Hierarchical Expert Recommendation on Community Question Answering Platforms

A thesis submitted to the
College of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements
for the degree of Master of Science
in the Department of Computer Science
University of Saskatchewan
Saskatoon

By
Amirabbas Jalali

Abstract

The community question answering (CQA) platforms, such as Stack Overflow, 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. Many recommendation methods have been proposed to match questions to potential good answerers. However, most existing methods have focused on modelling the user-question interaction — a user might answer multiple questions and a quetion might be answered by multiple users — using simple collaborative filtering approaches, overlooking the rich information in the question's title and body when modelling the users' expertise.

This project fills the research gap by thoroughly examining machine learning and deep learning approaches that can be applied to the expert recommendation problem. It proposes a Hierarchical Expert Recommendation (HER) model, a deep learning recommender system that recommends experts to answer a given question in the CQA platform. Although choosing a deep learning over a machine learning solution for this problem can be justified considering the degree of complexity of the available datasets, we assess performance of each family of methods and evaluate the trade-off between them to pick the perfect fit for our problem.

We analyzed various machine learning algorithms to determine their performances in the expert recommendation problem, which narrows down the potential ways for tackling this problem using traditional recommendation methods. Furthermore, we investigate the recommendation models based on matrix factorization to establish the baselines for our proposed model and shed light on the weaknesses and strengths of matrix-based solutions, which shape our final deep learning model. In the last section, we introduce the Hierarchical Expert Recommendation System (HER) [34] that utilizes hierarchical attention-based neural networks to represent the questions better and ultimately model the users' expertise through user-question interactions. We conducted extensive experiments on a large real-world Stack Overflow dataset and benchmarked HER against the state-of-the-art baselines. The results from our extensive experiments show that HER outperforms the state-of-the-art baselines in recommending experts to answer questions in Stack Overflow.

Acknowledgements

First, I would like to thank my supervisor, Dr. Roy Ka-Wei Lee, for his belief in me and his excellent advice and guidance during my M.Sc. program. His support and mentorship have guided me through all my research steps. Special thanks to Prof. Julita Vassileva, Prof. Michael Horsch, and Prof. Ralph Deters for their priceless comments on my research. I am thankful to my caring wife, Mina Mousavifar, for both her emotional and academic support.

Contents

List of Tables

List of Figures

List of Abbreviations

CDAE	Collaborative Denoising Autoencoder
CF	Collaborative Filtering
CQA	Community Question Answering
DeepFM	Deep Factorization Machine
DNN	Deep Neural Network
ERS	Expert Recommendation System
FM	Factorization Machine
HAN	Hierarchical Attention Network
HER	Hierarchical Expert Recommendation
HR	Hit Ratio
MF	Matrix Factorization
MRR	Mean Reciprocal Rank
NCF	Neural Collaborative Filtering
NDCG	Normalized Discounted Cumulative Gain
NLP	Natural Language Processing
SO	Stack Overflow
SVD	Singular Value Decomposition

1 Introduction

1.1 Motivation

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 goal of CQA platforms is to provide high-quality answers fast to questions raised by its users. However, it is challenging to achieve this goal and many questions remain unanswered or poorly answered because the experts did not receive the questions relevant to their expertise. This motivated the need for expert recommendation, which aims to find the relevant experts to answer a given question.

Expert recommendation is a widely studied research area. Existing studies have attempted to apply collaborative filtering [39] or simple content-based approaches such as modelling users' topical expertise [79, 83, 50] to perform expert recommendations. Nevertheless, the existing methods are ineffective in leveraging the rich information in the questions to model users' expertise. For instance, the existing methods assume the question's textual description as a bag of words or simple text sequences, ignoring the importance of different sentences and the context of the words used in the sentences.

In this project, first, we review machine learning techniques and traditional recommendation methods to shape the basis of our study. Machine learning methods usually need inputs processed with combinations of feature engineering techniques. On the other hand, deep learning methods take the raw input and process the input and adjust their respective influence on the final output during the learning procedure. We refer to the Matrix Factorization technique in collaborative and content-based filtering by traditional recommendation.

Then we aim to address the research gap by proposing the Hierarchical Expert Recommendation (HER) [34] model, a hierarchical attention-based network [80] that learns latent representations of questions and users' expertise to perform expert recommendations. Specifically, HER first learns latent user and question representations from their meta-data (e.g., question's vote counts, user's reputation scores, etc.) using a deep neural network. We published our architecture as "Hierarchical Expert Recommendation on Community Question Answering Platforms" in the "Proceedings of the Canadian Conference on Artificial Intelligence 2021".

The question title and main body contain rich information about the questions themselves and provide important information about its answerers' expertise. For instance, a user who answered a Python data processing-related question is likely to have the expertise in performing data processing with Python. To extract the rich information from the questions, HER first initializes each question's text features with Word2Vec embeddings [48] and feeds them to a hierarchical attention-based network [76] to learn an attended representation. The attended representation considers the importance of the sentences and words that best represent the question for the expert recommendation task. Finally, the hierarchically-attended question representation are combined with the latent representations learned from the questions and users' meta-data to predict if a given question is likely to be answered by a given user.

In order to justify the validity of the approach and the final proposed model to solve expert recommendation problem, we designed several experiments to help us analyze the strengths and weaknesses of the different methods and strategies to tackle this problem. We first examine the most suitable machine learning methods for expert recommendation and thoroughly explain why they are not proper candidates for this purpose. In the subsequent chapter, we discuss the traditional recommendation methods like matrix factorization and basic deep learning recommendation methods. Based on the results of our experiments, we found that using a complex network for the deep part of the model can gain a significant improvement to the final result. Finally, considering all the knowledge from previous experiments, we developed our novel architecture that utilizes state-of-the-art techniques in this field of study. We conducted extensive experiments and different ablation investigations to prove the supremacy of our proposed model on real-world data.

1.2 Research Goals

Knowledge acquisition is challenging, even considering the recent multi-domain social networks and community question answering (CQA) platforms available for individuals and businesses. Usually, the knowledge acquisition cycle has three main stages: (a) Asking a question or raising a concern and exposing it to other members; (b) Providing answers by the domain experts to address the stated problem; (c) Evaluating the provided answers and deciding on the best one. There are two main challenges in the knowledge acquisition cycle:

1. Finding the most appropriate candidate who is an expert in the domain that can potentially provide high-quality answers to the provided question. It can be tricky due to the numerous participants and virtually unlimited domains in an online community question answering platform like Stack Overflow.

2. Increasing question exposure to experts. Research [5] shows that approximately 7.5% of Stack Overflow questions remain unanswered, and one of the reasons is that the question can not attract experts. In the other word, experts are not exposed to the questions.

Recommendation systems can be used to recommend questions to experts. These recommenders usually are called Expert Recommendation Systems (ERS). ERS tries to create a convenient experience for users who ask or answer questions in CQA platforms by linking the most appropriate experts with each question and increasing the chance of answering the question correctly and adequately. These expert recommendation systems usually use the text available in questions as data input. However, recently researchers have found that using hierarchical attention-based networks can increase the recommended items' accuracy by taking into account the different importance of the words and sentences in the question. These networks consider the text as a hierarchy from word level to sentence level. This feature distinguishes the most important aspect of the available corpora. With this in mind, we state the following particular research goals.

- **Review expert recommendation systems and their varieties:** As we aim to design new and more effective ERS, it is crucial to have a substantial understanding of the expert recommendation systems. In order to achieve this, we review the existing literature on recommendation systems, particularly the ERS to gain a good perspective about the various available techniques in this field, how they work, and how they help users. Using hierarchical attention-based models in recommendation systems, particularly expert recommendation systems, is comparatively new. There are limited or inadequate definitions of hierarchical attention-based recommendations. We fill this gap by reviewing the basic definitions of the data's hierarchy and attention-based networks before developing a hierarchical attention-based expert recommendation system.

- **Examining the techniques in expert recommendation systems:** Matrix Factorization (MF), Factorization Machine (FM) and their variants like Deep Factorization Machine (DeepFM) are currently the state-of-the-art methods in recommendation systems. To fully understand these baseline methods and find their weaknesses, we start with their elementals and analyze their use in developing recommendation systems.

- **Justifying the proposed methodology:** It is always a controversial topic to which extend we should use deep learning to solve our problems. In this study, we aim to justify our approach by providing a thorough analysis of different techniques with different levels of complexity. The information gained by these experiments helps us shape our expectations of each technique and guide us to the best possible method.

- **Develop more effective expert recommendation system:** Current expert recommendation systems do not utilize texts or use text as a long sequence of words without considering their importance. The hierarchical attention-based expert recommendation system can use the text's hierarchy and extract the difference in importance of a question's words and sentences. This novel architecture can take this information into account to learn the various features that relate users to the questions. Our primary goal in developing a new ERS is to increase the question exposure to the domain experts and increase the accuracy of the recommendation.

- **Analyze the performance of the proposed expert recommendation system:** According to the expert recommendation systems' characteristics, by recommending a list of experts to answer a question, we evaluate the proposed expert recommendation systems' effectiveness in their ranking-based methods, e.g., Normalized Discounted Cumulative Gain (NDCG), Mean Reciprocal Rank (MRR) and Hit Ratio (HR). The accuracy of these models are compared to each other and the aforementioned baselines.

1.3 Structure of the Thesis

The thesis structure is as follows:

- **Chapter 2: Background and Related Works**
  A comprehensive study about the recommendation systems, their classification, and their various techniques is conducted. Furthermore, this chapter covers many recently published studies about applying these techniques in expert recommendation systems.

