Automatic Construction and Ranking of Topical Keyphrases on Collections of Short Documents

Marina Danilevsky* Chi Wang* Nihit Desai* Xiang Ren* Jingyi Guo† Jiawei Han*

Abstract

We introduce a framework for topical keyphrase generation and ranking, based on the output of a topic model run on a collection of short documents. By shifting from the unigram-centric traditional methods of keyphrase extraction and ranking to a phrase-centric approach, we are able to directly compare and rank phrases of different lengths. Our method defines a function to rank topical keyphrases so that more highly ranked keyphrases are considered to be more representative phrases for that topic. We study the performance of our framework on multiple real world document collections, and also show that it is more scalable than comparable phrase-generating models.

1 Introduction

Keyphrases have traditionally been defined as terms or phrases which summarize the topics in a document [19]. Keyphrase extraction is an important step in many tasks, such as document summarization, clustering, and categorization [13]. More recently, the definition has been expanded to include the notion of topical keyphrases - groups of keyphrases which summarize the topics in a given document, or document collection [11]. Most existing work on keyphrase extraction identifies keyphrases from either individual documents or a collection of long documents [18, 11]. However, recently there has been some interest in working with short texts, such as a collection of microblogs [23, 22], to summarize the topics in the collection.

Most current approaches to topic construction yield ranked lists of unigrams to represent topics. However, it has long been known that unigrams account for only a small fraction of human-assigned index terms [19]. Furthermore, a person who us unfamiliar with the topic may not be able to easily view unigrams, and automatically combine them into 'true' phrases. For example, a person completely unfamiliar with the topic of Machine Learning would not be able to know that the unigram list {'support', 'vector', 'machine'} should actually be transformed into the phrase 'support vector machine.' Therefore, in order to construct high quality keyphrases for a given topic, it is important to provide n-gram keyphrases rather than unigram keywords.

learning
support vector machines
reinforcement learning
feature selection
conditional random fields
classification
decision trees
:

Table 1: The topic of Machine Learning, automatically constructed from titles of computer science papers published in DBLP.

On the other hand, it is inappropriate to discard all unigrams when approaching this task, or in fact to demonstrate a bias towards any particular phrase length. For instance, consider that the unigram 'classification' and the trigram 'support vector machines' are both high quality topical keyphrases for the machine learning topic in the domain of computer science. It is also not ideal to present separate ranked lists of topical phrases of each length, since when people are asked to characterize topics, they do not limit themselves to e.g. listing only bigrams, but rather provide a set of relevant phrases with no regard for phrase length. We should therefore also be able to perform integrated ranking of mixed-length phrases in a natural way.

Table 1 presents one example of a topic that our framework is able to automatically construct from a collection of the titles of computer science papers published in DBLP. The topic can be clearly interpreted as that of Machine Learning, and is represented by a list of high quality mixed-length phrases, of which the top ranked are shown. These sample results motivate the main content of our work. Our main contributions, motivated by these sample results, are:

• We present KERT (Keyphrase Extraction and Ranking by Topic), a framework for topical keyphrase generation and ranking. By altering the steps in the traditional methods of unsupervised keyphrase extraction and combining the techniques of topic modeling, statistical generative models and frequent pattern mining, we are able to directly compare

*Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA, {danilev1,chiwang1,nhdesai2,xren7,hanj}@illinois.edu

†Department of Computer Science, University of Massachussets Amherst, Amherst, MA 01003, USA, jingyi@cs.umass.edu

phrases of different lengths, resulting in a natural integrated ranking of mixed-length keyphrases.

• We define a function to rank topical keyphrases so that more highly ranked keyphrases are considered to be better representative phrases for that topic.

• We demonstrate the scalability of KERT on various datasets, as compared to other phrase-generating topic modeling approaches.

We demonstrate the effectiveness of our approach on two real world short document collections. The first is the DBLP dataset, a collection of titles of recently published Computer Science papers in the areas related to Databases, Data Mining, Information Retrieval, Machine Learning, and Natural Language Processing. These titles come from DBLP,[1] a bibliography website for computer science publications. The second is the arXiv dataset, a sample of titles of physics papers published within the last decade. This collection of titles comes from arXiv,[2] an online archive for electronic preprints of scientific papers. Finally, we demonstrate the significant improvement in performance offered by our framework as compared to other topical models which directly yield phrase outputs.

2 Related Work

The state-of-the-art approaches to unsupervised keyphrase extraction have generally been graph-based, unigram-centric ranking methods, which first extract unigrams from text, rank them, and finally combine them into keyphrases. TextRank [15] constructs keyphrases from the top ranked unigrams in a document collection. Topical PageRank [11] splits the documents into topics and creates keyphrases from top ranked topical unigrams. Some previous methods have used clustering techniques on word graphs for keyphrase extraction [12, 5], relying on external knowledge bases such as Wikipedia to calculate term importance and relatedness. Barker and Cornacchia [1] use natural language processing techniques to select noun phrases as keyphrases. Tomokiyo and Hurst [18] take a language modeling approach, requiring a document collection with known topics as input and training a language model to define their ranking criteria.

Unlike most of these methods, which extract keyphrases from documents, we aim to extract keyphrases from a corpus of short texts. Zhao et al [23] uses one example of short text - microblogging data from Twitter. We work with document titles for the purpose of easier evaluation, but our method can also be applied to microblogging data. Yan et al [22] uses an NNMF approach to learn the topics of short documents, but their concern is with the documents, rather than with a high quality representation of the topics found in the collection.

