Representing Documents via Latent Keyphrase Inference

Jialu Liu† Xiang Ren† Jingbo Shang†
Taylor Cassidy‡ Clare R. Voss‡ Jiawei Han†

†Department of Computer Science, University of Illinois at Urbana-Champaign
‡Computational & Information Sciences Directorate, Army Research Laboratory

†{jliu64, xren7, shang7, hanj}@illinois.edu ‡{taylor.cassidy, clare.r.voss}.civ@mail.mil

ABSTRACT

Many text mining approaches adopt bag-of-words or n-grams models to represent documents. Looking beyond just the words, *i.e.*, the explicit surface forms, in a document can improve a computer's understanding of text. Being aware of this, researchers have proposed concept-based models that rely on a human-curated knowledge base to incorporate other related concepts in the document representation. But these methods are not desirable when applied to vertical domains (*e.g.*, literature, enterprise, *etc.*) due to low coverage of in-domain concepts in the general knowledge base and interference from out-of-domain concepts. In this paper, we propose a data-driven model named *Latent Keyphrase Inference* (*LAKI*) that represents documents with a vector of closely related domain keyphrases instead of single words or existing concepts in the knowledge base. We show that given a corpus of in-domain documents, topical content units can be learned for each domain keyphrase, which enables a computer to do smart inference to discover latent document keyphrases, going beyond just explicit mentions. Compared with the state-of-art document representation approaches, LAKI fills the gap between bag-of-words and concept-based models by using domain keyphrases as the basic representation unit. It removes dependency on a knowledge base while providing, with keyphrases, readily interpretable representations. When evaluated against 8 other methods on two text mining tasks over two corpora, LAKI outperformed all.

1. INTRODUCTION

Text data (*e.g.*, web queries, business reviews, product manuals) are ubiquitous and tasks like document grouping and categorization play an essential role in big data applications. At the heart of analysing text data is a unified representation of text input.

Related Work: The most common representation for texts is the bag-of-words [1] due to its simplicity and efficiency. This method however typically fails to capture word-level synonymy (missing shared concepts in distinct words, such as "doctor" and "physician") and polysemy (missing distinct concepts in same word, such as "Washington" can be either the city or the government). As a remedy, topic models [7, 3] try to overcome this limitation by positing a set of latent topics which are distributions over words, and assuming that each document can be described as a mixture of these topics. Nevertheless, the interpretability of latent space for topic models is not straightforward and perusing semantic meaning in inferred topics is difficult [4, 18]. Concept-based models [10, 23, 21, 12, 14] were proposed to overcome these barriers. The intuition is to map a text query into a high-dimensional vectorial representation where each dimension corresponds to a concept in a general Knowledge Base (KB), like Wikipedia or Freebase, making them easily interpretable. For example, the text sequence "*DBSCAN for knowledge discovery*" can be mapped to KB concepts like "*KB: data mining*", "*KB: density-based clustering*" and "*KB: dbscan*". Such methods take advantage of a vast amount of highly organized human knowledge. However, most of the existing knowledge bases are manually maintained, and are limited in coverage and freshness. Researchers have therefore recently developed systems such as Probase [25] and DBpedia [2] to replace or enrich traditional KBs. Neverthless, the rapid emergence of large, domain-specific text corpora (e.g., business reviews) poses significant challenges to traditional concept-based techniques and calls for methods of representing texts by interpretable units without requirement of a general KB.

Categories	Representation
Words	dbscan, method, clustering, process, ...
Topics	[k-means, clustering, clusters, dbscan, ...]
	[clusters, density, dbscan, clustering, ...]
	[machine, learning, knowledge, mining, ...]
KB Concepts	data mining, clustering analysis, dbscan, ...
Keyphrases	dbscan: [dbscan, density, clustering, ...]
	clustering: [clustering, clusters, partition, ...]
	data mining: [data mining, knowledge, ...]

Table 1: Representations for query "DBSCAN is a method for clustering in process of knowledge discovery." returned by various categories of methods.

In this paper, we study the problem of learning representations for domain-specific texts: Given massive training texts in a specific domain or genre, we aim to learn a systematic way to derive interpretable representations for any new in-domain documents without relying on a KB. When existing approaches are directly applied to this problem setting, they encounter several limitations:

• **Representation interpretability**: A lot of data-driven methods (*e.g.*, bag-of-words and topic models) lack straight-

WWW 2016, April 11–15, 2016, Montréal, Québec, Canada.
ACM 978-1-4503-4143-1/16/04.
Include the http://dx.doi.org/10.1145/2872427.2883088 .

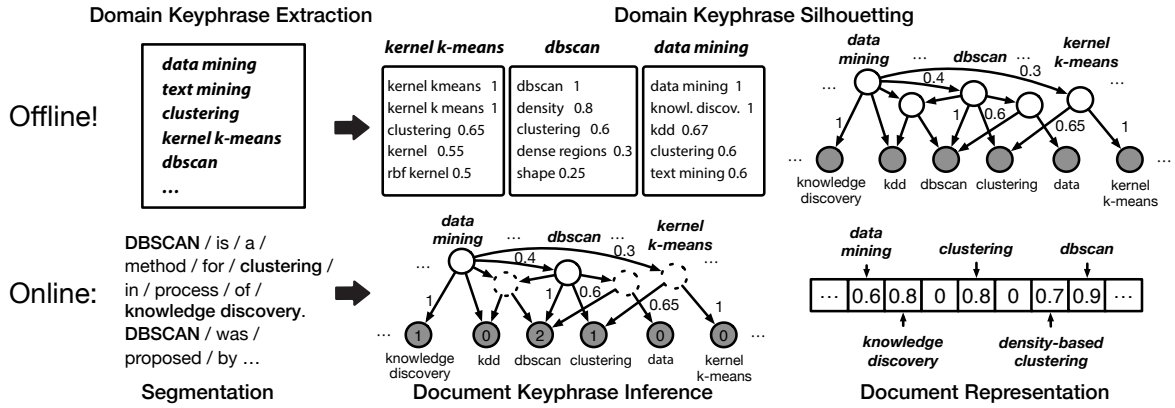

Figure 1: Overview of LAKI. White and grey nodes represent domain keyphrases and content units respectively.

forward interpretation for document representation, which is critical for model verification and for ensuring that the model is capturing user's intuitions about the text input [4].

• **Domain restriction**: For concept-based methods relying on a KB [10, 21, 23], the provided knowledge in KB usually suffers from limited coverage and freshness on specific, dynamic or emerging domains.

• **Cross-domain interference**: Due to the wide range of domains covered in a general KB, many words will have multiple possible KB referents even if they are unambiguous in the target domain, thus introducing noise and distortion in the text representation even when the vocabulary overlap between the target domain and knowledge base is small [12].

To address the above challenges, we exploit several intuitive ideas as follows. We first extract *domain keyphrases* from an in-domain corpus, which are both meaningful and interpretable, and instantiate them as dimensions in our text representation. Thus, a document is represented as a vector of keyphrases, in contrast to traditional methods using words, topics or concepts (see Table 1). Note that within a particular domain, a given keyphrase will likely have one meaning [11], making the representation relatively unambiguous. On one hand, limiting phrase extraction to the in-domain corpus helps eliminate cross-domain interference. However, many keyphrases relevant to a given document are not *explicit document keyphrases*, *i.e.*, they are not mentioned in the document. This presents a unique challenge, especially for short texts like paper abstracts and business reviews. We therefore propose to associate each domain keyphrase with a *silhouette*—a topically-cohesive set of words and phrases, which enables a computer to incorporate *latent document keyphrases* into text representation.

