Extracting Discriminative Keyphrases with Learned Semantic Hierarchies

Yunli Wang	Yong Jin	Xiaodan Zhu	Cyril Goutte
NRC Canada	U. New Brunswick	NRC Canada	NRC Canada
Scientific Data Mining	CS Faculty	Text Analytics	Multilingual Text Processing
Ottawa ON, Canada	Fredericton NB, Canada	Ottawa ON, Canada	Ottawa ON, Canada
Yunli.Wang@nrc.ca	Yong.Jin@unb.ca	Xiaodan.Zhu@nrc.ca	Cyril.Goutte@nrc.ca

Abstract

The goal of keyphrase extraction is to automatically identify the most salient phrases from documents. The technique has a wide range of applications such as rendering a quick glimpse of a document, or extracting key content for further use. While previous work often assumes keyphrases are a *static* property of a given documents, in many applications, the appropriate set of *keyphrases* that should be extracted depends on the set of *documents* that are being considered together. In particular, good keyphrases should not only accurately describe the content of a document, but also reveal what discriminates it from the other documents. In this paper, we study this problem of extracting *discriminative* keyphrases. In particularly, we propose to use the hierarchical semantic structure between candidate keyphrases to promote keyphrases that have the right level of specificity to clearly distinguish the target document from others. We show that such knowledge can be used to construct better discriminative keyphrase extraction systems that do not assume a static, fixed set of keyphrases for a document. We show how this helps identify key expertise of authors from their papers, as well as competencies covered by online courses within different domains.

1 Introduction

The purpose of keyphrase extraction is to automatically identify the most salient phrases from documents. Keyphrases (of which keywords are a special case) are widely used for providing a quick glimpse of various types of documents, such as news, technical documents, etc. Automatically extracting the relevant keyphrases therefore has a wide range of applications and accordingly has attracted much attention from the scientific community.

Previous work, however, often assumes that keyphrases are a *static* property of documents, that is, a given document would always produce a fixed set of keyphrases. Many approaches were developped for that purpose. For example, the Keyphrase Extraction Algorithm, or KEA (Witten et al., 1998), uses a supervised learning method (Naïve Bayes) to predict keyphrases based on their lexical features. Turney (2000) developed a genetic algorithm (GenEx) to extract keyphrases, and showed that this outperformed the well-known C4.5 algorithm. More recent work on supervised keyphrase extraction used, e.g., a combination of lexical and syntactic features (Hulth, 2003) or other statistical classifiers such as support vector machine (SVM) (Zhang et al., 2006) or conditional random fields (CRF) (Zhang et al., 2008). Unsupervised methods were also proposed, based on a graph-based ranking model (Mihalcea and Tarau, 2004), or using co-occurrences (Matsuo and Ishizuka, 2004), enriched with WordNet (Martinez-Romo et al., 2016). Unsupervised keyphrase extraction was also applied to shorter texts from twitter, using multiple random walks to topic context (Zhao et al., 2011) or unsupervised feature extraction (Marujo et al., 2015). The use of hierarchical information to extract keyphrases was explored in (Smatana and Butka, 2016; Berend, 2016). Although these supervised and unsupervised methods achieve improved performance, little work has been done to generate discriminative keyphrases based on other documents in the group.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 932–942, Osaka, Japan, December 11-17 2016.

In many applications, the appropriate set of keyphrases that should be generated depends on the context in which the document is considered, and particularly the *group* of documents being considered. For example, the keyphrases that represent the competencies of a researcher or a job hunter could depend on the groups of researchers or resumes that are being considered. Similarly, the keyphrases describing an online course depend on the set of courses under consideration: *Machine Learning* may be the appropriate descriptive keyphrase to describe a course within a set of Computer Science courses, but it is not very useful to a student considering a set of Machine Learning courses. Note that we differentiate the *collection* of documents, e.g. the online course descriptions, and the *group* of documents considered within the collection. For example, the subset of Machine Learning courses we use later is a *group* of 25 documents within the 1132 courses in the entire Coursera collection. Although previous work takes into account the specificity of terms within the *collection* (for example using inverse document frequency), they do not target discriminative keyphrases within a *group*. Restricting the collection and the extraction to the subset of documents in the group in order to use existing approaches has the important downside that it degrades the estimates of term/phrase frequency the extraction relies on. This is especially problematic for supervised approaches that require annotated documents.

