Yan *et al. Big Data Analytics* (2016) 1:7
DOI 10.1186/s41044-016-0007-z

Big Data Analytics

RESEARCH Open Access

Semantic indexing with deep learning: a case study

Yan Yan[1*], Xu-Cheng Yin[2], Bo-Wen Zhang[2], Chun Yang[2] and Hong-Wei Hao[2]

*Correspondence:
yanyanustb@126.com
[1]School of Mechanical Electronic and Information Engineering, China University of Mining and Technology, Beijing 100083, China
Full list of author information is available at the end of the article

Abstract

Background: Deep learning techniques, particularly convolutional neural networks (CNNs), are poised for widespread application in the research fields of information retrieval and natural language processing. However, there are very few publications addressing semantic indexing with deep learning. In particular, there are few studies of semantic indexing in biomedical literature because of several specific challenges including a vast amount of semantic labels from automatically annotating MeSH terms for MEDLINE citations and a massive collection with only the title and abstract information.

Results: In this paper, we introduce a novel CNN-based semantic indexing method for biomedical abstract document collections. First, we adaptively group word2vec categories into (coarse) subsets by clustering. Next, we construct a high-dimensional space representation with Wikipedia category extension, which contains more semantic information than bag-of-words. Thereafter, we design a hierarchical CNN indexing architecture for learning documents from a coarse- to fine-grained level with several multi-label training techniques. We believe that the low-dimensional representation of the output layer in CNNs should be more compact and effective. Finally, we perform comparative experiments for semantic indexing of biomedical abstract documents.

Conclusion: Experimental results on the MEDLINE dataset show that our model achieves superior performance than conventional models.

Keywords: Deep learning, Semantic indexing, Convolutional neural networks, Biomedical documents

Background

Over the last several years, deep neural networks (DNNs) [9] have emerged as a powerful machine learning technology that has achieved tremendous success in image classification, speech recognition, and natural language processing (NLP) tasks by showing significant gains over state-of-the-art shallow learning. In particular, convolutional neural networks (CNNs) [15] are a flexible neural network framework that can be used to reduce variations and exploit spatial correlations using weight sharing and local connectivity. Recently, CNNs have become more popular than fully-connected DNNs.

Semantic indexing [19, 21, 23] occupies an important position in document classification and information retrieval. Document ranking largely depends on measuring the semantic similarity of query-document pairs. Usually the query and the document must be mapped in a low-dimensional space and effectively learning this representation is a

crucial step. Moreover, with the sheer size of data available today, big data information brings great opportunities and potential for various sectors [3, 24]. DNNs have shown their superiority in NLP and deep learning is beginning to play a key role in providing big data predictive analytics solutions. Following some successful applications in this domain, CNNs have also been explored for semantic indexing.

Recently, researchers have applied CNNs to several NLP tasks and achieved significant progress. For example, Zeng et al. used CNNs for relation classification [26] and Dos Santos utilized CNNs for semantic analysis of text [6]. In contrast, the field of biomedicine MeSH (Medical Subject Heading) indexing poses several specific challenges. For example, there are many professional terms (multi-labels), only the title and abstract information of documents is provided, and there is a high correlation between different labels. Hence, little research has been published on this topic. In this paper, we use deep learning and explore ways to apply CNNs to biomedical abstract indexing. We also provide a comparison with several state-of-the-art methods.

We offer three major contributions in this paper. First, to the best of our knowledge, we are the first to present a case study of biomedical document semantic indexing using CNNs. Second, we design a hierarchical CNN-based indexing framework (Hierarchical Classification, HC) and use a suitable loss function for CNN training. We conduct the multi-label classification using a coarse-to-fine approach. Third, we use the MetaMap (in the biomedical field) keywords together with the Wikipedia category to enrich the document representation. Empirical results verify that this improved representation is more compact than bag-of-words (BOW).

Related work

Generally speaking, there are two major directions of research in the semantic indexing domain. On the one hand, shallow learning is the most primitive and direct method for matching correlated documents by comparing keywords. The BOW and term frequency-inverse document frequency (TF-IDF) models only contain information about word frequency, and in most cases the corresponding lexical match is very imprecise because a concept can be described with different words or different language modes. Several learning methods have been attempted for semantic indexing, such as latent semantic indexing (LSI), latent Dirichlet allocation (LDA)and probabilistic latent semantic indexing (pLSI) [2, 4, 11]. These methods are all based on topic models that use SVD to operate on a document vector matrix and remap it to a semantic space (always with a low-dimensional representation), where each dimension represents a "latent topic". However, these methods use linear function computation and are unsupervised, which makes them very limited for semantic representation. Recently, some researchers have attempted to use supervised techniques to learn document representation. Typically, supervised LDA [16] adds a response variable to LDA by generalizing linear models with respect to the EM algorithm associated with each document, and trains the model with the category or labels which is more fit to predict response values for new documents. However, the query and the document are always processed independently in this supervised approach.

