EFCNN:A Restricted Convolutional Neural Network for Expert Finding

Yifeng Zhao, Jie Tang, and Zhengxiao Du

Department of Computer Science and Technology, Tsinghua University, China
zhao-yf16@mails.tsinghua.edu.cn,jietang@tsinghua.edu.cn,
duzx16@mails.tsinghua.edu.cn

Abstract. Expert finding, aiming at identifying experts for given topics (queries) from expert-related corpora, has been widely studied in different contexts, but still heavily suffers from low matching quality due to inefficient representations for experts and topics (queries). In this paper, we present an interesting model, referred to as EFCNN, based on restricted convolution to address the problem. Different from traditional models for expert finding, EFCNN offers an end-to-end solution to estimate the similarity score between experts and queries. A similarity matrix is constructed using experts' document and the query. However, such a matrix ignores word specificity, consists of detached areas, and is very sparse. In EFCNN, term weighting is naturally incorporated into the similarity matrix for word specificity and a restricted convolution is proposed to ease the sparsity. We compare EFCNN with a number of baseline models for expert finding including the traditional model and the neural model. Our EFCNN clearly achieves better performance than the comparison methods on three datasets.

Keywords: Expert Finding · Convolution Neural Network · Similarity Matrix.

1 Introduction

Online question-and-answer (QA) has become a more popular way for users to share their experiences and to ask questions on the Internet. For example, Quora.com and Zhihu.com, the most popular websites for sharing and acquiring knowledge, attract users to answer millions of questions per day; Toutiao QA, an up-and-coming mobile social platform, has accumulated 580 million Toutiao users and 300 thousand professional writers (authors). The competitive advantage of the online QA platforms is that they provide high-quality answers for users and offers a new direction for professional knowledge sharing. However, at the same time, it also poses new challenges. One central challenge is finding a way to assign those new questions (queries) to potential experts, referred to as expert finding.

Expert finding has been studied by researchers from different communities. Several different methods have been proposed. These include keyword-based

modeling [2], language modeling [18, 1, 3], latent semantic indexing [6], and topic modeling [11, 15]. Most of these methods represent every expert as a document and cast the problem as a document matching problem. In language models, each document is represented by words with their term frequency-inverse document frequency (TF-IDF)[20] score. Latent semantic indexing learns a low-dimensional representation by decomposing the word feature space, and topic models such as Latent Dirichlet Allocation probabilistically group similar words into topics, and represent documents as distributions over these topics. Obviously, these existing methods mainly represent documents by the frequency or co-frequency of words, but ignore semantic information at the phrase and sentence level. Thus, how to capture and utilize semantic information at the word, phrase, and sentence level of documents remains a challenging problem.

Given a query paper and a candidate expert pool, we represent each expert as a set of documents he/she has written. As a result, how to estimate the similarity score between these documents and the query paper becomes the central issue. Inspired by the success of convolutional neural networks (CNN) [13] in image recognition, we cast this task as an "image recognition" and present a method based on restricted convolution. Specifically, the similarity between any word pair of two documents is calculated to generate a similarity matrix. However, this similarity matrix not only ignores the word specificity, but is also very sparse and position-related. Therefore we introduce IDF into the similarity matrix for word specificity and propose a restricted convolution layer to ease the problem of the sparse and position-related matrix. The experiments on three datasets show that our work performs better than the baselines.

Contributions In this paper, we define the problem of expert finding and propose a framework based on a restricted convolutional neural network. Based on the similarity matrix, we propose restricted convolution. Compared to a standard convolution, restricted convolution considers the importance of position, and penalizes for similarity far from the center of filters. For taking word specificity into accounts, we further construct a new similarity matrix by combining original similarity matrix and IDF. We prove that the proposed framework can capture and utilize semantic information from word-level to document-level. We compare our framework with several state-of-the-art approaches on three different datasets and experimental results show the effectiveness of our framework.