Topic modeling techniques such as PLSA (probabilistic latent semantic analysis) [7] and LDA (latent Dirichlet allocation) [3] take documents as input, model them as mixtures of different topics, and discover word distributions for each topic. Some previous work has developed topic modeling approaches to discover topical phrases comprised of consecutive words [20, 21, 10]. Our framework uses topic model output to perform initial word clustering into topics, and then extract and rank keyphrases. We do not restrict phrases to only word sequences explicitly found in the text, which gives our method an advantage when analyzing short documents.

Since we aim to transform a collection of short texts into sets of ranked keyphrases, the top-K keyphrases for each topic may also serve as that topic's labels. Therefore our work is tangentially related to automatic topic labeling [14].

3 Framework

A key aspect of our framework is that we do not follow the traditional unigram-centric approach of keyphrase extraction, where words are first extracted and ranked independently, and then combined to create phrases. Instead, we construct topical phrases immediately after clustering the unigrams. By shifting from a unigram-centric to a phrase-centric approach, we are able to extract topical keyphrases and implement a ranking function that can directly compare keyphrases of different lengths.

3.1 KERT Algorithm Overview When humans are asked to generate phrases that could represent a topic, the aspect of phrase length is seldom considered, except as a reasonable maximum for the considered topic (e.g. phrases in computer science generally do not consist of more than four words). Therefore, we should generate topical keyphrases in such a way as to be able to directly compare keyphrases of mixed lengths during the ranking step. We refer to this chracteristic as exhibiting the **comparability property**. For example, the keyphrases 'classification,' 'decision trees,' and 'support vector machines' should all be ranked highly in the integrated list of keyphrases for the Machine Learning topic, in spite of having different lengths.

Traditional probabilistic modeling approaches, such as language models or topic models do not have the comparability property. They can model the probability of seeing an n-gram given a topic, but the probabilities of n-grams with different lengths (unigrams, bigrams, etc) are not well comparable. These approaches simply find longer n-grams to have much smaller probability than shorter ones, because the probabilities of seeing every possible unigram sum up to 1, and so do the probabilities of seeing every possible bigram, trigram, etc. However, the total number of possible n-grams grows following a power law ($O(|V|^n)$), and ranking functions based on these traditional approaches invariably favor short n-grams. While previous work has used various heuristics to correct this bias during post-processing steps,

[1] http://www.dblp.org/
[2] http://arxiv.org

such as using a penalization term with respect to the phrase length [18, 23], our approach is cleaner and more principled.

The key to KERT exhibiting the comparability property is representing the random event $e_t(p) =$'seeing a phrase p in a random document with topic t'. With this definition, the events of seeing n-grams of various lengths in different documents are no longer mutually exclusive, and therefore the probabilities no longer need to sum up to 1. In order to do this, we must therefore first discover the phrase 'p' and then calculate $e_t(p) =$. There are two ways to discover keyphrases: either extract them from the text (as sequences of words which actually occur in the text), or to automatically construct them [6], an approach which is regarded as both more intelligent and more difficult [8, 16]. We must therefore clearly define our process of keyphrase discovery.

Like [23], our framework focuses on constructing topics from a collection of short documents. There are many cases where the full text of a document collection is not available, or is too noisy, for the desired task of topic discovery from the collection. Furthermore, we focus on documents which are information-rich, meaning that most of the document content is informative, not noisy, with the usual exception of function words. The documents may also be a mix of multiple topics, in spite of their short length. In this work we primarily evaluate our performance on two collections of scientific paper titles, which fit this criteria well. While our framework could technically be applied to a collection of noisy short documents such as tweets, it would require at least a transformation of the noisy documents into information-rich documents in order to perform well, and this is out of the scope of this work

When working with a short document, extracting phrases directly from the text is quite limiting as this approach is too sensitive to the caprices of various writing styles. For instance, consider that two computer science papers titles, one containing 'mining frequent patterns' and the other containing 'frequent pattern mining,' are clearly discussing the same topic, and should be treated as such. A keyphrase may also be separated by other words: a document containing 'mining top-k frequent closed patterns also contains the topic of frequent pattern mining, in addition to incorporating the secondary topics of top-k frequent patterns, and closed patterns. Therefore, within the context of this work we will define a *phrase* to be an order-free set of words appearing in the same document and must construct our phrases. As we work with information-rich documents, we can assume that most of the words in the document are informative and not noisy, resulting in the construction of informative phrases.

We are interested particularly in constructing *topical keyphrases*, which are phrases whose words all belong to the same topic. Therefore, before we are able to construct these topical keyphrases, we must first assign every word in every document to a topic. Algorithms that solve traditional topic modeling problems such as LDA [3] are particularly well suited to this task precisely because they provide these topic assigments, in addition to estimating the topic distributions. They are also widely used and well-understood, and can therefore be easily incorporated into our framework for the purpose of assigning words to topics.

Moving backwards through the above overview, our steps for topical keyphrase generation and ranking are therefore as follows:

Step 1. Given a collection of short texts, cluster words in the dataset into topics, using an appropriate unigram-generating topic model method.

Step 2. Construct candidate keyphrases for each topic according to the word topic assignments specified by the topic model output.

Step 3. Rank the keyphrases in each topic $t \in 1, \ldots, k$ according to a well-defined ranking function.

In the following sections we detail the methodology for each of these steps.