Our solution, called *Latent Keyphrase Inference* (LAKI), is developed to systematically integrate the above ideas. As shown in Fig. 1, it consists of an offline domain keyphrase learning phase and an online document keyphrase inference phase. In the offline phase, corpus-wide domain keyphrases are extracted and their silhouettes are learned through a hierarchical Bayesian network which optimizes the likelihood with respect to latent document keyphrases, given observed content units in the training corpus. The Bayesian network is essentially a directed acyclic graph (DAG) for modeling the dependency 1) between domain keyphrases and content units, and 2) between domain keyphrases themselves. In the online process, LAKI identifies the top-ranked latent document keyphrases for the input text by statistical inference on the Bayesian network. Consequently, LAKI is able to transform text string into a high-dimensional vector representation. Entries in the output vector quantify the relatedness between the document and the respective keyphrases. The major contributions of this paper are:

1. We propose to use latent keyphrases as document representation for domain-specific texts, which enhances representation interpretability, solves domain restriction and avoids cross-domain interference.
2. We develop a Bayesian network-based approach to model domain keyphrase silhouettes, which later helps infer latent document keyphrases and solves the rarity of explicit keyphrase mentions in the document.
3. Experiments on corpora of different domains show both the effectiveness and efficiency of the proposed solution.

2. BACKGROUND

This paper deals with the problem of *learning representation of domain-specific texts*. Given a document corpus as input, we aim to produce a set of keyphrases as the basis for vectorial representation of any text queries posed to this corpus. The task is formally defined as follows.

Definition 1 (Problem Definition). Given a document corpus D with specific focus on certain genres of content, we aim to automatically extract semantically keyphrases $K = \{K_1, \cdots, K_M\}$ from D, and derive a model that can generate high-dimensional vectorial representation $[P_1^{(q)}, \cdots, P_M^{(q)}]$ for any text query q from the same domain, where each entry of the vector quantifies relatedness between query and the corresponding keyphrase. ■

Ideally, these we would use domain-specific knowledge base concepts instead of keyphrases. But identifying and constructing these concepts purely from text data with satisfactory performance remains an unsolved problem. Instead, we instantiate them as *domain keyphrases* since the problem of extracting high-quality keyphrases is more tractable, and there exist many publicly available methods. A **domain keyphrase** is a phrase (which may be one or more words) of great significance in the domain extracted from a domain-specific corpus. They are meant to be natural, meaningful and likely unambiguous semantic units for representing domain-specific texts. However, one needs to overcome their low rate of explicit mentions in documents to which they are relevant. Our solution is to identify *latent document keyphrases* which are relevant to a given document, but are

not explicitly mentioned therein. A **document keyphrase** is a domain keyphrase that is relevant to a specific document, *i.e.*it serves as an informative word or phrase to indicate the content of that specific document. Actual mention of a document keyphrase is not necessarily required.

Following the above discussion, a good model should generate representations in terms of document keyphrases, together with numerical values indicating strength of relatedness. These representations could be used in various text mining tasks to attain outstanding performance in terms of application-specific evaluation metrics.

Our propsed text representation method is called *Latent Keyphrase Inference* (LAKI). Its core technical contribution is domain keyphrase silhouetting—an unsupervised learning process to mine *domain keyphrase silhouettes*.

Definition 2 (Domain Keyphrase Silhouette). Given a domain keyphrase K_m, its *keyphrase silhouette* S_m comprises a topically-cohesive bag of *content units* (*i.e.*, words and phrases) regarding K_m. Let the total set of content units in the corpus is denoted as $T = \{T_1, \cdots, T_L\}$. Then S_m is a non-negative vector $[S_{m1}, \cdots, S_{mL}]$ where each entry S_{ml} refers to a relatedness score between domain keyphrase K_m and content unit T_l. ∎

These silhouettes not only enable the LAKI to identify latent document keyphrases through statistical inference, but also enhance the interpretability of corresponding domain keyphrases. LAKI can be divided into two phases: (i) *the offline domain keyphrase learning phase*, which extracts domain keyphrases from the in-domain corpus and learns their silhouettes respectively, and (ii) *the online document/query keyphrase inference phase*, which derives vectorial representation for each query based on the domain keyphrase silhouettes, as outlined below.

- *Offline Domain Keyphrase Learning:*
  1. Extract domain keyphrases from a domain-focused document corpus; and
  2. Learn domain keyphrase silhouettes by iteratively optimizing a Bayesian network with respect to the unknown values, i.e., latent document keyphrases, given observed content units in the training corpus."
- *Online Document/Query Keyphrase Inference:*
  1. Segment input query into content units; and
  2. Do inference for document keyphrases given the observed content units, generating a high-dimensional vector where each entry quantifies relatedness between the input query and corresponding keyphrase.

Fig. 1 illustrates the above two phases with examples for each individual step.

3. DOMAIN KEYPHRASE EXTRACTION AND SEGMENTATION

Domain Keyphrase Extraction refers to automatically discovering salient terms from a domain-focused document corpus, whereas *segmentation* is to partition text into continuous segments and is more context dependent. Both components serve as the building blocks for LAKI and examples of their functionality are shown on the left side of Fig. 1.

Solving these two tasks is not the objective of this work as there are various works proposed in the literature doing them well. In general, they can be divided into two categories: data-driven and linguistic-based. The former [24, 8, 17] explore frequent n-grams in document collections and leverage a variety of statistical measures derived from a corpus to estimate phrase quality and doing segmentation. The latter [9, 27, 15] rely on linguistic features that include part-of-speech (POS) tags or parse trees, and usually require large training sets labeled by humans to identify phrase boundaries, which are very costly to obtain. As the rest of our framework is fully data-driven, for the sake of consistency, we adopt an algorithm named SegPhrase which was recently proposed to combine domain keyphrase extraction with segmentation in an iterative fashion [17]. The algorithm is motivated by the observation that domain keyphrase extraction and segmentation are mutually dependent and thus can benefit each other. More specifically, domain keyphrase extraction relies on segmentation to locate candidate mentions, which later helps rectify their corpus-level statistics for assessing quality. Simultaneously, segmentation has access to stored extracted keyphrases to guide the partitioning of the document text into content units.

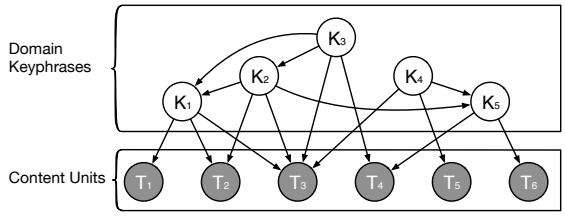

Figure 2: An illustrative Bayesian network for domain keyphrase silhouetting.

4. DOMAIN KEYPHRASE SILHOUETTING

In this section, we present our model to domain keyphrase silhouettes by optimizing a Bayesian network w.r.t. latent document keyphrases, given observed content units including both words and phrases in the training corpus. Recall that the *silhouette* of a domain keyphrase K_m consists of a bag of related content units for capturing the topic of K_m. Besides modeling dependency between keyphrases and content units, we consider interactions between keyphrases themselves and make the network hierarchical with DAG-like structure shown in Fig. 2. Content units are located at the bottom layer and domain keyphrases form the rest. Both types of nodes act as binary variables[1] and directional links between nodes depict their dependency. Specifically, the links connecting domain keyphrases to content units form the so-called *domain keyphrase silhouettes*.