In this paper, we study the problem of extracting *discriminative* keyphrases, that depend on the group of documents under consideration within a larger collection. We embed keyphrases in a semantic hierarchical structure using a *Deep Belief Network* (DBN) to characterize the relationship between pairs of phrases. We show that such knowledge can be used to build a discriminative keyphrase extraction system that adapts to the set of documents considered instead of returning a fixed set of keyphrases for a document. We test our approach on two tasks. First, using scientific articles, we extract keyphrases that identify authors expertise from the articles they published. Using a hierarchy of concepts learned from a scientific book, we show that this allows us to contrast researchers within different but related domains. The set of expertise keyphrases differs, for the same researcher, depending on the domain and the set of peers. In our second collection, we explore the problem of extracting keyphrases describing competencies taught by online courses. A semantic hierarchical structure of course phrases guides the extraction towards keyphrases that distinguish one course from the set of courses it is compared to. This is illustrated on two overlapping domains, showing that descriptive keyphrases for the same course may differ depending on the other courses within the domain.

2 Method

The discriminative keyphrase extraction relies on a keyphrase similarity decribed in Section 2.1, used to compute a similarity-based score (Section 2.2). We then extend that score with a semantic hierarchy learned using a Deep Belief Network, as described in Section 2.3.

2.1 Embedding-based Keyphrase Similarity

In order to measure the semantic similarity between two keyphrases p and q, we employ the widely used cosine similarity. This requires some kind of vector representation for both phrases. Learning representations for words, phrases or documents is central to natural language understanding. Vector representations learned using neural networks, a.k.a. embeddings, have recently shown to be effective in a wide range of tasks (Collobert et al., 2011; Mikolov et al., 2013). In our work, we use these low-dimensional vector representations to encode the meaning of each keyphrase.

Starting from word representations obtained from `word2vec`[1], we follow a standard approach to obtain a phrase representation, by averaging the vectors of each component word:

$$\mathbf{p} = \frac{1}{|p|} \sum_{w \in p} \mathbf{w}, \tag{1}$$

where p, w are the phrase and words respectively, $\mathbf{p}$ and $\mathbf{w}$ are their vector representations and $|p|$ is the number of words in p. For example, the embedding for *Machine Learning* is the average of the vector

[1] `https://code.google.com/archive/p/word2vec/`

representations for *Machine* and for *Learning*. The similarity between two phrases p and q is:

$$\text{cosine}(\mathbf{p}, \mathbf{q}) = \frac{\langle \mathbf{p}, \mathbf{q} \rangle}{\sqrt{\langle \mathbf{p}, \mathbf{p} \rangle \langle \mathbf{q}, \mathbf{q} \rangle}} = \frac{\sum_i p_i q_i}{\sqrt{(\sum_i p_i^2)(\sum_i q_i^2)}} \tag{2}$$

where i runs over the dimensions of the chosen embedding space, and $\langle \cdot, \cdot \rangle$ is the scalar product notation. Note that despite its simplicity, arithmetic average has been found to be very effective among many alternatives when combining word vectors to represent phrases (Mitchell and Lapata, 2008).

2.2 Similarity-based Discriminative Keyphrase Extraction

Now equipped with a similarity between keyphrases, we turn to extracting *discriminative* keyphrases for a document. In the similarity-based approach, we consider every candidate keyphrase p, and compare it to all other keyphrases from the group of document by computing the average similarity score between p and all other keyphrases q from all documents in the group. In our example, this would be all expertise keyphrases extracted for all researchers in the group considered:

$$sScore(p) = \frac{1}{C} \sum_{q \in \mathcal{K}} \text{cosine}(\mathbf{p}, \mathbf{q}), \tag{3}$$

where $\mathcal{K}$ is the set of keyphrases extracted from all documents, and $C = |\mathcal{K}|$ is the total number of keyphrases extracted in the group. We use $sScore(p)$ to rank all candidate keyphrases for the document (or researcher) under concern. We consider various ways to select the best discriminative keyphrases below and compare these different strategies in the experimental section.

Top: Pick the top N keyphrases, i.e. most similar with other candidates on average. These should be "safe bets" but not too specific;

Bottom: Pick the bottom N keyphrases, i.e. most dissimilar with other candidates on average. These should be very specific but also noisy;

Middle: Pick the middle N keyphrases, These may strike the right balance: some similarity with the rest, i.e. not noisy, but not too similar, i.e. specific to a document.

This separation is somewhat crude, but further investigation in Section 4.2 show that further refinements do not yield better performance.

2.3 Hierarchy-based Discrimintive Keyphrase Extraction

In order to use hierarchical semantic information to extract discriminative keyphrases, we first need to model the hierarchical information between keyphrases. We generalize the linear projection for hierarchical relations proposed by Fu et al. (2014), by using a Deep Belief Network (DBN) to model this relationship on keyphrase embeddings. A d-dimensional vector for each keyphrase is obtained using again `word2vec`, as in Section 2.1. The hierarchical information between two keyphrases p and q is then modeled as a binary classification problem: From a $2 \times d$ dimensional input containing the embeddings $\mathbf{p}$ and $\mathbf{q}$, the model predicts whether q is a child of p (positive class) or not (negative class).