3.2 Clustering Words using Topic Modeling The first step of KERT is to cluster every unigram in every document to a topic. LDA [3] and its extensions have been shown to be effective for modeling textual documents. In this paper we specifically work with one modification of the LDA [3] model, bLDA, which includes an additional background topic $z = 0$. This allows us to relax the assumption, previously introduced in Section 3.1 that every word is informative, and allow to relegate uninformative unigrams to a background topic, which effectively removes them from further consideration by KERT. At the same time, using the relatively simple model of bLDA rather than a more specialized model allows us to focus on the performance of the mining and ranking steps of KERT, which comprise our main contributions, rather than the unigram clustering step. However, our framework can easily incorporate the output of many topic models, such as PLSA [7], LDA [3], PY [17] and SAGE [4].

Formally, bLDA models each document by a mixture of the foreground topics $z = 1, \ldots, k$ and the background topic. For each word in a document, we decide if it belongs to the background topic or one of the foreground topics, and then choose the word from the appropriate distribution.

Formally, let ϕ^t denote the word distribution for topic $t = 0, \ldots, k$. Let θ^d be the topic distribution of document d. Let λ denote a Bernoulli distribution that chooses between the background topic and foreground topics. The generative process is as follows:

1. Draw $\phi^t \sim Dir(\beta)$, for $t = 0, \ldots, k$.
2. For each document $d \in D$,
   (a) draw $\theta^d \sim Dir(\alpha)$.

(b) for each word position i in d

i. draw $y_{d,i} \sim Bernoulli(\lambda)$

ii. if $y_{d,i} = 0$, draw $w_{d,i} \sim Multi(\phi^0)$, otherwise

A. draw topic $z \sim \theta^d$

B. draw $w_{d,i} \sim Multi(\phi^z)$

where α and β are Dirichlet prior parameters for θ and ϕ.

We use a collapsed Gibbs sampling method for model inference. We iteratively sample the topic assignment $z_{d,i}$ for every word occurrence $w_{d,i}$ in each document d until convergence. In traditional topic modeling tasks, the sampled topic assignments are mainly used to estimate the topic distribution ϕ^z. In our case, we are more interested in the topic assignments for the words in each document, because these values are the foundation of our topical keyphrase generation step, as described in the next section. We use the *maximum a posteriori* (MAP) principle to label each word occurrence: $z_{d,i} = \arg\max_{z_{d,i}=0,\dots,k} P(z_{d,i}|W)$.

3.3 Candidate Keyphrase Generation After the clustering step described in Section 3.2 is completed, each word $w_{d,i}$ in each document d has a topic label $z_{d,i} \in \{0, \dots, k\}$. If a set of words in a document d are assigned a common foreground topic $t > 0$, these words may comprise a topical phrase in t (e.g. {'frequent','pattern','mining'} in the topic of 'Data Mining'). If this set occurs in many documents with the topic label t, it is likely to be a good candidate keyphrase for that topic. As described in Section 3.1, within the context of this work, we define a topical keyphrase to be an order-free set of words appearing in the same document and belonging to the same topic. However, the frequent pattern mining approach to generating candidate topical keyphrases described in this section may be easily adapted to a stricter definition, such as requiring consistent word order, or allowing a phrase to only consist of consecutive words (no gaps).

For each topic t, we construct a topic-t word set $p_d^t = \{w_{d,i}|z_{d,i} = t\}$ consisting of the words with the topic label t for each document d. We unite all the topic-t word sets into the topic-t set $D_t = \{d|p_d^t \neq \emptyset\}$. We may then mine frequent word sets from $p_{D_t}^t = \{p_d^t|d \in D_t\}$ using any efficient pattern mining algorithm, such as FP-growth [6]. We require a good topical keyphrase to have enough topical support $f_t(p) > \mu$ in order to filter out some coincidental co-occurrences (where $f_t(p)$ denotes the frequency of the word set p in topic t).

We thus define a *candidate topical keyphrase* for topic t to be a set of words $p = \{w_1 \dots w_n\}$ which are simultaneously labeled with topic t in at least μ documents, as discovered by the frequent pattern mining step. We then rank the candidate topical keyphrases within each topic.

3.4 Ranking Keyhrases for Topic Representation

Our goal is to construct a ranking function to evaluate the quality of topical keyphrases. Such a ranking function must successfully represent human intuition for judging what constitutes a high quality topical keyphrase. We now present the characteristics that such a ranking function should have, together with the criteria that represent these characteristics (constructing and ranking keyphrases for Computer Science topics as a running example).

• **Coverage.** Coverage, which may be referred to by other names such as frequency, or importance, is the most basic criteria, required by every ranking function that tackles this same problem. *Example: 'information retrieval' has better coverage than 'cross-language information retrieval' in the Information Retrieval topic.* A keyphrase that is not frequent in a topic should never be highly ranked as being representative of that topic, regardless of its length, or its value according to any other criteria. This further suggests that the combined ranking function should be constructed in such a way that a topical keyphrase with low coverage is guaranteed to have a lower rank.

• **Purity.** A keyphrase is pure in a topic if it is only frequent in documents belonging to that topic and not frequent in documents within other topics. *Example: 'query processing' is more pure than 'query' in the Database topic.* Like Coverage, some version of this criterion is also present in most ranking functions.

