *cross-task document scoring*, our goal is to automatically learn a document scorer $\mathcal{F}(d, t)$, which could output the relevance of document d for task $t \in \mathcal{T}$.

In *the training phrase*, we are given a set of training documents $\mathbf{d} = \{d_1, d_2, ..., d_N\}$ and a list of document labels $\hat{\mathbf{y}} = \{\hat{y}_1, ..., \hat{y}_N\}$, where each d_i is labeled by $\hat{y}_i$ denoting the relevance of d_i for task $t_i \in \mathcal{T}$ (t_i and t_j could refer to the same task, indicating d_i and d_j are labeled for the same query or in the same domain). Here, $\hat{y}_i$ could be either binary (*e.g.*, 0–negative, 1–positive) or ordinal (*e.g.*, 0–irrelevant, 1–relevant or 2–perfectly relevant).

Based on the training documents, we aim at learning $\mathcal{F}$, which, in *the testing phase*, could predict the relevance of a new document d' for task $t' \in \mathcal{T}$. Specifically, as highlighted by "cross-task," we require that t' should differ from all the training tasks, *i.e.*, $t' \neq t_i$ for all i. That is because, in most applications, the target task set $\mathcal{T}$ could not be covered by finite training examples (*e.g.*, there are infinite possible queries in document retrieval) and thus, $\mathcal{F}$ should be adaptable to unseen queries or domains, *i.e.*, new tasks.

3.2 Challenge: Learning to Adapt Keyword Contribution

To score a document d for a task t, the scorer has to assess the content of document d. Formally, we use $\mathcal{V}(t)$ to represent the vocabulary of keywords that are considered in task t, where each keyword $w \in \mathcal{V}(t)$ could be 1-gram, 2-gram, noun phrases, *etc.* Following previous works [3, 22, 4], in verbose-query-based retrieval and entity-centric document filtering, $\mathcal{V}(t)$ covers keywords that are mentioned in the query or the entity identification page, while in cross-domain sentiment analysis, $\mathcal{V}(t)$ includes all the keywords. We then define $\mathcal{W}(d,t) \subseteq \mathcal{V}(t)$ as the content of document d that is related with task t.

Based on the document content $\mathcal{W}(d,t)$, the relevance of document d could be viewed as the accumulation of its containing keywords' contribution. Formally, if we use *Conrib*(w,d,t) to denote the contribution of keyword w to document d with respect to task t, $\mathcal{F}(d,t)$ takes the form of

$$\mathcal{F}(d,t) \propto \sum_{w \in \mathcal{W}(d,t)} \textit{Conrib}(w,d,t) \tag{2}$$

As Eq. 2 shows, the challenge of cross-task document scoring becomes how to learn *Conrib*(w,d,t) to determine the keyword contribution for new tasks t' which do not appear in the training data, *i.e.*, *learning to adapt keyword contribution*.

3.3 Insight: Keyword Scoring Principle

Due to the requirement of "cross-task," the learning of *Conrib*(w,d,t) is non-trivial. If our target is a single-task document scoring problem (*i.e.*, tackling the same task in both training and testing phases), as a common solution [8], we can define *Conrib*$(w,d,t) = \alpha_w * \textit{TF}_w(d)$, where each feature function $\textit{TF}_w(d)$ represents how many times document d contains a specific keyword w, and α_w, as its weighting, characterizes the importance of keyword w (*e.g.*, in text categorization , if the target topic is about finance, keyword "stock" should have value of α_w). The scorer is then defined by

$$\mathcal{F}(d,t) = \sum_{w \in \mathcal{W}(d,t)} \alpha_w * \textit{TF}_w(d) \tag{3}$$

where α_w could be learned by standard learners. However, such a design could not be applied in cross-task document scoring, because, as we mentioned in Section 1, one keyword usually has very different importance for different tasks, and thus, α_w learned from training data could not be adapted to new tasks.

Traditional learning to rank frameworks (*e.g.*, RankSVM [10]) manage to tackle different tasks (*i.e.*, queries); however, they circumvent the problem of learning *Conrib*(w,d,t), which define the scorer by

$$\mathcal{F}(d,t) = \sum_{k=1}^{N} \beta_k f_k(d,t) \tag{4}$$

where $f_{k \in \{1 \ldots N\}}(d,t)$ is a set of handcrafted sub-scorer features and β_k is learned to represent the confidence of $f_k(d,t)$, *e.g.*, in document retrieval [5], $f_k(d,t)$ could be vector space model, BM25 and language model. As we discussed in Section 1, such frameworks require designers to manually determine the keyword contribution in the feature design, laying heavy burden on designers.

In order to automatically learn the contribution of keywords, as our key insight, we propose the keyword scoring principle:

Definition 1 (Keyword Scoring Principle). If we use $\mathcal{R}_t(w,t)$ to denote the importance of keyword w for task t, and $\mathcal{R}_d(w,d)$ to denote the importance of keyword w for document d, the keyword contribution *Contrib*(w,d,t) should take the following form:

$$\textit{Contrib}(w,d,t) : \mathcal{R}_t(w,t), \mathcal{R}_d(w,d) \rightarrow \mathbb{R} \tag{5}$$

3.4 Abstract: Feature Decoupling

As we introduced in Section 1, towards automatically learning the keyword contribution *Contrib*(w,d,t) based on the principle, we propose the concept of *feature decoupling*, which suggest "decoupling" the original scorer-as-feature design (*i.e.*, $f_k(d,t)$ used in Eq. 4) into two types of more elementary features:

Definition 2 (Feature Decoupling). To learn *Contrib*(w,d,t), we propose to design, first, *meta-features* $f_k^{(M)}(w,t)$, which characterize task-t-related properties of keyword w, for determining the importance of keyword w for task t, *i.e.*, defining $\mathcal{R}_t(w,t)$ by

$$\mathcal{R}_t(w,t) : \langle f_1^{(M)}(w,t), f_2^{(M)}(w,t), ... \rangle \rightarrow \mathbb{R} \tag{6}$$

and, second, *intra-features* $f_k^{(I)}(w,d)$, which describe how document d contains keyword w, for determining the importance of keyword w for document d, *i.e.*, defining $\mathcal{R}_d(w,d)$ by

$$\mathcal{R}_d(w,d) : \langle f_1^{(I)}(w,d), f_2^{(I)}(w,d), ... \rangle \rightarrow \mathbb{R}. \tag{7}$$

Figure 1 lists all the features we use for verbose-query-based retrieval, entity-centric document filtering, and cross-domain sentiment analysis.