2 Problem Formulation

Let $S = \{(v_i, d_i)\}_{i=1}^{N}$ denote the set of experts and his/her documents, where v_i is a candidate expert, d_i is the set of support documents authored by (or associated with) expert v_i and N is the expert size. The input of our problem also includes a query d_q, which can be also viewed as a document. There can be various kinds of documents in different applications. For example, in an academic network, the documents could be papers published by researchers, while in a Quora-like website, the documents could be the questions (or answers) that users have asked

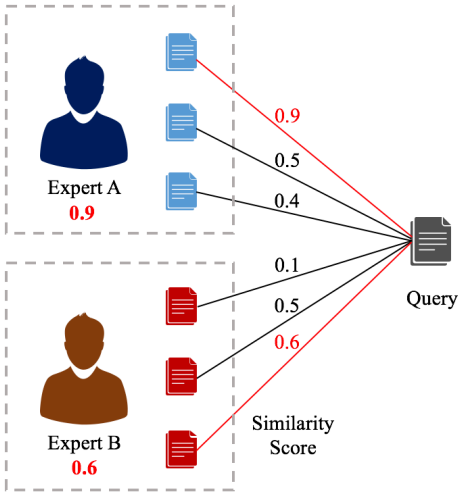

Fig. 1. An example of expert finding: each expert has a set of documents; each document has a similarity score with respect to query. Intuitively, expert A is more relevant than expert B regarding the query .

(or answered). Given this, we can formally define the expert finding problem as follows:

Definition 1 ***Expert Finding.*** *Given a set S and a query d_q, the objective here is to learn a function f using documents d_i in each expert v_i and the query d_q, in order to predict a ranked list $R(v_i, d_i) \subset S$ with $|R| = k$, which is the top-k relevant experts in S with respect to d_q.*

One challenge here is that experts and query documents are two different kinds of entities. This means that they cannot be represented in a common space and the relevance between an expert and query documents cannot be measured directly. An alternative method is to measure the relevance based on expert v_i's documents d_i, where experts with a more relevant document to the query should be ranked higher. The central problem is how to model the representations for documents and queries so that the similarity score can be easily estimated. In this paper, we propose a model based on the similarity matrix and restricted convolution to address this problem.

3 Our model

The basic idea of our model is to cast this problem as an image recognition problem. Our model first constructs a similarity matrix using the embedding of words contained in the document and the query. Viewing the similarity matrix as an image, a restricted convolutional neural network is employed to learn the representations and also predict the relevance score of a candidate to the query.

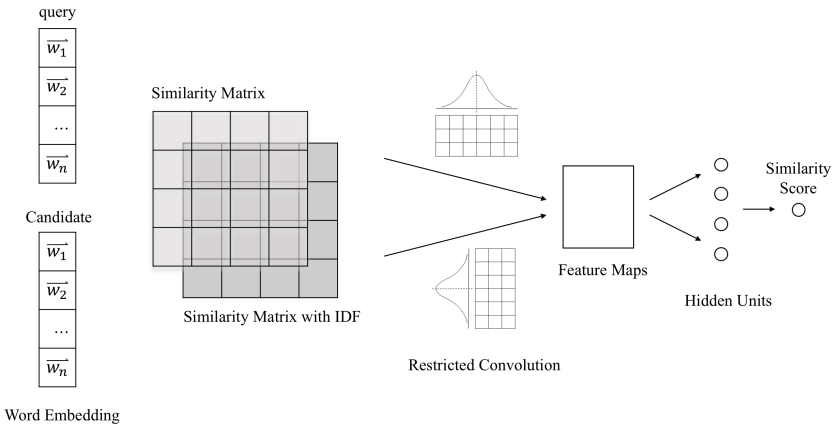

Fig. 2. The overall architecture of EFCNN

3.1 Word Embedding and Similarity Matrix

Word Embedding It is easy to understand that both the query d_q and candidate document d_i can be represented as word sequences. In order to construct a similarity matrix, a variety of methods can be used to compute the similarity between words. For example, the similarity can be simply defined as 1 or 0 to indicate whether two words are identical; however, that ignores the semantic information between two similar words. Considering the semantic information, we use the word embedding technology, i.e., the Word2Vec model [14], to represent each word as a multi-dimensional vector, and then compute the similarity.