• **Phraseness.** Since KERT constructs phrases as word sets, this criterion evaluates the candidate topical keyphrases discovered in the previous step, somewhat following the intuition in [18]. A group of words should be combined together as a keyphrase if they co-occur significantly more often than the expected chance co-occurrence frequency, given that each term in the phrase occurs independently. *Example: 'active learning' is a better phrase than 'learning classification' in the Machine Learning topic, since the latter simply combines two unigrams which are frequent in the topic.* It is of course possible for both a unigram, and a word set containing that unigram to both be reasonable topical keyphrases, for example as with 'learning' and 'reinforcement learning' for the topic of Machine Learning.

• **Completeness.** A keyphrase is not complete if it is a subset of a longer keyphrase, in the sense that it rarely occurs in a document without the presence of the longer keyphrase. *Example: 'support vector machines' is a complete phrase, whereas 'vector machines' is not because 'vector machines' is almost always accompanied by 'support'.*

The combination of Phraseness and Completeness take the place of other criteria that have been used by approaches which perform phrase extraction rather than phrase construction, such as part-of-speech tagging [12, 11]. The Phraseness and Completeteness ranking criteria together embody the characteristic of a phrase being 'understandable' to people

[12]. We also do not go the route of evaluating this characteristic by calculating semantic relatedness between words, which has been used in other approaches using knowledge bases such as Wikipedia [12]. This is because we wish to avoid relying on external information which may possibly be unavailable, or not relevant for a particular (e.g., domain-specific) dataset.

The step of generating candidate topical keyphrases allows us to represent the random event $e_t(p)$ = 'seeing a phrase p in a random document with topic t.' The KERT algorithm therefore exhibits the *comparability property* and is able to directly compare phrases of any length, according to the ranking criteria. Formally, the probability of $e_t(p)$ is defined to be $P(e_t(p)) = \frac{f_t(p)}{|D_t|}$. In the subsequent sections we define our measures representing the 4 criteria of coverage, purity, phraseness, and completeness using quantities related to this probability. Because we work with relatively short documents, which we assume to be uniformly informative, we do not consider criteria based on document structure, or the phrase location in the document.

3.4.1 Coverage A representative keyphrase for a topic should cover many documents within that topic. For example, 'information retrieval' has better coverage than 'cross-language information retrieval' in the topic of Information Retrieval. We directly quantify the coverage measure of a phrase as the probability of seeing a phrase in a random topic-t word set $p_d^t \in D_t$:

$$\pi_t^{cov}(p) = P(e_t(p)) = \frac{f_t(p)}{|D_t|} \tag{3.1}$$

3.4.2 Purity A phrase is pure in topic t if it is only frequent in documents about topic t and not frequent in documents about other topics. For example, 'query processing' is a more pure keyphrase than 'query' in the Databases topic.

We measure the purity of a keyphrase by comparing the probability of seeing a phrase in the topic-t collection of word sets and the probability of seeing it in any other topic-t' collection ($t' = 0, 1, \ldots, k$, $t' \neq t$). A reference collection $D_{t,t'} = D_t \cup D_{t'}$ is a mix of topic-t and topic-t' documents. If there exists a topic t' such that the probability of $e_{t,t'}(p)$ ='seeing a phrase p in a reference collection $D_{t,t'}$' is similar, or even larger than the probability of seeing p in D_t, the phrase p indicates confusion about topic t and t'. The purity of a keyphrase compares the probability of seeing it in the topic-t collection and the maximal probability of seeing it in any topic-t' reference collection:

$$\begin{aligned} \pi_t^{pur}(p) &= \log \frac{P(e_t(p))}{\max_{t' \neq t} P(e_{t'}(p))} \\ &= \log \frac{f_t(p)}{|D_t|} - \log \max_{t' \neq t} \frac{f_t(p) + f_{t'}(p)}{|D_{t'}|} \end{aligned} \tag{3.2}$$

3.4.3 Phraseness A group of words should be grouped into a phrase if they co-occur significantly more frequently than the expected co-occurrence frequency given that each word in the phrase occurs independently. For example, while 'active learning' is a good keyphrase in the Machine Learning topics, 'learning classification' is not, since the latter two words co-occur only because both of them are popular in the topic.

We therefore compare the probability of seeing a phrase $p = \{w_1 \ldots w_n\}$ and seeing the n words $w_1 \ldots w_n$ independently in topic-t documents:

$$\begin{aligned} \pi_t^{phr}(p) &= \log \frac{P(e_t(p))}{\prod_{w \in p} P(e_t(w))} \\ &= \log \frac{f_t(p)}{|D_t|} - \sum_{w \in p} \log \frac{f_t(w)}{|D_t|} \end{aligned} \tag{3.3}$$

3.4.4 Completeness A phrase p is not complete if a longer phrase p' that contains p usually co-occurs with p. For example, while 'support vector machines' is a complete phrase, 'vector machines' is not, as 'support' nearly always accompanies 'vector machines.'

We thus measure the completeness of a phrase p by examining the conditional probability of observing p' given p in a topic-t document:

$$\begin{aligned} \pi_t^{com}(p) &= 1 - \max_{p' \supsetneq p} P(e_t(p')|e_t(p)) \\ &= 1 - \max_{w} P(e_t(p \cup \{w\})|e_t(p)) \\ &= 1 - \frac{\max_w f_t(p \cup \{w\})}{f_t(p)} \end{aligned} \tag{3.4}$$

3.4.5 Keyphrase Topical Quality Ranking Function We can combine these 4 measures into a comprehensive function representing the quality of a topical keyphrase by viewing them within an information theoretic framework, and representing the relationships between coverage, phraseness, and purity using two pointwise Kullback-Leibler(KL)-divergence metrics.