Supervised semantic indexing (SSI) [1] focuses on pairwise preferences, which account for the correlations between words (queries and target documents), and then uses learning to rank and choose the best combination of features from a large feature set generated from all word pairs. To deal with the large feature space, memory, speed, and capacity

control issues, a low-rank representation is used in the SSI model. Although this feature embedding can improve document-document and query-document retrieval, the linear word embedding is not sufficient for capturing the document semantics.

On the other hand, deep learning techniques have also been investigated for semantic indexing. Salakhutdinov and Hinton [21] proposed a novel representation method for extending semantic indexing. This method uses the deep auto-encoder model, where the higher layer is encoded with binary codes and the lower layer is generated based on word-count vectors. They also introduced the constrained Poisson model to deal with documents with varying lengths. Mirowski [19] improved the deep auto-encoder model by introducing a dynamic variable using gradient-based MAP inference. This dynamic variable is capable of not only calculating the encoder and decoder cross-entropy, but also training classifiers together with document label. Thus, this method can predict the document categories, determine a better optimization objective function, improve document semantic indexing, and determine the number of steps required by the difference between the current value and the previous dynamic variables. Also, the topic model complexity can be reduced to a cross-entropy loss function model. Wu [23] introduced a deep architecture composed of restricted Boltzmann machines RBMs (which exploits nonlinear embedding and is thus different from other DNNs) to compute the semantic representation of documents. In this low-dimensional semantic space, the nonlinear features embedded through the deep semantic embedding model can achieve a better compact representation. The model also uses discriminative fine-tuning and modifies the calculation of rank scores of relevant and irrelevant documents. Therefore, the indexing performance is improved by this model.

In this paper, considering biomedical document semantic indexing as a case study, we propose a novel indexing approach with deep learning. To deal with the large number of professional terms in biomedical documents, we propose a hierarchical CNN-based coarse-to-fine indexing framework (HC) and design a suitable loss function for CNN training. Considering the high correlation of different labels, we formulate the indexing problem as a multi-label classification system. Because only the title and abstract information of documents is provided, we introduce a rich document representation using both the MetaMap keywords and the Wikipedia category.

Methods

Hierarchical CNN-based framework for semantic indexing (HC)

LeCun [15] introduced error gradients and the BP algorithm into the CNNs structure, making CNNs more extensively applicable. Using the principle of weight parameters with respect to sharing and local receptive fields, the number of trainable parameters can be minimized in the network, which makes the neural network structure simpler and more flexible.

Considering the numerous classes of the documents and the uneven distribution of samples, we introduce a hierarchical CNN-based framework (HC) to conduct biomedical document semantic indexing for both multiple labels and correlated labels. The architecture of our proposed framework is summarized in Fig. 1. The model consists of three parts: feature representation, CNN model, and multi-label hierarchical classification. Hierarchical indexing achieves far superior performance compared with flat classification when processing large number of classes. Moreover, the coarse clustering step is an

