Enhancing Categorization of Computer Science Research Papers using Knowledge Bases

Shashank Gupta
shashank.gupta@research.iiit.ac.in

Priya Radhakrishnan
priya.r@research.iiit.ac.in

Manish Gupta*
manish.gupta@iiit.ac.in

Vasudeva Varma
vv@iiit.ac.in

Umang Gupta
umangup@microsoft.com

International Institute of Information Technology, Hyderabad, India
Microsoft, India

Abstract

Automatic categorization of computer science research papers using just the abstracts, is a hard problem to solve. This is due to the short text length of the abstracts. Also, abstracts are a general discussion of the topic with few domain specific terms. These reasons make it hard to generate good representations of abstracts which in turn leads to poor categorization performance. To address this challenge, external Knowledge Bases (KB) like Wikipedia, Freebase etc. can be used to enrich the representations for abstracts, which can aid in the categorization task. In this work, we propose a novel method for enhancing classification performance of research papers into ACM computer science categories using knowledge extracted from related Wikipedia articles and Freebase entities. We use state-of-the-art representation learning methods for feature representation of documents, followed by learning to rank method for classification. Given the abstracts of research papers from the Citation Network Dataset containing 0.24M papers, our method of using KB, outperforms a baseline method and the state-of-the-art deep learning method in classification task by **13.25%** and **5.41%** respectively, in terms of accuracy. We have also open-sourced the implementation of the project[4].

1 Introduction

One of the difficulties faced in categorization of the papers in the conference proceedings is automatically identifying the categories of the research papers from the standard ACM computing classification system[1]. It is important to find the right category of the paper submitted by authors for several purposes, which includes sending the paper to a panel with relevant reviewers according to the category, publishing papers under the correct category and so on. Given the limited amount of content in the abstract and a very high level discus-

* The author is also an applied scientist at Microsoft

In: L. Dietz, C. Xiong, E. Meij (eds.): Proceedings of the First Workshop on Knowledge Graphs and Semantics for Text Retrieval and Analysis (KG4IR), Tokyo, Japan, 11-Aug-2017, published at http://ceur-ws.org

[1] http://www.acm.org/about/class/2012

sion of the topic with few domain specific terms, finding the category of the paper using just the abstract is a challenging and a hard problem.

In this paper we address this problem and propose a novel method to leverage external Knowledge Bases (KB) to improve the performance of short text categorization, using the learning to rank framework [Liu09].

We evaluate our method on a large dataset of abstracts with the aim of classifying the paper into one of the 24 ACM computer science categories. Our method outperforms the baseline method, which does not use any external information, by **13.25%** and outperforms the existing state-of-the-art model in text categorization by **5.41%**, in terms of accuracy.

2 Related Work

Traditional methods for text classification, work by representing document using human curated features like TF-IDF features, followed by a linear classifier like SVM [Joa98]. Due to the bag-of-words assumption and sparsity induced by high dimensionality in the TF-IDF feature vector, these methods do not perform very well. Another approach to this problem is using dimensionality reduction methods on the TF-IDF feature vector to overcome the sparsity problem. These methods include Latent Semantic Allocation (LSA) [DDF$^+$90] and Latent Dirichlet Allocation (LDA) [BNJ03].

Recent advancements in distributional representations of text resulted in better representation schemes for the document. Some examples of such techniques are word2vec [MSC$^+$13], paragraph2vec [LM14] and GloVe [PSM14]. Mikolov et al. [MSC$^+$13] demonstrated the superiority of distributed representation methods over classical representation methods in the sentiment analysis task.

The success of deep learning methods in the field of computer vision and speech processing, inspired their applications in Natural Language Processing (NLP) [CWB$^+$11]. Combined with the superior representation learning methods, these methods have proven to be state-of-the-art in a variety of NLP tasks like sentiment analysis [dSG14], document similarity task [LL13], etc. For the text categorization problem, current state-of-the-art models are based on Convolutional Neural Networks (CNN) [Kim14].

Our main contribution lies in using the learning to rank framework to combine KB with the text using state-of-the-art representation learning methods for text.

2.1 Learning to Rank

In a typical setting, the learning to rank method is defined as follows. We are given a query $q_i \in Q$ and a set of N candidate documents $(d_{i1}, d_{i2}, ..., d_{iN})$. For each document d_{ij}, there is a binary relevance label y_{ij} such that $y_{ij} \in \{0, 1\}$, where a label of 1 indicates that the candidate document is relevant and 0 otherwise. Given this information, the goal of learning to rank method, is to learn a function h, that assigns a higher score to relevant documents than to the non-relevant documents. Formally, learning to rank tries to learn the function h defined as follows.

$$h(w, \psi(q_i, d_{ij})) \rightarrow \mathbb{R} \tag{1}$$

where w is the set of parameters for the function, and ψ generates a feature vector representation of the query and the document combined.