- **Chapter 3: Research Methodology and Implementations**
  This chapater begins with our dataset analysis and the way this data is gathered. It covers comprehensive explanations of the proposed recommendation systems. The intention behind choosing each of the several machine learning algorithms in the expert recommendation is explained in detail. It introduces two types of recommendation systems. One is based on traditional techniques, and the other is a fusion of traditional and deep recommendation systems. This section thoroughly analyzes their respective characteristics in tackling this problem. At last, our main contribution, the hierarchical expert recommendation system, is introduced in detail.

- **Chapter 4: Experiment Results and Discussion**
  We focus on explaining the experimental settings for each method and analyzing the results of all the proposed recommendation systems, including the related discussions.

- **Chapter 5: Conclusion and Future Directions**
  It concludes the primary research subjects of the thesis. In addition, the potential future paths for further research are noted.

2 Background and Related Works

2.1 Recommendation Systems

Nowadays, we are overwhelmed by data we can access over the internet. Every second, million terabytes of data are generated, and finding what we need has become more complicated. This massive flow of data can not be filtered or validated before it becomes available for consumers; thus, lots of spam and not related content can appear in user queries. Recommendation systems are designed to help us reach the most appropriate content that conveniently fulfills our needs. For example, finding a movie to watch or finding a good place to dine, even finding someone who can accurately answer a question can be achieved using recommendation systems. In this section, we will review the traditional recommendation systems and their various recommending techniques. As this research study focuses on developing deep hierarchical attention-based expert recommendation systems, we will also review the deep learning-based recommendation systems and hierarchical attention-based networks.

2.1.1 Introduction to Recommendation Systems

Recommendation systems are tools designed to provide personalized suggestions to system users for specific items [11]. Items generally refer to products or services, and users are the individuals or businesses making queries to find their desirable items [59]. Recommendation systems are a necessity in the information space nowadays, and they are crucial to face particular challenges in the platform, such as information overload [3]. Information overload can be defined as the amount of data that an individual or, in other words, a system user cannot entirely observe when he or she wants to make a decision. This term conveys a similar meaning to various fields such as knowledge overload systems [22] that mostly occurs in CQA platforms. For instance, the number of available questions on the CQA platforms is increasing exponentially. This rapid growth in the accessible questions may confuse experts, and leads to frustrating experiences since the overwhelming number of questions makes finding the answerable ones too complicated and even impossible. Recommendation systems can help users or, in our case, experts avoid confusion and frustration caused by searching for a desirable question by providing an exclusive and personalized list of available choices related to them. Recommendation systems are used to provide suggestions to the end-users wherever users interact with items or other users, independent from the domains. For example, movies and TV series [26], music [15], books [70] and foods [24] are some of the application domains of the recommendation systems. Consequently, recommendation systems are crucial parts of popular online platforms and social applications such as Netflix, Spotify, Amazon, and Yelp. Bobadilla et al. [10] mention several different cases that recommendation systems perform a vital role in shaping a satisfying experience for the end-users and enhancing the performance and revenue of various businesses which utilize them in their respected platforms. As discussed so far, Recommendation systems are indispensable parts of interactive social platforms and are implemented to provide a list of suggested personalized and related items to end-users. However, the domains in which recommendation systems are used can be utterly different from each other. For example, the principles for recommending a restaurant and recommending a scientific paper are totally different; accordingly, researchers proposed various types of recommendation systems depending on the domain. The most popular classification separates recommendation systems into content-based, collaborative, demographic, and hybrid algorithms [2]. This classification is shown in figure 2.1. In the following sections, the stated recommendation strategies are explained.

2.1.2 Traditional Recommendation Systems

There are innumerable research in recommendation systems, and consequently, many diverse recommendation techniques have been introduced so far. Recommendation methods can be categorized into memory-based and model-based approaches from an algorithmic standpoint. Memory-based approaches rely on heuristics to compare objects and determine how relevant they are to a specific end-user. These methods typically work with the user and item interaction matrix. As a result, a memory-based approach is easy to implement due to its simple logic and focus on a single task. However, this simplicity comes at a cost, and it is definitely less flexible compared to a model-based approach. On the other hand, model-based methods employ more complex algorithms than memory-based methods, such as various machine learning or deep learning tools, to build a predictor model from the data. These methods create a predictor model that produces item suggestions for a specific end-user using information from the recommendation systems. In figure 2.1, the well-known memory-based and model-based methods are listed. Other classifications can further categorize recommendation systems, and as previously discussed, the most commonly used classification separates all proposed methods into three main categories [71]:

- Content-based recommendation systems: suggest an item or a group of items to a particular user based on the previously selected items by the user. The previously selected items shape the user preference, and the suggested items are the most similar items to the user's preferences.
- Collaborative filtering-based recommendation systems: suggest an item or a group of items to a particular user based on the rating similarities between the user and other users. The user rating to the items shapes the user's preferences. The suggested items are the items that have high ratings by users who have similar rating patterns to the user's rating pattern.
- Hybrid recommendation systems: These recommendation systems are combinations of other recommendation systems and usually are seen as a combination of collaborative with content-based approach [6].

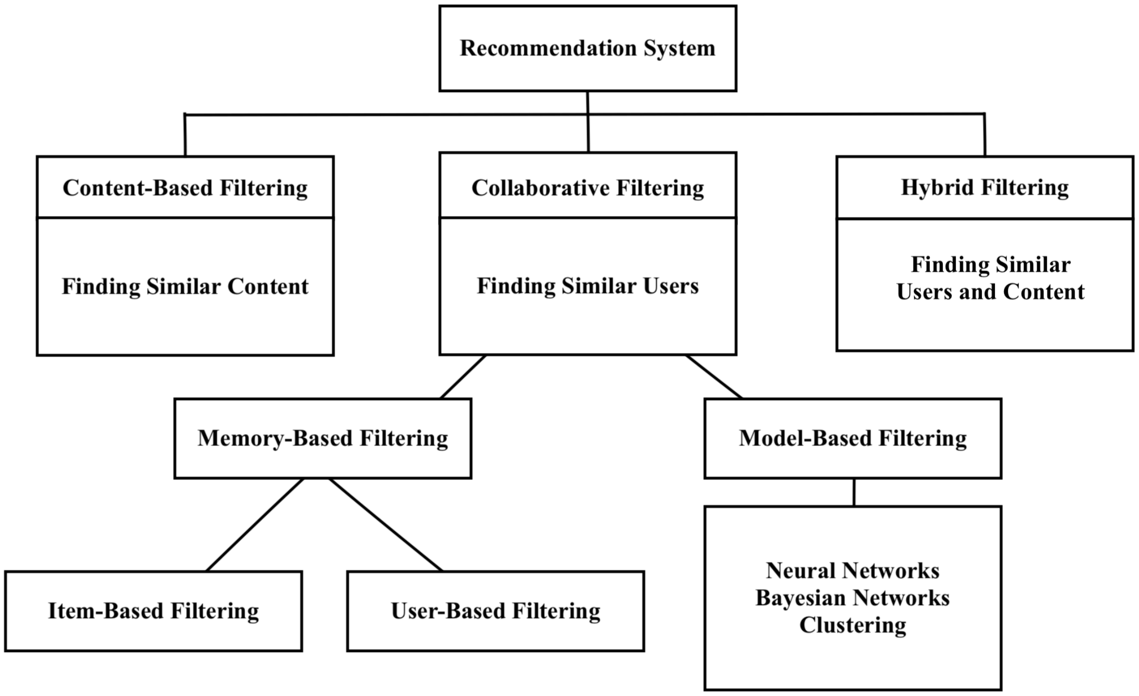


Figure 2.1: Classifications of recommendation systems

They suggest items to users based on information about other users and user preferences simultaneously.

In a content-based recommendation system, commonalities among the users and their related items are extracted and shape the user preferences, then similar items to the users which have the same feature set are recommended to the end-users [46]. Recent studies show that using natural language processing techniques can effectively improve the user feature extraction and, as a result, the accuracy of the recommendation [82]. Vector Space Models (VSM) are the most popular techniques for finding similarities between users and items. In this method, each item can be mapped to a vector in a k-dimensional vector space. Hence, each user can be seen as an aggregation of his or her related item vectors. For example, consider $V_i = v_1, v_2, v_3, ..., v_k$ as an item vector for item i. v_j where $0 < j < k$ reflects the importance of feature j of the item i. If user n has a set of related items $R_n = v_1, v_2, v_3, ..., v_m$, the user n vector is an aggregation of the R_n vectors that is $U_n = agg(R_n) = u_1, u_2, u_3, ..., u_k$. agg can be a function that inputs the item vectors and output the user vector. In the last step, we need to find the relatedness of the user to each item using similarity measures like Pearson correlation or Cosine Similarity. In our study, we mainly used Cosine similarity measurement, and it is defined as:

$$cos(\vec{V_i}, \vec{U_n}) = \frac{\vec{V_i} \cdot \vec{U_n}}{||\vec{V_i}|| \times ||\vec{U_n}||} \tag{2.1}$$

In a collaborative filtering-based recommendation system, recommendations are based on the assumption that similar users make similar choices. Consequently, this method recommends items to an end-user based