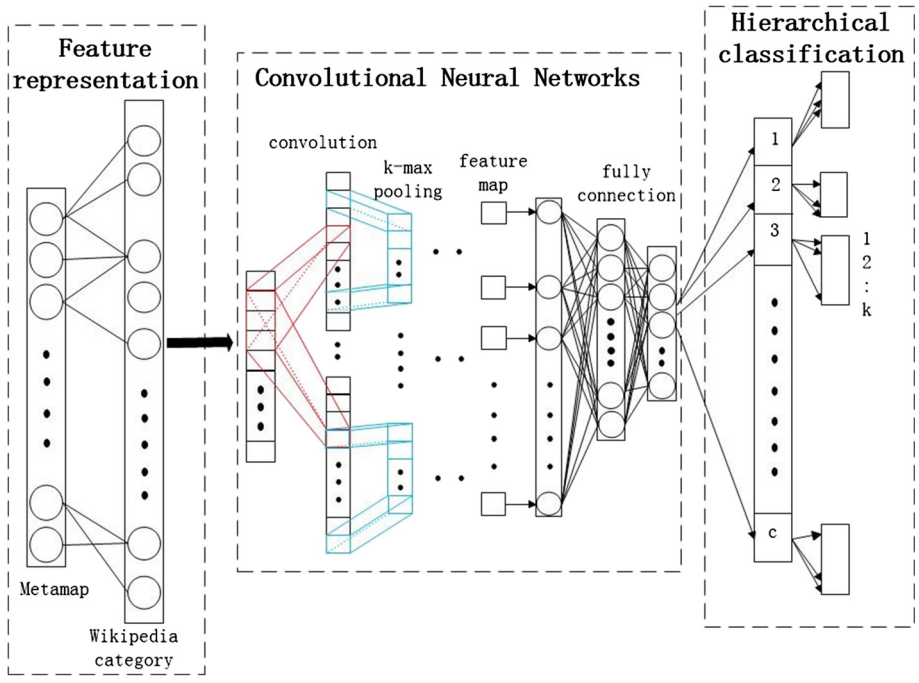

Fig. 1 A hierarchical CNNs-based framework with multi-label classification for semantic indexing (HC)

effective way to remove noise from the unevenly distributed samples. In addition, we also design suitable loss functions for the learning of this framework. The details of these three parts are described in the following subsections.

Semantic feature representation

Wikipedia is a multi-language encyclopedia, covering a wide range of information on individual wiki pages. Wikipedia is an actual corpus, and with rapid changes in social information, it is constantly expanded and updated. There is a vast amount of information about any particular word provided on its Wikipedia page, but the format of each page is not exactly the same.

Considering the shortcomings of BOW, combined with the challenges associated with the indexing task for medicinal abstract documents, we effectively extract information from the Wikipedia page, then use the Wikipedia category information to expand document representation based on MetaMap keywords. MetaMap is a widely applied open-source toolkit which extracts concepts in the UMLS metathesaurus from biomedicine texts. Gabrilovich [7, 8] proposed a feature construction method to enrich document representation. With each document scanned by the feature generation, the relevant Wikipedia concept can be expanded to the document instead of using BOW features. We thus also use this encyclopedia (Wikipedia) to expand the document representation.

We download the Wikipedia corpus up to the year 2012 and development toolkit for the appropriate word links to Wikipedia. When we scan one document, this toolkit can link words that appear in the Wikipedia corpus. These words are also called anchor text words. The process of constructing the document representation is as follows. First, we find the words of the document based on MetaMap, then automatically link the anchor

text word (from the MetaMap words) corresponding to Wikipedia categories based on the toolkit. Words without links to MetaMap words are called non-anchor text words. For these non-anchor text words, we compute the distribution of category-words (non-anchor text words) using the known distribution of words (anchor text words)-documents and categories-documents. This process is referred to as latent Dirichlet allocation (see Fig. 2).

Parameter description: n is the number of distinct words in the documents, m is the total number of classes in all documents, and t is the number of Wikipedia categories. Matrix P_1 represents the distribution of words and documents, where the row w_i is the probability that the word w_i will appear in $j(1, \ldots m)$ classes of documents, denoted as w_{ij}. Matrix P_2 represents the distribution of words and Wikipedia categories, where the row w_i is the probability that a word w_i will correspond to the Wikipedia category $k(1, \ldots t)$, denoted as w_{ik}. Matrix P_3 represents the distribution of Wikipedia categories and document classes, where the row c_k is the probability that the category c_k will appear in the documents, denoted as c_{kj}.

Because the number of Wikipedia categories corresponding to each anchor text word is not fixed, the corresponding number varies greatly. Some of the words may correspond to only one category (e.g., "biomolecular"), some may have three corresponding categories (e.g., "protein"), and some may have seven (e.g., "disease"); this leads to unequal extension lengths for each document. To reduce the differences in document length and also to avoid introducing excessive wiki category extensions (some Wikipedia categories contribute very little and even introduce noise when predicting document labels), we compute the extended Wikipedia categories based on the MetaMap, and remove some wiki categories (e.g., time and date categories) that do not have enough positive impact on the document representation. We also remove some stop words and function words, as the removal of these words does not affect the understanding of the article. We then statistic the occurrence frequency of the categories, retaining only the top 3000 categories. After post-processing, each document is represented as a vector of fixed length 3000*1. Each element in the document representation vector is the word frequency corresponding to the wiki category appearances; that is, the vector is the input to our model.

We were able to use the Wikipedia categories for the corresponding anchor words to calculate the Wikipedia categories of non-anchor words (P_2). This form of document representation combines global (the proportion of the words in all documents categories) and local (anchor words in each document category) information. Hence, through the text representation, we were able to learn more semantic information via the model.