DBNs are deep learning models consisting of multiple layers of hidden variables, often used to obtain abstract representations (e.g., features) for raw inputs. They were shown to be effective in many problems (Bengio, 2009). We use a typical DBN architecture composed of two hidden layers between one input and one output layer. Pairs of adjacent layers in the DBN are trained in a greedy layer-wise fashion as described in (Hinton et al., 2006). The principle of greedy layer-wise unsupervised training is widely applied to train DBNs with Restricted Boltzmann Machines (RBMs) as the building blocks for each layer. It mainly consists of two steps: (1) train each RBM in an unsupervised way to obtain the initial weights; and (2) starting from these initial weights, train the network in a supervised way using backpropagation. A RBM is a type of undirected graphical model (Hinton, 2010). Given a vector of visible (input) binary units $\mathbf{v} \in \{0, 1\}^{|\mathbf{v}|}$ and a vector of binary hidden units $\mathbf{h} \in \{0, 1\}^{|\mathbf{h}|}$, connections

NIPS	**DL**			**ML**		
	Researchers	keyphrases	gold.std	Researchers	keyphrases	gold.std
	10	206	77	10	173	70
Coursera	**MLC**			**CSDS**		
	Courses	keyphrases	gold.std	Courses	keyphrases	gold.std
	25	436	101	31	1190	292

Table 1: The number of total keyphrases and gold standard keyphrases in NIPS and Coursera datasets

between the visible and hidden units are weighted by the $|\mathbf{h}| \times |\mathbf{v}|$ matrix $\mathbf{W}$. Given bias terms $\mathbf{a}$ and $\mathbf{b}$ for the visible and hidden units, respectively, the energy function is given by:

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{a}^T\mathbf{v} - \mathbf{b}^T\mathbf{h} - \mathbf{h}^T\mathbf{W}\mathbf{v} \tag{4}$$

The joint probability distribution over visible and hidden units $P(\mathbf{v}, \mathbf{h})$, and the marginal distribution $P(\mathbf{v})$ over the visible units are defined as:

$$P(\mathbf{v}, \mathbf{h}) = \frac{1}{Z}e^{-E(\mathbf{v},\mathbf{h})} \quad \text{and} \quad P(\mathbf{v}) = \frac{1}{Z}\sum\nolimits_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \tag{5}$$

with $Z = \sum_{\mathbf{v}}\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}$ the partition function. The RBM is trained using Gibbs sampling, alternatively sampling $\mathbf{h}$ given $\mathbf{v}$, and $\mathbf{v}$ given $\mathbf{h}$ from the conditional probabilities:

$$P(h_j = 1|\mathbf{v}) = \sigma(b_j + \mathbf{W}_{j\cdot}\mathbf{v}), \quad \text{and} \quad P(v_i = 1|\mathbf{h}) = \sigma(a_i + \mathbf{W}_{\cdot i}^T\mathbf{h}) \tag{6}$$

where $\sigma(x) = 1/(1 + \exp(-x))$ is the sigmoid function, $\mathbf{W}_{j\cdot}$ the j-th row and $\mathbf{W}_{\cdot i}$ the i-th column of weight matrix $\mathbf{W}$. Gibbs sampling allows us to get unbiased samples of the expectation of $v_i h_j$ under the distribution specified by the model, from which the RBM can be learned using contrastive divergence (Hinton et al., 2006).

Once the DBM model has been trained, we use it to predict the hierarchical relationship between pairs of candidate keyphrases. Specifically, given a candidate keyphrase, we form pairs with all the other keyphrases from the considered group of documents. For each pair, a prediction is made using the trained DBN model, indicating the hierarchical relationship between the two keyphrases in the pair. From these predictions, we estimate the number of children M of the candidate keyphrase in the group. This hierarchical information allows us to estimate the position of the candidate keyphrase in the semantic hierarchy. A large M indicates that the keyphrase is relatively high in the tree. Otherwise, the keyphrase is likely located at a lower level in the hierarchy. We incorporate this information in a hierarchy-based score by combining it with the discriminative score from Eq. 3 and modulating the trade-off through an exponent α:

$$hScore^{\alpha}(p) = sScore \times \left(\frac{1}{M}\right)^{\alpha} \tag{7}$$

3 Experiments

Two collections were used in this study: NIPS (Neural Information Processing Systems) conference papers and Coursera courses. The collections are described below and summarized in Table 1.

3.1 The NIPS Data

The NIPS data was obtained from `http://www.cs.nyu.edu/~roweis/data.html`. The dataset contains papers published at the NIPS conference from 1987 to 1999. All texts from volumes 0 to 12 were combined as one corpus to train the word embeddings using the `word2vec` tool. We set the window size to 8 and vector dimension to 200 so that each word is represented by a 200-dimensional numerical vector.