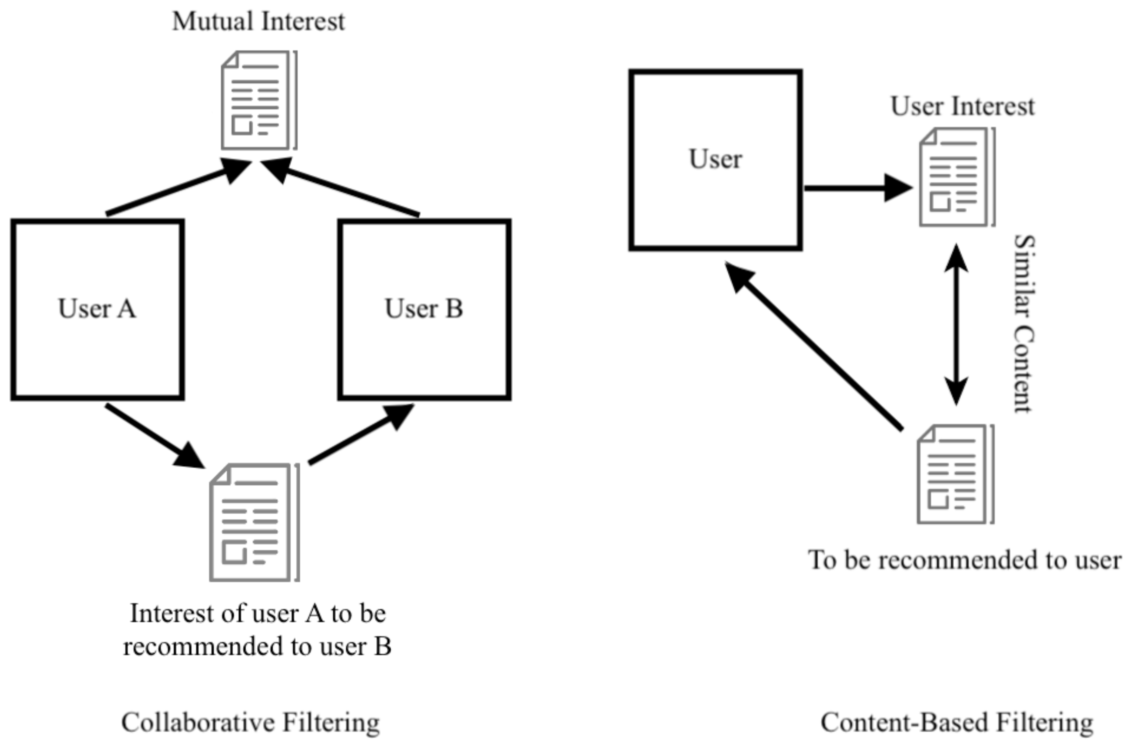


Figure 2.2: Content Based vs. Collaborative Based Recommendation

on the other users' choices who have similar interactions in the system. For instance, a group of users liked a set of movies, and our specific user liked most of them. Following the collaborative filtering logic, the system should recommend the unwatched movies from that set of movies to the user [67]. Similar users are found based on their histories of rating or interacting with the same items. One of the main benefits of collaborative filtering is that it just uses interaction patterns. As a result, we do not need a thorough understanding of the system domain and the item features, and we do not need any content for the items either.

One of the biggest challenges in recommendation systems is cold-start. Cold-start means that there is not enough data available to extract fine-grained user and item features. It can be observed as a sparse dataset. This problem results from lacking user interaction with the system. In this situation, the recommendation system can not extract users' preferences and recommend items accurately.

Hybrid recommendation systems have been introduced to tackle cold-start and high data sparsity. These methods combine content-based and collaborative filtering recommendation techniques to increase the effectiveness of the model. This combination can be implemented in different ways, such as mixing the results of both methods to increase the recommendation accuracy or concatenate the features extracted from those methods and recommend based on the combined feature sets.

The cold-start problem can not be solved entirely using hybrid recommendation methods. Deep learning methods usually perform better when it comes to sparse data and dense feature extraction. These methods also help us extract as much as possible data out of the available context in the dataset. Moreover, it should

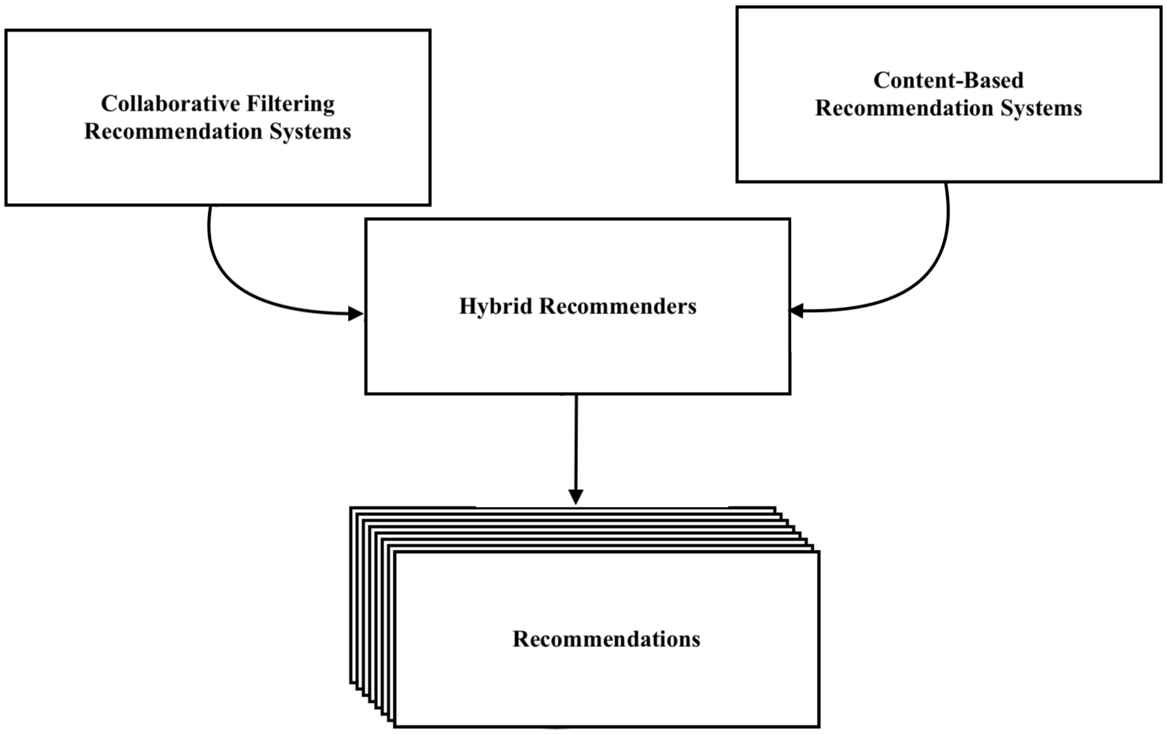

Collaborative Filtering Recommendation Systems
Content-Based Recommendation Systems
Hybrid Recommenders
Recommendations

Figure 2.3: Hybrid Recommendation Systems Architecture

be noted that all of the available information in the dataset are not equally important to consider when it comes to a recommendation. Deep learning can also help us find the most important aspect of the feature set and also help us reduce the noises introduced by spammers or low-quality content in the dataset.

2.2 Expert Recommendation Systems

In this section, we first elaborate on what expert recommendation is, then we review the related work on expert recommendation and its application in community question and answering (CQA) platforms. We will also discuss related studies in the attention-based deep recommendation systems as it is a core component in our proposed Hierarchical Expert Recommendation (HER) model.

2.2.1 Introduction to Expert Recommendation Systems

Expert recommendation systems in CQA platforms can be categorized as a specific type of recommendation system that aims to extract the users' expertise and recommend a set of problems to domain experts to achieve the best possible answer for each problem [78]. An abstraction of an expert recommendation system can be found in figure 2.4.

With the exponential growth of available data in CQA platforms, lots of research have been focused on improving the quality of CQA services. One of the biggest challenges that CQA platforms are currently

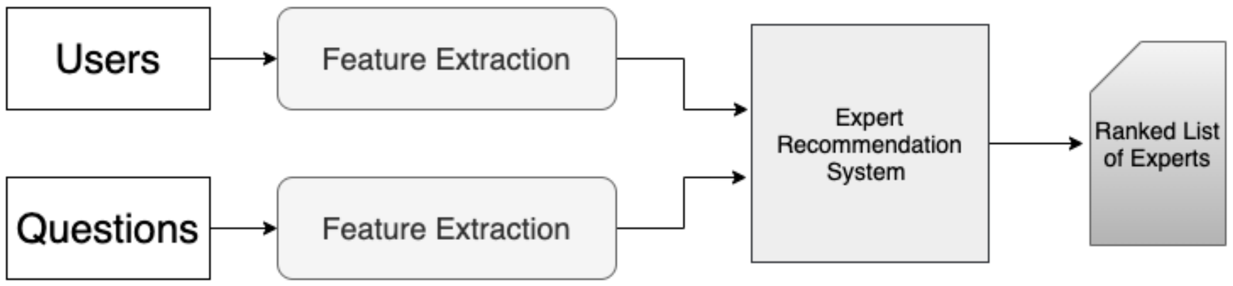


Figure 2.4: Expert Recommendation Systems Schema

facing is the ever-increasing volume of unanswered questions. There are several reasons behind this problem. For example, the question body or the tags are not descriptive enough to catch the experts' attention. As a result, the domain experts are not exposed to the specific question, and the question remains unanswered or poorly answered. Expert recommendation aims to solve this problem by finding the expertise of users and extract the question domain as accurately as possible to recommend the best possible question to the related domain experts and reduce the severity of the aforementioned problem.

One of the strategies used in CQA platforms to ensure the provision of timely and quality answers is the expert recommendation. The issue of recommending experts in these platforms has over the years attracted the attention of researchers. According to previous research [9], people join these platforms because they get timely and high-quality responses to questions posted on them. Recommendation on these platforms involves linking questions to users who are knowledgeable in the domain of the question so that they can provide high-quality answer to the question. Several research works have tried to find experts for questions in different CQA platforms using various approaches such as graph-based algorithms [61], social network analysis or link analysis [44], ranking metrics [86], and statistical models [58] and reported reasonable success. However, achieving high accuracy in recommending experts in CQA platforms is still posing challenging problems. As a result, other methods such as machine learning algorithms are exploited to improve accuracy. Machine learning algorithms have been shown to improve classification and prediction problems in various areas, but limited research exists on assessing state-of-the-art machine learning algorithms for expert finding in CQA.