Pointwise KL-divergence is a distance measure between two probabilities that takes the absolute probability into consideration, and is more robust than pointwise mutual information when the relative difference between probabilities need to be supported by sufficiently high absolute probability.[3] The product of coverage and purity, $\pi_t^{cov}(p)\pi_t^{pur}(p) = P(e_t(p)) \log \frac{P(e_t(p))}{P(e_{t'}(p))}$ is equal to the pointwise KL-divergence between the probabilities of $e_t(p)$

[3]Note that Tomokiyo *et al.* [18] uses KL-divergence metrics to derive their ranking function as well, but they require an annotated foreground corpus and a background corpus as input. Furthermore, they consider language models only for consecutive ngrams and do not exhibit the comparability property. So their ranking function behaves quite differently from ours.

and $e_{t'}(p)$. Likewise, the product of coverage and phraseness, $\pi_t^{cov}(p)\pi_t^{phr}(p)$ is equivalent to the pointwise KL-divergence between the probability of $e_t(p)$ under different independence assumptions. We combine these two metrics with a weighted summation, and implement the completeness criterion as a condition to filter out incomplete phrases: (3.5)

$$Rank_t(p) = \begin{cases} 0 & \pi_t^{com} \le \gamma \\ \pi_t^{cov}[(1-\omega)\pi_t^{pur} + \omega\pi_t^{phr}](p) & \text{o.w.} \end{cases}$$

Here, $\gamma \in [0,1]$ controls how aggressively we prune incomplete phrases. $\gamma = 0$ corresponds to ignoring the completeness criterion and retaining all *closed* phrases, where no supersets have the same topical support. As γ approaches 1, more phrases will be filtered and eventually only *maximal* phrases (no supersets are sufficiently frequent) will remain. The other three criteria rank keyphrases that pass the completeness filter.

Although $Rank_t(p)$ is a combination of two metrics, the coverage criterion is a factor in both. Thus, when $P(e_t(p))$ is small, phrase p has low support, and thus the estimates of purity and phraseness will be unreliable and their role should be limited. When $P(e_t(p))$ is large, phrase p has high support, and magnifies the cumulative effect (positive or negative) of the purity and phraseness criteria.

The relative weights of the purity and phraseness criteria are controlled by $\omega \in [0,1]$. Both measures are log ratios on comparable scales, and can thus be balanced by a weighted summation. As ω increases, we expect more topic-independent, but common phrases to be ranked higher.

For each topic t, we construct a topic-t word set $p_d^t = \{w_{d,i}|z_{d,i} = t\}$ consisting of the words with the topic label t for each document d. We unite all the topic-t word sets into the topic-t set $D_t = \{d|p_d^t \neq \emptyset\}$. We may then mine frequent word sets from $p_{D_t}^t = \{p_d^t|d \in D_t\}$

4 Experiments

4.1 Judging Topical Keyphrase Quality In our first set of experiments, we use the DBLP dataset - a collection of paper titles in Computer Science - to evaluate the ability of our method to construct topical keyphrases that appear to be high quality to human judges, via a user study. The titles were minimally pre-processed by removing all stopwords, resulting in 33,313 documents consisting of 18,598 unique words. We first describe the methods we used for comparison, and then present a sample of the keyphrases actually generated by these methods and encountered by participants in the user study. We then explain the details of our user study, and present quantitative results.

4.1.1 Ranking Methods for Comparison We use bLDA, introduced in Section 3.2, to perform the word clustering step and create input for all the methods that we compare. We resort to a Newton-Raphson iteration method [16] to learn the hyperparameters, and empirically set $\lambda = 0.1$, which yields coherent results for our dataset.[4]

To evaluate the performance of KERT, we implemented several variations of the function, as well as two baseline functions. The baselines come from Zhao et al [23], who focus on topical keyphrase extraction in microblogs, but claim that their method can be used for other text collections. We implement their two best performing methods: kpRelInt* and kpRel.[5] We also construct variations of KERT where the keyphrase extraction steps are identical, but each of the four ranking criteria is ignored in turn. We refer to these versions as KERT$_{\text{-cov}}$, KERT$_{\text{-pur}}$, KERT$_{\text{-phr}}$, and KERT$_{\text{-com}}$.

These variations nicely represent the possible settings for the parameters γ and ω, which are described in Section 3.4.5. In KERT we set $\gamma = 0.5$ and $\omega = 0.5$. KERT$_{\text{-com}}$ sets $\gamma = 0$ to demonstrate what happens when we retain all *closed* phrases. As γ approaches 1, more phrases will be filtered but a very small number of *maximal* phrases (no supersets are frequent) will not be. KERT$_{\text{-phr}}$ sets $\omega = 0$ and KERT$_{\text{-pur}}$ sets $\omega = 1$, to demonstrate the effect of ignoring phraseness for the sake of maximizing purity, and ignoring purity to optimize for phraseness, respectively. For Step 2 of every KERT variation, we set $\mu = 5$.

4.1.2 Qualitative Results Table 2 shows the top 10 ranked topical keyphrases generated by each method for the topic of Machine Learning. kpRel and kpRelInt* yield very similar results, both clearly favoring unigrams. However, kpRel also ranks several keyphrases highly which are not very meaningful, such as 'learning classification' and 'selection learning.' Removing coverage from our ranking function yields the worst results, confirming the intuition that a high quality keyphrase must at minimum have good coverage. Without purity, the function favors bigrams and trigrams that all seem to be meaningful, although several high quality unigrams such as 'learning' and 'classification' no longer appear. Removing phraseness, in contrast, yields meaningful unigrams but very few bigrams, and looks quite similar to the kpRelInt* baseline. Finally, without completeness, phrases such as 'support vector' and 'vector machines' are improperly highly ranked, as both are sub-phrases of the high quality trigram 'support vector machines.'