For completeness, we give a brief introduction to the Word2Vec model. Word2Vec employs a neural network to learn word embedding for each word. The neural network architecture (the skip-gram model) consists of an input layer, a projection layer, and an output layer. The objective is to maximize the probability of surrounding words for an input word in the corpus. Therefore, the objective can be written as:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{w_j \in nb(w_t)} \log p(w_j|w_t)$$

where T is the size of the corpus, $nb(w_t)$ is the set of surrounding words of w_t, and $|nb(w_t)|$ is determined by the window size in training. The probability $p(w_j|w_t)$ is the hierarchical softmax of the word embedding of w_j and w_t. The authors demonstrate that semantic relationships are often preserved in vector operations on word embeddings, e.g., vec("*King*") − vec("*Man*") + vec("*Woman*") results in a vector that is closest to the vector representation of the word "*Queen*." Due to its high quality and low computational cost, we use Word2Vec embedding as our preferred embedding.

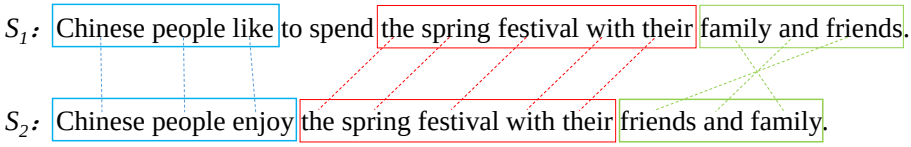

Fig. 3. An illustration of two similar sentences.

Similarity Matrix based on Embedding Given word embeddings, there are many measures to obtain the similarity score, such as euclidean distance, cosine metric and dot product. In this paper, cosine metric is adopted to compute the similarity score between two embeddings. Therefore, the similarity matrix M can be written as:

$$M_{i,j} = \frac{\text{vec}(w_i)^T \text{vec}(w'_j)}{\|\text{vec}(w_i)\| \cdot \|\text{vec}(w'_j)\|} \tag{1}$$

where w_i is the i-th word in query d_q, w'_j is the j-th word in document d_i, $\text{vec}(w)$ is the Word2Vec embedding of the word w, and $\|\cdot\|$ is the norm of Word2Vec embedding.

In this way, similarity matrix M can provide meaningful matching information between query and document at word, phrase, and sentence level. Take two sentences in Figure 2 as an example, we find that these two sentences are similar at all three mentioned levels. At the word level, these sentences not only have identical word pairs, e.g., "*Chinese-Chinese*," but also have similar word pairs, e.g., "*like-enjoy*." At the phrase level, sentences can be broken down into three matching phrase pairs, e.g., "*(Chinese people like)-(Chinese people enjoy)*." These three mentioned phrase pairs roughly construct sentences, which indicates the similarity at the sentence level.

Similarity Matrix with IDF Similarity matrix mainly focuses on the word similarity of two documents, ignoring how specific and distinctive a word is. Take two sentences in Figure 2 as an example again, "chinese" and "festival" are always more specific than "people" and "enjoy" and should be given more attention. TF-IDF is the most common measurement for scoring word specificity. In this paper, similarity matrix already contains the whole words in the document, so only the IDF needs to be taken into account. IDF is the logarithmically scaled inverse fraction of the documents that contains the word. There are a whole family of inverse functions, and here we choose the smooth IDF:

$$IDF(w) = log(1 + \frac{N}{n_w})$$

where n_w is the number of documents containing the word w and N is the total number of documents.

Similarity matrix with IDF can be regarded as the supplementary of similarity matrix for word-specific information, but it cannot completely replace the

similarity matrix, which is discussed in Section 4.2 . Formally, it can be written as:

$$M_{i,j}^{IDF} = M_{i,j} * IDF(w_i) * IDF(w'_j)$$

where the smooth IDF will not change the sign of word similarity.

3.2 EFCNN: Expert Finding with Restricted CNN

Restricted Convolution Inspired by the great success of the CNN in image recognition, [17] views the similarity matrices as images, which can be the input for CNN. However, as shown in Figure 4, the similarity matrix is slightly different from the traditional image, with its sparse value and regional discontinuity.