2.2.2 Machine/Deep Learning Based Expert Recommendation Systems

Currently, researchers are exploring machine learning and deep learning models for recommendation systems. Research [85] has shown that using traditional language models for recommending experts on CQA could lead to word mismatch in question routing as a result of data sparseness. Liu et al. [45] addressed the issue of expert findings in an online CQA Wenwo. The authors explored expert findings through the construction of a binary classifier using post style (for questions and answers) and profile-based features. They reported that the approach resulted in a better routing of questions as many active and new users responded to routed questions. Pal et al. [51] investigated the use of selection bias of questions in grouping users in a CQA platform. Using the Gaussian classification model, the authors evaluated their model with the Stack Overflow (SO) dataset. The results of the research reveal that the approach can effectively differentiate experts among other users.

Dijk et al.[20] transformed early detection of topical expertise issues into a classification problem. They extracted textual, behavioural, and time-aware features from the SO dataset to predict whether a user will be an expert in the future. According to the authors, experts are users with ten or more accepted answers on the SO platform. Their goal was to classify users as experts or non-experts. The authors created topic profiles using tags associated with questions. The questions, answers, and comments associated with topics were profiled and ranked using the td-idf scoring system. They implemented and evaluated three classification algorithms: Gaussian Naive Bayes, Random Forest, and Linear Support Vector. The results of their research revealed that Random Forest performed better than Gaussian Naive Bayes and Linear Support Vector. Differing from Dijk et al.s [20], the proposed system in this study utilizes user and question features included time-based features. To the best of our knowledge, no existing research with similar features construction has evaluated the five mentioned machine learning models using the SO dataset.

Term frequency inverse document frequency (TF-IDF) [63] consists of two statistics, term frequency and inverse document frequency. Term frequency(tf) measures frequency of a word in a document, which is the number of occurrences of a term in a document divided by the whole number of terms in the document for normalization to remove the dependency of frequency on document length. Inverse document frequency(idf) measures the importance of the document in the corpus. Document frequency is the number of occurrences of a term in the documents in the corpus divided by the number of documents for normalization. The inverse of document frequency shows the rareness of a term in the corpus. For instance, if the word is very common and appears in many documents, this number will approach 0. Otherwise, it will approach 1.

$$\mathrm{tf}(t,d) = \frac{f_{t,d}}{\sum_{t'} f_{t',d}} \tag{2.2}$$

$$\mathrm{df}(t) = \sum_{d'} f(t,d') \tag{2.3}$$

$$\mathrm{idf}(t,D) = \log(\frac{N}{1+df(t)}) \tag{2.4}$$

$$\text{tf-idf}(t,d,D) = tf(t,d) \cdot idf(t,D) \tag{2.5}$$

Where f is frequency, t and d denotes term and document respectively, N is the total number of documents in the corpus, and D shows the corpus. At last, these TF-IDF vectors are normalized by ”Euclicidian Norm”.

Based on existing research, the performance of expert finding models is highly dependent on feature selection. For example, [44], [84] in their research established that reputation and domain knowledge of users are relevant features that could be employed in expert finding models. Research [44] derived the knowledge profile of users from the history of previously accepted questions and answers of the users. In their research, question-answer pairs were converted to vectors using the tf-idf approach considering the relevance of past questions to the target question.

Existing recommender systems learn users and items latent representations based on the user's ratings on items. For example, Koren et al. [39] proposed a Singular Value Decomposition (SVD) based Matrix Factorization (MF) method to capture latent user and item representations from the interaction matrix between users and items. This technique will be thoroughly covered in the next section. In CQA platforms, there is an enormous number of users and questions. Therefore, the interaction matrix between users and questions becomes very sparse. It is not practical to learn accurate user and question representations solely based on collaborative filtering-based methods in these platforms.

Recommendation on CQA platforms has mainly centred on linking questions to users proficient in a particular question domain. Various research efforts have attempted to find experts for questions in different CQA platforms using several approaches such as graph-based algorithms [61], statistical models [58], social network analysis or link analysis [44], and ranking metrics [86]. However, recommending experts in CQA platforms with Machine Learning algorithms can not capture the complicated structure of these networks. Dijk et al. [20] transformed early detection of topical expertise issues into a classification problem. They extracted textual, behavioural, and time-aware features from the SO dataset to predict whether a user will be an expert - users with ten or more accepted answers - in the future. They implemented and evaluated the following algorithms: Gaussian Naive Bayes, Random Forest, and Linear Support Vector. Their research revealed that Random Forest performed better than Gaussian Naive Bayes and Linear Support Vector, which verify the needs for more complex models to deal with this problem.

Recent works [21, 68, 73, 75, 81] have investigated deep learning models for the recommendation. They primarily used Deep Neural Networks (DNN) for modelling auxiliary information, such as the textual description of items [73], acoustic features of music [68, 75], cross-domain behaviours of users [21], and the valuable data in knowledge bases [81]. The most related work is [77], which presents a Collaborative Denoising Autoencoder (CDAE) for Collaborative Filtering (CF) with implicit feedback. In contradiction to the DAE-based CF [66], CDAE also plugs a user node to autoencoders' input to build the user's ratings. CDAE is equal to the variation of Singular Value Decomposition called SVD++ model introduced in [38] when the identity function is applied to activate the hidden layers of CDAE. Separated from CDAE, NCF [31] utilizes a two-pathway architecture, modelling user-item interactions with a multi-layer feed-forward neural network. This allows NCF to learn an arbitrary function from the data, capturing more complexity and expressive than the fixed inner product function. However, the application of this method in SO, which does not yield proper recommendations using Machine Learning algorithms, has not been explored yet.

To address the limitation of the collaborative filtering-based approach, researchers set their eyes on content-based approaches. Specifically, researchers considered expert recommendation as a user or item modelling problem. Liu et al. [79] introduced the CQARank model, which utilized both users' latent topic

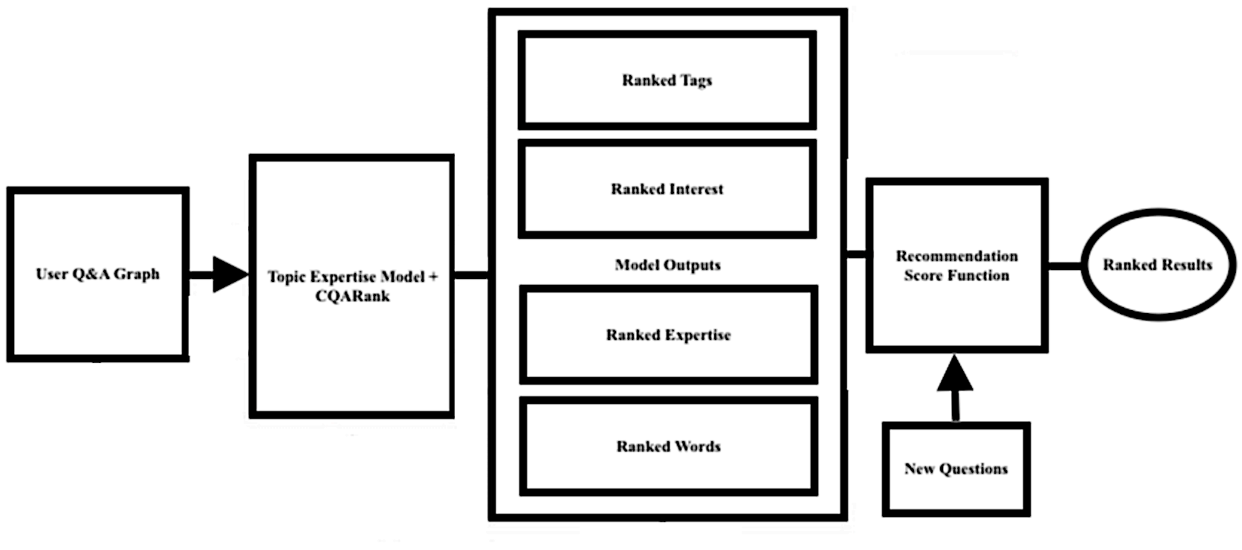


Figure 2.5: CQARank Model Structure

and expertise estimation to recommends experts with similar topical preferences and high-level topical expertise. The high level architecture is shown in 2.5. In this study, only topics are extracted from the available text in the dataset. Topics do not have any information about the importance of a word in comparison to others. The importance of each sentence is completely ignored. A word in the question's title can have more impact on the recommendation than another word somewhere else not in the title. This situation can be troublesome for this kind of model. For example, a user mentioned 'python' in the title, but the provided code is in 'javascript .' the user put this only as an example to further describe his/her problem.

Zhao et al. [83] introduced a topic-level expert learning framework that concurrently identifies experts on each topic and recommends questions by modelling both questions and experts. They incorporate link analysis and topic modelling into the content-based expert recommendation system. They first create a model to generate topics and model the experts over those topics. This study focuses on link analysis in the CQA network and is based on the intuition that no one is an expert in all fields. As a result, they assign these experts to their domain of expertise using topic modelling. This study, like the CQARank model, [79] suffers from in-depth text analysis, and they treat all the available texts in the dataset as series of words without considering the different importance between each word and each sentence.