First, for *meta-features*, the designs vary a lot across applications:
a) In verbose-query-based retrieval, meta-features are designed to identify important keywords in the query. For example, *QueryPos* is used, as keywords mentioned earlier in the query tend to be more important. Following previous works [3], we model not only unigram, but also adjacent bigrams in the query. To discriminate their importance, we design separate features for them, *e.g.*, *QueryPos*[unigram] and *QueryPos*[bigram].
b) In entity-centric document filtering, we only model unigrams, because the query is already very long, and we find that considering bigrams does not help improve the performance. In additional to the features used in verbose-query-based retrieval, we also design meta-features like *TFInTitle*, *TFInInfoBox*, to leverage the structure of Wikipedia pages for identifying importance keywords.
c) In cross-domain sentiment analysis, the design of meta-features is very different from the other applications. In order to realize the insight proposed by Blitzer *et al.* [4] (introduced in Section 2), given a domain-specific keyword w, we propose to calculate a list of meta-features *Corr*$[w_p]$, each of which measures the Pearson correlation between w and one particular pivot keyword w_p – the correlation is positive if two keywords are positively correlated, negative if negatively correlated and zero if independent.

Second, *intra-features* characterize how a document contains a keyword. As discussed in Section 1, our designed intra-features describe how a keyword appears in different positions of a document (*e.g.*, title, anchor text) by different representations of term frequency (*e.g.*, term frequency normalized by document length). In verbose-query-based retrieval, as suggested by Bendersky *et al.* [3], we also design *WindowTF* specifically for bigrams, which counts the number of times that two keywords appear within a fixed size of window.

Name	Description
Meta-Feature $f_k^{(M)}(w,t)$	
QueryTF	Term Frequency of w is in the query (or the Wiki page).
IDF	The inverse term frequency of w
IsNoun (Vb/ Adj/ Adv/Num)	If w is a noun/verb/adjective/adverb/number
QueryPos	The first position where w is mentioned.
TFInTitle (InfoBox/OpenPara /AnchorEntity)	Term Frequency of w in the Wikipedia title/ the InfoBox/ the Wikipedia opening paragraph/ the anchor entities.
Corr[w_p]	Pearson Correlation between w and one pivot keyword w_p
Intra-Feature $f_k^{(I)}(w,d)$	
DocTF(Raw/Normalized/Log -Scaled)	Term frequency of w in d, which is represented by three features: raw frequency, frequency normalized by length, and logarithmically scaled frequency computed by log (f(w,d) + 1).
AnchorTF(Raw/...)	Term frequency of w in the anchor text of d
TitleTF(Raw/...)	Term frequency of w in the title of d
WindowsTF(Raw/...)	Only for Bigram. The number of times two keywords appear within a fixed size of window.
Doc-Feature $f_k^{(D)}(d)$	
DocLen	The document length of d
PivotTF[w_p]	Term Frequency of pivot keyword w_p in document d

Type	Feature List
Verbose-Query-based Retrieval	
Meta-Feature	[UniGram/Bigram] *QueryTF, IDF, QueryPos* [UniGram] *IsNoun(Vb/...)*
Intra-Feature	[UniGram/Bigram] *DocTF(Raw/...), AnchorTF(Raw/...), TitleTF(Raw/...)* [Bigram] WindowTF (Raw/...)
Doc-Feature	*DocLen*
Entity-centric document Filtering	
Meta-Feature	*QueryTF, IDF, IsNoun(Vb/ ...), QueryPos, TFInTitle(InfoBox/...)*
Intra-Feature	*DocTF(Raw/...), DocAnchorTF(Raw/...), DocTitleTF(Raw/...)*
Doc-Feature	*DocLen*
Cross-domain Sentiment Analysis	
Meta-Feature	*Corr[w_p]*
Intra-Feature	*DocTF(Raw/...)*
Doc-Feature	*DocLen, PivotTF[w_p]*

Figure 1: Feature design for three applications.

Training Phase:
Given: Training documents $\boldsymbol{d} = \{d_1, d_2, ..., d_N\}$, each d_i is labeled by y_i denoting the relevance of d_i with respect to task t_i, represented by its content $\mathcal{W}(d_i, t_i)$, and characterized by meta-features $f_k^{(M)}(w, t_i)$, intra-features $f_k^{(I)}(w, d_i)$ and doc-features $f_k^{(D)}(d_i)$
Output: A document scorer $\mathcal{F}(d, t)$ learned from training data.
Testing Phase:
Given: A new task t' ($t' \neq t_i$, for all i), document d' represented by $\mathcal{W}(d', t')$, features $f_k^{(M)}(w, t')$, $f_k^{(I)}(w, d')$ and $f_k^{(D)}(d')$
Output: The relevance of d' with respect to t', i.e., $\mathcal{F}(d', t')$

Figure 2: Cross-task document scoring.

Finally, we design doc-features $f_k^{(D)}(d)$, which characterize document features that have the same weightings across different tasks. For example, in cross-domain sentiment analysis, we design *PivotTF*$[w_p]$ for each pivot keyword w_p (*e.g.*, "good," "bad"), which represents the same sentiment in all domains. The design of doc-features is not the main focus of this paper, as they are the same with the features used in traditional learning models.

Given such a decoupled feature design, the cross-task document scoring problem is summarized in Figure 2.

4. TREE-STRUCTURED RESTRICTED BOLTZMANN MACHINE

Towards realizing learning to adapt keyword contribution, we require that our framework fulfill the requirements of *inferred sparsity* and *distant supervision*. As our solution, we propose *Tree-Structured Restricted Boltzmann Machine* (T-RBM) to learn a two-stage document scorer based on the decoupled features.

4.1 Requirement: Inferred Sparsity & Distant Supervision

As we motivated in Section 1, in many applications, the keyword vocabulary $\mathcal{V}(t)$ we model is very noisy. If such noisy keywords are not appropriately filtered, their inferred values of *Contrib*(w, d, t), once accumulated, will severely affect the prediction accuracy of $\mathcal{F}(d, t)$. Therefore, we require that our framework achieve *inferred sparsity*:

Requirement 1 (Inferred Sparsity for Keyword Contribution). *Contrib*(w, d, t) should equal to zero for unimportant keywords w.

This concept of inferred sparsity is closely related with feature sparsity, which traditional sparse learning works [20] aim to achieve. The goal of feature sparsity is to learn a sparse feature weighting vector, in which only a few features have non-zero weightings, for the purpose of reducing model complexity and increasing prediction accuracy. One common technique for achieving feature sparsity is l_1 regularization. For example, in single-task document scoring (which defines $\mathcal{F}(d, t)$ by Eq. 3), we can add an l_1 regularization term $|\alpha_w|_1$ to the objective function to learn sparse feature weightings α_w.

