Hierarchical Expert Recommendation on Community Question Answering Platforms

Amirabbas Jalali^{†,*}, Roy Ka-Wei Lee[‡]

[†] University of Saskatchewan

[‡] Singapore University of Technology and Design

Keywords: Deep Learning, Expert Recommendation, Hierarchical Attention-Based Networks

1. Introduction

The community question answering (CQA) platforms, such as Stack Overflow (SO), have become the primary source of answers to most questions in various topics. CQA platforms offer an opportunity for sharing and acquiring knowledge at a low cost, where users, many of whom are experts in a specific topic, can potentially provide high-quality solutions to a given question. The main problem in CQA platforms is delivering the appropriate content to the best possible audience. Usually, many questions remain unanswered or poorly answered because experts do not receive related questions. In this project, we aim to solve this problem using a combination of deep neural networks with a hierarchical attention-based network [1]. First, we learn the latent user and question features from the deep neural network. Next, we learn each question's text feature with the Word2Vec embedding model [2] and feed it to a hierarchical attention-based network to create a dense embedding of the question text. This network utilizes two separate attention layers to learn the importance of words and sentences [3]. Finally, we combine these features to perform the task of recommending questions to the users. The latent user and question embeddings are fused using our novel deep learning architecture. Our preliminary results showed that our approach improved expert recommendation and outperformed the baseline methods.

2. Methods

This section proposes the Hierarchical attention-based Expert Recommendation (HER) framework, which learns the question representation based on the question tags, words, and sentences and determines the user representation based on the interactions between users and questions and available static features in the dataset. The architecture of HER is shown in 1.

HER uses a deep neural network (DNN) to capture user and question static features and interactions to create a dense representation of non-text data available in the Stack Overflow dataset. The deep neural network (DNN) component consists of several hidden layers to extract the latent features and a dropout layer to help the model generalize and avoid over-fitting. This DNN component introduces two major hyper-parameters in our model: the dimension of hidden layers and dropout rate, which was optimized using a validation dataset. After learning the user and question latent features using the user and question metadata, we learn the question texts' features using a hierarchical approach. We first train a Word2Vec [2] model that enables us to find semantically similar words by computing the cosine similarity between word vectors. We use the gensim [4] library to train Word2Vec model. We set the length of the vector to be 100, as [5] suggests. We leave this vector

* amirabbas.jalali@usask.ca

Figure 1. HER Architecture

trainable in the learning phase to have a fine-tuned vector based on our data. Now we have all the inputs available. As shown in Figure 1, HER has two separate parts with different inputs. The first part is DNN to find a dense representation of the user and question static features and interactions we call it Dense Encoder. The second part is learning a dense representation of question texts. We call it Hierarchical Encoder. In the following paragraphs, we will describe these two parts.

2.1. Dense Encoder

HER utilizes a deep neural network (DNN) to extract the latent user and question features. It contains a large set of hidden layers, each has a specific number of units to help extract latent features. We set the number of hidden layers to 1024, and for each layer, we decide to have 1024 units. The dropout rate for the dropout layer is set to 0.2. These hyperparameters are tuned based on the validation set. The output of this DNN is as same as the units per layer. Therefore, we added a fully connected layer on top to have one number as the final output. We used the sigmoid activation function to bound this number and made it comparable to the other part of our architecture.

$$MLP(V, AFunc) = AFunc(WV + b) \quad (2.1)$$

Our DNN consist of multiple Multi-layer Perceptron (MLP), Where V is the input vector, W is the weight matrix, b is the bias, and $AFunc$ is the activation function.

2.2. Hierarchical Encoder

The first layer in Hierarchical Encoder is word embedding. We use the Word2Vec model to convert each word to a vector. Because Stack overflow has a limit of 150 characters for question title and virtually no limit on the question body, we choose 12 words as the maximum length of each sentence and a total of ten sentences in every question. Our experiments showed us the essential words in every question usually happen in the first few sentences. The next layer is a Bidirectional Gated Recurrent Unit (GRU) [6] to extract the information of each word and give us a dense representation of words in each sentence. To emphasize the most important words in every sentence, we add an attention layer to maximize the words' information. Now, we have a dense representation of words in each sentence. Therefore, we add another Bidirectional GRU layer to do the same on the sentences and narrow the model output toward the most important sentences using another attention layer. Finally, To make it comparable to user features, we flatten the output, feed it to a fully connected layer, and bind it with the sigmoid function.