To ease this problem, we adopt an intuitive method named restricted convolution to edit the convolution structure to produce position-based filters for each layer. Specifically, the closer a position is to the central axis, the higher its weight is. In this paper, we use two decay functions, including linear and exponential decay.

linear decay is a common decay function. If the weight attenuation follows linear decay, the weights $w^{(1,k)} \in R^{m \times n}$ of k-th filter in the l-th restricted convolutional layer can be written as:

$$w_{i,j}^{(l,k)} = (\alpha + \frac{\left|\lfloor \frac{n}{2} \rfloor - j\right|}{\lfloor \frac{n}{2} \rfloor} * (1-\alpha)) * w_{i,\lfloor \frac{n}{2} \rfloor}^{(l,k)} \tag{2}$$

where α is the decency coefficient of linear decay.

exponential decay considers the smooth of decay. If the weight attenuation follows exponential decay, the weights are computed as following:

$$w_{i,j}^{(l,k)} = e^{-\beta |j - \lfloor \frac{n}{2} \rfloor|} * w_{i,\lfloor \frac{n}{2} \rfloor}^{(l,k)} \tag{3}$$

where β is the decency coefficient of exponential decay. Obviously, comparing to standard convolutional layer, each filter in restricted convolution only has one column variable in central axis, which will accelerate the training process. In addition, the restricted convolution can also be transposed as shown in Figure 2.

Forward Network Same with standard convolution, the k-th filter in restricted convolutional layer is used to compute dot product between its weights $w^{(l,k)}$ and regions in the input $z^{(l-1)}$. An element-wise activation function δ is applied to obtain a non-linear feature map $z^{(l,k)}$. Formally, we have:

$$\begin{aligned} z^{(0)} &= M \oplus M^{IDF} \\ z_{x,y}^{(l,k)} &= \delta\Big(\sum_{t=0}^{c^{(l-1)}-1} \sum_{i=0}^{m_t-1} \sum_{j=0}^{n_t-1} w_{i,j}^{(l,k)} \cdot z_{x+i,y+j}^{(l-1,k)} + b^{(l,k)} \Big) \end{aligned} \tag{4}$$

where m_t, n_t denotes the size of t-th filter, $c^(l)$ denotes the number of filters in the l-th layer and $b^{(l,k)}$ is a bias term.

In addition to reducing the spatial size of the feature maps, max-pooling layers also operate independently on every output of convolutional layers and resize them spatially. Therefore, the output $z^{(l,k)}$ of the max-pooling layer can be written as:

$$z_{x,y}^{(l,k)} = \max_{0 \le i < r_k} \max_{0 \le j < r_k} z_{x \cdot r_k + i, y \cdot r_k + j}^{(l-1,k)} \tag{5}$$

where r_k denotes the size of the k-th pooling filter, which is set to 2 in our model.

The final feature maps are then turned into a vector and passed through an MLP with several hidden layers. In this paper, we use only two fully-connected layers. For the final output, a single unit is connected to all units of the last hidden layer:

$$s = W_2 \delta(W_1 \cdot z + b_1) + b_2 \tag{6}$$

where W_1 and W_2 are the weights of fully-connected layers with b_1 and b_2 are the bias terms.

Optimization and Model Training As the task is formalized as a ranking problem, we can utilize pairwise ranking loss such as hinge loss for training. Given a triple (d_q, d_+, d_-), where document d_+ is ranked higher than document d_- with respect to query document d_q, the loss function is defined as:

$$Loss(d_q, d_+, d_-) = max(0, 1 - s(d_q, d_+) + s(d_q, d_-))$$

where $s(d_q, d_+)$ and $s(d_q, d_-)$ are the corresponding predicted similarity scores.

Since the size of expert-finding datasets we use is relatively small, for experiments on these datasets we train our model on a task called citation prediction. Given the abstracts of three documents, the model needs to give higher rank to the document that has citation relationship with the query. Obviously, to complete the task, the model also needs to compute the relevance of two documents. The training dataset of our model is collected from an academic search system Aminer[21].

Training is done through stochastic gradient descent over mini-batches, with the Adagrad update rule [5]. It achieves good performance with a learning rate of 0.001. For regularization, we employ dropout [7] on the penultimate layer, which prevents co-adaptation of hidden units by randomly dropping out, i.e., set to zero. To avoid over-fitting, we apply an early-stop strategy [19].