In contrast with feature sparsity which aims at learning sparse feature weightings, in inferred sparsity, we would like to "sparsify" the inferred value of a function, *i.e.*, *Contrib*(w, d, t). Traditional feature sparsity techniques simply do not work, because the value of *Contrib*(w, d, t) is jointly determined by a set of decoupled features – setting some of their weightings to be zero can not make the whole function become sparse.

Besides inferred sparsity, due to the lack of keyword labels, the learning framework should also achieve *distant supervision*:

Requirement 2 (Distant Supervision by Document Labels). The learning of keyword contribution *Contrib*(w, d, t) should be distantly guided by only document labels $\hat{\boldsymbol{y}}$.

As we introduced in Section 2, some existing works [3, 11, 22] on verbose-query-based retrieval and entity-centric document filtering also adopt the concept of meta-features. Although we can extend their works to support multiple intra-features, their models still have different limitations in fulfilling our requirements.

First, the concept weighting model proposed by Bendersky *et al.* [3], the Markov random field model proposed by Lease *et al.* [11] and the linear weighting model proposed by Zhou *et al.* [22] all belong to same the category, which define *Contrib*$(w, d, t) = [\sum_i \beta_i * f_i^{(M)}(w, t)] * \textit{DocTF}(w, d)$ as a linear function over meta-features. Since *DocTF*(w, d) is just one type of intra-feature, we can easily extend it to support multiple intra-features, by defining *Contrib*$(w, d, t) = \sum_{i,j} \beta_{ij} * f_i^{(M)}(w, t) * f_j^{(I)}(w, d)$, and the scorer becomes

$$\mathcal{F}(d, t) = \sum_{i,j} \beta_{ij} \sum_{w \in \mathcal{W}(d,t)} f_i^{(M)}(w, t) * f_j^{(I)}(w, d) \qquad (8)$$

where each $\sum_{w \in \mathcal{W}(d,t)} f_i^{(M)}(w, t) * f_j^{(I)}(w, d)$ could be treated as a generated feature, and β_{ij} could be learned by standard learners.

Such a linear design of *Contrib*(w, d, t) fails to fulfill the requirement of inferred sparsity. Actually, except the trivial solution which sets all β_{ij} to be 0, for other assignments of β_{ij}, we can only get a small number of keywords (no more than the number of β_{ij}) which have zero contribution. The drawback of failing to achieve inferred sparsity could be observed by checking the semantics of the generated features. Take meta-feature *QueryPos* and intra-feature *DocTF* as example. The generated feature denotes the *QueryPos* summation of keywords from a document, which fails to capture the intuition that

keywords with low *QueryPos* are more important, and documents containing more low-*QueryPos* keywords are more relevant.

Second, realizing the limitations of linear models, Zhou *et al.* [22] propose BoostMapping, which first adopts a boosting framework to generate a set of clusters $c_1, c_2, ...$, with each cluster c_i containing keywords sharing similar meta-features, *e.g.*, "*IDF* > 10 and *QueryPos* < 5," and then assumes that keywords from the same cluster c_i share the same contribution $\textit{Contrib}_i(w,d,t) = \gamma_i * \textit{DocTF}(w,d)$. Similar to the linear weighting model, we can extend it to support multiple intra-features by defining $\textit{Contrib}_i(w,d,t) = \sum_j \gamma_{ij} * f_j^{(I)}(w,d)$, and the scorer becomes

$$\mathcal{F}(d,t) = \sum_{i,j} \gamma_{ij} \sum_{w \in \mathcal{W}(d,t) \cap c_i} f_j^{(I)}(w,d). \quad (9)$$

BoostMapping manages to achieve inferred sparsity, as the learner will assign $\gamma_{ij} = 0$ for unimportant clusters. However, it is difficult to extend the learning algorithm of BoostMapping (specifically, for generating clusters c_i) to support multiple intra-features. Furthermore, as reported in the paper [22] and confirmed in our experiment, BoostMapping might overfit to training tasks in some specific settings, which would lead to poor generalization capability.

4.2 Proposal: Two-Stage Scoring Model

To fulfill these two requirements, we develop a *two-stage* scoring procedure. Specifically, to achieve inferred sparsity, we *discretize* the importance of a keyword into different levels, and explicitly define $\textit{Contrib}(w,d,t) = 0$ for unimportant keywords; to realize distant supervision, we unify the two stages in one single model, and learn the feature weightings by only document labels. The model is described in the following:

• In the first stage, we build a keyword classifier $\mathcal{C}(w,t) \in \{0,1,...,L\}$ based on meta-features $f_k^{(M)}(w,t)$ to discretize the importance of keywords, where $\mathcal{C}(w,t) = 0$ indicates that w is a noisy keyword for task t, and $\mathcal{C}(w,t) \in \{1,...,L\}$ represents keywords of different importance levels.

• In the second stage, we determine contribution of each keyword w based on its importance level $\mathcal{C}(w,t)$. For important keywords with $\mathcal{C}(w,t) = l \neq 0$, the value of $\textit{Contrib}(w,d,t)$ should depend on intra-features $f_k^{(I)}(w,d)$, while for unimportant ones (*i.e.*, $\mathcal{C}(w,t) = 0$), we set $\textit{Contrib}(w,d,t) = 0$ to fulfill the requirement of inferred sparsity.

Formally, such a two-stage model defines $\textit{Contrib}(w,d,t)$ as,

$$\textit{Contrib}(w,d,t) = \begin{cases} \mathcal{U}_l(w,d) & \text{if } \mathcal{C}(w,t) = l \neq 0; \\ 0 & \text{if } \mathcal{C}(w,t) = 0. \end{cases} \quad (10)$$

where $\mathcal{U}_l(w,d)$ is a function defined over intra-features $f_k^{(I)}(w,d)$.

4.3 Solution: Tree-Structured Restricted Boltzmann Machine

In order to realize the design of $\textit{Contrib}(w,d,t)$ in Eq. 10, the model should, first, characterize how keyword classifier $\mathcal{C}(w,t)$ depends on meta-features $f^{(M)}(w,t)$, second, determine how the contribution of important keywords, *i.e.*, $\mathcal{U}(w,d)$, is defined based on intra-features $f^{(I)}(w,d)$, and, third, be capable of learning $\mathcal{C}(w,t)$ in the absence of keyword labels. To achieve these three goals, we propose a novel *Tree-structured Restricted Boltzmann Machine* (T-RBM), in the framework of Markov Network, as our solution. Restricted Boltzmann Machine (RBM) [19] refers to one type of bipartite Markov network, which is widely used in many applications (*e.g.*, deep learning [6], topic modeling [18]). As a simplified

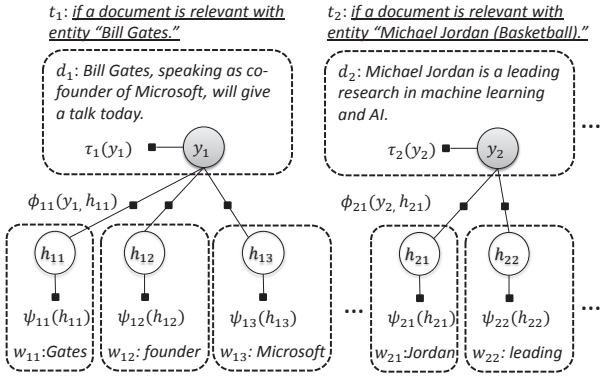

Figure 3: Tree-structured Restricted Boltzmann Machine.