4.1.3 User Study and Quantitative Results To quantitatively measure keyphrase quality, we invited people to judge

[4]The learned $\alpha = 1.0$ is smaller than the typical setting due to the nature of our short text, and $\beta = 0.07$ is larger because in our dataset, different topics often share the same words.

[5]Their main ranking function kpRelInt considers the heuristics of phrase interestingness and relevance. As their interestingness measure is represented by re-Tweets, a concept that is not appropriate to our dataset, we reimplement the interestingness measure to be the relative frequency of the phrase in the dataset, and we therefore refer to our reimplementation as kpRelInt*. kpRel considers only the relevance heuristic.

Method	Top 10 Topical Keyphrases
kpRelInt*	learning / classification / selection / models / algorithm / feature / decision / bayesian / trees / problem
kpRel	learning / classification / learning classification / selection / selection learning / feature / decision / bayesian / feature learning / trees
KERT$_{\textbf{-cov}}$	effective / text / probabilistic / identification / mapping / task / planning / set / subspace / online
KERT$_{\textbf{-pur}}$	support vector machines / feature selection / reinforcement learning / conditional random fields / constraint satisfaction / decision trees / dimensionality reduction / constraint satisfaction problems / matrix factorization / hidden markov models
KERT$_{\textbf{-phr}}$	learning / classification / selection / feature / decision / bayesian / trees / problem / reinforcement learning / constraint
KERT$_{\textbf{-com}}$	learning / support vector machines / support vector / reinforcement learning / feature selection / conditional random fields / vector machines/ classification / support machines / decision trees
KERT	learning / support vector machines / reinforcement learning / feature selection / conditional random fields / classification / decision trees / constraint satisfaction / dimensionality reduction / matrix factorization

Table 2: Top 10 ranked keyphrases in the Machine Learning topic by different methods.

the generated topical keyphrases generated by the different methods. Since the DBLP dataset generates topics in Computer Science, we recruited 10 Computer Science graduate students - who could thus be considered to be very knowledgeable judges in this domain - for a user study.

We generated five topics from the DBLP dataset. As can be seen from the bLDA output in Section 3.2, Topic 5 is very mixed and difficult to interpret as representing just one research area. We discarded it, leaving four topics which are clearly interpretable as Machine Learning, Databases, Data Mining, and Information Retrieval. For each of the four topics, we retrieved the top 20 ranked keyphrases by each method. These keyphrases were gathered together per topic and presented in random order. Users were asked to evaluate the quality of each keyphrase on a 5 point Likert scale.

To measure the performance of each method given the user study results, we adapted the **nKQM@K measure** (normalized keyphrase quality measure for top-K phrases) from [23], which is itself a version of the *nDCG* metric from information retrieval [8]. We define nKQM@K for a method *M* using the top-K generated keyphrases:

$$nKQM@K = \frac{1}{T}\sum_{t=1}^{T}\frac{\sum_{j=1}^{K}\frac{score_{aw}(M_{t,j})}{log_2(j+1)}}{IdealScore_K}$$

Here T is the number of topics, and $M_{t,j}$ refers to the j^{th} keyphrase generated by method M for topic t. Unlike in [23], we have more than 2 judges, so we define $score_{aw}$ as the *agreement-weighted* average score for the $M_{t,j}$ keyphrase, which is the mean of the judges' score multiplied by the weighted Cohen's κ. This gives a higher value to a keyphrase with scores of (3,3,3) than to one with scores of (1,3,5), though the average score is identical. Finally, $IdealScore_K$ is calculated using the scores of the top K keyphrases out of all judged keyphrases.

Method	nKQM$_{@5}$	nKQM$_{@10}$	nKQM$_{@20}$
KERT$_{\textbf{-cov}}$	0.2605	0.2701	0.2448
kpRelInt*	0.3521	0.3730	0.3288
KERT$_{\textbf{-phr}}$	0.3632	0.3616	0.3278
kpRel	0.3892	0.4030	0.3767
KERT$_{\textbf{-com}}$	**0.5124**	**0.4932**	**0.4338**
KERT	**0.5198**	**0.4962**	**0.4393**
KERT$_{\textbf{-pur}}$	**0.5832**	**0.5642**	**0.5144**

Table 3: nKQM@K (methods ordered by performance)

Table 3 compares the performance across different methods.[6] The top performances are clearly variations of KERT with different parameter settings. As expected, KERT$_{\text{-cov}}$ exhibits the worst performance. The baselines perform slightly better, and it is interesting to note that kpRel, which is smoothed purity, performs better than kpRelInt*, and even slightly better than KERT$_{\text{-phr}}$. This is because kpRelInt* adds in a measure of *overall* keyphrase coverage in the entire collection, which hurts rather than helps for this task. Removing completeness appears to have a very small negative effect, and we hypothesize this is because high-ranked incomplete keyphrases are relatively rare, though very obvious when they do occur (e.g. 'vector machines'). KERT$_{\text{-pur}}$ performs the best - which may reflect human bias towards longer phrases - with an improvement of at least 50% over the kpRelInt* baseline for all reported values of K.