Table 1. Dataset Summary

Feature	Min	Mean	Max
Experts Answers	21	72	4507
Reputation	1	25,278	1,166,685
Views	23	3,551	1,911,062
Upvote	0	1,218	53,163
Downvote	0	716	77,537

Table 2. User & Question Count

# Users	# Questions	Sparsity
5,448	392,798	99.98%

$$L_{w|s} = MLP(V_{w|s}, \tanh) \quad (2.2)$$

$$A_{w|s} = \frac{\exp(L_{w|s}^T E_{w|s})}{\sum_{W|S} \exp(L_{w|s}^T E_{w|s})} \quad (2.3)$$

$$S_{w|s} = \sum_{W|S} A_{w|s} V_{w|s} \quad (2.4)$$

In the first step of attention, We feed the extracted word or sentence vector V through a one-layer MLP to get L as a latent representation of the word or sentence. Secondly, we compute the word or sentence’s attention value as the similarity of V with and the embedding E to get a normalized attention value A after feeding it to a softmax function. Lastly, we measure the summarize vector S as a weighted sum of the word or sentence vectors based on the attention values. The embedding vectors E are initialized based on pre-learned values and jointly fine-tuned during the training process.

2.3. Feature Combination

Now we have two dense representations, and we have to relate the model outputs to the target. To find the relatedness, we add our two outputs together and connect them to one cell with a sigmoid activation function to tell us how related a user and a question are to each other. Finally, we rank and find the most appropriate user to recommend. For the learning phase, we put our binary labels as output to train our model. We use binary cross-entropy as our loss function.

$$p(\gamma, \bar{\gamma} | \mathbf{U}, \mathbf{Q}, \theta) = \Pi_{(u,q) \in \gamma} \hat{y}_{uq} \Pi_{(u,q) \in \bar{\gamma}} (1 - \hat{y}_{uq}) \quad (2.5)$$

$$L = - \sum_{(u,q) \in \gamma \cup \bar{\gamma}} y_{uq} \log \hat{y}_{uq} + (1 - y_{uq}) \log (1 - \hat{y}_{uq}) \quad (2.6)$$

Where Q and U denote the set of questions and answers, $\hat{y}_{uq}$ is the predicted relatedness, y_{uq} is the actual label, γ is related questions, $\bar{\gamma}$ is negative samples and θ is our model parameters.

3. Results

In this section, we first introduce our experiment setup, then we show our results. We use the Stack Overflow dataset, including one year from 2018 to 2019. We run our experiments on users who have answered more than 15 questions.

We used the Word2Vec model to convert each word in a question title to a vector. We learnt this model on our corpus to have domain-specific vectors that enable us to capture similarities in this context better than other pre-trained models.

Table 3. Main Data-set Results

Model	HR ¹ @1	HR@5	HR@10	HR@50	NDCG ² @5	NDCG@10	NDCG@50	MRR ³ @100
MF	0.0100	0.0498	0.1001	0.5000	0.0295	0.0456	0.1291	0.0420
PNN	0.0429	0.1227	0.1850	0.5662	0.0833	0.1033	0.1835	0.0692
DeepFM	0.0508	0.1285	0.1926	0.5828	0.0900	0.1106	0.1928	0.0729
HER	0.0527	0.1375	0.1999	0.5884	0.0959	0.1159	0.1983	0.0758

¹ Hit Rate, ² Normalized Discounted Cumulative Gain, ³ Mean Reciprocal Rank

The goal of our recommendation task is to recommending a more relevant user (expert) to the specific question. Therefore, we only care about the presence of the expert in our recommended user list and a rank-based measures are sufficient for our purpose. As we do not have any negative samples on our dataset, we randomly generate 100 users as our negative samples. We put the expert between these negative samples to see how our model can rank the expert higher than the others.