RBM, our proposed T-RBM takes a tree structure to characterize the dependency between documents and keywords.

We highlight three important features of our proposed T-RBM model. First, to tackle the lack of keyword labels, T-RBM models the importance of keywords as *hidden variables*, and only uses document labels to guide the learning. Second, T-RBM adopts a novel *hidden variable regularization* idea, which allows the model to control the inferred sparsity of keyword contribution. Third, taking a tree structure, T-RBM could be *efficiently learned* by standard optimization techniques.

4.3.1 Model Overview

Figure 3 shows a T-RBM model designed for entity-centric document filtering. Generally, T-RBM is composed of N tree-structured Markov Networks, each of which corresponds to one document. There are no edges between trees, indicating that the documents are independent with each other. Each tree contains two types of nodes y_i and $\mathbf{h_i} = \{h_{i1}, ..., h_{i|\mathbf{h_i}|}\}$, with $y_i \in \{0,1\}$ denoting the relevance of document d_i ($y_i = 1$ if d_i is relevant, and $y_i = 0$ otherwise) and $h_{ij} \in \{0,1,...,L\}$ is a hidden variable denoting the importance of w_{ij} for task t_i (for notational convenience, we use $\mathbf{w_i} = \{w_{i1}, w_{i2}, ...\}$ to represent document content $\mathcal{W}(d_i,t_i)$, where w_{ij} denotes the j-th keyword in document d_i). More specifically, $h_{ij} = 0$ indicates that w_{ij} is unimportant, and $h_{ij} \in \{1,...,L\}$ corresponds to different importance levels of keywords, *e.g.*, in sentiment analysis, both "good" and "bad" are important keywords, but have different contribution to the document sentiment.

Three types of factors $\psi_{ij}(h_{ij})$, $\phi_{ij}(y_i, h_{ij})$ and $\tau_i(y_i)$ are defined in T-RBM. Specifically, $\psi_{ij}(h_{ij})$ characterizes whether keyword w_{ij} is important for task t_i, $\phi_{ij}(y_i, h_{ij})$ models how keyword w_{ij} contributes its importance to document d_i, and $\tau_i(y_i)$ models the effect of other document features $f_k^{(D)}(d_i)$. Based on the factors, the conditional probability $P(y_i, \mathbf{h_i}|d_i, \mathbf{w_i}, t_i)$ is defined to represent the probability of a specific assignment of y_i and $\mathbf{h_i}$, given in the following,

$$P(y_i, \mathbf{h_i}|d_i, \mathbf{w_i}, t_i) \propto \tau_i(y_i) \prod_{j=1}^{|\mathbf{w_i}|} \phi_{ij}(y_i, h_{ij}) \psi_{ij}(h_{ij}) \quad (11)$$

We are interested in document scorer $\mathcal{F}(d_i, t_i)$, which, in the language of probability, is formally described by $P(y_i = 1|d_i, \mathbf{w_i}, t_i)$. Based on Eq. 11, we can represent $P(y_i = 1|d_i, \mathbf{w_i}, t_i)$ in the form of graph factors, by marginalizing $P(y_i = 1, \mathbf{h_i}|d_i, \mathbf{w_i}, t_i)$ over all possible assignments of $\mathbf{h_i}$,

$$P(y_i = 1|d_i, \mathbf{w_i}, t_i) = \sum_{\mathbf{h_i}} P(y_i = 1, \mathbf{h_i}|d_i, \mathbf{w_i}, t_i) \quad (12)$$

To enforce inferred sparsity, we also want to control the percentage of noisy keywords in the model learning. In the language of probability, the inferred sparsity of keyword contribution could be represented by $P(\mathbf{h_i} = 0|d_i, \mathbf{w_i}, t_i)$, defined as follows,

$$P(\mathbf{h_i} = 0|d_i, \mathbf{w_i}, t_i) = \sum_{y=0}^{1} \prod_{j=1}^{|\mathbf{w_i}|} P(y_i = y, h_{ij} = 0|d_i, \mathbf{w_i}, t_i) \tag{13}$$

4.3.2 Factor Design

In T-RBM, we design factors $\psi_{ij}(h_{ij})$, $\phi_{ij}(y_i, h_{ij})$ and $\tau_i(y_i)$ to characterize the dependency required by Eq. 10.

First, $\psi_{ij}(h_{ij})$ characterizes how keyword classifier $\mathcal{C}(w_{ij}, t_i)$ judges if w_{ij} is an important keyword for task t_i, which should depend on meta-features $f_k^{(M)}(w_{ij}, t_i)$:

$$\psi_{ij}(h_{ij}) = \begin{cases} e^{\sum_k \theta_{kl}^{(M)} f_k^{(M)}(w_{ij}, t_i)} & \text{if } h_{ij} = l > 0; \\ 1 & \text{if } h_{ij} = 0. \end{cases} \tag{14}$$

Eq. 14 characterizes different levels of important keywords by defining L sets of feature weightings $\theta_{k,l}^{(M)}$. Note that we set $\tau_i(0) = 1$ since only relative value between $\tau_i(l)$ and $\tau_i(0)$ matters in the Markov Network.

Second, $\phi_{ij}(y_i, h_{ij})$ models how noisy keywords are filtered, and how the contribution of important keywords $\mathcal{U}_l(w, d)$ depends on intra-features $f_k^{(I)}(w_{ij}, d_i)$,

$$\begin{aligned} &\phi_{ij}(y_i, h_{ij}) \\ = &\begin{cases} \exp[\sum_k \theta_{kl0}^{(I)} f_k^{(I)}(w_{ij}, d_i)] & \text{if } h_{ij} = l \ \&\ y_i = 0; \\ \exp[\sum_k \theta_{kl1}^{(I)} f_k^{(I)}(w_{ij}, d_i)] & \text{if } h_{ij} = l \ \&\ y_i = 1; \\ 1 & \text{if } h_{ij} = 0. \end{cases} \end{aligned} \tag{15}$$

We set $\phi_{ij}(0, 0) = \phi_{ij}(1, 0) = 1$ to represent that if w_{ij} is an unimportant keyword, w_{ij} would not contribute any score to document d_i. When w_{ij} is important, *i.e.*, $h_{ij} = l > 0$, the value of $\phi_{ij}(y_i, h_{ij})$ depends on how w_{ij} appears in document d_i, *i.e.*, intra-features $f_k^{(I)}(w_{ij}, d_i)$. It should be noted that, even if we assume only one keyword importance level with $L = 1$, such a factor design can still discriminate keywords of different importance, because the contribution of keyword w_{ij} is the marginalization result of $P(y_i, h_{ij} = 1|d_i, \mathbf{w_i}, t_i)$ and $P(y_i, h_{ij} = 0|d_i, \mathbf{w_i}, t_i)$.