Before diving into the technical details, we motivate our multi-layered Bayesian network approach to the silhouetting problem. First, it is better to have our model infer more than explicit document keyphrases. For example, even if the text only contains "html" and "css", the word "web page" comes to mind. But more than that, a multi-layered network will activate ancestor keyphrases like "world wide web" even they are not directly linked to "html" or "css", which are content units in the bottom layer.

Meanwhile, we expect to identify document keyphrases with different strength scores. Reflected in this Bayesian model from a top-down view, when a parent keyphrase is activated, it is more possible for its children with stronger connection to get activated.

[1]For multiple mentions of a content unit, we choose to make several copies of that node together with its links.

Furthermore, this formulation is quite flexible. We allow a content unit to get activated by each connected domain keyphrase as well as by a random noise factor (not shown in Fig. 2), which behaves like a *Noisy-OR*, *i.e.*, a logical OR gate with some probability of having "noisy" output. This increases robustness of the model especially when training documents are noisy.

We define the conditional distribution in our Bayesian network in line with the above motivations. Mathematically, we use $K = \{K_1, K_2, \ldots, K_M\}$ and $T = \{T_1, T_2, \ldots, T_L\}$ to denote domain keyphrases and content units respectively. For notational convenience, we use a unified symbol Z to denote K and T such that $K = \{Z_1, \ldots, Z_M\}$ and $T = \{Z_{M+1}, \ldots, Z_{M+L}\}$. A child node Z_j is Noisy-OR [13] with its parent nodes $Pa(Z_j) = \{Pa_j^1, Pa_j^2, \ldots\}$ as:

$$p(Z_j = 1 \big| Pa(Z_j)) = 1 - exp\Big(-W_{0j} - \sum_i W_{ij} \mathbb{1}_{Pa_j^i}\Big) \quad (1)$$

where W denotes link weight and $\mathbb{1}$ is an indicator function returning 1 if its associated node state is true. In this way, larger weight of a link will make its child node more likely to be activated. Note that the leak term W_{0j} allows for the possibility of a node to be true even if all parents are false. Intuitively, it can be explained as a prior for a domain keyphrase node and a noise for a content unit. Meanwhile, leak terms $\{W_{0*}\}$ for all nodes can be naturally transformed to link weights by positing a latent factor Z_0 with $p(Z_0 = 1) = 1$, where notationally convenient. An example of such a notation is shown on the left side of Fig. 3 for a tiny family.

In the following subsections, we first discuss how to learn link weights given the Bayesian network structure and then discuss how the initialization is done to decide this structure and to set initial link weights.

4.1 Model Learning

To effectively learn link weights during domain keyphrase silhouetting, Maximum Likelihood Estimation is adopted. The intuition is to estimate parameters by maximizing the likelihood of observed content units together with partially-observed document keyphrases[2]. Suppose we have N documents in the corpus where each document d uses a binary vector $t^{(d)}$ to represent the states of content units (*i.e.*, observed or not), the log-likelihood of the corpus is:

$$L(D) = \sum_{d=1}^{N} \log \sum_{k \in \Omega^{(d)}} p(K = k, T = t^{(d)}) \quad (2)$$

where $\Omega^{(d)}$ is the space of all possible combinations of document keyphrase states. This space changes for different documents and will be discussed later in this subsection.

It is difficult to directly optimize Eq. 2 due to the latent states for the rest keyphrases. Instead we resort to the Expectation-Maximization (EM) algorithm which guarantees to give a local optimum solution. The EM algorithm starts with some initial guess at the maximum likelihood parameters and then proceeds to iteratively generate successive estimates by repeatedly applying the E-step (Expectation-step) and M-step (Maximization-step). For general Bayesian networks, normally $p(Z_j, Pa(Z_j)|T = t)$ must be computed for all state combinations between Z_j and $Pa(Z_j)$ in the E-step. In our case due to the presence of Noisy-OR as in

[2]Explicit document keyphrases can be identified by applying existing keyphrase extraction methods like [24].

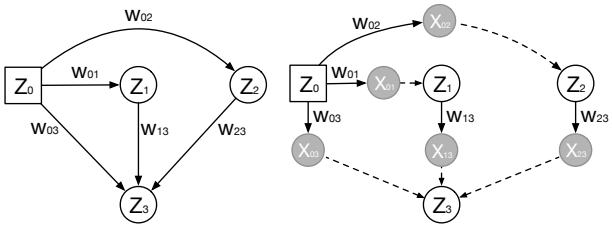

Figure 3: An alternative representation of Noisy-Or Bayesian network. We assume parents $Pa(Z_3)$ of Z_3 are $Pa_3^1 = Z_1$ and $Pa_3^2 = Z_2$ respectively.

Eq. 1, we can dramatically reduce the complexity by dividing the whole family into parent-child pairs and computing each pair separately. To demonstrate this, we show an alternative representation of the left Noisy-OR network in Fig. 3 by adding grey nodes for each link. A grey node X_{ij} is true with probability $1 - \exp(-W_{ij})$ only if the parent node Pa_j^i is true. The original child white node on the left now becomes deterministic-OR of parent grey nodes. Based on this representation, one can still derive the same probability distribution as Eq. 1.

Expectation Step: Since the child white node always performs OR operation on its parents, we only need to focus on the grey nodes with single parent. This reduces a lot of storage consumption and transforms our task to computing $R_{ij}^{(d)} = P(X_{ij} = 1, Pa_j^i = 1 | T = t^{(d)}, \Omega^{(d)})$ together with $P_m^{(d)} = P(K_m = 1 | T = t^{(d)}, \Omega^{(d)})$ where:

- $R_{ij}^{(d)}$ refers to the probability of a grey node X_{ij} to be activated as well as that both of its parent Pa_j^i and child node Z_j are present; and
- $P_m^{(d)}$ refers to the probability of a domain keyphrase node K_m to be activated, *i.e.*, becoming a document keyphrase.

Unfortunately, to compute these two exactly, one still needs to enumerate all state combinations of domain keyphrases. This is NP-hard for a Bayesian network like ours [5]. We therefore constrain the search space $\Omega^{(d)}$ such that non-related keyphrases are directly excluded before applying EM. That is, we only allow keyphrase ancestors of observed content units to change states during the inference. We are essentially forcing certain elements of K to be fixed in their states during the enumeration of $\Omega^{(d)}$. The corresponding analytical expressions are derived following Bayes rules:

$$R_{ij}^{(d)} = \frac{\sum_{c \in \Omega^{(d)}} p\big(Z = \{k, t^{(d)}\}\big) \frac{P(X_{ij}=1|Pa_j^i)}{p(Z_j=z_j|Pa(Z_j))} \mathbb{1}_{z_j} \mathbb{1}_{Pa_j^i}}{\sum_{z \in \Omega^{(d)}} p\big(Z = \{k, t^{(d)}\}\big)}$$

$$P_m^{(d)} = \frac{\sum_{z \in \Omega^{(d)}} p\big(Z = \{k, t^{(d)}\}\big) \mathbb{1}_{k_m}}{\sum_{z \in \Omega^{(d)}} p\big(Z = \{k, t^{(d)}\}\big)}$$

For longer text, there will still be quite many keyphrases left for inference following the above strategy, making inference intractable at a large scale. We therefore resort to approximate inference by applying a stochastic sampling technique for generating samples from the joint probability distribution over Z. This type of approximation technique is also used for online inference, introduced in the next section.