4 Experiments

4.1 Experimental Setup

To evaluate the proposed model, we conduct the experiments of expert finding problem on three datasets.

Datasets As the paper-reviewer assignment is private and interest-related, there is no publicly labeled dataset. It is difficult to create one as well. Therefore, for the purpose of evaluation, we collect three datasets from an online system and human judgments.

Paper-Reviewer: This dataset comes from an online system which connects journal editors with qualified journal reviewers. The system recommends journal submissions that are posted by journal editors to qualified reviewers who are willing to review. It includes 540 papers submitted to ten journals and 2,359 experts' invitation responses. Among these responses, 953 are "agree", while the rest are viewed as "decline" (including "unavailable" and "no response"). Basically, we consider "agree" as relevant and "decline" as irrelevant.

Topic-Expert: This dataset is based on papers from Aminer[21]. It consists of 86 papers with 20 candidate experts for each query. In this dataset, we follow a traditional expertise matching setting such as [22, 4].

Patent-Relevance: This dataset is based on documents of patents, which comes from the Patent Full-Text Datasets of the United States Patent and Trademark Office. It consists of 67 patent queries with 20 candidate patents for each query.

In Topic-Expert and Patent-Relevance, we gather relevance judgments from college students and experts on patent analysis as the ground truth. The relevance is simply expressed as binary: relevant or irrelevant. In these three datasets, only the first 64 words are chosen for the abstracts of the papers or the patent documents.

Comparison Methods We compare the following methods in the experiment:

- **BM25**: The relevance score between query q and document d is measured by the BM25 score, where each word in query q is considered as a keyword. The relevance score is defined as:

$$\text{BM25}(q,d) = \sum_{w \in q} \text{IDF}(w) \cdot \frac{\mathcal{N}_d^w \cdot (k_1+1)}{\mathcal{N}_d^w + k_1 \cdot (1 - b + b \cdot \frac{\mathcal{N}_d}{\lambda})}$$

  where $\text{IDF}(\cdot)$ is inverse document frequency, $\mathcal{N}_d^w$ is word w's frequency in document d, and $\mathcal{N}_d$ is the length of d. We set $k = 2$, $b = 0.75$, and λ as the average document length;
- **MixMod[22]**: While another setting is the same as BM25, the relevance score between query q and expert e is defined as:

$$P(q|e) = \sum_{d_j \in D_i} \sum_{m=1}^{k} \prod_{t_i \in q} P(t_i|\theta_m) P(\theta_m|d_j) P(d_j|e)$$

  where $P(t_i|\theta_m)$ denotes the probability of generating a term given a theme θ_m and $P(\theta_m|d_j)$ denotes the probability of generating a theme given a document d_j;
- **Doc2Vec[12]**: We represent each document via Paragraph Vector model. The similarity score between two documents is produced by the cosine metric of their representations;

Table 1. Results of relevance assignment(%). NG is the simplify of NDCG.

Method	**Paper-Reviewer**			**Topic-Expert**			**Patent-Relevance**		
	NG@1	NG@3	NG@5	NG@3	NG@5	NG@10	NG@3	NG@5	NG@10
BM25	36.9	40.4	41.6	64.2	63.7	66.2	49.8	57.7	62.4
MixMod	35.3	39.9	42.5	57.6	58.2	58.0	45.6	51.3	56.5
Doc2Vec	34.5	41.3	43.4	60.9	63.8	66.2	44.2	48.0	54.7
WMD	41.2	47.7	49.8	62.5	64.5	66.8	57.4	58.5	61.9
LSTM-RNN	34.1	38.6	42.1	58.6	60.5	63.7	58.3	58.1	65.0
MatchPyramid	40.0	47.8	48.8	66.3	66.0	68.7	57.6	59.4	62.9
EFCNN	**43.4**	**49.6**	**52.3**	**67.7**	**67.1**	**70.8**	**59.8**	**61.4**	**65.8**

- **WMD[10]**: We apply the Word Mover's Distance (WMD) to measure the similarity between two documents. The WMD is the minimum distance required to transport the words from one document to another based on word embeddings;
- **LSTM-RNN[16]**: [16] adopts an LSTM to construct sentence representations and uses cosine similarity to output the similarity score;
- **MatchPyramid[17]**: The MatchPyramid is a standard CNN built on the standard similarity matrix to get the similarity score.