The DBN classifier modelling the hierarchical semantic information was trained from the table of content (TOC) of a machine learning book (Bishop, 2006). All words were converted to lowercase and unrelated keyphrases (e.g., from introduction and exercises) were removed. A phrase pair (p, q) was marked as positive if p was the ancestor of q in the TOC tree. Negative pairs were sampled from keyphrases without hierarchical relationship (e.g., from different chapter titles). The labeled training data includes 424 positive and 576 negative examples, used to train the DBN.

We selected two groups of researchers from the NIPS authors: one for deep learning (DL) and one for machine learning (ML). We selected ten researchers in each group, and collected reference expertise keyphrases from the following resources: homepages, resumes, Google Scholar webpages, LinkedIn profiles, as well as other related webpages. From the extracted list of keyphrases, four human annotators manually selected the gold standard expertise keyphrases from candidate keyphrases for each researcher and each group. Note that researchers who appear in both groups can have two distinct sets of reference discriminitive keyphrases, depending on which researcher group is considered.

3.2 The Coursera Data

Coursera is one of largest online education platforms, providing thousands of massive open online courses. One purpose of extracting discriminative keyphrases from course descriptions is to help students choose courses according to their interests. Discriminitive keyphrases among similar courses are more useful and meaningful than general keyphrases. We collected the course information of 1132 courses using the Coursera API[2] (Coursera, 2016).

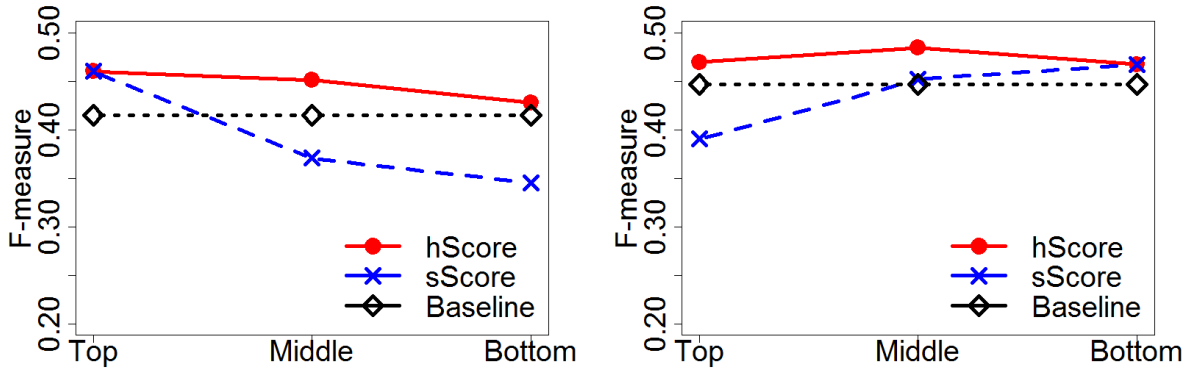
To obtain the semantic keyphrase hierarchy from the Coursera data, word vectors were trained using `word2vec` based on the whole Coursera corpus, and the DBN model was built based on the hierarchical pairs of phrases from courses. Phrases extracted from the course titles and course descriptions formed positive example pairs, while pairs of keyphrases occurring in the course description of the same course were negative examples. In total, 1945 positive and 455 negative keyphrase pairs were used to train a DBN model on the Coursera data.

Groups of similar courses were identified by clustering courses based on their textual descriptions. We removed punctuation, stop words and numbers and used tf-idf to generate document profiles. Thirty clusters were generated using k-means. The second largest cluster was identified as grouping a computer science and data science (CSDS) courses. We selected 31 courses with more than 30 candidate keyphrases from that CSDS cluster. We also selected 25 courses in the Machine Learning (MLC) subdomain under Data Science. Courses in MLC are more homogeneous than in CSDS, but may have fewer keyphrases (Table 1). In both groups (CSDS and MLC), candidate keyphrases were extracted from the course names and course descriptions. Part-of-speech patterns based on Brill's part-of-speech tagger (Turney, 1997) were used to extract candidate keyphrases. Two researchers picked the reference discriminative keyphrases from the set of candidates, for each course in CSDS and MLC (Summary in Table 1). Note again that the same course can have different reference discriminative keyphrases, depending on whether it is considered in the MLC or CSDS groups.

3.3 NIPS Data Results

Similarity- and hierarchy-based scores were used to extract the discriminative keyphrases from all articles of each researcher. Eight keyphrases in the DL group and seven keyphrases in the ML group were extracted for each researcher. The experimental results for both groups are illustrated in Figure 1. We measure performance using the F_1 score (Van Rijsbergen, 1979). The baseline method corresponds to randomly selecting keyphrases within the set of candidates. Its performance is the expected F_1 score under a uniform probability of extraction. For similarity- and hierarchy-based methods, the $sScore$ and $hScore$ were computed for each candidate expertise keyphrase for each researcher, and ranked in descending order of score. We measure the performance when selecting the top, middle or bottom keyphrases from the ranked list. Results from Figure 1 show that the hierarchy-based method ($hScore$) always outperforms both the baseline and the similarity-based approach ($sScore$). The similarity-based