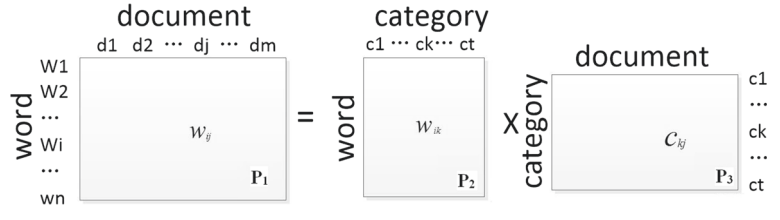

Fig. 2 Words mapping with Wikipedia

CNNs training

CNNs have the advantage of the transformation invariance features of the visual system structure. The layers in the network interleave one-dimensional convolutional layers and k-max pooling layers. In convolution networks, each neuron is only connected to the local area of the lower layer, instead of the whole layer of neurons. That is, it will extract partial information or features. Each neuron detects features from the local receptive field (which may or may not overlap), depending on the operation the neuron performs. Convolution neurons act as feature detectors, and the characteristics of the reaction (i.e., the degree of the reaction) depend on the weights of the neural connections. Each layer of a convolution neural network contains several feature maps. Each feature map is generated by one convolution kernel and detects a certain local feature at every position of the preceding layer. All of the nodes on the same map share a set of convolution kernel parameters.

ReLU [14] is the abbreviation of Rectified Linear Unit, which can be used to increase the nonlinear properties of the network, as well as the sparsity, without affecting the receptive fields of the convolution layer. The ReLU function, which is also the neuron's output, is: $f(x) = max\ (0, x)$. The function signifies that the neuron outputs zero for an input value of less than zero, otherwise the neuron outputs the original input value. The sigmoid function, which is used widely in deep learning with excessive layers, suffers from the vanishing gradient problem, making the training process difficult. Using the ReLU instead of the sigmoid function in a deep network ensures that the network neurons are modestly sparse after training, thus eliminating the issue of vanishing gradients along with the paths of active hidden units.

Besides, the advantage of ReLU is that the network trains faster and can still achieve the sigmoid pre-training effect without pre-training or advanced optimization strategies. Max pooling acts as a feature mapping layer and its operator is a nonlinear subsampling function that returns the maximum of a set of values [15].

In each convolution layer, we filter every input sample vector using multiple one-dimensional filters, and extract the local receptive field feature by setting the size of the sliding window. A convolution layer followed by an ReLU and a k-max pooling forms a feature map. At the end of the network, there are the fully-connected layers such that each feature map connects to all of the subsampling maps from the previous layer. Multiple convolution kernels, which are computed for each convolution layer using multiple feature maps with different filters, enrich the representation of the documents.

The recently introduced technique called "dropout" can reduce over-fitting by decreasing the complexity of co-adaptation of data [10]. Neurons are "dropped out" by randomly setting 50 % of the nodes in each hidden layer in the network to 0, which means that 50 % do not participate in the forward pass or back-propagation processes. In our system, "dropout" is used in all convolutional and fully-connected layers.

Hierarchical indexing

In general, multiple labels are not independent. Thus, effectively using the dependencies among labels can improve the accuracy of multi-label classification [5, 17]. For dealing with thousands of labels, clustering labels into coarse subsets may greatly improve the efficiency of classification. Recently, word2vec [18] representation in continuous space has been found to represent a good relationship between words, and this process can make related words and groups of words appear next to each other in the representation

space. Based on this theory, Ioannis Pavlopoulos trained the word representation for biomedicine labels[1], which greatly improved our research. Based on this word2vec representation, we designed an indexing structure that introduces label (word) embedding for multi-label classification (shown in the third part of Fig. 1).

There are two error updates in our model: coarse classification and sub-classification. We use Eq. 1 to calculate the coarse classification, which is equivalent to the preliminary estimate of the root node of document labels. Then, based on Eq. 2 to calculate the document labels, we calculate the fine classification of the documents. This forms our hierarchical classification approach.

In our model, the error function of the clustered coarse classes (categories) and the weight updates are performed as follows.

$$E = \frac{1}{NC} \sum_{C} \sum_{N} \left(f(x, w) - y_c \right)^2, \quad \Delta w = \frac{\partial E}{\partial w} \tag{1}$$

where C is number of coarse clustering categories, N is the number of training samples, x is the output of CNNs, w is the connection weights between the output layer and coarse cluster layer, $f(x, w)$ is the output value of the model and y_c is the target. For each C, the hierarchical classification function is

$$\Delta w_k = \frac{\partial J_k}{\partial w}, \quad J_k = \frac{1}{n_k} \sum \left(f(x, w_k) - y \right)^2 \tag{2}$$

where k is the number of categories for each coarse subset of the corresponding sub-classification. We use word2vec to cluster the labels into coarse subsets, and subdivide each clustered documents to make the final label judgment.