[6]Although we cannot directly evaluate which differences between the nKQM values are statistically significant, we examined the differences between the distributions of mean judge scores. We found, for example, that the preference for phrases generated by KERT$_{\text{-pur}}$ was statistically significant, whereas the difference between KERT and KERT$_{\text{-com}}$ was not.

4.2 Maximizing Mutual Information In order to perform a quantitative evaluation, we use a dataset which, unlike the DBLP collection, has been labeled. In the arXiv dataset, each physics paper title is labeled by its authors as belonging to the subfield of Optics, Fluid Dynamics, Atomic Physics, Instrumentation and Detectors, or Plasma Physics. We minimally pre-processed the titles by removing all stopwords, resulting in 9,722 titles evenly sampled from the specified 5 physics subfields, and consisting of 9,648 unique words. Since the titles are labeled, we can explore which method maximizes the mutual information between phrase-represented topics and titles. As the collection has 5 categories, we set T=5.

For each method, we do multiple runs for various values of K (the number of top-ranked phrases per topic considered), and calculate the mutual information MI_K for that method as a function of K. To calculate MI_K, we label each of the top K phrases in every topic with the topic in which it is ranked highest. We then check each paper title to see if it contains any of these top phrases. If so, we update the number of events "seeing a topic t and category c" for $t = 1 \dots T$, with the averaged count for all those labeled phrases contained in the title; otherwise we update the number of events "seeing a topic t and category c" for $t = 1 \dots T$ uniformly, where c is the Primary Category label for the paper title in consideration. Finally, we compute mutual information at K:

$$MI_K = \sum_{t,c} p(t,c)\ log_2 \frac{p(t,c)}{p(t)p(c)}$$

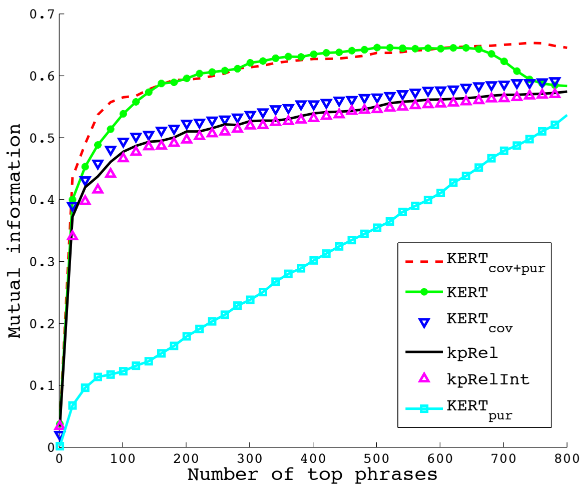

Figure 1: Mutual Information at K (MI_K) for various K. Methods in legend are ordered by performance, high to low.

We compare the baselines (kpRelInt* and kpRel), KERT, and variations of KERT where only coverage ($KERT_{cov}$), only purity ($KERT_{pur}$), and only coverage and purity ($KERT_{cov+pur}$) are used in the ranking function. We use the output of bLDA with the same parameter settings as specified in Section 4.1.1. Figure 1 shows MI_K for each method for a range of K.

It is clear that for MI_K, coverage is more important than purity, since $KERT_{pur}$ is by far the worst performer. Both baselines perform nearly as well as $KERT_{cov}$, and all are comfortably beaten by $KERT_{cov+pur}$ ($> 20\%$ improvement for K between 100 and 600), which uses our coverage and purity measure. It is interesting and important to note that adding in the phraseness and completeness measures yields no improvement in MI_K. However, the experiments with the DBLP dataset demonstrate that these measures - particularly phraseness - are very helpful in the eyes of expert human judges. In contrast, while MI_K is definitely improved with the addition of the purity measure, people seem to prefer that this metric not affect the keyphrase ranking. Although we outperform other approaches in both evaluations, these observations show interesting differences between theory-based and human-centric evaluation metrics.

5 Scalability and Performance

Although we mention several unigram-generating topic model choices in Section 3.2, there are also topic models that directly yield phrase outputs, such as Turbo Topics [2], frequent pattern enriched topic model [9], TNG [21], Bigram Model [20], and PDLDA [10]). However, the performance of these models is significantly worse than KERT, especially on larger document collections, as we show in this section.

We pick two well-known phrase-generating topic model approaches, PDLDA and TurboTopics, and compare their performance to KERT on four datasets. Together with the DBLP dataset described in Section 4.1 and the arXiv dataset described in Section 4.2, we consider two more collections of short documents: ACL_{title}, comprising the titles of 13,638 papers from the ACL Anthology Network Dataset,[7], and $DBLP_{full}$, a full collection of the titles available in DBLP, consisting of over 1.8 million paper titles.[8]

	$DBLP_{full}$	**DBLP**	**arXiv**	**ACL_{title}**
# documents	*1,832,469*	*33,313*	*9,722*	*13,638*
Turbo Topics	27d	4h 58m	10m 58s	9m 22s
PDLDA	1d 5h 17m 13s	15m 32s	4 m35s	12m 17s
KERT*	**1h 46m 1s**	**27.57s**	**13.1s**	**23.3s**

*For KERT we set γ=10 for $DBLP_{full}$; we set γ=5 for DBLP, arXiv, and ACL_{title}

Table 4: Comparing runtimes of KERT, PDLDA and Turbo Topics.