Finally, we define $\tau_i(d_i)$ to incorporate document features $f_k^{(D)}(d_i)$ that are generalizable across tasks:

$$\tau_i(y_i) = \begin{cases} \exp[\sum_k \theta_k^{(D)} f_k^{(D)}(d_i)] & \text{if } y_i = 1; \\ 1 & \text{if } y_i = 0. \end{cases} \tag{16}$$

We use $\boldsymbol{\theta} = \{\theta^{(M)}_{1..K^{(M)},1..L}, \theta^{(I)}_{1..K^{(I)},1..L,0..1}, \theta^{(D)}_{1..K^{(D)}}\}$ to denote the set of parameters that need to be determined in T-RBM, where $K^{(M)}$, $K^{(I)}$ and $K^{(D)}$ denote the number of designed meta-features, intra-features and doc-features respectively, and the total number of parameters is $K^{(M)} * L + K^{(I)} * L * 2 + K^{(D)}$.

4.3.3 Model Learning

Following the maximal likelihood principle, our objective is to learn $\boldsymbol{\theta}$ to maximize the likelihood $\mathcal{L}(\boldsymbol{\theta}; \hat{\mathbf{y}})$ of observing document labels $\hat{\mathbf{y}}$, regularized by $\mathcal{G}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log P(\mathbf{h_i} = \mathbf{0}|d_i, \mathbf{w_i}, t_i)$ to control the inferred sparsity of all the keywords. Formally the likelihood is defined by

$$\begin{aligned} &\operatorname*{argmax}_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}; \hat{\mathbf{y}}) + \lambda \mathcal{G}(\boldsymbol{\theta}) \\ = &\operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{i=1}^{N} \log P(y_i = \hat{y}_i|d_i, \mathbf{w_i}, t_i) + \lambda \log P(\mathbf{h_i} = \mathbf{0}|d_i, \mathbf{w_i}, t_i) \\ = &\operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{i=1}^{N} \log \sum_{\mathbf{h_i}} \tau_i(\hat{y}_i) \prod_{j=1}^{|\mathbf{w_i}|} \phi_{ij}(\hat{y}_i, h_{ij}) \psi_{ij}(h_{ij}) \\ &+ \lambda \log \sum_{y=0}^{1} \tau_i(y) \prod_{j=1}^{|\mathbf{w_i}|} \phi_{ij}(y, 0) \psi_{ij}(0) + (1 + \lambda) \log \sum_{y=0}^{1} \sum_{\mathbf{h_i}} \tau_i(y) \\ &\cdot \prod_{j=1}^{|\mathbf{w_i}|} \phi_{ij}(y, h_{ij}) \psi_{ij}(h_{ij}) \end{aligned} \tag{17}$$

where λ controls the inferred sparsity of keyword contribution. Specifically, when $\lambda > 0$, the model will favor more sparse keyword contribution, and vice versa. Here we first study document scoring as a classification problem by assuming that document labels are all binary (*i.e.*, $\hat{y}_i \in \{0, 1\}$), and will later extend our solution to document ranking problems which accept ordinal labels.

To optimize the objective function, we use the Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm, which is a popular gradient-based method for solving unconstrained nonlinear problems – specifically, we adopt the LibLBFGS library [15], a c-implementation of L-BFGS. As the optimization routine, in each step, we compute the current gradient value of the objective function at $\boldsymbol{\theta}$, and LibLBFGS will update $\boldsymbol{\theta}$ according to the gradient value. The optimization routine stops until the objective function converges.

The gradient value of the objective function at $\boldsymbol{\theta}$, as required by LibLBFGS, is computed as follows,

$$\begin{aligned} \frac{\partial \mathcal{L}(\boldsymbol{\theta}; \hat{\mathbf{y}})}{\partial \theta_{kl}^{(M)}} = &\sum_{i=1}^{N} \sum_{j=1}^{|\mathbf{w_i}|} [P(h_{ij} = l|y_i = \hat{y}_i, d_i, \mathbf{w_i}, t_i) \\ &-(1 + \lambda) P(h_{ij} = l|d_i, \mathbf{w_i}, t_i)] f_k^{(M)}(w_{ij}, t_i) \end{aligned} \tag{18}$$

$$\begin{aligned} \frac{\partial \mathcal{L}(\boldsymbol{\theta}; \hat{\mathbf{y}})}{\partial \theta_{kly}^{(I)}} = &\sum_{i=1}^{N} \sum_{j=1}^{|\mathbf{w_i}|} [P(h_{ij} = l, y_i = y|y_i = \hat{y}_i, d_i, \mathbf{w_i}, t_i) \\ &-(1 + \lambda) P(h_{ij} = l, y_i = y|d_i, \mathbf{w_i}, t_i)] f_k^{(I)}(w_{ij}, d_i) \end{aligned} \tag{19}$$

$$\begin{aligned} \frac{\partial \mathcal{L}(\boldsymbol{\theta}; \hat{\mathbf{y}})}{\partial \theta_{k}^{(D)}} = &\sum_{i=1}^{N} [P(y_i = 1|y_i = \hat{y}_i, d_i, \mathbf{w_i}, t_i) \\ &-(1 + \lambda) P(y_i = 1|d_i, \mathbf{w_i}, t_i)] f_k^{(D)}(d_i) \end{aligned} \tag{20}$$

In Eq. 18, Eq. 19 and Eq. 20, all the probabilities could be computed by belief propagation [17]. As T-RBM is a tree-structured graph, belief propagation could efficiently compute the exact result.

In general, the time complexity of T-RBM is $O[T * (N_d * K^{(D)} + N_e * K^{(I)} + N_w * K^{(M)})]$, where T denotes the number of total iterations, N_d, N_e and N_w represent the number of documents, keyword-document pairs and keywords respectively. Such time complexity stems from the computation of factors $\psi_{ij}(h_{ij})$, $\phi_{ij}(y_i, h_{ij})$ and $\tau_i(y_i)$ given current model parameters $\boldsymbol{\theta}$. After computing the factors, the time complexity of belief propagation is $O(N_e)$, and updating parameter $\boldsymbol{\theta}$ takes $O(|\boldsymbol{\theta}|)$, which are much faster than the factor computation and could be ignored.

