Enhancing classification of Research papers to categories of Computer Science domain using Wikipedia articles

No Author Given

No Institute Given

Abstract. We propose a novel method for enhancing classification performance of Research papers to Computer Science categories using Wikipedia articles of the categories. We use state-of-the-art representation learning methods for embedding documents followed by Learning to Rank method for classification. Given the abstracts of research papers from Citation network dataset our method outperforms the state-of-the-art by **3%** accuracy. Our method is also faster to train as compared to state-of-the-art for text classification.

Keywords: Text classification; Representation Learning; Learning to Rank

1 Introduction

Finding categories of research papers automatically from its abstract is a relatively tough problem due to limited context and content presented at a relatively higher level of abstraction than the rest of the paper. This problem comes under standard text classification problem. Traditional methods for text classification works by representing document as human curated features like tf-idf features and then applying a linear classifier like SVM on top of it [6]. Due to Bag-of-Words nature of the representation, they are only able to capture the term frequencies and no semantic information is captured. Unsupervised representation learning methods like LSA [4], LDA [1] overcome these limitations by embedding the documents in a low dimensional space such that some semantic information is preserved in that space. Representing documents using these features yields better performance in classification task [1].

Recent advancements in representation learning for textual data is based on the distributional hypothesis, which states -"words which are similar in meaning occur in similar context" [13] . Recent (popular) examples following this distributional hypothesis are : word2vec [11], paragraph2vec [8] and Glove [12]. [9] demonstrates that these models which are based on distributional hypothesis are better than traditional representation learning methods like SVD (LSA) on word analogy tasks. Following the success of these methods, in this work we use these models for representation of the document **(check if appropriate to write about LTR** followed by Learning to Rank method for text classification.

One problem with all the methods described above is that they only use the patterns in word occurrence or co-occurence statistics to learn the function mapping document to categories. This is not very effective when we are dealing with short texts like abstracts of research papers in our case. [5] have demonstrated that by using external Knowledge Base like Wikipedia, performance of text classification systems can be improved. Our model also leverages external Knowledge Base (Wikipedia) to augment more knowledge into the system and hence enhancing the performance. To combine the two Wikipedia articles with research papers we use Learning to Rank from Information Extraction domain [10]. We briefly describe Representation Learning and Learning to Rank method in the next section.

2 Definitions

2.1 Word2vec

Given a word sequence $(w_1, w_2, ..., w_N)$, the objective of word2vec with skip-gram model is to maximize the average log probability given by:

$$\frac{1}{N}\sum_{t=1}^{N}\sum_{j=-c}^{c}\log p(w_{t+j} \mid w_t)$$

where c is the size of the training context given by the user. The probability $p(w_{t+j} \mid w_t)$ in the skip-gram model is given by :

$$p(w_{t+j} \mid w_t) = \frac{exp(v_{w_{t+j}}^T v_{w_t})}{\sum_{j=1}^{W} exp(v_{w_j}^T v_{w_t})}$$

where $v_{w_{t+j}}$ and $v_{w_t} \in \mathcal{R}^d$ are center word and context word's representation respectively . Word representations are learned during optimization of this objective function described above. We use word embeddings released by Google [1] trained on Google News Dataset.

2.2 Paragraph2vec

Let $(w_1, w_2, ..., w_N)$ be the words in a document, the objective of the paragraph2vec [11] distributed memory (PV-DM) model is to maximize the average log probability defined as follows:

$$\frac{1}{N}\sum_{n=1}^{N}\log p(w_n|w_1, ..., w_{n-1}, s, t)$$

where s is the size of context window and t is the title. To generate the paragraph vector, the model simply treats the title as a special word and generates the

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

representation of title, which can be used as representation for the document. We use genim [2] for implementation of PV-DM model with dimension of document vector set to 300.

2.3 Learning to Rank

Given a query q from a set of queries Q, a set of candidate documents $(d_1, d_2, ..., d_N)$ and a set of relevance labels R, $(y_1, y_2, ..., y_N)$ which is set to 1 if the candidate document is relevant to the query and 0 otherwise. Given this information, the goal of Learning to Rank method is to learn a function **h**, which assigns relevant document higher than non-relevant documents given query and a set of candidate documents. Formally Learning to rank can be defined as learning a function h defined as:

$$h(w, \psi(q, D)) \rightarrow R$$

where w is the parameter of the function, and ψ generates feature vector representation of query and the document combined. Learning to Rank can be broadly divided into three categories:

Pointwise Approach Simplest method to learn a ranking function which takes triplets of the form (q, D, y) and learns a relavance function in the form of a binary classifier which predicts y, given feature vector representation of (q, d). It is widely used in practice due to its simplicity and effectiveness.

Pairwise Approach It is slightly more advanced form of ranking than pairwise method, where the model is explicitly trained to score correct pairs higher than incorrect pairs with a fixed margin. Formally it learns a function h such that:

$$h(w, \psi(q, D_i)) \geqslant h(w, \psi(q, D_j)) + C$$

where (q, D_i) is the correct pair (from training data), and (q, D_j) is the incorrect or corrupt pair (constructed by randomly sampling D_j).

Listwise Approach One limitation with Pointwise and pairwise approach is that both ignore the fact that ranking is a prediction problem list of objects (documents). Listwise approach [2], treats q query with its list of candidates as a single learning instance, thus capturing significantly more information about the ground truth ordering of input candidates.

3 Our Method

Our problem directly translates to Learning to Rank problem. For each paper (research article) d_p in our case we have a set of candidate Wikipedia articles corresponding to each of the catergories $(w_{c1}, w_{c2}, ..., w_{ck})$ with set of labels

[2] https://radimrehurek.com/gensim/models/doc2vec.html

Method	Accuracy
Tf-Idf + SVM Baseline	58.8281%
CNN baseline	69.07827%
Our model	**71.2512%**

Table 1.