The Learning to rank framework has these two broad categories:

- `Pointwise Approach`, where the training instances are (q_i, d_{ij}, y_{ij}) and a binary classifier is trained over input pairs (q_i, d_{ij}) defined formally as: $h(w, \psi(q_i, d_{ij})) \rightarrow y_{ij}$ with the goal to predict, whether the document d_{ij} is relevant to the query q_i or not.

- `Pairwise Approach` where the model is trained to score correct pairs higher than incorrect pairs with a fixed margin.

While it can be argued that pairwise models can give better results than pointwise models, the primary focus of this work is on generating a good combined representation for the abstract and the corresponding KB entity which can be used for classification, rather than capturing different aspects of similarity for ranking. Hence in this paper, we adopt pointwise method of the learning to rank framework.

3 Our Method

Let $(w_{c1}, w_{c2}, ..., w_{cK})$ denote the set candidate KB entities corresponding to each of the K categories. Let $(y_1, y_2, ..., y_K)$ denote the set of K labels each corresponding to a category. For each paper (research article) d_p in the dataset, there is an associated value for each y_i such that y_i is 1 if d_p belongs to category c_i and 0 otherwise. For each paper d_p, our goal is to rank KB entity relevant to its category w_{ci} higher in score than other KB entities.

Formally the model in Eq. 1 can be specified as follows.

$$h(W, \psi(d_p, w_{ci})) = f(W \begin{bmatrix} R(d_p) \\ R(w_{ci}) \end{bmatrix} + b) \tag{2}$$

where R is the feature representation function (word2vec or paragraph2vec), w_{ci} is the KB entity corresponding to the category c_i, $W \in \mathbb{R}^{1*2d}$ is the linear transformation matrix, d is the embeddings dimension and $b \in \mathbb{R}$ is the bias parameter. We use sigmoid function to realize f.

Our input consists of triplets (d_p, w_{ci}, y_i), where y_i is 1 if w_{ci} is the KB entity corresponding to the category of the document d_p and 0 otherwise. Since we are training a discriminative classifier, for each positive pair (d_p, w_{ci}), we sample a negative pair (d_p, w_{cj}) with label 0. The model is trained to optimize the Binary Cross-Entropy (BCE) loss function defined as follows.

$$BCE(y_i, h(W, \psi(d_p, w_{ci}))) = -(y_i \log h(W, \psi(d_p, w_{ci})) + (1 - y_i) \log(1 - h(W, \psi(d_p, w_{ci})))) \tag{3}$$

For testing on a new document, we generate the feature representation of the document and combine it with the feature representation of each category in accordance with Eq. 2, and select the category for which a label of 1 is predicted. If for multiple categories, we get a label of 1, we randomly select one category out of all predicted categories.

We use word2vec and paragraph2vec for the feature vector representation R. To generate the feature vector representation using word2vec, we compute the average of the word vectors corresponding to each token in the document. To account for out-of-vocabulary words, a similar strategy described in [SM15] is followed and they are replaced with a randomly sampled vector of same dimension as a vector drawn from a uniform distribution $U[-0.25, 0.25]$. For the feature vector representation using paragraph2vec, we follow the inference mechanism as described in [LM14].

4 Experiments

4.1 Dataset

We use the Citation Network Dataset [CST+13] which contains 236565 papers, with each article categorized into one of the 24 ACM categories. We randomly split these articles into 80% training instances and 20% testing instances, and used a one-vs-rest logistic regression Eq.2[2] for classification. For external knowledge base, we select two popular knowledge bases: Wikipedia and Freebase (FB). We use pre-trained embeddings for freebase entities, trained on Google News Corpus[3] to initialize each entity's representation corresponding to each category in text. We experiment with word2vec and paragraph2vec to generate embeddings of categories with Wikipedia.

4.2 Experimental Setup

To generate document embeddings we use gensim [ŘS10], an open-source python library with embedding size set to 300. For text pre-processing, we remove all stop-words, punctuations and non-ASCII characters from abstracts of all articles. We then lower-case all text. To generate the TF-IDF representation of articles, we consider each article's abstract as a document and collection of all abstracts as document collection, and term frequency and inverse-document-frequency is computed accordingly. We use a dimension of 300 for Freebase entity embedding. For more details about experimental settings, interested readers can refer to the open-source implementation code[4]

[2] `http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`
[3] https://code.google.com/archive/p/word2vec/
[4] https://github.com/shashankg7/Knowledge-Base-integration-with-Text-Classification-pipeline

Method	Accuracy	Training Time (in secs)
TF-IDF + SVM Baseline	59.30%	-
CNN baseline [Kim14] with Avg. Padding	64.76%	-
CNN baseline [Kim14] with Max. Padding	67.14%	259.446
Our model	**72.55%**	59.82

Table 1: Performance Comparison with Baseline Methods

4.2.1 Category to Entity Mapping

Knowledge-Base (KB) consists of entities and their relations. To generate semantic representation of the categories, we need to map them to their corresponding entities in the KB. Each category is mapped to its corresponding entity in KB by matching their corresponding description text. For example, the category 'Machine Learning and Pattern Recognition' is mapped to the entity '/en/machine_learning in Freebase and to the entity 'Machine learning' in Wikipedia. More details can be found in the open-source implementation of the project[4].