Mumtaz et al. [50] utilized the state-of-the-art word embedding technique to extract domain-specific semantics and latent features that match user expertise latent features with question content latent features. This study proposed an embedding-based approach that integrates users' textual content from the questions and answers in a unified framework. This framework utilizes an embedding-based approach to find the most relevant users to a specific question by finding the most similar experts based on extracted expertise features and ranking them based on community feedback. The framework is shown in 2.6. To extract user representation from textual data, they calculate the average of all words related to the specific user and

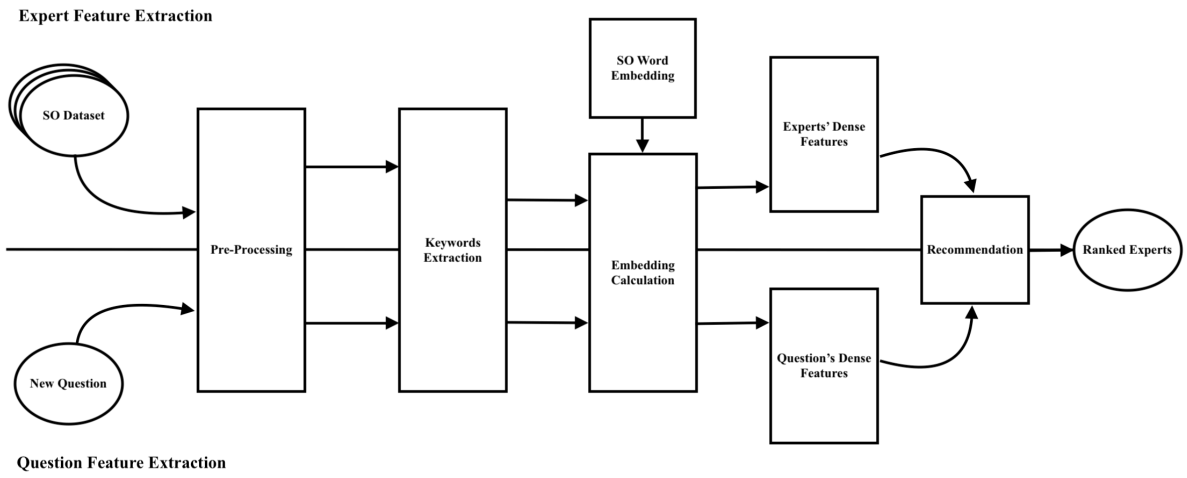


Figure 2.6: Expert2Vec Model Structure

create the expertise features. This method can not distinguish between important and unimportant words and does not consider sentence structure in creating expertise features for users.

Recent studies have attempted to combine Factorization Machine [57] and Deep Neural Networks (DNN) to learn user and item representations for the recommendation. For example, Guo et al. [27] proposed DeepFM, which combined FM for recommendation and DNN for feature learning to achieve a vast improvement in recommendation performance in click-through rate prediction. As one can see in figure 2.7, this model utilizes two fundamental parts in its architecture, the wide part that inputs the raw data to upper levels and the deep part that embeds the same input to extract the latent features. The factorization machine component in this model takes the output of the wide part of the model - the part that leverage the data before passing it through the deep part - and extracts more refined features than other methods like logistic regression. This component can increase the overall accuracy in click-through rate (CTR) prediction.

The DNN approach also has limitations. Qu et al. [54] addressed the insensitive gradient issue in DNN-based models and proposed a Product-based Neural Network (PNN), which adopts a feature extractor to explore feature interactions to predict user responses in the scenario of click prediction. An example of PNN application in CTR prediction can be seen in the figure 2.8.

Nevertheless, most of these methods were not designed to learn text-based features. A naive approach to model text-based features using DeepFM and PNN is to concatenate the texts from the same user or item into a long document and learn the text embeddings. However, such an approach neglects the difference between informative entries and uninformative ones. Wu et al. [76] proposed a hierarchical framework called Hierarchical User and Item Representation with Three-Tier Attention (HUITA) to learn user and item representations from reviews for the recommendation. HUITA contains three main components. First of all, the sentence encoder learns sentence representations from words. Then, a review encoder uses the output of the sentence encoder to learn review representations from sentences and, at last, the user/item encoder uses

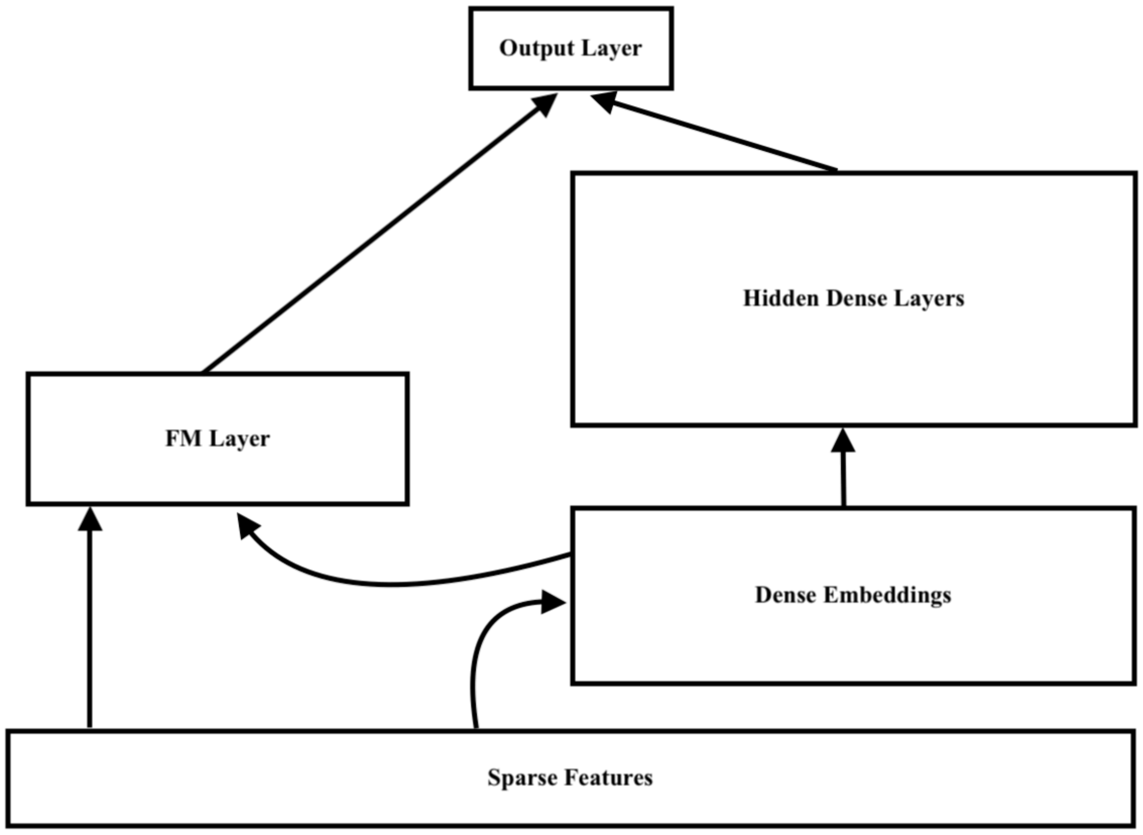


Figure 2.7: DeepFM Model Structure

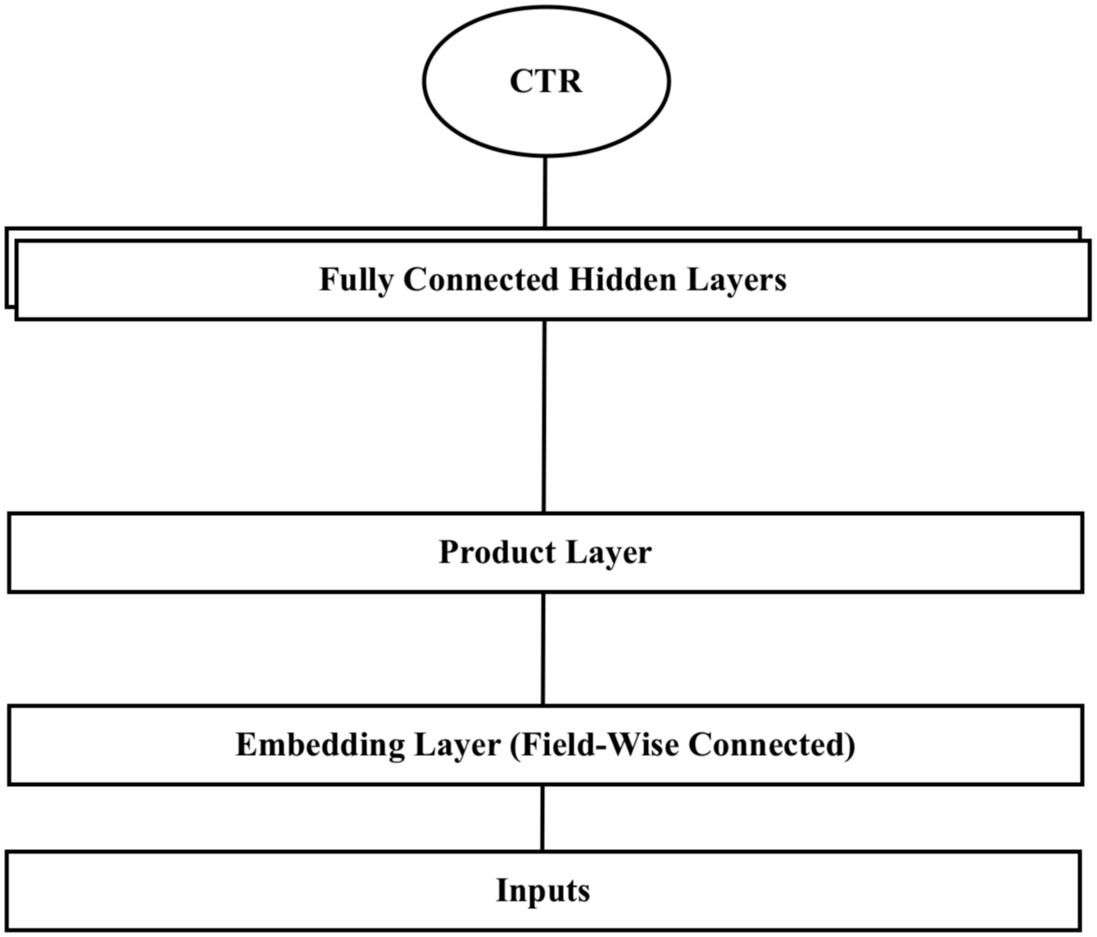

CTR
Fully Connected Hidden Layers
Product Layer
Embedding Layer (Field-Wise Connected)
Inputs