4.3.4 Extension to Ranking Problem

This section discusses how to extend T-RBM to ranking problems, where training labels are ordinal instead of binary, *e.g.*, $\hat{y}_i \in \{0, 1, 2\}$, denoting {"irrelevant", "relevant", "perfectly relevant"}.

As the solution, we apply the cumulative logits approach [12] to convert the ranking problem back to a binary classification problem. Assume that each $\hat{y}_i \in \{0, 1, ..., V\}$, we will construct V different binary document classifier separately. For the v-th document classifier, we partition the data into two groups: $\{y_i < v\}$ and $\{y_i \geq v\}$, and learn a document classifier $P(y_i \geq v|d_i, \mathbf{w_i}, t_i)$ based on the learning algorithm in Section 4.3.3. Given the results of the classifiers, we can compute the expected relevance as the ranking score of each document, given as follows,

$$\mathcal{F}(d_i, t_i) = \sum_{v=1}^{V} v * P(y_i = v|d_i, \mathbf{w_i}, t_i) \quad (21)$$

$$= \sum_{v=1}^{V} v * [P(y_i \geq v|d_i, \mathbf{w_i}, t_i) - P(y_i \geq v - 1|d_i, \mathbf{w_i}, t_i)] \quad (22)$$

5. EXPERIMENT

In this section, we compare the overall performance of T-RBM with four state-of-the-art baselines, to demonstrate the effectiveness of applying T-RBM for cross-task document scoring problems.

5.1 Experiment Setting

To demonstrate the capacity of T-RBM on general cross-task document scoring problems, we study three different applications – verbose-query-based retrieval, entity-centric document filtering and cross-domain sentiment analysis – on different datasets with specification shown in Figure 4.

1. *Robust (Verbose-query-based Retrieval)*. In order to construct a large dataset, we combined two newswire collections released by TREC 2004 and 2005 Robust tracks. Unlike most datasets that contain only short keyword queries, these two collections are the largest publicly available datasets that have detailed query descriptions, which could be used as verbose queries. Following previous verbose query works [3], we only used the $\langle desc \rangle$ portions of TREC queries, and ignored the $\langle title \rangle$ portions. Given one query, each document is labeled as 2–perfectly relevant, 1–relevant and 0–irrelevant.

2. *TREC-KBA (Entity-centric Document Filtering)*. This dataset includes 29 Wikipedia entities covering living persons from different domains and a few organizations. For each entity, 100∼3000 candidate documents are collected and labeled as garbage, neutral, relevant or central. Following the same procedure in the previous work [22], we got binary labels by viewing central and relevant documents as positive, and others negative.

3. *Review (Cross-domain Sentiment Analysis)*. This dataset was constructed by Blitzer *et al.* [4] through selecting Amazon product reviews from four different domains: books, DVDs, electronics and kitchen appliances. Each domain contains 1000 positive, 1000 negative and 3000∼5000 unlabeled reviews. We selected 60 pivot keywords by mutual information as suggested by Blitzer *et al.* [4], and learned pivot predictors (for SCL) and Pearson Correlation (for T-RBM) based on unlabeled reviews.

Among the three applications, verbose-query-based retrieval was studied as a ranking problem in previous works, while entity-centric document filtering and cross-domain sentiment analysis were treated as classification problems. Following such conventions, we used ranking-oriented metrics like NDCG@k and MAP (Mean Average Precision) for verbose-query-based retrieval, and classification-oriented metrics including precision, recall, F1-measure and accuracy for the other two tasks.

Dataset	Num of Tasks	Num of Docs	Num of Positive Docs	Classification / Ranking
Robust	249 Queries	349,093	Perfectly Relevant: 3,821 Relevant: 20,147	Ranking
TREC-KBA	29 Entities	52,238	24,704	Classification
Review	4 Domains	8,000	4000	Classification

Figure 4: Dataset specification.

To confirm the confidence of the comparison, for both verbose-query-based retrieval and entity-centric document filtering, we divided queries into 5 sets, and adopted 5-fold cross validation with standard t-test. For cross-domain sentiment analysis, we applied the evaluation method in [4], which trained one model for each domain, adapted the model to the other three domains and reported the average ranking performance over 12 sets of results.

To deal with unbalanced data, we reweighed the training instances to balance positive and negative data. All features were first standardized to reduce the impact of different feature scales. In those baselines which require classifier or ranker learning, we used SVM-Light [9] and RankSVM [10] to learn the feature weightings with default parameters. In T-RBM, we empirically set the number of keyword importance level L to be 2 and regularization factor λ to be 0.005, and will later investigate their impact.

5.2 Quantitative Evaluation

5.2.1 Overall Performance

We designed different baselines to incrementally validate the performance of our proposed T-RBM model. First, we experimented the StandardLearner baseline to demonstrate the necessity of realizing learning to score keywords. Second, we compared T-RBM with LinearMapping [22, 3, 11] and BoostMapping [22] to verify the effectiveness of T-RBM on learning to adapt keyword contribution. In cross-domain sentiment analysis, we also compared T-RBM with SCL [4] to demonstrate that T-RBM well realizes the domain adaptation insight proposed by Blizter *et al.* [4].

1. *Standard Learning Model* (StandardLearner). This baseline solves the problems by standard learning to rank/classify techniques, which defines the scorer by Eq. 4. In the feature design, for verbose-query-based retrieval, we followed the LETOR benchmark [14] to extract ranking features such as TFIDF, BM25 and Language model for each query-document pair; for entity-centric document filtering, as the input query is an entity document, we designed features to calculate the document similarity using different metrics introduced in [7], *e.g.*, Euclidean distance, cosine similarity; in cross-domain sentiment analysis, we used all keywords as features and disregarded domain differences.

2. *Linear Weighting Model* (LinearWeight). As discussed in Section 4.1, the models proposed by Bendersky *et al.* [3], Lease *et al.* [11], and Zhou *et al.* [22] all belong to the same category. This baseline extends such models to support multiple intra-features, which defines the scorer by Eq. 8.

3. *Boosting Mapping Model* (BoostMapping). This baseline adopts the boosting framework proposed by Zhou *et al.* [22], which defines the document scorer by Eq. 9. Since learning keyword clusters based on multiple intra-features is difficult, we first learned the keyword clusters c_i based on only $DocTF(w, d)$, and then applied the SVM or RankSVM again to re-learn the feature weightings γ_{ij}.