Coarse classification is a multi-class classification problem in our model, and this layers labels are independent of each other; that is, each document belongs to one of these major categories. The category number (C) is determined based on the word2vec cluster results, where different datasets have different results. Sub-classification is a multi-label classification problem, where each label can have some relationship with respect to the other labels under the same coarse category. Thus, these subcategories are not independent. Although our prediction approach entails two parts, the fine-tuning process is combined. First, we predict the test document label according to the weight parameters obtained from the training set, and then we calculate the error with respect the real label. From Eqs. 1 and 2, we calculate the error and back-propagate it through the neural network. We continuously update these connection weights until the objective function is optimized.

Experiments

Dataset and experimental setup

We use a biomedical document dataset from the BioASQ task challenge for semantic indexing[2]. We download data up to the year 2013 from the website, and select one million articles. Each article contains only the article title and abstract, and there are five to 20 classes (called Medical Subject Headings in the medical field, MeSH) for each article on average. Altogether, there are 27,149 MeSH headings. The National Library of Medicine uses MeSH to index articles in MEDLINE. Because of the extremely uneven distribution

of the dataset, only about 150 MeSH headings appear in more than 1 % of the entire MEDLINE database. Yepes' paper [25] selected the top 10 most frequent MeSH headings to avoid the extremely unbalanced distribution of the dataset. With the same motivation, as well as to extend the experiments, we introduce some unbalanced samples and select the top 2000 most frequent MeSH headings in which each category sample appears more than 500 times. We apply the widely-used cuda-convnet package[3]to train our model on a single NVIDIA GPU.

We randomly select 80 % of the data as training samples and 20 % as test samples, and perform 10-fold cross-validation in our experiments.

Details

The dimensionality of the input, which is the feature representation of our HC model, is 3000 by 1. The main parameters affecting model performance are the size of the convolutional layer sliding window, the parameter k of k-max pooling, the number of fully-connect layers, and each layer's node number. A large sliding window entails smaller connection weight parameters between the layers, increased information redundancy, and a smaller difference between the documents. A smaller sliding window entails greater connection weight parameters and more complexity in the weight calculation and the update process. k-Max pooling was initially proposed to solve the problem of variable length inputs such that the selected max k nodes can be used to form the variable transform into a fixed-length problem. Because the sliding window size settings directly determine the number of nodes in the pooling layer, we introduce k-max pooling to fix the number of nodes to better adjust the parameters of the fully connected layers. This is done to improve training efficiency and also to better observe the influence of the model parameters.

Each training sample (article) contains about 50 to 100 words. In our experiments, the training samples involves mapping the MetaMap keywords based on UMLS, then using the Wikipedia category information to expand the document representation. From all of the data in the sample set, we include the 3000 most frequent categories to represent the document in the form of a vector, in which each category is regarded as one dimension. To reduce the number of hidden units, we experimented with different size settings for the sliding window of the first layer. The maximum number of steps of the window slide is 3000, then followed by ReLU and pooling layer; that is, the document is used in its entirety to capture the feature information. We also vary the sliding window size (from 3000 to 10) to determine the best model. We compare the performance of our best model (with five pooling layers, each of which follows a convolutional layer, and three fully-connected layers at the end of the CNN) with existing methods.

In our experiment, we use matconvernet [22] to implement the CNN model. Our model achieved its best results when the parameter settings are as follows. The sliding window size is set to 200 steps, the learning rate is set to 0.001, the batch size is set to 400, the numepochs is set to 50, and the weight decay is set to 0.003. In the fully-connected layers, we choose the three layers corresponding to the classic LeNet5 [20] layer. The number of neural nodes is 400, 300, and 200 in the first, second, and third fully-connected layer respectively. The following experimental results and analysis are based on these parameter settings.

Quantitative evaluation

We use the micro and macro Precision (P), Recall (R), F_1-measure (F_1) and Similarity (S) as the evaluation criteria [12].