4.2.2 Results and Discussion

For comparisons, we select the state-of-the-art, Convolution Neural Network (CNN) based method for text categorization [Kim14]. CNN used in the paper operates at a sentence level, so we concatenate all sentences in the abstract into a single sentence. The abstracts in our case are of variable length with different number of tokens, so there is a need to convert them to fixed length sequences. To achieve this, we employ two strategies: average and maximum length padding. In the case of average length padding, we consider the size of each abstract in the corpus and calculate the average length across the corpus. Each abstract in the dataset is then converted to the average size by either padding with a 'null' token or by truncating the sequence, depending on the length of the abstract relative to the average length. Similar strategy is employed with maximum length padding. For padding using the maximum document length method, we removed all documents in the training set with document length greater than 150. We use the code made available by authors[5] and run it on our dataset with the best settings reported in the paper.
We present the results of all the methods in Table 1. For evaluating the performance, micro-averaged accuracy is used as a measure. It is clear that our method outperforms the CNN baseline in accuracy by **5.4%**. Since abstracts are short texts, adding external information to the model clearly gives us an advantage over current methods.

Method	Acc. without KB	Acc. with Wiki.	Acc. with FB	Acc. with Wiki. + FB
para2vec	12.86%	68.04%	**72.55%**	72.54%
word2vec	50.95%	71.38%	71.51%	71.13%

Table 2: Performance Comparison with Different KBs

4.2.3 Quantitative Analysis

We compare the results of our method with baseline methods that do not use KB in Table 2. Paragraph2vec and word2vec combined with Freebase gives an increase of **59.69%** and **20.56%** respectively in terms of accuracy over the baseline methods. It can also be observed that the performance of the method is consistent across different KBs. It can be inferred that the common factor which gives an increase in the performance is the knowledge from the KB, and not the type of KB itself. Further, we select two popular KBs with different semantics and one KB as combination of these two. The performance across these KBs provides us with evidence that the method can be generalized across different KBs. The two KBs differ in the sense that, Wikipedia contains mostly textual information about entities and topics, while Freebase has relation information between entities too and is different in nature than Wikipedia. We also present the training time comparison between our method and deep learning based state-of-the-art method in the rightmost column of the Table 1. The results are shown for 25 epochs of training for CNN (best setting reported by the authors) and 100 (determined using cross-validation) epochs of training for Logistic Regression. It is clear that our method is faster to train as compared to the CNN owing to mostly linear operations in our model, as compared to more complex feed-forward and back-propagation methods in CNNs.

[5]https://github.com/harvardnlp/sent-conv-torch

5 Conclusion

We propose a novel method to combine information from external knowledge bases, using learning to rank framework to enhance classification performance in short texts like abstracts of research papers. We empirically demonstrate that our method outperforms deeper networks in terms of accuracy. Performance with different KBs was also studied, and it provides a strong evidence that the method can be generalized across different KBs. The only requirement is the presence of a high quality external KB. We also demonstrate that our method is faster to train than the baseline method.

In future, we plan to extend this work for different NLP tasks like topic modeling, named entity recognition etc. It would be interesting to observe the effect of addition of domain specific KBs instead of general purpose KBs. It would also be interesting to observe the effect of addition of information from KBs in the setting when number of classes are very large.

References

[BNJ03] D. Blei, A.Y. Ng, and M.I. Jordan. Latent dirichlet allocation. *JMLR*, 2003.

[CST+13] T. Chakraborty, S. Sikdar, V. Tammana, N. Ganguly, and A. Mukherjee. Computer science fields as ground-truth communities: Their impact, rise and fall. In *ASONAM*, pages 426–433, 2013.

[CWB+11] R. Collobert, J. Weston, Lé. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *JMLP*, pages 2493–2537, 2011.

[DDF+90] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, page 391, 1990.

[dSG14] Cí.N. dos Santos and M. Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING*, pages 69–78, 2014.

[Joa98] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *ECML*, pages 137–142, 1998.

[Kim14] Y. Kim. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751, 2014.

[Liu09] T.Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, pages 225–331, 2009.

[LL13] Z. Lu and H. Li. A deep architecture for matching short texts. In *Advances in Neural Information Processing Systems*, pages 1367–1375, 2013.

[LM14] Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML*, pages 1188–1196, 2014.

[MSC+13] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[PSM14] J. Pennington, R. Socher, and C.D. Manning. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–43, 2014.

[ŘS10] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[SM15] A. Severyn and A. Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In *SIGIR*, pages 373–382. ACM, 2015.