4. *Structural Correspondence Learning* (SCL) [4]. This baseline implements the SCL algorithm (introduced in Section 2) proposed by Blitzer *et al.* We used SVMLight to train pivot predictors and the final classifiers, and adopted the same parameter setting used in [4].

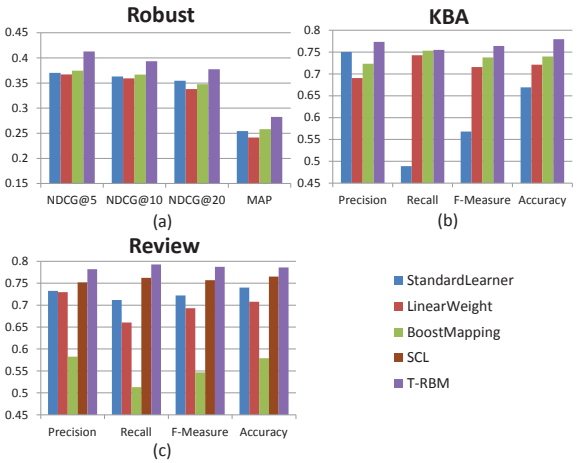

Figure 5: Performance comparison with baselines.

Figure 5 demonstrates T-RBM consistently outperforms other baselines in three applications. Specifically, we observe that T-RBM achieves encouraging improvement against runner-up BoostMapping in verbose-query-based retrieval (NDCG@5 $+10.2\%$, NDCG@10 $+7.3\%$, NDCG@20 $+8.5\%$ and MAP $+9.4\%$) and entity-centric document filtering (F-Measure $+3.5\%$ and accuracy $+5.4\%$) with p-value< 0.05, while in cross-domain sentiment analysis, T-RBM outperforms runner-up SCL (F-Measure $+4.4\%$ and accuracy $+2.7\%$). We analyze the performance differences between T-RBM and other baselines in the following.

First, for the StandardLearner baseline, its performance largely depends on the quality of the adopted features. The results in Figure 5 show that, intuitive feature designs fail to achieve satisfactory results for general cross-document scoring problems. Specifically, in verbose-query-based retrieval, using traditional document scorers as features does not perform well because of the existence of noisy keywords. In entity-centric document filtering, since there exist many relevant documents that are not similar to the identification page (*e.g.*, they might discuss only one or two aspects of the entity), StandardLearner based on document similarity fails as well. In cross-domain sentiment analysis, the result confirms the necessity of adapting keyword importance for different domains.

Second, LinearWeight does not perform well. As we discussed in Section 4.1, the linear definition of keyword contribution function makes it vulnerable to noisy keywords. From the result, we can observe that LinearWeight achieves relatively better performance in verbose-query-based retrieval, which is a task that involves less noisy keywords compared with the other two. The results confirm the importance of fulfilling the inferred sparsity requirement.

Third, BoostMapping fulfills the inferred sparsity requirement, by eliminating the contribution of keywords from noisy clusters; however, it might easily overfit the dataset in some specific settings. In the experiment, we can observe that BoostMapping outperforms LinearWeight over Robust and KBA datasets, but achieves very poor performance on the Review dataset. That is because, unlike the first two applications where the training dataset contains many different tasks (*i.e.*, queries or entities), in cross-domain sentiment analysis, the training data only contains one domain at a time. As the result, BoostMapping severely over-fits the training domain and fails to be generalized to the other domains.

Fourth, SCL outperforms other baselines, and the result is consistent with the performance reported in [4]. However, SCL relies on the idea of pivot predictors to realize domain adaptation, which could not be generalized to other cross-task document scoring problems that do not have the concept of pivot keywords. Moreover, in the experiment, we discover that the performance of SCL is sensitive to the model used for training the pivot predictors – different pivot predictors (*e.g.*, trained by SVM and logistic regression) tend to result in very different prediction performance.

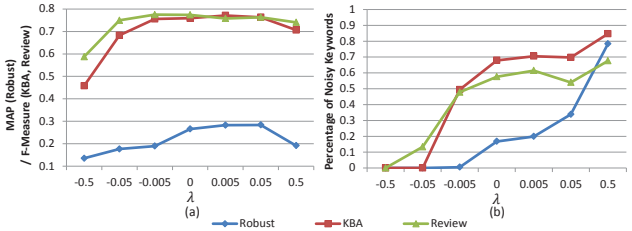

Figure 6: Effect of different parameters.

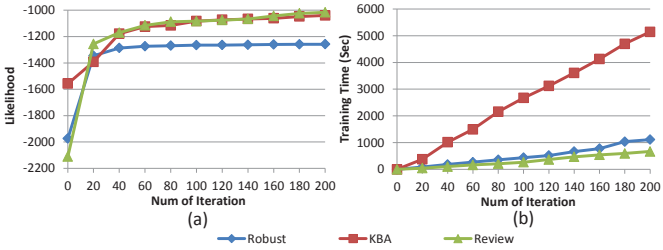

Figure 7: Learning efficiency.

Finally, T-RBM outperforms all the four baselines on three different applications. Essentially, the insight of T-RBM is very similar to BoostMapping, which classifies keywords into groups based on their meta-features, and learns the importance of different groups. However, different from BoostMapping which adopts a greedy algorithm to generate clusters, T-RBM takes advantage of Markov Network to jointly learn the weightings for meta-features and intra-features, which greatly reduces the risk of over-fitting. The results demonstrate that T-RBM achieves satisfactory performance on both learning to rank and domain adaptation.

5.2.2 *Different Parameters*

In T-RBM, we have to manually determine two parameters: the number of keyword importance levels L and regularization parameter λ. We experimented different values of L ranging from 1 to 5, and discovered that the performance of T-RBM is not sensitive to L. That is because, as we discussed in Section 4.3.2, even if we set $L = 1$, T-RBM can still model different levels of keyword importance by the marginalization of hidden variables.

In this section, we only investigate the impact of λ. Figure 6 shows how different values of regularization parameter λ affect the performances of T-RBM (in Figure 6 (a)) and the number of keywords that are classified as noise (in Figure 6 (b)). In Figure 6 (a), we can discover that, when λ is negative, the performance of T-RBM decreases significantly for all three datasets. That is because, as Figure 6 (b) illustrates, a negative value of λ will tempt the model to judge more keywords as important, making the scorer vulnerable to noisy keywords. The result again confirms the importance of fulfilling the requirement of inferred sparsity. T-RBM usually achieves the best performance when λ is around 0.005 and 0.05. Continually increasing the value of λ will over-regularize the model and lower down the performance.