As for neural models, we can see that Doc2Vec and LSTM-RNN are all sentence representation models, while WMD, MatchPyramid and EFCNN are the interaction-based model.

Parameter settings In our model, there are two restricted convolutional layers, both having 64 filters. All filters are set to 3×7 and Batch Normalization[8] is adding to all restricted convolutional layers. The number of hidden units of the fully-connected layer is set to 256. And the hyperparameters α and β are set to 0.2 and 2.0 respectively, which is discussed in Section 4.2.

The Word2Vec embedding is learned on AMiner data [21]. The embedding is trained using the Skip-gram architecture [14]. For a fair comparison, word embedding of all comparison models is the same as that of the proposed model and the dimension number of word embedding is set to 150.

Evaluation Metric Formalized as a ranking problem, the output is a ranked list of experts, where the order depends on the maximum similarity score of their documents regarding the query. The goal is to rank the positive one higher than the negative ones. Therefore, we use NDCG@n [9] as an evaluation metric. Formally, we have:

$$\text{NDCG@}n = \frac{\sum_{i=1}^{n} \frac{2^{r_i}-1}{\log_2(i+1)}}{\sum_{i=1}^{|R|} \frac{2^{R_i}-1}{\log_2(i+1)}}$$

where r_i is the relevance of i-th expert in the output and R represents the list of experts (ordered by their relevant) in the length of n.

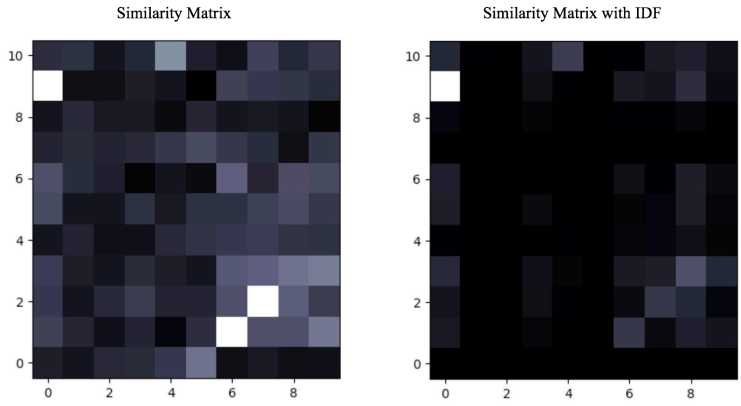

Fig. 4. The visualization result of two matrices based on a document pair. The brighter the pixel is, the larger value it has. The document pair is as follows: D1: Privacy is an enormous problem in online social networking sites. D2: While online social networks encourage sharing information, they raise privacy issues.

4.2 Results and Discussion

Performance Analysis We compare the performance of all methods on three datasets. Table 1 shows the ranking accuracy of different methods in terms of NDCG, where measures are averaged for all queries on each dataset. Roughly speaking, the neural methods (such as WMD and EFCNN) outperform the traditional methods in most cases. Only taking the exact word matching into account, traditional methods will lose important information easily, while neural methods based on word embedding can learn better representations and deal with the mismatch problem effectively. As for neural models, we can see that interaction based models, such as WMD and MatchPyramid, perform better than representation based models. This is mainly because these model can capture more detailed information from the interaction of documents.

On Paper-Review and Topic-Expert, we also see that our model achieves significant improvement compared to all the baselines. On Paper-Reviewer, EFCNN clearly outperforms the comparison methods in all by 2.2% and 1.9% ($p-$value $\ll 0.01$ in both cases by t-test) in terms of NDCG@1 and NDCG@5 respectively. It indicates that restricted convolution deals with sparse but position-depended signals effectively.

How the Similarity Matrix with IDF works To have a better understanding of how the similarity matrix with IDF works, we show the pixel images of matrices without/with IDF in Figure 4. From the pixel image, we can see that similarity matrix focuses more on word similarity. While the similarity matrix with IDF focuses more on details, it will only show the significant result when two words are similar and when both of them are important to the document.