Maximization Step: Based on the sufficient statistics collected by the Expectation step, one can update each link weight W_{ij} between node Pa_j^i and X_{ij} with the following closed-form solutions:

$$W_{ij} = -\log\Big(1 - \frac{\sum_d R_{ij}^{(d)}}{\sum_d P_{Pa_j^i}^{(d)}}\Big); \quad W_{0j} = -\log\Big(1 - \frac{\sum_d R_{0j}^{(d)}}{|N|}\Big)$$

Expectation and maximization steps are iterated until the model changes minimally.

4.2 Model Initialization

Like other EM frameworks, the parameter estimation will suffer from the problem of local maximum, making the results vary with different network structures and initialized link weights. Therefore, to obtain a good initialization before model training is important for our task.

Specifically, there are two sub-problems for the model initialization:

1. How to decide the topological order among domain keyphrase nodes and to build links among them?
2. How to build links between domain keyphrase and content units?

For the former, a reasonable topological order of DAG should be similar to that of a domain ontology. The links among domain keyphrase nodes should reflect IS-A relationships [26]. Ideally, documents and queries which are describing specific topics will first imply some deep keyphrase nodes being activated. Then the ontology-like topological order ensures these content units have the chance of being jointly activated by general keyphrases via inter-keyphrase links. Many techniques [26, 20, 6] have been previously developed to induce an ontological structure over keyphrases. It is out of scope of our work to specifically address these or evaluate their relative impact in our evaluation. We instead use a simple data-driven approach, where domain keyphrases are sorted based on their counts in the corpus, assuming keyphrase generality is positively correlated withs its number of mentions [20]. Thus, keyphrases mentioned more often are higher up in the graph. Links are added from keyphrase K_i to K_j if K_i has more counts and they are closely related and frequently co-occured:

$$p(K_i|K_j) \geq \alpha, \quad sim(K_i, K_j) \geq \beta$$

where α is a threshold reflecting the confidence about the IS-A relationship and β requires two domain keyphrases to be related. And $sim(K_i, K_j)$ is computed based on the cosine similarity between word2vec embeddings of keyphrases K_i and K_j. In our work, we empirically set α and β to be 0.5 and 0.3 respectively. Note that the latter score is also used to detect equivalence between keyphrases (*i.e.*, acronyms or inflectional variants) and we merge them to alleviate the duplication problem. We remark that some more sophisticated work can be applied here to help detect different lexical semantic and syntactic relations. We leave this for future work.

Once the topological order among domain keyphrase nodes has been decided, one can concatenate all content units right after the sorted keyphrases and link higher-ranked keyphrase nodes to lower-ranked content units when $sim(K_m, T_j) \geq \beta$. We initialize link weights between nodes to be their $sim(\cdot, \cdot)$ scores. As for the leak terms of nodes, they are simply set to be the probability of observing them in the corpus.

5. ONLINE INFERENCE

The online inference is designed to efficiently quantify the relatedness between the text query and its potential document keyphrases. Inspired by the sufficient statistics collected in E-step, we are particularly interested in computing $P_m^{(q)} = p(K_m|T = t^{(q)})$, *i.e.*, the activation probability for a certain keyphrase K_m, as the non-negative weight for the m_{th} entry in our output vectorial representation.

Notice that in the online inference phase, efficiency is usually a big challenge and the inference previously mentioned for the E-step will be intractable if the document becomes too long. In this regard, we resort to an approximate sampling approach. This technique can be applied to compute both $P_m^{(q)}$ and $P_m^{(d)}$. At the same time, $R_{ij}^{(d)} = p(X_{ij}, Pa_j^i|T = t^{(d)})$ needed in the E-step can also be benefited.

In the last section, we discussed about exact inference in the E-step where enumeration over potential document keyphrase states are necessary to help compute the above terms. In fact, they can be more efficiently approximated by use of Monte Carlo sampling methods, which are a set of computational techniques for generating samples from a target distribution like the joint probability $p(K, T = t)$ in our setting. Among the Monte Carlo family, we apply Gibbs sampling in this work to sample keyphrase variables during each inference procedure. Given content unit vector t representing a document d or query q, we proceed as follows:

1. Start with initial setting: only observed content units and explicit document keyphrases are set to be true, denoted by $\{k^{(0)}, t\}$
2. For each $s \in \{1, \ldots, S\}$, sequentially sample all keyphrase nodes following conditional distribution $p(K_m|K_{-m} = \{k_1^{(s)}, \ldots, k_{m-1}^{(s)}, k_{m+1}^{(s-1)}, \ldots, k_M^{(s-1)}\}, T = t)$, denoted as $k^{(s)}$.

where

$$p(K_m = 1|K_{-m} = k_{-m}; T = t) = \frac{p(K_m = 1; K_{-m} = k_{-m}; T = t)}{\sum_{i=0}^{1} p(K_m = i; K_{-m} = k_{-m}; T = t)} \tag{3}$$

To compute Eq. 3 efficiently, for every domain keyphrase node, we maintain the following probability ratio:

$$\frac{p(K_m = 1, K_{-m} = k_{-m}, T = t)}{p(K_m = 0, K_{-m} = k_{-m}, T = t)} \tag{4}$$

where K_{-m} refers to all the keyphrase nodes except K_m. Given the above ratio, one can easily compute Eq. 3 needed for sampling K_m.

Now the problem becomes how to maintain Eq. 4 for each keyphrase node during the sampling process. In fact, according to the chain rule in Bayesian network, we have

$$\frac{p(K_m = 1, K_{-m} = k_{-m}, T = t)}{p(K_m = 0, K_{-m} = k_{-m}, T = t)} = \frac{p(K_m = 1|Pa(K_m))}{p(K_m = 0|Pa(K_m))} \times \prod_{Z_j \in Ch(K_m)} \frac{p(Z_j|Pa_{-K_m}(Z_j), K_m = 1)}{p(Z_j|Pa_{-K_m}(Z_j), K_m = 0)}$$

where $Ch(K_m)$ refers K_m's children. From the above equation, one can conclude that Eq. 4 for node K_m should be updated whenever nodes in its Markov blanket[3] change states.

The above Gibbs sampling process ensures that samples approximate the joint probability distribution between all keyphrase variables and content units. Such sampling is performed over all the original nodes in the network but does not include the grey nodes (see Fig. 3) in the alternative representation for the sake of sampling efficiency. One can easily compute the probability distribution over each of the grey nodes given a domain keyphrase state combination.

[3]Markov blanket for a node in a Bayesian network composed of its parents, children and children's other parents.

Method	Semantic Space	Input Source	Toolkit
ESA	KB concepts	KB	ESAlib
KBLink	KB concepts	KB	WikiBrain
BoW	Words	-	scikit-learn
ESA-C	Documents	Corpus	ESAlib
LSA	Topics	Corpus	scikit-learn
LDA	Topics	Corpus	MALLET
Word2Vec	-	Corpus	gensim
EKM	Explicit Keyphrases	Corpus	-
LAKI	Latent Keyphrases	Corpus	-

Table 2: Comparisons among different methods

By marginalizing over necessary domain keyphrase variables in the Bayesian network, the following approximate equations can be derived:

$$\hat{R}_{ij} = \frac{\sum_{s=1}^{S} \frac{p(X_{ij}=1|Pa_j^i)}{p(Z_j=z_j^{(s)}|Pa(Z_j))} \mathbb{1}_{z_j^{(s)}} \mathbb{1}_{Pa_j^i}}{S}; \quad \hat{P}_m = \frac{\sum_{s=1}^{S} \mathbb{1}_{k_m^{(s)}}}{S}$$

For online inference, the ultimate representation for text query q is a high-dimensional vector $[\hat{P}_1^{(q)}, \hat{P}_2^{(q)}, \cdots, \hat{P}_M^{(q)}]$ where non-zero entries indicate document keyphrases.