5.2.3 *Training Efficiency*

In this section, we are going to investigate the training efficiency of T-RBM. Figure 7(a) shows that the likelihood value converges very quickly within around 100 iterations in all three applications, which, as Figure 7(b) displays, takes 358 seconds for verbose-query-

Data set	Task	Unimportant (L=0)	Important (L=1)
Robust 2005	Query: Marine Vegetation	as, of, purpose	Drug purpose, marine vegetation, harvest
	Query: Abuse of E-Mail	by, in, through	prevent excess, many people, mail
KBA	Entity: Jim Steyer	2011, http, media	privacy, manager, expert, professor
	Entity: Douglas Carswell	http, UK, time	donate, foundation, enthusiast, localist
Review	Domain: DVD	was, you, just	amazing, great performance, great dvd
	Domain: electronics	what, get	inexpensive, works great, fit into, faster

(a) Examples of important/unimportant keywords.

Data set	Meta-Feature ($\theta_{k1}^{(M)}$)	Intra-Feature ($\theta_{k11}^{(I)} - \theta_{k10}^{(I)}$)
Robust 2005	IDF[Uni](1.118), IDF[Bi](0.535), QueryTF [Bi](0.301), IsVerb[Uni](0.209)	DocTF_Log[Uni](0.744), HeadTF_Log [Uni](0.178), WindowTF_Raw[Bi](0.103)
KBA	QueryPos (-0.829), IsNum(-0.353), IDF (0.169), TextTF (0.082), TFInInfoBox(0.075)	DocTF_Noramlized(0.082), TitleTF_Log (0.034) AnchorTF_Raw(-0.072)
Review	Cor[a_wonerful](14.662), Cor [easy_to](9.097), Cor[poorly] (-7.746), Cor[enjoable](7.694)	DocTF_Raw(0.0362), DocTF_Log(0.023) DocTF_Normalized(0.005)

(b) Features with highest absolute weightings.

Figure 8: Case study.

based retrieval, 2672 seconds for entity-centric document filtering, and 211 seconds for cross-sentiment analysis. In general, the training speed is very fast to get a stable solution.

5.3 Case Study

To give an intuitive illustration of how T-RBM identifies important/unimportant keywords for cross-task document scoring applications, in Figure 8 (a), we perform case studies by showing 2 example tasks per dataset, and listing some of their unimportant ($L = 0$) and important keywords ($L = 1$) based the value of $P(h_{ij})$ derived by T-RBM. To analyze the effect of our designed features, we list the meta-features and intra-features which have the maximal absolute weightings in Figure 8 (b).

As Figure 8 displays, in terms of meta-features, in verbose-query-based retrieval, it meets our expectation that *IDF*, for both unigrams and bigrams, has the highest weightings; the result also includes *QueryTF[Bigram]*, implying that bigrams (*e.g.*, "drug purpose," "prevent excess") usually have higher importance than unigrams. In entity-centric document filtering, *QueryPos* is the most important meta-feature, demonstrating the position of a keyword is a very useful signal for identifying important keywords from a long Wikipedia page; *TFInfoBox* also has high weighting, which verifies our intuition of using Wikipedia page structure to identify important keywords. In cross-domain sentiment analysis, the result shows that the correlation with keywords such as "a_wonderful" and "easy_to" is helpful for identifying the sentiment of a keyword, demonstrating that our framework successfully realizes the insight proposed by Blizter *et al.* [4]. In terms of intra-features, we can observe *DocTF* always has the highest weighting, whereas, different applications favor different term frequency representations. For example, in entity-centric document filtering, *DocTF_Normalized* has higher weighting because the TREC-KBA dataset contains a large number of short documents (*e.g.*, tweets), and without normalization, the scorer would severely bias towards long documents.

6. CONCLUSION

In this paper, we proposed to study the cross-task document scoring problem – given some labeled documents for some user queries or data domains, how to learn a scorer which could predict the relevance of a new document for a new query or domain. As the key insight, towards facilitating the feature design, we proposed the idea of feature decoupling. To design a document scorer based on the decoupled features, we proposed a novel T-RBM model as our solution. Experiments on three applications confirmed the effectiveness of T-RBM.

7. REFERENCES

[1] M. Bendersky and W. B. Croft. Discovering key concepts in verbose queries. In *Proc. of SIGIR '11*, page 491. ACM, 2008.

[2] M. Bendersky, D. Metzler, and W. B. Croft. Learning concept importance using a weighted dependence model. In *Proc. of WSDM '10*, pages 31–40. ACM, 2010.

[3] M. Bendersky, D. Metzler, and W. B. Croft. Parameterized concept weighting in verbose queries. In *Proc. SIGIR '11*, pages 605–614. ACM, 2011.

[4] J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proc. of ACL '07*, number 1, page 440, 2007.

[5] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon. Adapting ranking svm to document retrieval. In *Proc. SIGIR '06*, pages 186–193. ACM, 2006.

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

[7] A. Huang. Similarity measures for text document clustering. In *Proc. of NZCSRSC '08*, pages 49–56, 2008.

[8] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *LNCS*, volume 1398, pages 137–142, 1998.

[9] T. Joachims. Making large scale svm learning practical. 1999.

[10] T. Joachims. Optimizing search engines using clickthrough data. In *Proc. SIGKDD '02*, pages 133–142. ACM, 2002.

[11] M. Lease. An improved markov random field model for supporting verbose queries. In *Proc. of SIGIR '09*, pages 476–483. ACM, 2009.

[12] P. Li, Q. Wu, and C. J. Burges. Mcrank: Learning to rank using multiple classification and gradient boosting. In *Proc. of NIPS '07*, pages 897–904, 2007.

[13] T. Li, V. Sindhwani, C. Ding, and Y. Zhang. Knowledge transformation for cross-domain sentiment classification. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 716–717. ACM, 2009.

[14] T.-Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *Proc. of SIGIR '07 workshop*, pages 3–10, 2007.

[15] N. Okazaki. Liblbfgs: a library of limited-memory broyden-fletcher-goldfarb-shanno (l-bfgs). *URL http://www. chokkan. org/software/liblbfgs/index. html*, 2011.

[16] S. J. Pan, X. Ni, J.-T. Sun, Q. Yang, and Z. Chen. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th international conference on World wide web*, pages 751–760. ACM, 2010.

[17] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[18] R. Salakhutdinov and G. E. Hinton. Replicated softmax: an undirected topic model. In *NIPS '09*, pages 1607–1614, 2009.

[19] P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. 1986.

[20] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[21] V. Vapnik. Statistical learning theory, 1998.

[22] M. Zhou and K. C.-C. Chang. Entity-centric document filtering: boosting feature mapping through meta-features. In *Proc. of CIKM '13*, pages 119–128. ACM, 2013.