[2] `https://building.coursera.org/app-platform/catalog/`



(a) DL

(b) ML

Figure 1: Performance of discriminative keyphrase extraction using similarity-based ($sScore$) and hierarchy-based ($hScore$) methods from the top ($\alpha = 0$), middle ($\alpha = 0.25$), and bottom ($\alpha = -2$) keyphrases in the NIPS DL (a) and top ($\alpha = -1$), middle ($\alpha = 3$), and bottom ($\alpha = 0$) in ML (b) groups of researchers. Baseline is a random choice among candidate keyphrases.

method sometimes performs worse than the baseline, which performs quite well as the candidate lists are small. These results show that the hierarchical information is clearly beneficial for discriminative keyphrase extraction. The best performance is achieved by mid-level keyphrases in the ML dataset and top level keyphrases in the DL dataset. This suggests that the hierarchy-based method does a good job pushing the relevant discriminative keyphrases towards the top in the narrower DL domain.

3.4 Coursera Data Results

The same setup was used to extract keyphrases that represent the concepts covered in Coursera courses. Ten keyphrases were extracted from the CSDS group, since each course has at least 30 keyphrases. For the MLC group, we extracted only 5 keyphrases as there are fewer candidates. Examples are given below for 2 course:

Course #1—Machine Learning: Clustering & Retrieval

1. MLC: mixed membership, expectation maximization, dirichlet allocation, other documents, latent dirichlet

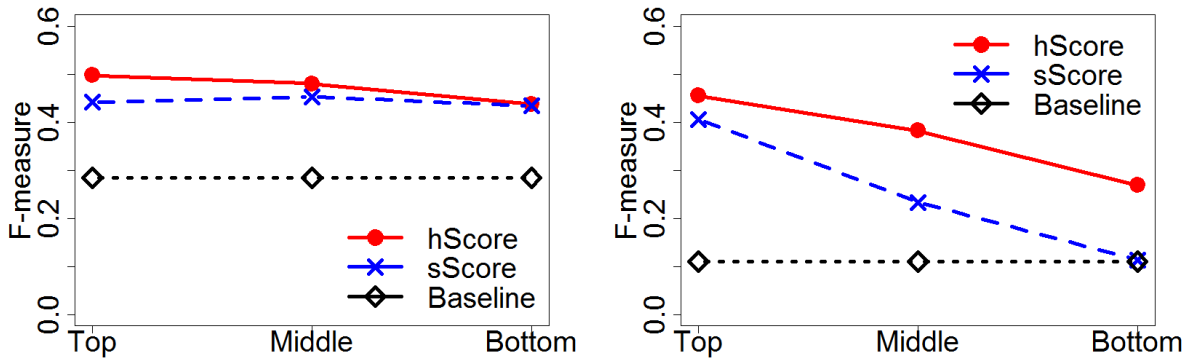
2. CSDS: document retrieval, similar documents, membership modeling, mapreduce learning, case study

Course #2—Machine Learning Capstone : An Intelligent Application with Deep Learning

1. MLC: product recommender, deep features, deep learning, intelligent application, pretrained models

2. CSDS: product recommender, learning classifiers, neural network, activation functions, pre-trained models

Results are presented in Figure 2, using the F_1 score for the top, middle, and bottom keyphrases. They show that both the similarity- and hierarchy-based methods outperform the baseline by a large margin in most situations. This is in part due to the fact that there are many more candidate keyphrases than in the NIPS data, so that the baseline's expected performance is much lower. The middle keyphrases yield the best performance for $sScore$ on the MLC group, otherwise the top keywords reach the best performance. This suggests again that both scores generally do a good job pushing the most discriminative



(a) MLC

(b) CSDS

Figure 2: Performance of discriminative keyphrase extraction using similarity-based ($sScore$) and hierarchy-based ($hScore$) methods from the top ($\alpha = -7$), middle ($\alpha = 0.5$), and bottom ($\alpha = 5$) keyphrases in the Coursera MLC (a) and the top ($\alpha = -7$), middle ($\alpha = 0.25$), and bottom ($\alpha = 10$) CSDS (b) groups of courses. Baseline is a random choice among candidate keyphrases.

competency keyphrases to the top of the list. The hierarchy-based method again consistently outperforms the similarity-based method. This suggests that the semantic hierarchy brings an important information that is useful for extracting discriminative keyphrases and provides a clear boost in performance.