	Positive	Negative	Predicted
Positive	True Positive (TP)	False Negative (FN)	Actual Positive
Negative	False Positive (FP)	True Negative (TN)	Actual Negative
Actual	Predicted Positive	Predicted Negative	

TP is the number of true positives, FN is the number of false negatives, FP is the number of false positives, and TN is the number of true negatives. Finally, precision (P) and recall (R) are defined as

$$P = \frac{TP}{(TP + FP)} \tag{3}$$

$$R = \frac{TP}{TP + FN} \tag{4}$$

F_1 and Similarity (S) are comprehensive assessment metrics and defined as

$$F_1 = \frac{2 * P * R}{(P + R)} \tag{5}$$

$$S = \frac{TP}{(TP + FP + FN)} \tag{6}$$

Comparison methods

In the experiments, we compare six approaches: our proposed hierarchical classification (**HC**) system with CNNs, a directly binary classification (**DBC_flat**) method, support vector machines (**SVM**)[4], latent Dirichlet allocation (**LDA**) [2], naive Bayesian (**NB**) [13], and logistic regression ((**LR**)) [13].

Directly Binary Classification (**DBC_flat**): In our experiments, to verify the validity of the hierarchical classification in our HC model, we compared our method to flat classification based on the CNN model, called DBC_flat. The total number of document classes in this approach is 2000. Regarding each class as a binary classification entails extending each of the 2000 nodes into two nodes. Thus, the last layer of the model has 4000 nodes. The label (1,0) signifies that the document belongs to this class; otherwise, its label is (0,1). Through the output value of the sigmoid function computing nodes, the value of the two nodes is compared to determine whether it belongs to the class. The output value of the nodes is computed by the sigmoid function.

Support Vector Machines (**SVM**): We consider three kernel types for the SVM model: linear kernel, RBF function, and sigmoid function. The linear kernel is used for linearly separable cases and is the simplest kernel with the least parameters. The RBF function is used mainly for linearly inseparable cases but entails multiple parameters. Our experimental results signify that the RBF function is preferable for our task, and we use cross-validation on the training set to select the RBF parameters. The multi-label classification is regarded as multiple binary classification problems in SVM. The samples of each

class in the dataset are unevenly distributed. For each class, positive samples are defined as current class samples, and the negative samples are two to three times more numerous than the positive samples. We thus train 2000 classifiers. In the test phase, each test sample is judged by 2000 classifiers to predict whether it belongs this class.

Results and discussion

Experiments with different models

Table 1 and Fig. 3 shows the results of biomedical abstract semantic indexing. Experimental analysis indicates that DBC_flat classification using CNNs (where each class as a binary classification problem) has very poor performance. By analyzing the results from the test documents, we found that most test samples were predicted to be negative by the model. Further evaluation of the training stage revealed very few class numbers of positive samples. Too many nodes were connected in the model and the adjustment of their weights was not updated in the fine-tuning process. Specifically, more weights were updated in the negative samples compared with the positive samples. In response to this problem of update weights, we try the solution of dropout strategy as we used in HC to reduce the number of weights update, but it has little impact on the results. In response to this problem of update weights, we include the dropout strategy as we used in HC to reduce the number of weight updates, but this has little impact on the results.

Our proposed hierarchical semantic indexing method (HC) greatly increased the classification precision for the positive samples in the first layer. Figure 4 shows the label embedding representation of coarse clusters(partly). This hierarchical semantic framework can effectively reduce the negative impact of negative samples, and mainly updates the weights connected with previous layer nodes during the next layer classification. The updating of weights of nodes that are connected with different previous layer nodes are independent from one other (see Eq. 2), which greatly improve the efficacy of this model for training of positive samples.

We also compare ReLU with the sigmoid function in our CNNs model. The network neurons with ReLU are reasonably sparse after training; thus, vanishing gradients do not exist along with paths of active hidden units in an arbitrarily deep network. We also find that "dropout" weight updating eliminates dependencies on the interaction relationship among the hidden nodes. The "dropout" method encourages each individual hidden unit to learn useful features without relying on other specific hidden units to correct its mistakes.

Overall, the effectiveness of our approach is not only the result of our hierarchical indexing architecture but is also positively influenced by the feature representation. MetaMap

Table 1 Biomedicine semantic indexing results on Precision (P), Recall (R), F1 metrics

Method	MiP	MiR	MiF_1	MaP	MaR	MaF_1
LR	0.353	0.385	0.368	0.316	0.341	0.328
NB	0.382	0.426	0.402	0.339	0.361	0.350
SVM	0.477	0.513	0.495	0.464	0.509	0.485
LDA	0.487	0.534	0.509	0.483	0.519	0.500
DBC_flat	0.548	0.482	0.513	0.519	0.445	0.479
HC	**0.640**	**0.579**	**0.608**	**0.618**	**0.544**	**0.579**