Table 4 reports the runtime for generating topical models on these document collections, as well as the size of each

[7] http://clair.eecs.umich.edu/aan/index.php

[8] This DBLP snapshot was constructed on Sept 29, 2012.

collection, comparing the performances of KERT, PDLDA, and Turbo Topics. KERT clearly outperforms both PDLDA and Turbo Topics on every dataset. On the three smaller collections of short texts (DBLP, arXiv, and $\text{ACL}_{\text{title}}$), KERT takes less than half a minute, while PDLDA takes up to 15 minutes, and Turbo Topics takes as long as five hours (on the DBLP dataset). The runtime of Turbo Topics quickly becomes unacceptable, requiring nearly a month to run on $\text{DBLP}_{\text{full}}$. PDLDA shows a significant improvement over Turbo Topics as the datasets increase in size, taking a day where Turbo Topics took a month, and two hours where Turbo Topics took two days. However, KERT's improvement over Turbo Topics is nearly as great, requiring 8 minutes to PDLDA's 2 days, and finishing the full DBLP dataset in under two hours. Therefore, the KERT framework demonstrates a significant practical advantage over other comparable phrase-generating approaches.

6 Conclusion

In this work we introduce a framework for topical keyphrase generation and ranking, building on the output of a topic model run on a collection of short documents. Unlike existing techniques, our phrase-centric approach is able to construct candidate topical keyphrases in such a way as to allow our ranking function to directly compare the quality of keyphrases of different lengths. Our method yields high quality topical keyphrases, with over 50% improvement over a baseline method according to human judgement and over 20% improvement according to mutual information. Finally, we show that our method performs significantly better than comparable phrase-generating topic modeling approaches, especially as the collection size scales up.

In the future we would like to further explore why human judgement appears to be consistently biased against the purity criteria, in contrast to quantitative metrics such as mutual information. We are also interested in analyzing a wider variety of datasets, and formally extending our approach to analyzing longer texts.

7 Acknowledgments

This work was supported in part by the U.S. National Science Foundation grants IIS–0905215, U.S. Army Research Laboratory under Cooperative Agreement No. W911NF–09–2–0053 (NS-CTA). Chi Wang was supported by a Microsoft Research PhD Fellowship. Marina Danilevsky was supported by a National Science Foundation Graduate Research Fellowship grant NSF DGE 07-15088. We are grateful to the authors of PDLDA for sharing their implementation with us.

References

[1] K. Barker and N. Cornacchia. Using noun phrase heads to extract document keyphrases. In *Canadian AI*, 2000.
[2] D. M. Blei and J. D. Lafferty. Visualizing topics with multi-word expressions. *arXiv preprint arXiv:0907.1013*, 2009.
[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3, 2003.
[4] J. Eisenstein, A. Ahmed, and E. P. Xing. Sparse additive generative models of text. In *ICML*, 2011.
[5] M. Grineva, M. Grinev, and D. Lizorkin. Extracting key terms from noisy and multitheme documents. In *WWW*, 2009.
[6] J. Han, J. Pei, Y. Yin, and R. Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Min. Knowl. Discov.*, 8(1):53–87, Jan. 2004.
[7] T. Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Mach. Learn.*, 42(1-2):177–196, Jan. 2001.
[8] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, Oct. 2002.
[9] H. D. Kim, D. H. Park, Y. Lu, and C. Zhai. Enriching text representation with frequent pattern mining for probabilistic topic modeling. *Proc. Am. Soc. for Inf. Sci. and Tech.*, 49(1):1–10, 2012.
[10] R. V. Lindsey, W. P. Headden, III, and M. J. Stipicevic. A phrase-discovering topic model using hierarchical pitman-yor processes. In *EMNLP-CoNLL '12*, 2012.
[11] Z. Liu, W. Huang, Y. Zheng, and M. Sun. Automatic keyphrase extraction via topic decomposition. In *EMNLP*, 2010.
[12] Z. Liu, P. Li, Y. Zheng, and M. Sun. Clustering to find exemplar terms for keyphrase extraction. In *EMNLP*, 2009.
[13] C. D. Manning and H. Schütze. *Foundations of statistical natural language processing*. MIT Press, Cambridge, MA, USA, 1999.
[14] Q. Mei, X. Shen, and C. Zhai. Automatic labeling of multinomial topic models. In *KDD*, 2007.
[15] R. Mihalcea and P. Tarau. Textrank: Bringing order into texts. In *EMNLP*, 2004.
[16] T. P. Minka. Estimating a dirichlet distribution, 2000.
[17] I. Sato and H. Nakagawa. Topic models with power-law using pitman-yor process. In *KDD*, 2010.
[18] T. Tomokiyo and M. Hurst. A language model approach to keyphrase extraction. In *Proc. ACL 2003 Workshop on Multiword expressions*, 2003.
[19] P. D. Turney. Learning algorithms for keyphrase extraction. *Inf. Retr.*, 2(4):303–336, May 2000.
[20] H. M. Wallach. Topic modeling: beyond bag-of-words. In *ICML*, 2006.
[21] X. Wang, A. McCallum, and X. Wei. Topical n-grams: Phrase and topic discovery, with an application to information retrieval. In *ICDM*, 2007.
[22] X. Yan, J. Guo, S. Liu, X. Cheng, and Y. Wang. Learning topics in short texts by non-negative matrix factorization on term correlation matrix. In *SDM*, 2013.
[23] W. X. Zhao, J. Jiang, J. He, Y. Song, P. Achananuparp, E.-P. Lim, and X. Li. Topical keyphrase extraction from twitter. In *HLT*, 2011.