To further improve the efficiency of Gibbs sampling, one can follow the idea of E-step to reduce the number of sampled nodes. Intuitively, only a small portion of domain keyphrases are related to the text query. There is no need to sample all keyphrase nodes since most of them do not have chance to get activated. That is to say, we can skip majority of them based on a reasonable relatedness prediction before conducting Gibbs sampling. Suppose content unit vector $T' = \{T'_1, \cdots, T'_l\} \subseteq T$ contains only observed content units. We pick the following scoring function:

$$p(T' = \{1, \cdots, 1\} | Z_j = 1)$$

This score can be viewed as the probability of generating the observed content units when domain keyphrase Z_j is activated. Computing $p(T' = \{1, \cdots, 1\}|Z_j = 1)$ is still challenging because keyphrases are connected and enumeration over state combinations of connected keyphrases are unavoidable. Thus we adopt a local arborescence structure inspired by [22] to approximate the probability by keeping the path from Z_j to each content unit T'_r for which the activation probability product along the path is maximum among all paths from Z_j to T'_r. In this way, the activation probability for each content unit is independent given Z_j is activated. The associate probability of the approximate link between keyphrase node Z_j and content unit T_r is denoted as $\tilde{p}(T'_r = 1|Z_j = 1)$. We then have:

$$p(T^0 = \{1; \cdots; 1\}|Z_j = 1) \approx \prod_r \Big(1 - \big(1 - p(T_r^0 = 1|Z_0)\big) \times \big(1 - \tilde{p}(T_r^0 = 1|Z_j = 1)\big)\Big)$$

In addition, the score can be naturally and efficiently propagated from children to parents in the network recursively:

$$\tilde{p}(T'_r = 1|Z_j = 1) = \max_i \ \tilde{p}(T'_r = 1|Ch_j^i = 1)\, p(Ch_j^i = 1|Z_j = 1)$$

where Ch_j^i refers to the i_{th} child node of Z_j. The above equation can be computed efficiently following reverse topological order of domain keyphrase nodes.

6. EXPERIMENTAL STUDY

In this section, experiments were conducted to demonstrate the effectiveness of the proposed LAKI in generating high quality vectorial representations of text queries. We begin with the description of datasets.

Dataset	#Docs	#Words	Content type
Academia	0.43M	28M	title & abstract
Yelp	0.47M	98M	review

Table 3: Dataset statistics

6.1 Datasets

Two real-world data sets were used in the experiments and detailed statistics are summarized in Table 3.

- The **Academia** dataset[4] is a collection of major computer science publications. We use both paper titles and abstracts in venues of database, data mining, machine learning, natural language processing and computer vision.
- The **Yelp** dataset[5] provides reviews of 250 businesses. We extract all the reviews belong to the "restaurant" category and each individual review is considered as a document.

6.2 Compared Methods

The proposed method is compared with an extensive set of methods. They are briefly described as follows.

- Explicit Semantic Analysis (**ESA**) [10] views each text query as a weighted vector of KB entries, where the values on each dimension denote the similarity scores computed between the query and the associated KB entries
- **KBLink** [23] first detects related KB entries in the query and then represents it using the hyperlink structures of the KB centered at the identified entries
- **BoW** stands for bag-of-words method where each dimension reflects word frequency in the query
- **ESA-C** extends ESA by replacing the general KB with a domain-specific corpus where each document is considered to be a KB entry in the original ESA framework
- Latent Semantic Analysis (**LSA**) [7] is a topic modeling technique learning word and document representations by applying Singular Value Decomposition to the words-by-documents co-occurrence matrix
- Latent Dirichlet Allocation (**LDA**) [3] is a probabilistic topic model assuming words in each document were generated by a mixture of topics, where a topic is represented as a multinomial probability distribution over words
- **Word2Vec** [19] computes continuous distributed representations of words by training a neural network, with the desideratum that words with similar meanings will map to similar vectors
- Explicit Keyphrase Mentiosn (**EKM**) uses bag of explicit domain keyphrase mentions
- Latent Keyphrase Inference (**LAKI**) is proposed in this work to derive document representation via inferring latent keyphrases in the text.

Table 2 provides more details about the differences among the above methods. The first two methods utilize knowledge base as the input source to train models in order to represent any new text query. In contrast, the rest methods except BoW require a domain-specific corpus as the training source.

6.3 Experimental Setting

For domain keyphrase extraction, we set the minimum keyphrase support as 10 and the maximum keyphrase length as 6, which are two parameters required by SegPhrase. The

[4] http://aminer.org/billboard/AMinerNetwork
[5] https://www.yelp.com/academic_dataset

process generates 33 and 25 thousand domain keyphrases on Academia and Yelp respectively. Among them, only 6367 and 4996 exist in Wikipedia, which implies its knowledge incompleteness.

LAKI was initialized and trained following Sec. 4.2 on both datasets, whose model information is listed in Table 4. EKM uses the same set of domain keyphrases. The Yelp model contains more content units but has fewer domain keyphrases and links, indicating the knowledge underlying Yelp reviews is less structured. Regarding the inference process of LAKI, we run the Gibbs sampler for 5000 iterations per query. The pruning strategy introduced in Sec. 5 is applied to select at most 200 keyphrase nodes as candidates for sampling. Settings of these two parameters will be discussed later in this section. EM steps are repeated until the change on training perplexity is small enough.

Regarding other methods, ESA and KBLink need a general KB as the input source. Wikipedia is chosen since it is rich in both text content and structural links. For LSA and LDA, both require users to specify number of topics before training. During our experiments, various numbers of topics ranging from 10 to 1000 have been tested and the best got reported. Word2Vec has two main learning algorithms: continuous bag-of-words and continuous skip-gram. We compared both of them in our tasks and discovered the former generally worked better. For the sake of convenience, results of the continuous bag-of-words algorithm with context window size of 5 are reported. As suggested in the paper, negative sampling was activated and vector dimensionality was set to be 300.

In addition, TF-IDF is applied to re-weight terms for BoW, LSA and EKM. Stop words are removed for all the methods. Other parameters required by contrasting methods were set as the default values according to the toolkits.

6.4 Evaluation

The goal of our experiments is to quantitatively evaluate how well our method performs in generating vectorial representations of online queries. We introduce our empirical evaluation in two problem domains: phrase relatedness and document classification.

Phrase Relatedness: For a set of phrase pairs, method performance is evaluated by how well the generated relatedness scores correlate with the gold scores. The gold score for each phrase pair is based on human judgements of the pair's semantic relatedness.

To create these phrase pairs, we first generate 100 pairs of frequently co-occurred and potentially realated phrases from concept mentions in Wiki articles. Then we expanded the set by randomly combining phrases from different pairs and the size of final evaluation set is 300. Each pair was carefully assigned a relatedness score from 0 to 3 where higher score indicates stronger similarity. We used Pearson's linear correlation coefficient to compare computed relatedness scores with human judgements.

Document Classification: In this classification task, we wish to classify a document into several mutually exclusive classes. A challenging aspect of the document classification problem is the choice of features. Considering different vectorial representations as document features, we expect the classification accuracy can well reflect the discriminative power of each method.