4 Discussion

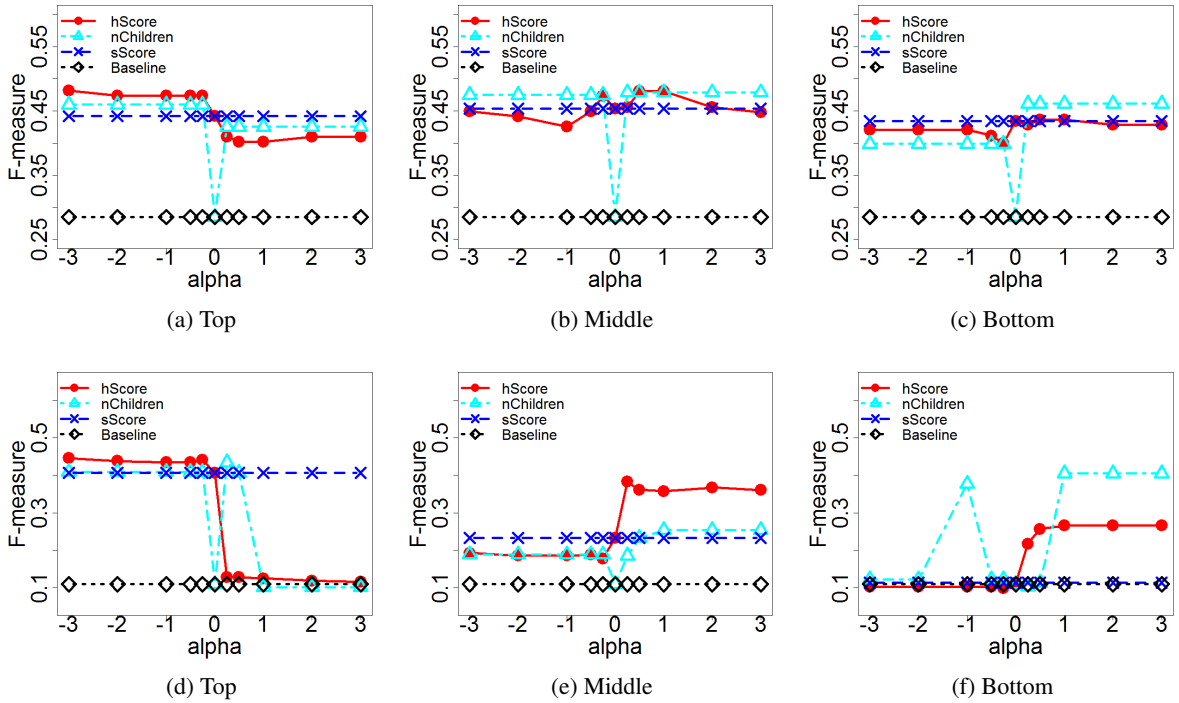
We analyze and discuss below two additional issues on the Coursera datasets: the effect of the hierarchical information on the hierarchy-based score, and the optimal selection of the discriminative keyphrases.

Note also that in this study, we used DBN to learn the hierarchical relationship between keyphrase pairs. Other classifiers could be used to model this relationship. However, DBN are expected to perform well on high dimensional word-embedding and are able to model non linear relationships between word pairs.

4.1 Effect of Hierarchical Information on the Hierarchy-based Scores

We saw that the hierarchy-based method outperforms the similarity-based method in all experiments. The $hScore$ is a trade-off between the number of children M and the $sScore$, to which it reduces when $\alpha = 0$ (Eq. 7). To further investigate the role of the hierarchical information in the $hScore$, we vary the value of α and compare it to the $sScore$, the baseline, and a new score using only the number of children, $nChildren = M^{-\alpha}$. We show their performance on the top, middle, and bottom level keyphrases in Figure 3. Note that negative α promote keyphrases with more children (more general keyphrases), while positive α push these keyphrases down the ranked list and favours more specific (less children) keyphrases. When $\alpha = 0$, $hScore = sScore$ and $nChildren$ is constant and performs the same as the random selection baseline.

On both course groups (MLC and CSDS), the best performance is achieved by the $hScore$ using negative α, on the top level keyphrases. Although the number of children behaves similarly in that regime, its performance is slightly lower, indicating that the keyphrase similarity still plays a key role. The difference in behaviour between $hScore$ and $nChildren$ is more pronounced on the bottom and middle level keyphrases. In particular, the $nChildren$ clearly outperforms the $hScore$ for positive α, but the resulting performance is still lower than what $hScore$ achieves on the top level keyphrases. Note also that the flexibility provided by the α parameter allows $hScore$ and $nChildren$ to always outperform the other two scores ($sScore$ and random baseline) for at least some value of α, again confirming the positive role of the hierarchical semantic information. In conclusion, this confirms that the best performance is usually obtained using the combination of similarity and hierarchy information implemented in the



(a) Top (b) Middle (c) Bottom

(d) Top (e) Middle (f) Bottom

Figure 3: Performance of several discriminative scores with varying trade-off parameter α, on the top (a), middle (b), and bottom (c) keyphrases for the Coursera MLC (top) and CSDS (bottom) course groups.