For baselines, we use the MF, Deep Factorization Machine (DeepFM) [7], and Product-Based Neural Network (PNN) [8]. DeepFM utilizes Factorization Machine (FM) [9] to extract latent features for questions and users based on their interaction and features. It also uses a Dense Multi-layer Perceptron to capture high order relevancy between users and questions to improve the result further. PNN tries to address the problems associated with the Deep part of the DeepFM, and it showed that it could slightly improve the results over DeepFM. We use the same setting for all of our experiments to evaluate these methods and compare them. Table 3 shows the preliminary experiment results. We observe that HER outperforms all the baselines. DeepFM outperforms PNN. However, The HER achieved better results on the experiment, and it shows that the Hierarchical part can significantly improve the results.

4. Discussion and Future Works

In the current recommendation setting, we assume that every user has an equal boolean relationship to each question. i.e., each user has the same opportunity to answer a given question. However, the real-world setting may be more complicated than this. For instance, it is unlikely a user would answer a question who has been answered by another user, especially when the provided answer has been accepted by the questioner. Therefore, we might need to consider the sequential, temporal, and dynamic attributes in future recommendation.

We would need to perform more ablation studies to understand the effectiveness of different components in HER model and test the model’s robustness by conducting experiments on other CQA datasets. We could also design experiments to highlight the explainability of our model; the hierarchical attention could be visualized to showcase which questions or keywords helped to decide if we should recommend a particular question to the user.

References

- [1] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy. “Hierarchical Attention Networks for Document Classification”. In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, June 2016, pp. 1480–1489. DOI: [10.18653/v1/N16-1174](https://doi.org/10.18653/v1/N16-1174). URL: <https://www.aclweb.org/anthology/N16-1174>.
- [2] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. “Distributed Representations of Words and Phrases and their Compositionality”. In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and

- K. Q. Weinberger. Curran Associates, Inc., 2013, pp. 3111–3119. URL: <http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf>.
- [3] C. Wu, F. Wu, J. Liu, and Y. Huang. “Hierarchical User and Item Representation with Three-Tier Attention for Recommendation”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 1818–1826. DOI: [10.18653/v1/N19-1180](https://doi.org/10.18653/v1/N19-1180). URL: <https://www.aclweb.org/anthology/N19-1180>.
 - [4] R. Rehůřek and P. Sojka. “Software Framework for Topic Modelling with Large Corpora”. English. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. <http://is.muni.cz/publication/884893/en>. Valletta, Malta: ELRA, May 2010, pp. 45–50.
 - [5] Y. Kim. “Convolutional Neural Networks for Sentence Classification”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. 2014, pp. 1746–1751. URL: <http://aclweb.org/anthology/D/D14/D14-1181.pdf>.
 - [6] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation”. In: 2014. arXiv: [1406.1078](https://arxiv.org/abs/1406.1078) [cs.CL].
 - [7] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He. “DeepFM: A Factorization-Machine based Neural Network for CTR Prediction”. In: 2017. arXiv: [1703.04247](https://arxiv.org/abs/1703.04247) [cs.IR].
 - [8] Y. Qu, H. Cai, K. Ren, W. Zhang, Y. Yu, Y. Wen, and J. Wang. “Product-based Neural Networks for User Response Prediction”. In: 2016. arXiv: [1611.00144](https://arxiv.org/abs/1611.00144) [cs.LG].
 - [9] S. Rendle. “Factorization Machines”. In: *Proceedings of the 2010 IEEE International Conference on Data Mining. ICDM '10*. USA: IEEE Computer Society, 2010, 995–1000. ISBN: 9780769542560. DOI: [10.1109/ICDM.2010.127](https://doi.org/10.1109/ICDM.2010.127). URL: <https://doi.org/10.1109/ICDM.2010.127>.