Dataset	#Nodes	#Links	#Domain Keyphr. (in Wiki)
Academia	237K	1.40M	33K (6,367)
Yelp	292K	1.14M	25K (4,996)

Table 4: Model information of LAKI

For each compared method, we first derived vectorial representations for all the training and testing documents without reference to their true class label. Then a support vector machine with both linear and radial basis function kernels was trained on the training set (only the best was reported). Classification accuracy on the testing set is reported to measure the performance of the method.

For the methods trained on the Academia dataset, we created a held-out set of 500 publications authored by *five researchers in different research areas*. We sampled each author's publications to avoid the problem of class imbalance. Regarding the Yelp dataset, we followed the above procedure and extracted *1000 reviews for 10 chain restaurants*. As for the classification details, we used LibSVM software package and created the training set with 70% of all documents. Five-fold cross validation was conducted to decide the proper parameter of the kernel. Five test runs were conducted on different randomly partitioned training and test sets. The average performance is reported.

6.5 Results

The correlations between computed relatedness scores and human judgements are shown in Table 5. Let's first ignore the numbers in parenthesis and we can discover that the trends on two datasets are similar.

Both BoW and EKM are not reported because queries (*i.e.*, phrases) are too short and they tend to return 0 most of the time, making it inappropriate for this task. Among other existing work, Word2Vec has the best performance due to its effective modeling of word representations in low dimensional space. Meanwhile, Word2Vec is good at representing very short text by simply computing arithmetic mean of word embeddings. LDA performs the worst in this task, suffering from the severe data sparsity in short text inference scenario. This phenomenon is consistent with recent studies on short text topic modeling. As for the two KB-based methods, both behave relatively poor in spite of their extraordinary performance reported on open domain in [10, 23]. KBLink is noticeably lower on Academia dataset due to the sparsity of hyperlinks between academic concepts. This is probably due to the reasons about domain restriction and out-of-domain noise. We can verify this assumption to some extent by comparing it with ESA-C. The latter method replaces KB with a domain corpus, which can be considered as a weak domain adaptation by viewing each document as a KB entry. ESA-C thus outperforms ESA slightly but its performance is still not satisfactory compared to LSA and Word2Vec. This implies enormous potential if a method is able to handle domain adaptation well, and meanwhile to bring KB-like semantics into each dimension. Our proposed method, LAKI, first extracts domain keyphrases from a corpus and then represents queries in the space of these keyphrases, fulfilling the above two targets simultaneously. Consequently, it outperforms the second best method (*i.e.*, Word2Vec) by 0.083 on Academia and 0.0466 on Yelp dataset, which is even more statistically significant than the gap between the second and the third best methods.

Table 6 shows evaluation results for the document classification task. LAKI still achieves much improvement com-

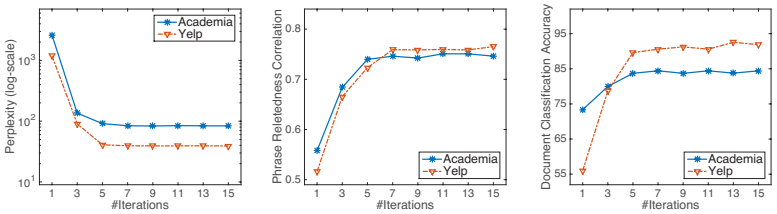

Figure 4: Training perplexity and performance of LAKI versus increasing iterations of EM

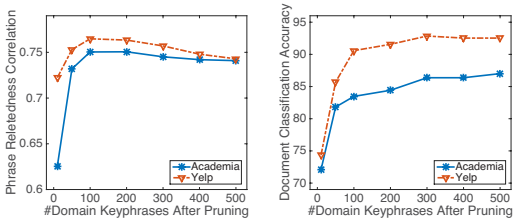

Figure 5: Performance variations of LAKI with increasing number of keyphrases

Method	Academia (w/ phrase)	Yelp (w/ phrase)
ESA	0.4320 (-)	0.4567 (-)
KBLink	0.1878 (-)	0.4179 (-)
ESA-C	0.4905 (0.5243)	0.4655 (0.5029)
LSA	0.5877 (0.6383)	0.6700 (0.7229)
LDA	0.3610 (0.5391)	0.3928 (0.5405)
Word2Vec	0.6674 (0.7281)	0.7143 (0.7419)
LAKI	**0.7504**	**0.7609**

Table 5: Phrase relatedness correlation

Method	Academia (w/ phrase)	Yelp (w/ phrase)
ESA	37.61 (-)	46.56 (-)
KBLink	36.37 (-)	35.94 (-)
BoW	48.05 (45.60)	51.26 (45.97)
ESA-C	39.75 (42.20)	49.13 (54.51)
LSA	72.50 (79.22)	66.55 (78.57)
LDA	77.27 (80.52)	75.55 (82.65)
EKM	45.46	40.57
LAKI	**84.42**	**90.58**

Table 6: Document classification accuracy (%)

pared with the competitors. The difference is especially noticeable on Yelp, where LAKI achieves 90.58% accuracy and the second best is 75.55%. It supports our claim that the proposed LAKI method is quite useful in representing both short-text and long-text documents. Among the other methods, LDA generates the best results since each document now contains more words than the previous short text scenario, making it easier to do inference by utilizing the word co-occurrence patterns. BoW is not performing well mainly due to the over-sparsity problem. Though documents are relatively long (100 words for Academia and 200 words for Yelp on average), some documents from the same category still share only a few words, which makes the classifier easy to misclassify. Similar to BoW, EMK performs much worse than LAKI due to the sparsity of keyphrase mentions, indicating the superiority of using latent keyphrases over explicit mentions. ESA-C beats ESA once again, reflecting the shortcoming of using general KB on a topic-focused domain. Word2Vec is omitted from the table due to its poor performance when applied to long text. Simply computing arithmetic or geometric mean of word embeddings results in very poor performance (close to random guess). We have also tried applying gradient descent to learn the document representations as introduced in [16]. But the performance is still far below our expectation.

Another significant difference between LAKI and the existing work is its support for multi-word phrases. By segmenting queries into content units, LAKI views each input query as a bag of phrases instead of words. Some other methods including BoW, ESA-C, LSA, LDA and Word2Vec can be modified to support such phrase-based input just like LAKI. Therefore, we have conducted experiments to show whether using the same input as LAKI can help boost their performance. Numbers within parenthesis in Tables 5 and 6 indicate the corresponding performance after switching to the phrase-based input. The highest correlation scores obtained by Word2Vec increase to 0.7218 and 0.7419 on the two datasets, which are still beated by LAKI. For the document classification task, LDA achieves the best accuracy among all contrasting methods, which are still 3.9% and 7.93% lower than LAKI for the two datasets respectively. It is interesting to see that BoW fails to utilize the phrase-based input. Our explanation is that BoW is lack of a training process and it relies fully on the content units in the input. It usually becomes more difficult for queries to share phrases than words due to the decrease in number of content units after segmentation.

6.6 Model Selection

In this subsection, we study the model behavior under different experimental settings. We begin with an empirical convergence study of the EM algorithm. Fig. 4 presents the training perplexity of LAKI with its performance versus iteration. Due to the good initialization discussed in Sec. 4.2, the perplexity becomes quite stable after the fifth iteration. This help save a lot of training time in practice. The perplexity is not monotonically decreasing because of the approximations resulted from pruning and sampling.