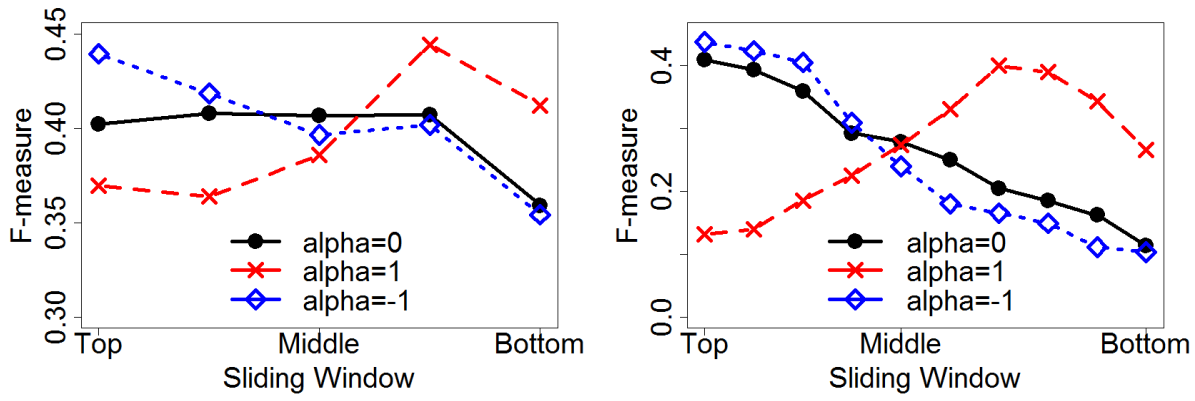
$hScore$, and that picking the keyphrases with the top scores works best.

4.2 Selection of Discriminative Keyphrases

We previously selected the discriminative keyphrases from the top, middle and bottom of the ranked candidate keyphrases. Although picking the top keyphrases usually works best, in the NIPS ML group, the middle level performed better. We investigate the influence of the location of the keyphrases in the list by computing the F_1 score of keyphrases in a sliding window on the MLC and CSDS course groups. Results are shown in Figure 4. Discriminative keyphrases were selected using five windows of five keyphrases (from top to bottom)[3] for the MLC group, and ten windows of ten keyphrases (from top to bottom) for the CSDS group. Results show that candidate keyphrases with high similarities and more children (top on the list when $\alpha = 0$ and $\alpha = -1$) perform very well for both groups. When $\alpha = 1$, keyphrases with more children were pushed down to the end of the ranked list, and they were selected as "lower intermediate" level keyphrases and performed better in both groups. There is a balance between the similarity with other keyphrases and number of children in the hierarchy in these two datasets, and $hScore$ is able to find the optimal parameter for extracting discriminative keyphrases.

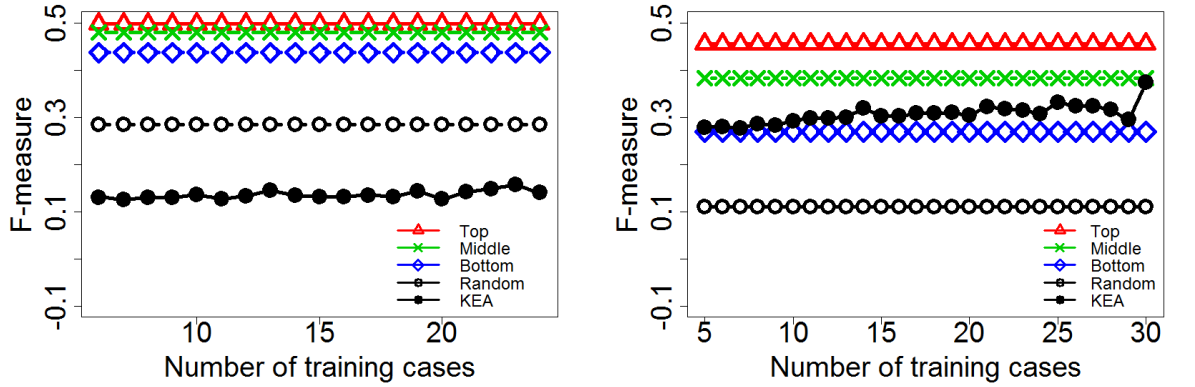
We also compared the performance of our method to KEA using the R package RKEA, in Figure 5. Whereas our method extracts discriminative keyphrases from the candidate keyphrases in a totally unsupervised manner (once the hierarchy is estimated), KEA uses a supervised learning methods to directly extract keyphrases from the text. In order to estimate the performance, we therefore average F_1 over 50 random choices of (labelled) training examples. As KEA does not use the same candidate keyphrases as our method, any partial match between keyphrases extracted by KEA and the gold standard is counted as a positive. We see from Figure 5 that the performance of KEA improves as the number of training cases increases, for both MLC and CSDS. However, on MLC is does worse than the random baseline, likely because it picks keyphrases that are not even among the candidate keyphrases. This is due to the fact that MLC contains courses with very short course description (sometimes as short as a

[3]For example, for a ranked list of 25 candidates, Figure 4(a) would show the performance of picking keyphrases in ranks 1–5 (top), ranks 6–10, ranks 11–15 (middle), ranks 16–20 and ranks 21–25 (bottom).