$(y_1, y_2, ..., y_l)$ such that y_i is 1 if d_p belong to category c_i and 0 otherwise. For each paper d_p we want to rank wikipedia article relevant to its category w_{ci} higher than other wikipedia articles.

We experiment with pointwise and pairwise approaches for ranking and found that pointiwse to be better. Listwise approach is not applicable in our case, since the problem is not a ranking problem.

Our input consists of triplets (d_p, w_{ci}, y_i), where y_i is 1 if w_{ci} is the Wikipedia article corresponding to category of the 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.

We use word2vec and paragraph2vec for feature vector representation. We follow Bag-of-Words approach for feature vector representation of document with word2vecc. We average the vectors corresponding to tokens in the document. To account for Out-of-vocabulary words, we replace them with a randomly sampled vector of same dimension as other vector sampled from a uniform distribution $\tilde{U}[-0.25, 0.25]$. For feature vector representation using paragraph2vec, we follow inference mechanism described in [8]. We use a linear classifier for the function h. We define function ψ as :

$$\psi(d_p, w_{ci}) = (f(d_p), f(w_{ci}))$$

where f is the feature vector representation function which takes a document as input and generates the feature vector (word2vec or paragraph2vec). We use simple concatenation of the two feature vector representation as function ψ

4 Experiments

4.1 Dataset

We use citation network dataset [3] which contains 247543 papers with each article categorised into one of 24 ACM categories. Since for our method we are assuming that labels of the paper have some Wikipedia article with same title, we select only those categories which have some Wikipedia [3] page with same title. We found out that out of 24 ACM categories, 23 categories have a corresponding Wikipedia page.

[3] Dump of October 2015

Method	Accuracy
Baseline (Paragraph2vec)	13.4722%
Pairwise model (Paragraph2vec)	13.4722%
Pairwise model (Word2vec)	45.1655%
Baseline (Word2vec)	51.0185%
Pointwise model (Word2vec)	70.9804%
Pointwise model (Paragraph2vec)	**71.2512%**

Table 2.

That leaves us with 236565 articles. We randomly split these articles into 80% training instances and 20% testing instances. Training data contains 189290 papers and testing data contains 47275 papers categorised into one of 23 categories. We use linear classifier (binary logistic regression for classification. [4]

Results and Discussion: For benchmarking we select recent state-of-the-art in word embedding based text classification method [7]. Since this baseline uses CNN which requires fixed length input sequence, we convert the papers into fixed length documents by padding them till average sequence length in training corpus. We use the code made available by authors (footnote) and ran it on our dataset with the best settings reported.

We present the result of our method in Table 1. For evaluating performance we use accuracy as measure. It is clear that our method outperforms the baseline in accuracy by **2.17 %**. Since abstracts are short texts, adding external information to the model clearly gives us an advantage over current methods.

Qualitative Analysis We also compare results of our method with simple baseline methods (without using Wikipedia). Paragraph2vec combined with Wikipedia gives **57.779%** increase in accuracy. Word2vec combined with Wikipedia gives an increase of **19.9619%** in the performance over using only word2vec. This can be attributed to the fact that abstracts are short texts and may not contain full information about the topic, while Wikipedia contains detailed information related to category (title), which results to a better performance.

Paragraph2vec fails to outperform word2vec based method due to limited context in the abstract. Paragraph2vec relies on presence of richer context to generate good features, so it fails to perform well on baseline without external information.

With external information Paragraph2vec outperforms word2vec baseline by **0.2708%**, since in this case we have a larger and richer context present due to addition of Wikipedia article, hence Paragraph2vec results in better feature representation than word2vec, and hence outperforms it.

From the point of view of efficiency, our model uses pre-trained feature representation followed by simple linear classifier, so it is faster to train as compared

[4] `http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`

Method	Time
CNN	266.708
Our Method	500

Table 3.

to Convolution Neural Network based baseline. We present in (table 3) the comparison of our method with CNN for training time. We only report the training times for both algorithms excluding pre-processing tasks. It is clear that our method is faster to train than CNN, since it does not involves computationally expensive operations like Convolutions,and computationally expensive Backpropagation steps for training.

5 Conclusion

We propose a novel method to combine information from external Knowledge Bases like Wikipedia through Learning to rank framework to enhance classification performance in short texts. We emphirically demonstrate that our method outperforms more complex Deep Learning based state-of-the-art methods in accuracy. We also demonstrate that our method is faster to train than baseline method, due to complex model in Neural Nets.

Future work in this direction can be to use Deep Learning models instead of simple Linear classifier in Learning to rank method.

References

1. David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.
2. Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136. ACM, 2007.
3. Tanmoy Chakraborty, Sandipan Sikdar, Vihar Tammana, Niloy Ganguly, and Animesh Mukherjee. Computer science fields as ground-truth communities: Their impact, rise and fall. In *Advances in Social Networks Analysis and Mining (ASONAM), 2013 IEEE/ACM International Conference on*, pages 426–433. IEEE, 2013.
4. Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391, 1990.
5. Evgeniy Gabrilovich and Shaul Markovitch. Overcoming the brittleness bottleneck using wikipedia: Enhancing text categorization with encyclopedic knowledge. In *AAAI*, volume 6, pages 1301–1306, 2006.
6. Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.
7. Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

8. Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196, 2014.
9. Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185, 2014.
10. Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.
11. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.
12. Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43, 2014.
13. Magnus Sahlgren. The distributional hypothesis. *Italian Journal of Linguistics*, 20(1):33–54, 2008.