Figure 2.8: PNN Model Structure in Predicting CTR

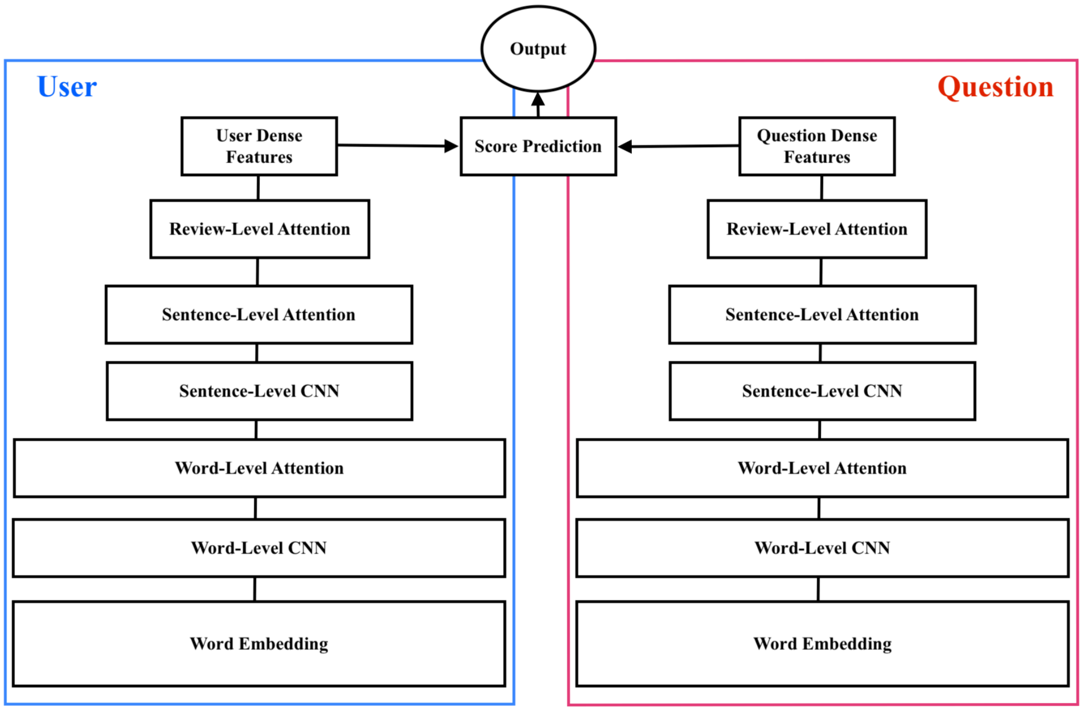


Figure 2.9: HUITA Architecture

the previous output to learn user/item representations from reviews. In addition, they utilize a three-tier attention network to capture the importance of each word, sentence and review. They used user and item embeddings based on IDs as the final representations to capture the latent features of users and items to make the recommendation. Their architecture can be found in figure 2.9.

Chen et al. [16] developed an attention-based model that extracts user intention in each available category and facilitates the traditional FM model for a top-k recommendation system. Zhu et al. [87] proposed a new personalized recommendation system with a deep attention neural network able to consider the user's sequential data from the past and the user's current preference. We are inspired by the approach proposed by Wu et al. [76], and adapted the approach to learning better representations for questions to improve expert recommendation hierarchically.

2.3 Summary

The importance of recommendation systems is undeniable. Nowadays, most available social platforms provide their users with a variant of recommendation systems that help them find items related to their preferences. On the other side, recommendation systems help information providers maximize their content exposure to the right users. In most cases, these recommendation systems include a variation or even combination of collaborative filtering and content-based recommendation to understand users' characteristics and recommend items based on that.

The expert recommendation in community question answering (CQA) platform is no exception. These recommendation systems inherit most of their characteristics from other recommendation systems except a fundamental difference. CQA platforms heavily rely on textual data alongside user interactions. Extracting fine-grained features from these textual data is not an easy task. It usually needs specific techniques to alleviate the complexity of grammar and meaning of the words and make the data more calculatable and manageable for mathematical equations.

The most recent publications in this field aimed to avoid the textual data or use more readily available data such as the question tags as their primary source of data. Despite the recent improvements in recommending questions to domain experts, studies show that there are still questions that never receive proper answers, and there is room for improvements. These questions are mostly not well described, or available tags can not effectively categorize them.

Attention mechanisms and hierarchical approaches to extract features from texts are getting popular because of their performance. These techniques helped recommendation systems to make precise recommendations considering the complexity of the textual data. Moreover, illustrating the most crucial aspect of the textual data can easily be achieved by these methods, and they also improve the explainability of the models by providing multi-level importance of each sentence in the dataset.

3 Research Methodology and Implementation

One of the main research goals of this project is to develop an effective expert recommender system, which utilizes various deep learning techniques to increase the overall accuracy of the existing recommendation systems. We are using the state of art techniques, such as deep factorization machines and hierarchical attention networks, to develop deep expert recommendation systems.

Using deep learning to solve our problems always comes with certain drawbacks. For example, deep learning models, in most cases, are impossible to understand or explain. They act as black boxes that take an input and gives the best possible answer without explaining why it is the correct answer. These models usually never tell about what are the parameters and why they have specific values. Moreover, deep learning models take lots of computing power and resources to be optimized. In comparison, machine learning methods are much less resource-intensive. In addition, machine learning algorithms can be more explainable and easier to understand and usually have less computation and memorization to find the correct answer. The decision between choosing deep learning over machine learning is not always a trivial task, and it needs justifications.

In this study, we first use state-of-the-art machine learning algorithms to explore the possibility of solving this problem efficiently and then exploring our findings to understand which family of methods, e.g. machine learning or deep learning, will be the most promising to solve our problem. Furthermore, we combine those methods to examine to what extent we want to drive our final result based on machine learning or deep learning methods. Finally, we explore and implement deep learning techniques and utilize them in a novel architecture aiming to outperform the state-of-the-art expert recommendation systems.

In this chapter, we first outline our domains and the dataset. The proposed methodology contains three parts. The first section includes different machine learning algorithms for expert recommendation systems. In the second section, a combination of the matrix factorization method and deep neural network is proposed to make a hybrid expert recommendation. Finally, the last section introduces our novel architecture based on our findings in previous sections that further improves the accuracy of current expert recommendation systems. The evaluation and results of these methodologies will be discussed in the next chapter.

3.1 Dataset Description

In this project, we use the Stack Overflow dataset[1], which has approximately 1 million users and 19 million posts. Due to massive number of posts available on this dataset, we only use the year 2019 for collecting the

[1]https://archive.org/details/stackexchange

data, and consider users who have answered more than 15 questions during this year as experts. Furthermore, we only consider the accepted answer for a proposed question, because most of the unaccepted answers have a low quality, which do not contribute to expert identification. The following paragraph provides summary of questions' and experts' features. The tables 3.1 and 4.2 also provide summary of our dataset.

- 392,798 Questions
  - Vote: Voting is how the community indicates which questions and answers are most useful and appropriate, which can be either affirmative or negative(as upvote and downvote).
  - Tags: Binary representation of presence of Top 50 tags in SO.
  - Score: The difference between question upvotes and downvotes, between -25 to 491.
  - View Count: Number of times this question has been viewed on SO, between 6 to 94,473.
  - Texts: String of question title and question texts (codes are removed).
- 5,448 Users (Experts)
  - Reputation: a user is awarded 10 reputation points for receiving an up vote on an answer given to a question and 10 points for the up vote of a question.
  - Views: Number of times this user has been viewed on SO.
  - Upvote: Number of affirmative votes the user has obtained in previous questions and answers.
  - Downvote: Number of negative votes the user has obtained in previous questions and answers.
- Question and Expert Interactions
  - Interaction Matrix: It is matrix representing the questions and users interactions where each column represents a question and each row represents a user. If a user answer a question, the corresponding item in the matrix would be 1 and all other values are zero.

We train our Word2Vec model, and we have a minimal vocabulary in comparison to available pre-trained models on the internet. The main reason to train the Word2Vec model on our vocabulary is to emphasize the relation between words in the programming community instead of the general domain. Some words have a completely different meaning in this domain, and it may result in inaccuracy.

The majority of our selected users have answered between 15 and 25 questions. Considering the number of questions and users, the sparsity of the interaction matrix between users and question is more than 99%. This sparsity makes the recommendation a challenging task and we need a capable method to extract latent features in this sparse space.

Negative sampling[49] is a method used for machine learning and deep learning models that have significantly more negative observations than positive ones. Furthermore, we do not have records of experts who

Table 3.1: 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 3.2: User & Question Count

# Dataset	# Users	# Questions	Sparsity
Main	5,448	392,798	99.98%
Python	2,211	54,471	99.95%
JavaScript	2,315	35,160	99.95%

did not answer a question directly. Consequently, we apply negative sampling to obtain negative interactions, such that our machine learning methods would learn both positive and negative interactions. For each question, we randomly sample 4 experts who did not answer the question as negative sample.

3.2 Expert Recommendation Using Machine Learning

We implemented four recommendation systems based on existing machine learning algorithms to recommend experts in CQA platforms. As part of our implementation, we engineered novel features to represent the questions and the user's expertise and used them as input for the machine learning algorithms. Finally, we conducted extensive experiments on a large real-world dataset and benchmarked the performance of the these machine learning algorithms on the experiment recommendation task.