There are two parameters in our LAKI method: number of domain keyphrases after pruning and sample size for Gibbs sampler. The former parameter decides the number of keyphrase nodes involved in the later sampling process. A smaller value will make the final output sparser while a larger one has risk in incorporating more unrelated keyphrases to undermine the performance. Fig. 5 shows how the performance varies with changes in this parameter. We can observe a peak around 150 keyphrases for the phrase relatedness task. In contrast, the curve is generally going up for the document classification task and becomes stable around 400. Our explanation is that the queries for the latter task contain more observed content units and there should exist more document keyphrases for longer text queries. This suggests a way to dynamically decide the number of pruned keyphrases. But in this work we simply fix its value to 200 and plan to explore this idea in the future.

For the Gibbs sampling size, a larger value usually leads to more accurate and stable estimation of the probability distribution. We show the experiment in Fig. 8 where both mean and standard deviation are reported. The curves are consistent with our expectation and they become relatively flat after 5000. Based on this observation we set the sampling size to 5000 for the sake of saving time consumption. Another experiment has been conducted to show the benefit by considering more domain keyphrases into the Bayesian network. As reported in Table 4, there is a large portion of keyphrases not existing in Wikipedia. To justify the domain restriction of Wiki, it is interesting to compare the

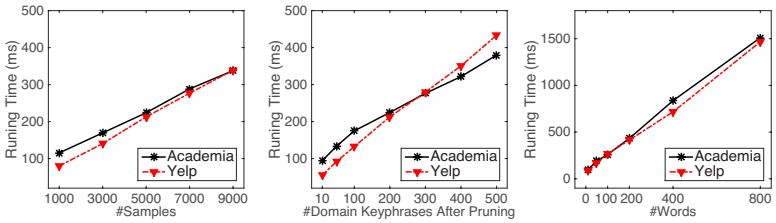

Figure 6: Impact of sample size, number of keyphrases, and word counts on query processing time

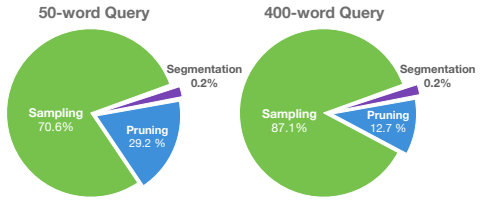

Figure 7: Breakdown of query processing time

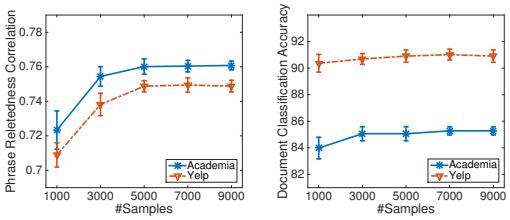

Figure 8: Performance with increasing sample size

standard LAKI model with the simplified version trained only on Wiki-covered keyphrases (see its definition in [17]). The bar plots in Fig. 9 demonstrate that with more domain keyphrases involved, LAKI is able to achieve outstanding improvement.

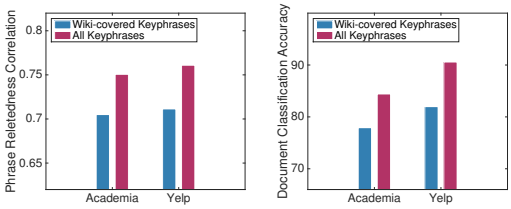

Figure 9: Performance with different keyphrase sets

6.7 Scalability Study

To understand the run-time complexity of our framework, we first analyse the execution time of online query inference phase. The experiments were conducted on a machine with Intel Core i7-2600K and 16GB memory. The code is implemented in C++. As shown in Fig. 6, LAKI grows linearly proportional to the sampling size, the number of domain keyphrases after pruning and the length of the query. Besides this, the pies in Fig. 7 show ratios of different components of our framework. One can observe that the pruning and sampling steps occupy most of the runtime. Moreover, as query size increases, the sampling part consumes relatively more. Fortunately, almost all components of our frameworks can be easily parallelized, because of the nature of independence between documents and queries. For the offline keyphrase learning phase, as the very first step, keyphrase extraction shares similar time complexity as reported in [17]. The most time consuming part is domain keyphrase silhouetting, which is an EM algorithm and each individual inference in E-step has similar performance compared to Figs. 6 and 7. The only difference is the fewer pruning time for silhouetting because the search space is constrained as discussed in Sec. 4.1.

6.8 Case Study

Previous experiments are focused on evaluating representation quality and time complexity quantitatively. In this subsection, we first present several queries with their top-ranked document keyphrases in Table 7 generated from the online phase of LAKI. Overall we see that LAKI can handle both short and long queries quite well. Most document keyphrases are successfully identified in the list. Relatedness between keyphrase and queries generally drops with ranking lowers down. Meanwhile, both general and specific document keyphrases exist in the ranked list. This provides results from LAKI with more discriminative power when someone is applying it to text mining applications like document clustering and classification. Moreover, LAKI has the ability to process ambiguous queries like "lda" based on contextual words "topic". We attribute this to the well-modelled domain keyphrase silhouettes and we show some examples of them in Table 8. As a domain keyphrase silhouette might contain many content units, we only demonstrate ones with the most significant link weights. For ease of presentation, link weights are omitted in the table.

7. CONCLUSIONS

In this paper, we have introduced a new research problem of learning representation for domain-specific texts. We propose a novel method called Latent Keyphrase Inference which integrates domain keyphrase extraction and silhouetting to help infer latent document keyphrases and solves the rarity of explicit keyphrase mentions in the query. The generated high-dimensional representations of documents are shown to significantly boost performance in potential text mining tasks compared to state-of-art methods. Meanwhile, entries in the document vector are highly self-explanatory through automatically learned domain keyphrase silhouettes.

A number of open problems need to be solved to allow further development of LAKI. One direction is to simultaneously model structured data such as meta information associated with the documents, including named entity, authorship, publishers, *etc.*. An alternative is to improve the model initialization by modeling more sophisticated relationship between domain keyphrases and considering more robust structured learning method. Regarding the scalability, it would be preferable if one can work out a more efficient inference algorithm. For example, we can use a deterministic module like neural network and train it using inference results from our current framework.

Acknowledgements

Research was sponsored in part by the U.S. Army Research Lab. under Cooperative Agreement No. W911NF-09-2-0053 (NSCTA), National Science Foundation IIS-1017362, IIS-1320617, and IIS-1354329, HDTRA1-10-1-0120, and grant 1U54GM114838 awarded by NIGMS through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative (www.bd2k.nih.gov), and MIAS, a DHS-IDS Center for Multimodal Information Access and Synthesis at UIUC.