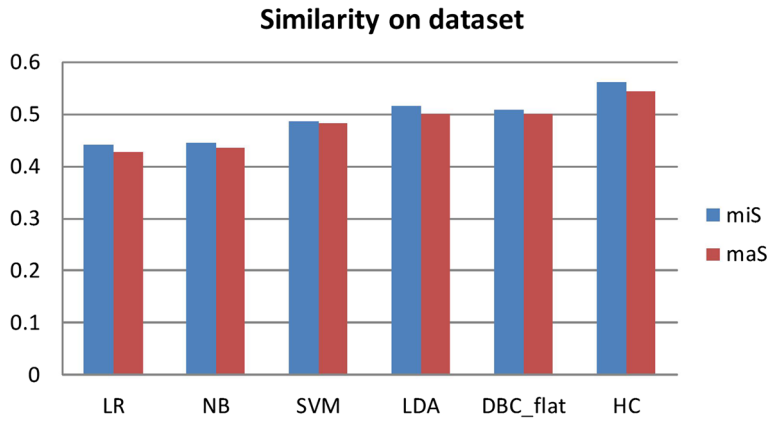

Fig. 3 Similarity measure on dataset

can extract concepts that appear in the UMLS from biomedical text. These vocabularies can provide a better representation for retrieving relevant MEDLINE citations.

Significance test

Figure 5 shows the F_1-measure performance comparison of the our method (HC) with other shallow and deep learning approaches. From this figure, we observe the following: (1) the deep learning methods have better performance than the shallow learning methods; and (2) the hierarchical indexing framework is better than the flat learning methods.

Conclusion

This paper presents a new model with CNNs for learning semantic representations to solve biomedical abstract indexing. Our proposed hierarchical CNN-based indexing architecture shows performance compared with existing methods. This architecture can also be easily extended to other multi-label indexing tasks. Because of time limitations,

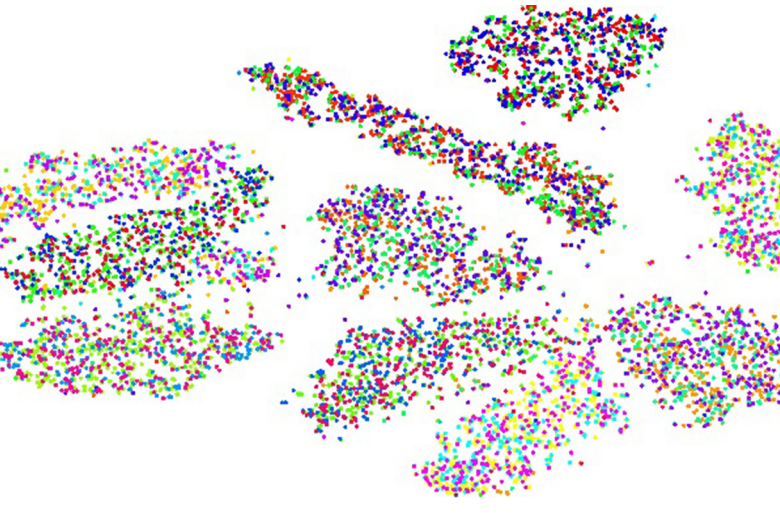

Fig. 4 Coarse clusters of label embedding

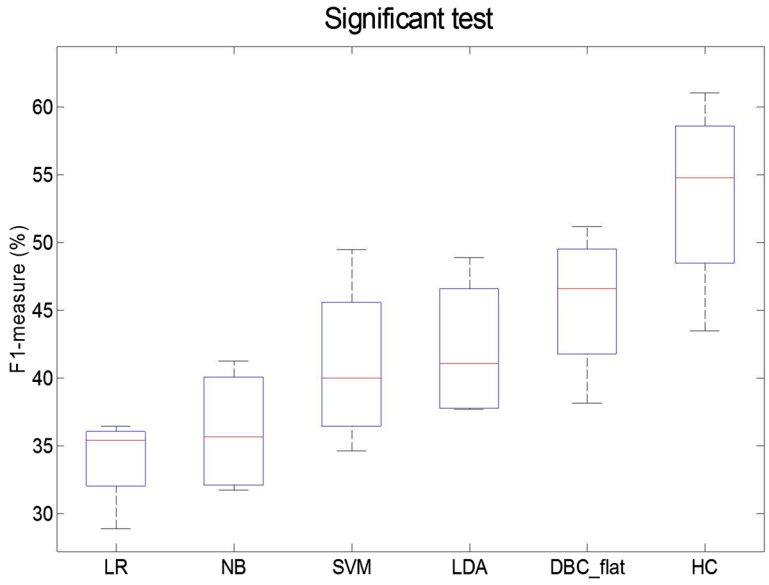

Fig. 5 Significant test

we present limited experiments and related results on 2000-category biomedical document indexing in this paper with the aim of a proof-of-concept. Hence, in the near future we will continue to investigate and improve our proposed approach for larger amounts of semantic labels and more biomedical documents. In addition, based on the GPU platform we have already set up, we can do parallel computing to deal with the big data and better optimize our model. We are thus prepared to meet the challenge of big data using our approach.