The main concern in the SO platform is the absence of successful coordination among questions and the potential competent answerers, which unfavourably influences effective knowledge acquisition. From one side, a question may receive several low-quality answers without receiving any proper answer in a limited timeframe. Similarly, an expert might encounter various new inquiries without being able to distinguish their question of interest rapidly. Therefore, expert recommendation is a promising approach. According to previous studies [1], most answers and knowledge in the communities originate from only a minority of users. A recent study on Stack Overflow and Quora [4] demonstrates that these platforms involve profoundly committed domain experts, who target not only the requester question but also a long-lasting answer to a broader crowd. Consequently, expert recommendation would bring experts attention to relevant questions effectively and instantly, leading to stronger communities in CQA. The purpose of the expert recommender system is to suggest relevant experts for a given question. Ranking experts based on experts features, questions features and their interaction require effectively adjusting the importance of these features and discovering proper ranking. However, due to the complexity of this problem, we cast our recommendation problem into a classification problem, in which the recommender would predict whether a question will be answered by an expert or not. Classification methods can easily utilize multiple aspects of features from the expert, question, and answer compared to language-based and network-based models which only consider characteristics extracted from text and social network relation respectively [74].

We applied methods such as Naive Bayes [20], Support Vector Machine [51], Logistic Regression [41], and Random Forest [18] for expert classification on our dataset, using question and answer features(e.g., community score, text similarity, tags, view count) and expert features such as (e.g., community badges, reputation, upvote).

We summarize this section objectives as follows:

- We proposed a traditional machine learning pipeline for expert recommendation in online social collaborative platform. As part of the proposed pipeline, we investigated and designed features that are representative of users and questions in SO for expert recommendations.
- We evaluated our proposed pipeline by conducting extensive experiments on a large real-world dataset.

3.2.1 Feature Engineering

Evaluation of similarity in questions and experts answers, which is a textual data, is one of the most critical tasks for expert recommendation in SO. Question textual features were extracted based on the question title. The user textual features were extracted based on the titles of questions that the user has previously answered. The raw data, as a sequence of symbols with variable length and sequential dependency, cannot be fed directly into the ML methods. On one hand, if we treat this text as nominal data, many methods would expect numerical feature vectors with a fixed size. On the other hand, if we manage this data as categorical input, we would have numerous features without any measure of importance between them. Hence, we use methods from information retrieval domain.

The most popular technique for text feature extraction is term frequency inverse document frequency (TF-IDF) [63], which provides a numerical statistic to show importance of a word to a document in a corpus. This model, converts the textual representation of information into sparse features. In this project, we extract textual features from question titles. To obtain a more meaningful corpus, we first tokenize question titles, then remove stop words and finally stem the words to their root using Porter Stemmer algorithm [52].

3.2.2 Machine Learning Models

We first present the feature engineering process, in which we extract features from questions and answers textual data. Then, we elaborate on different classification methods that would predict whether a question will be answered by an expert or not.

The input to our machine learning models is the feature vector, that we denote by $\mathbf{x}_i$, representing vector of question features $\mathbf{q}_m$, the vector of expert features $\mathbf{u}_n$, and the label indicating whether an expert has answered the question r_i.

$$\mathbf{x}_i = (\mathbf{q}_m, \mathbf{u}_n, r_i) \tag{3.1}$$

Naive Bayes

This method is a simple probabilistic model that applies Bayes theorem with the strong assumption of ”conditional independence” between features. This model fits our problem for its high scalability and linear time complexity because the number of parameters is linear with the number of features. We only have nominal features, so we use Gaussian Naive Bayes for our binary classification.

$$P(y_j|\mathbf{x}_i) = \frac{P(\mathbf{x}_i|y_j)P(y_j)}{P(\mathbf{x}_i)} \tag{3.2}$$

Where $\mathbf{x}_i$ is the feature vector of samples i, $i \in 1, 2, ..., n$, and y_j is the notation of class j, $j \in 0, 1$. Classes are binarized 1 or 0. 1 means expert will answer question , and 0 otherwise. $P(\mathbf{x}_i|y_j)$ is the probability of observing sample $\mathbf{x}_i$ given that it belongs to class y_j.

Further, the class condition probabilities of individual features d are as follows because of conditional independence assumption.

$$P(\mathbf{x}|y_j) = P(x_1|y_j)...P(x_d|y_j) = \Pi_{k=1}^{d} P(x_k|y_j) \tag{3.3}$$

Where d is the total number of features and x_k represent each feature.

Finally, the decision rule is:

$$\hat{y} = \arg\max \ P(y_j|\mathbf{x}_i) \quad for \ j = 0, 1 \tag{3.4}$$

Random Forest

This approach consists of a large number of individual decision trees [14]. Decision tree [13] is a supervised learning method that aims to predict the target variable by inferring decision rules from features. In this tree, leaves are the class labels and branches represents the decision rule. The tree is built from the top by a feature that best splits the set of items. There are two methods for measuring better split, Gini impurity[13] and information gain [55]. Gini impurity measures the ratio of errors when an item is labelled randomly based on the distribution of labels in the set. Entropy is computed as follows:

$$I_G(p) = \sum_{i=1}^{J} p_i(1 - p_i) = 1 - \sum_{i=1}^{J} p_i^2 \tag{3.5}$$

Where J is the number of classes, and p_i is the fraction of items labelled with class i in the set.

Information gain measures the amount of information a feature conveys about the class, which indicates the expected amount of information required for labelling an item. This information is measured by entropy that shows the ratio of each class in the child node obtained from the split in the tree. This method is computed as follows:

$$H(T) = -\sum_{i=1}^{J} p_i \log_2 p_i \tag{3.6}$$

Where T is the tree, H indicates entropy and p_i shows the ratio of each class in the child node.

$$S_a(v) = \{\mathbf{x} \in T | x_a = v\} \tag{3.7}$$

Where $S_a(v)$ is the set of training samples of T training set, in which feature a is equal to the value of v.

$$I_G(T, a) = H(T) - H(T|a) \tag{3.8}$$

$$= H(T) - \sum_{v \in vals(a)} \frac{|S_a(v)|}{|T|} H(S_a(v)) \tag{3.9}$$

Where $I_G(T, a)$ is the mutual information on the training set, which measures the total entropy for a feature, in case a unique classification can be made based on each value of the feature.

The set of decision trees operate as an ensemble, which uses multiple learning algorithms to obtain better predictive performance compared to each learning algorithm separately. This algorithm aims to utilize the wisdom of crowds in which each tree in the random forest outputs a class prediction, and the class with the most votes becomes the models prediction. Random forest allows each tree to randomly sample from the dataset with replacement, which results in different trees. These different trees would use diverse sets of features for prediction and add variability to this model. The essential characteristic in the random forest method is obtaining trees with low correlation with each other, which leads to better performance. While the algorithm via feature randomness tries to establish these low correlations, the selection of features and the hyper-parameters also impact this correlation. The hyper-parameters in random forest are number of trees, maximum depth of tree, maximum number of features for trees to randomly sample from, maximum number of nodes allowed as leaf nodes, minimum sample split which indicates number of samples needed to split a node, bootstrap and bootstrap features which shows sampling with or without replacement.

Logistic Regression

This method is a supervised binary classifier that uses a logistic function to model a binary dependent variable based on numeric features [47]. Logistic Regression presumes a linear relationship between the features and the log-odds of each class.

$$Z = \mathbf{w}^T \mathbf{x} = w_0 + w_1 x_1 + \cdots + w_n x_d \tag{3.10}$$

$$P(Y=1|\mathbf{x}_i) = \frac{1}{1+e^{-Z}} = \frac{1}{1+e^{-(w_0+w_1x_1+\cdots+w_nx_d)}} \tag{3.11}$$

Where w_i is a weight parameter for each feature tuned during learning. In the learning process, the model aims to minimize the error between the predicted and original values using gradient descent algorithm [62]. This error can be both calculated by Manhattan norm ($l1$) or Euclidean norm ($l2$). Gradient descent is an optimization algorithm for approaching a local minimum by taking steps proportional to the negative value of the gradient of the function at the current point.

$$l2\,norm:\ error(\mathbf{w},\mathbf{X}) = \sum_{i=1}^{n}(y_i-\hat{y}_i)^2 \tag{3.12}$$

Where i denotes training set samples and $\hat{y}_i$ is the predicted value.

$$\mathbf{w} \leftarrow \mathbf{w} - \text{gradient of error} = \mathbf{w} - [\alpha\mathbf{x}(y_i-\hat{y}_i)\odot\hat{y}_i\odot(1-\hat{y}_i)] \tag{3.13}$$

For optimizing the error function, various approaches can be applied. For instance, *"Newton's Method"* [33] considers both first and second partial derivatives for optimization; however, this method is computationally expensive due to the second derivatives calculation. *"Limited-memory Broyden Fletcher Goldfarb Shanno Algorithm"(LBFGS)* [43] method is similar to Newton's method, but the second derivatives are approximated by gradient evaluations. This approach has the best performance in small datasets but might not converge in large datasets. *"A Library for Large Linear Classification"(LIBLINEAR)* [23] approach utilizes a coordinate descent algorithm that performs approximate minimization along coordinate directions or coordinate hyperplanes. This method is recommended for high dimensional datasets but lacks in parallel execution. *"Stochastic Average Gradient descent"(SAG)* [65] method optimizes the sum of a finite number of smooth convex functions. This method is a faster solver for large datasets because of utilizing a memory of previous gradient values. Due to the large size and high dimensionality of our data, we examined both LIBLINEAR and SAG solver.