Table 2. The effect of hyperparameters α and β on Topic-Expert.

	NG@3	NG@10
EFCNN-Lin(α=0.2)	67.7	70.8
EFCNN-Lin(α=0.5)	66.2	68.7
EFCNN-Lin(α=1.0)	64.5	66.4
EFCNN-Exp(β=1.0)	66.3	68.7
EFCNN-Exp(β=1.5)	66.5	68.0
EFCNN-Exp(β=2.0)	67.3	69.5

These two matrices support each other, and we will lose some information if we drop any one of them.

Sensitivity Analysis of Hyperparameters Since there are two different decay functions, our model has two versions, denoted as EFCNN-Lin and EFCNN-Exp. There are two hyperparameters α and β in EFCNN-Lin and EFCNN-Exp, respectively. We further study the effect of different choices of α and β. The experimental result is listed in Table 2. The results indicate that the best model setting is always encouraging weight decay. Therefore, the consideration of weight decay along with positions in convolution is necessary.

5 Conclusions

In this paper, we study the problem of expert finding. We formalize the problem and propose a deep learning model based on restricted convolutional neural networks. We prove that the proposed model can capture the relevant information between two documents. Compared to several state-of-the-art models, our model can significantly improve the performance of expert finding.

The problem of expert finding represents an interesting and important research direction. In future work, it would be intriguing to investigate a deep architecture to learn expert representations directly. It would also be interesting to study how to incorporate both network information and content information together to better learn the expert representations.

References

1. Balog, K., Azzopardi, L., De Rijke, M.: Formal models for expert finding in enterprise corpora. In: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 43–50. ACM (2006)
2. Basu, C., Hirsh, H., Cohen, W.W., Nevill-Manning, C.: Recommending papers by mining the web (1999)
3. Cao, Y., Liu, J., Bao, S., Li, H.: Research on expert search at enterprise track of trec 2005. In: TREC (2005)

4. Deng, H., King, I., Lyu, M.R.: Formal models for expert finding on dblp bibliography data. In: ICDM'08. pp. 163–172 (2008)
5. Duchi, J.C., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research **12**, 2121–2159 (2011)
6. Dumais, S.T., Nielsen, J.: Automating the assignment of submitted manuscripts to reviewers. In: Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 233–244. ACM (1992)
7. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580 (2012)
8. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
9. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of ir techniques. ACM Transactions on Information Systems (TOIS) **20**(4), 422–446 (2002)
10. J.Kusner, M., Sun, Y., I.Kolkin, N., Q.Weinberger, K.: From word embeddings to document distances. In: International Conference on Machine Learning. vol. 15, pp. 957–966 (2015)
11. Karimzadehgan, M., Zhai, C., Belford, G.: Multi-aspect expertise matching for review assignment. In: Proceedings of the 17th ACM conference on Information and knowledge management. pp. 1113–1122. ACM (2008)
12. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: International Conference on Machine Learning. pp. 1188–1196 (2014)
13. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
14. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
15. Mimno, D., McCallum, A.: Expertise modeling for matching papers with reviewers. In: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 500–509. ACM (2007)
16. Palangi, H., Deng, L., Shen, Y., Gao, J., He, X., Chen, J., Song, X., Ward, R.: Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP) **24**(4), 694–707 (2016)
17. Pang, L., Lan, Y., Guo, J., Xu, J., Wan, S., Cheng, X.: Text matching as image recognition. In: AAAI. pp. 2793–2799 (2016)
18. Petkova, D., Croft, W.B.: Hierarchical language models for expert finding in enterprise corpora. International Journal on Artificial Intelligence Tools **17**(01), 5–18 (2008)
19. Prechelt, L.: Automatic early stopping using cross validation: quantifying the criteria. Neural Networks **11**(4), 761–767 (1998)
20. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Information processing & management **24**(5), 513–523 (1988)
21. Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., Su, Z.: Arnetminer: extraction and mining of academic social networks. In: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 990–998. ACM (2008)
22. Zhang, J., Tang, J., Liu, L., Li, J.: A mixture model for expert finding. In: PAKDD'08. pp. 466–478 (2008)