(a) MLC (b) CSDS

Figure 4: Performance of selecting discriminative keyphrases from a sliding window in Coursera data. Th x-axis indicates the position of the selected keyphrases in the ranked list of candidates (top to bottom).



(a) MLC (b) CSDS

Figure 5: Comparison with KEA in Coursera data.

couple sentences). KEA performed relatively better on CSDS, as that group contains courses with longe descriptions. Overall, our methods performed better than KEA on both datasets.

5 Conclusions

We propose a novel approach to keyphrase extraction, with a goal of finding phrases that both describe a document and differentiate it from a set of texts it is compared with. Previous work often assumes keyphrases are a *static* property of a document, while this work allows us to go beyond most state-of-the-art algorithms and generate keyphrases that depend on the set of documents under consideration, to generate discriminative descriptions of documents. This is done by learning the hierarchical semantic relation between concepts, and using this hierarchy to inform the keyphrase extraction process. We illustrate this on two datasets: a collection of scientific articles from which we extract keyphrases describing the expertise of authors in two related fields, and a collection of on-line courses from which we extract keyphrases describing the competencies covered by the courses, within two domains. Our experiments show that our method can extract domain-specific keyphrases, and that the hierarchical semantic information is useful for extracting the discriminative keyphrases from a group of similar articles or courses.

Acknowledgements

This work was funded by the NRC program Learning and Performance Support Systems (LPSS).

References

Yoshua Bengio. 2009. Learning deep architectures for AI. *Foundations and Trends in Machine Learning archive*, 2:1–127.

Gábor Berend. 2016. Exploiting extra-textual and linguistic information in keyphrase extraction. *Natural Language Engineering*, 22:73–95, 1.

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

Ronan Collobert, Jason Weston, Leon Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR*, 12:2493–2537.

Coursera. 2016. Catalog APIs. Accessed: 2016-03-18.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning semantic hierarchies via word embeddings. In *Proceedings of the 52th Annual Meeting of the Association for Computational Linguistics*.

Geoffrey Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural computation*, 18.

Geoffrey Hinton. 2010. A practical guide to training Restricted Boltzman Machines. UTML TR 2010-03, Dept. Computer Science, University of Toronto, August.

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

Juan Martinez-Romo, Lourdes Araujo, and Andres Duque Fernandez. 2016. Semgraph: Extracting keyphrases following a novel semantic graph-based approach. *Journal of the Association for Information Science & Technology*, 67(1):71–82.

Luis Marujo, Wang Ling, Isabel Trancoso, Chris Dyer, Alan W Black, Anatole Gershman, David Martins de Matos, João Neto, and Jaime Carbonell. 2015. Automatic keyword extraction on twitter. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 637–643.

Y. Matsuo and M. Ishizuka. 2004. Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence Tools*, 13:2004.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing Order into Texts. In *Conference on Empirical Methods in Natural Language Processing*, Barcelona, Spain.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, June.

M. Smatana and P. Butka. 2016. Extraction of keyphrases from single document based on hierarchical concepts. In *2016 IEEE 14th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pages 93–98, Jan.

Peter D. Turney. 1997. Extraction of keyphrases from text: Evaluation of four algorithms. *National Research Council Technical Report*, pages 1–31.

Peter D. Turney. 2000. Learning algorithms for keyphrase extraction. *Information Retrieval*, 2(4):303–336.

C. J. Van Rijsbergen. 1979. *Information Retrieval*. Butterworth, 2nd edition.

Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin, and Craig G. Nevill-Manning. 1998. KEA: Practical automatic keyphrase extraction. In *IN PROCEEDINGS OF THE 4TH ACM CONFERENCE ON DIGITAL LIBRARIES*, pages 254–255.

Kuo Zhang, Hui Xu, Jie Tang, and Juanzi Li. 2006. Keyword extraction using support vector machine. In *Proceedings of the 7th International Conference on Advances in Web-Age Information Management*, WAIM '06, pages 85–96, Berlin, Heidelberg. Springer-Verlag.

Chengzhi Zhang, Huilin Wang, Yao Liu, Dan Wu, Yi Liao, and Bo Wang. 2008. Automatic keyword extraction from documents using conditional random fields. *Journal of Computational Information Systems*, 4:1169–1180.

Wayne X. Zhao, Jing Jiang, Jing He, Yang Song, Palakorn Achananuparp, Ee P. Lim, and Xiaoming Li. 2011. Topical keyphrase extraction from Twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 379–388, Stroudsburg, PA, USA. Association for Computational Linguistics.