Endnotes

[1]http://participants-area.bioasq.org/.

[2]http://www.bioasq.org/participate/challenges.

[3]http://deeplearning.net/software/pylearn2/library/alex.html.

[4]https://www.csie.ntu.edu.tw/~cjlin/libsvm/.

Authors' contributions

YY carried out the model studies and drafted the manuscript. X-CY mend this paper. B-WZ participated in its design. CY participated in its experiments. H-WH is the instructor. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Received: 13 November 2015 Accepted: 4 August 2016
Published online: 30 August 2016

References

1. Bai B, Weston J, Grangier D, Collobert R, Sadamasa K, Qi Y, Chapelle O, Weinberger K. Supervised semantic indexing. In CIKM. 2009;187–96.
2. Blei DM, Ng AY, Jordan MI. Latent dirichlet allocation. J Mach Learn Res. 2003;3:993–1022.
3. Chen XW, Lin X. Big data deep learning: Challenges and perspectives. Access, IEEE. 2014;2:514–25.
4. Deerwester SC, Dumais. Indexing by latent semantic analysis. JASIS. 1990;41(6):391–407.
5. Dekel O, Keshet J, Singer Y. Large margin hierarchical classification. In: Proceedings of the twenty-first international conference on Machine learning. ACM; 2004. p. 27.
6. dos Santos CN, Gatti M. Deep convolutional neural networks for sentiment analysis of short texts. In: COLING; 2014.
7. Gabrilovich E, Markovitch S. Overcoming the brittleness bottleneck using wikipedia: Enhancing text categorization with encyclopedic knowledge. In: AAAI; 2006. p. 1301–6.

8. Gabrilovich E, Markovitch S. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In: IJCAI; 2007. p. 1606–11.
9. Hinton GE, Salakhutdinov RR. Reducing the dimensionality of data with neural networks. Science. 2006;313(5786): 504–7.
10. Hinton GE, Srivastava. Improving neural networks by preventing co-adaptation of feature detectors. 2012. arXiv preprint arXiv:1207.0580.
11. Hofmann T. Probabilistic latent semantic indexing. In: SIGIR; 1999. p. 50–7.
12. Huang SC, Chen BH. Highly accurate moving object detection in variable bit rate video-based traffic monitoring systems. Neural Netw Learn Syst, IEEE Trans. 2013;24(12):1920–31.
13. Jordan A. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. Adv Neural Inf Process Syst. 2002;14:841.
14. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems; 2012. p. 1097–105.
15. LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proc IEEE. 1998;86(11):2278–324.
16. Mcauliffe JD, Blei DM. Supervised topic models. In: NIPS; 2008. p. 121–8.
17. McCallum A, Rosenfeld R, Mitchell TM, Ng AY. Improving text classification by shrinkage in a hierarchy of classes. In: ICML; 1998. p. 359–67.
18. Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. 2013. arXiv preprint arXiv:1301.3781.
19. Mirowski P, Ranzato M, LeCun Y. Dynamic auto-encoders for semantic indexing. In: NIPS 2010 Workshop on Deep Learning; 2010.
20. Mrázová I, Kukačka M. Hybrid convolutional neural networks. In: Industrial Informatics, 2008. INDIN 2008. 6th IEEE International Conference on, IEEE; 2008. p. 469–74.
21. Salakhutdinov R, Hinton G. Semantic hashing. Int J Approx Reason. 2009;50(7):969–78.
22. Vedaldi A, Lenc K. Matconvnet: Convolutional neural networks for matlab. In: Proceedings of the 23rd Annual ACM Conference on Multimedia Conference. ACM; 2015. p. 689–92.
23. Wu H, Min MR, Bai B. Deep semantic embedding. In: SIGIR 2014 Workshop on Semantic Matching in Information; 2014.
24. Wu X, Zhu X, Wu GQ, Ding W. Data mining with big data. Knowl Data Eng, IEEE Trans. 2014;26(1):97–107.
25. Yepes AJ, MacKinlay A, Bedo J, Garnavi R, Chen Q. Deep belief networks and biomedical text categorisation. In: Australasian Language Technology Association Workshop; 2014. p. 123.
26. Zeng D, Liu K, Lai S, Zhou G, Zhao J. Relation classification via convolutional deep neural network. COLING. 20142335–44.