Query	*LDA*	*BOA*
Document Keyphrases	linear discriminant analysis, latent dirichlet allocation, topic models, topic modeling, face recognition, latent dirichlet, generative model, topic, subspace models, ...	boa steakhouse, bank of america, stripsteak, agnolotti, credit card, santa monica, restaurants, wells fargo, steakhouse, prime rib, bank, vegas, las vegas, cash, cut, dinner, bank, money, ...
Query	*LDA topic*	*BOA steak*
Document Keyphrases	latent dirichlet allocation, topic, topic models, topic modeling, probabilistic topic models, latent topics, topic discovery, generative model, mixture, text mining, topic distribution, ...	steak, stripsteak, boa steakhouse, steakhouse, ribeye, craftsteak, santa monica, medium rare, prime, vegas, entrees, potatoes, french fries, filet mignon, mashed potatoes, texas roadhouse, ...
Query	*SVM*	*deep dish pizza*
Document Keyphrases	support vector machines, svm classifier, multi class, training set, margin, knn, classification problems, kernel function, multi class svm, multi class support vector machine, support vector, ...	deep dish pizza, chicago, deep dish, amore taste of chicago, amore, pizza, oregano, chicago style, chicago style deep dish pizza, thin crust, windy city, slice, pan, oven, pepperoni, hot dog, ...
Query	*Mining Frequent Patterns without Candidate Generation*	*I am a huge fan of the All You Can Eat Chinese food buffet.*
Document Keyphrases	mining frequent patterns, candidate generation, frequent pattern mining, candidate, prune, fp growth, frequent pattern tree, apriori, subtrees, frequent patterns, candidate sets, ...	all you can eat, chinese food, buffet, chinese buffet, dim sum, orange chicken, chinese restaurant, asian food, asian buffet, crab legs, lunch buffet, fan, salad bar, all you can drink, ...
Query	*Text mining, also referred to as text data mining, roughly equivalent to text analytics, refers to the process of deriving high-quality information from text. High-quality information is typically derived through means such as statistical pattern learning.*	*It's the perfect steakhouse for both meat and fish lovers. My table guest was completely delirious about his Kobe Beef and my lobster was perfectly cooked. Good wine list, they have a lovely Sancerre! Professional staff, quick and smooth.*
Document Keyphrases	text analytics, text mining, patterns, text, textual data, topic, information, text documents, information extraction, machine learning, data mining, knowledge discovery, ...	kobe beef, fish lovers, steakhouse, sancerre, wine list, guests, perfectly cooked, lobster, staff, meat, fillet, fish, lover, seafood, ribeye, filet, sea bass, risotto, starter, scallops, steak, beef, ...
	Academia	Yelp

Table 7: Examples of document representation by LAKI with top-ranked document keyphrases in the vector (relatedness scores are ommited due to the space limit).

Domain Keyphr.	*linear discriminant analysis*	*boa steakhouse*
Silhouette	linear discriminant analysis, lda, face recognition, feature extraction, principle component analysis, uncorrelated, between class scatter, ...	boa steakhouse, boa, steakhouse, restaurant, dinner, strip steak, craftsteak, santa monica, vegas, filet, ribeye, new york strip, sushi roku, ...
Domain Keyphr.	*latent dirichlet allocation*	*ribeye*
Silhouette	latent dirichlet allocation, lda, topics, perplexity, variants, subspace, mixture, baselines, topic models, text mining, bag of words, ...	ribeye, steak, medium rare, medium, oz, marbled, new york strip, well done, prime rib, fatty, juicy, top sirloin, filet mignon, fillet, ...
Domain Keyphr.	*support vector machines*	*deep dish*
Silhouette	support vector machines, svm, classification, training, classifier, machine learning, prediction, hybrid, kernel, feature selection, ...	deep dish, pizza, crust, thin crust pizza, chicago, slice, pepperoni, deep dish pizza, pan style, pizza joints, oregano, stuffed crust, chicago style, ...
Domain Keyphr.	*fp growth*	*chinese food*
Silhouette	fp growth, algorithm, apriori like, mining, apriori, frequent patterns, mining association rules, frequent pattern mining, fp tree, ...	chinese food, food, chinese, restaurants, americanized, asian, orange chicken, chow mein, wok, dim sum, panda express, chinese cuisine, ...
Domain Keyphr.	*text mining*	*mcdonalds*
Silhouette	text mining, text, information retrieval, machine learning, topics, knowledge discovery, text data mining, text clustering, nlp, ...	mcdonalds, drive through, fast food, mcnugget, mcflurry, fast food chain, sausage mcmuffin, big bag, mcmuffin, burger king, ...
Domain Keyphr.	*database*	*sushi*
Silhouette	database, information, set, objects, storing, retrieval, queries, accessing, relational, indexing, record, tables, query processing, transactions, ...	sushi, rolls, japanese, sushi joint, seafood, ayce, sushi rolls, salmon sushi, tuna sushi, california roll, sashimi, sushi lovers, sushi fish, ...
	Academia	Yelp

Table 8: Examples of domain keyphrase silhouettes (from offline domain keyphrase learning). Link weights are ommited.

8. REFERENCES

[1] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. 1999.

[2] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. Dbpedia-a crystallization point for the web of data. *Web Semantics: science, services and agents on the world Wide Web*, 7(3):154–165, 2009.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[4] J. Chang, J. Boyd-Graber, C. Wang, S. Gerrish, and D. M. Blei. Reading tea leaves: How humans interpret topic models. In *Advances in Neural Information Processing Systems 22*, 2009.

[5] G. F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42(2):393–405, 1990.

[6] M. Y. Dahab, H. A. Hassan, and A. Rafea. Textontoex: Automatic ontology construction from natural english text. *Expert Systems with Applications*, 34(2):1474–1480, 2008.

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

[8] A. El-Kishky, Y. Song, C. Wang, C. R. Voss, and J. Han. Scalable topical phrase mining from text corpora. *Proceedings of the VLDB Endowment*, 8(3):305–316, 2014.

[9] K. Frantzi, S. Ananiadou, and H. Mima. Automatic recognition of multi-word terms:. the c-value/nc-value method. *International Journal on Digital Libraries*, 3(2):115–130, 2000.

[10] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, pages 1606–1611, 2007.

[11] W. A. Gale, K. W. Church, and D. Yarowsky. One sense per discourse. In *Proceedings of the Workshop on Speech and Natural Language*, pages 233–237. Association for Computational Linguistics, 1992.

[12] T. Gottron, M. Anderka, and B. Stein. Insights into explicit semantic analysis. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pages 1961–1964, 2011.

[13] Y. Halpern and D. Sontag. Unsupervised learning of noisy-or bayesian networks. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, pages 272–281, 2013.

[14] S. Hassan and R. Mihalcea. Semantic relatedness using salient semantic analysis. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[15] A. Hulth. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 216–223, 2003.

[16] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning*, pages 1188–1196, 2014.

[17] J. Liu, J. Shang, C. Wang, X. Ren, and J. Han. Mining quality phrases from massive text corpora. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1729–1744, 2015.

[18] Q. Mei, X. Shen, and C. Zhai. Automatic labeling of multinomial topic models. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 490–499. ACM, 2007.

[19] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at ICLR*, 2013.

[20] M. Sanderson and B. Croft. Deriving concept hierarchies from text. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 206–213, 1999.

[21] Y. Song, H. Wang, Z. Wang, H. Li, and W. Chen. Short text conceptualization using a probabilistic knowledgebase. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, pages 2330–2336, 2011.

[22] C. Wang, W. Chen, and Y. Wang. Scalable influence maximization for independent cascade model in large-scale social networks. *Data Mining and Knowledge Discovery*, 25(3):545–576, 2012.

[23] I. Witten and D. Milne. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *Proceedings of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy*, 2008.

[24] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning. Kea: Practical automatic keyphrase extraction. In *Proceedings of the Fourth ACM Conference on Digital Libraries*, pages 254–255. ACM, 1999.

[25] W. Wu, H. Li, H. Wang, and K. Q. Zhu. Probase: A probabilistic taxonomy for text understanding. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 481–492, 2012.

[26] X. Yin and S. Shah. Building taxonomy of web search intents for name entity queries. In *Proceedings of the 19th International Conference on World Wide Web*, pages 1001–1010, 2010.

[27] Z. Zhang, J. Iria, C. A. Brewster, and F. Ciravegna. A comparative evaluation of term recognition algorithms. *Proceedings of the Sixth International Conference on Language Resources and Evaluation*, 2008.