We should note that logistic regression outputs the probability of the class, so we should use a threshold to map probabilities higher than this threshold to the current class and lower probabilities to the other class.

$$P(Y=1|\mathbf{x}_i) > 0.5 \rightarrow \text{expert will answer the question} \tag{3.14}$$

$$P(Y=1|\mathbf{x}_i) \leq 0.5 \rightarrow \text{expert won't answer the question} \tag{3.15}$$

Support Vector Machine [12]

SVM is a non-probabilistic supervised classifier, intending to find a hyperplane in D-dimensional space(D the number of features) that classifies the class values. SVM is capable of both linear and non-linear classification. In this algorithm, we aim to maximize the margin between the data points and the hyperplane. So our objective is minimizing the following function by taking partial derivatives with respect to the weights to find the gradients.

$$Z = [\frac{1}{n}\sum_{i=1}^{n} max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i))] + \lambda \left\|\mathbf{w}\right\|^2 \tag{3.16}$$

$$\frac{\partial}{\partial w_k}\lambda \left\|\mathbf{w}\right\|^2 = 2\lambda w_k \tag{3.17}$$

$$\frac{\partial}{\partial w_k} max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i)) = \{0, -\mathbf{y_i} \odot \mathbf{x_i}\} \tag{3.18}$$

where λ is the regularization parameter for preventing overfitting by penalizing for additional and higher-order weights, and *alpha* is the learning rate. So the weights would be updated based on the gradients.

$$No\, misclassification \rightarrow \mathbf{w} = \mathbf{w} - \alpha \cdot 2\lambda\mathbf{w} \tag{3.19}$$

$$Misclassification \rightarrow \mathbf{w} = \mathbf{w} - \alpha \cdot (2\lambda\mathbf{w} - \mathbf{y_i} \odot \mathbf{x_i}) \tag{3.20}$$

SVM model utilizes kernel functions for the decision function. Kernel is an approach for computing the dot product of two vectors in the transformed space. The idea is mapping the non-linearly separable dataset into a higher dimensional space such that a separating hyperplane might be found.

$$K(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^{\mathbf{T}}\phi(\mathbf{y}) \tag{3.21}$$

where $\mathbf{x}$ and $\mathbf{y}$ are vectors of features in the input space. This means if we use a mapping function that maps our data into a higher-dimensional space, then the maximization and decision rule depends on the dot products of the mapping function for different samples. So, we need to know K and not the mapping function itself. This function is known as Kernel function, and it reduces the complexity of finding the mapping function.

SVM supports linear, polynomial and radial basis function kernel. Linear kernel is the simple inner product of the input space, which is suitable for linearly separable data.

$$K(\mathbf{x}, \mathbf{y}) = \mathbf{x}^{\mathbf{T}}\mathbf{y} + c \tag{3.22}$$

where c is the regularization parameter as the sum of a regularization term and the misclassification rate. On the one hand, for large values of c, a smaller margin is specified for classifying all training points correctly. This results in a lower misclassification rate on the training rate and possibly overfitting. On the other hand, a smaller c encourages a more considerable margin, therefore a more straightforward decision function in a trade-off with lower training accuracy.

Polynomial kernel [12] is useful when the data points are not linearly separable. This kernel represents the similarity of vectors in a feature space over polynomials of the original variables, allowing the learning of non-linear models. However, this kernel requires normalized training data.

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^{\mathbf{T}}\mathbf{y} + c)^d \tag{3.23}$$

where d is the order of the kernel, and c is a parameter that allows the trade-off between the influence of higher-order terms versus lower-order terms in the polynomial. Higher-order kernels tend to overfit the training data and thus do not generalize well.

Radial Basis function[72] kernel (RBF) is a general-purpose kernel appropriate for when there is no prior knowledge about the data. We use the RBF kernel for our SVM model because of its generalization and capacity to handle our unbalanced dataset.

$$K(\mathbf{x}, \mathbf{y}) = exp(-\frac{||\mathbf{x} - \mathbf{y}||^2}{2\sigma^2}) \tag{3.24}$$

where $||\mathbf{x} - \mathbf{y}||^2$ is the squared Euclidean distance between the two feature vectors, and σ is a parameter that acts as a smoothing parameter that determines the influence of each of the points. This parameter indicates the range of influence of a single training sample with low values meaning far and high values meaning close. σ can be seen as the inverse of the radius of influence of samples selected by the model as support vectors. When σ is low, the curve of the decision boundary is shallow, and thus the decision region is vast. When σ is high, the curve of the decision boundary is high. So this parameter adjusts the curvature of the decision boundary.

3.3 Deep Expert Recommendation Based on Matrix Factorization

In community question and answer (CQA) platforms, such as Stack Overflow(SO), recommending relevant experts for answering questions is important. Personalized recommendation models users' preferences for items based on their past interactions, known as Collaborative filtering(CF)[64]. The most popular collaborative filtering technique is matrix factorization (MF), which utilizes a vector of latent features to represent a user or an item that can model users interaction on an item as the inner product of their latent vectors[40].

Expert recommender systems are algorithms aimed at suggesting relevant questions to experts. Collaborative filtering(CF)[64] models experts preference for questions based on their past interactions. These past expert-question interactions are sufficient to detect similar experts or similar questions and make predictions based on these estimated proximities. MF assumes that each dimension of the latent space is independent of each other and linearly combines them with the same weight. However, a neural network recommender(NCF) [31] consists of two separate models, Generalized Matrix Factorization(GMF) and Multi-Layer Perceptron(MLP) to generalize the MF method and capture the non-linearity in the latent space respectively.

This method aims to learn the complex structure of expert-question interaction data by allowing varying importance of latent dimensions and learning non-linear functions for the interaction. The objectives of this section are as follows:

- Examining state-of-the-art algorithms for collaborative filtering based recommendation in Stack Overflow by conducting extensive experiments on a large real-world dataset.
- Exploring the importance of latent features against the interaction decomposition function and highlighting the critical aspects of future deep recommenders in Stack Overflow.
- Highlighting the ultimate path to take for this study.

The model introduced as NeuMF[31] provides a general framework that is generic and can express and generalize matrix factorization. This model is the fusion of two separate models, Generalized Matrix Factorization(GMF) and Multi-Layer Perceptron(MLP). Since NeuMF solely focuses on collaborative filtering, it uses the interaction matrix, which is the one-hot encoding — an array of zeros and ones where ones show the presence of interaction — of expert and question IDs as input. So the interaction matrix is a sparse matrix, where ui is the interaction between a specific expert u and a question i. The embedding layer is after the input layer, a dense layer that decomposes the sparse matrix into expert and question latent vectors. The target value y_{ui} is a binarized 1 or 0, as 1 means expert u has answered question i, and 0 otherwise. The prediction score $\hat{y}_{ui}$ then represents how likely the question i will be answered by the expert u. The training is performed by minimizing the pointwise loss between the predicted score of $\hat{y}_{ui}$ and its target value of y_{ui}. Then the likelihood function is defined as binary cross-entropy loss, also known as log loss. So, the recommendation problem can be addressed as a binary classification problem.

$$p(\gamma, \gamma^{-}|\mathbf{P}, \mathbf{Q}, \theta_f) = \Pi_{(u,i)\in\gamma}\hat{y}_{ui}\Pi_{(u,i)\in\gamma^{-}}(1 - \hat{y}_{ui}) \tag{3.25}$$

where $\mathbf{P} \in R^{M\times K}$ and $\mathbf{Q} \in R^{N\times K}$ denotes the latent factor matrix for experts and questions and θ_f denotes the model parameters of the interaction function f.

By taking the negative logarithm of the likelihood:

$$L = -\sum_{(u,i)\in\gamma} log(\hat{y}_{ui}) - \sum_{(u,i)\in\gamma^{-}} log(1 - \hat{y}_{ui}) \tag{3.26}$$

$$= -\sum_{(u,i)\in(\gamma\cup\gamma^{-})} y_{ui}log(\hat{y}_{ui}) + (1 - y_{ui})log(1 - \hat{y}_{ui}) \tag{3.27}$$

3.3.1 Generalized Matrix Factorization

GMF aims to generalize the MF method [40] by allowing the interaction function determined from a large family of factorization models[57]. The mapping function of the first layer of this model is:

$$\phi(\mathbf{p}_u, \mathbf{q}_i) = \mathbf{p}_u \odot \mathbf{q}_i \tag{3.28}$$

where, the expert latent vector $\mathbf{p}_u$ and the question latent vector $\mathbf{q}_u$ equals $\mathbf{P}^T\mathbf{v}_u^U$ and $\mathbf{Q}^T\mathbf{v}_v^V$ respectively. Then the output layer of this model is:

$$\hat{y}_{ui} = a_{out}(\mathbf{h}^T(\mathbf{p}_u \odot \mathbf{q}_i)) \tag{3.29}$$

Where the output layer uses the sigmoid activation function with the binary cross-entropy loss function. If we use the uniform vector 1 as $\mathbf{h}$ and the identify function for a_{out}, this model performs the same as MF. Therefore, this model is the generalized version of MF.

For training GMF, mini-batch Adaptive Moment Estimation(ADAM)[36] is used due to its efficiency in tuning the learning rate for each parameter by slightly changing the learning rate for frequent parameters and significantly altering learning rate for the infrequent parameter. Moreover, I initialized the parameters using a random initialization from a Gaussian distribution (with a mean of 0 and a standard deviation of 0.01).

3.3.2 Multi-layer Perceptron

GMF concatenate experts and questions features. However, this concatenation may not capture the interactions between expert and question features. So, I utilize a standard multilayer perceptron to learn the interaction between experts and questions latent features. For learning the interaction between expert and question latent features $\mathbf{p}_u$ and $\mathbf{q}_i$, instead of learning the element-wise product in GMF, the following method will be applied:

$$\mathbf{z}_1 = \phi(\mathbf{p}_u, \mathbf{q}_i) = \begin{bmatrix} \mathbf{p}_u \\ \mathbf{q}_i \end{bmatrix} \tag{3.30}$$

$$\phi_2(\mathbf{z}_1) = a_2(\mathbf{W}_2^T\mathbf{z}_1 + \mathbf{b}_2) \tag{3.31}$$

$$\ldots \tag{3.32}$$

$$\phi_2(\mathbf{z}_{L-1}) = a_L(\mathbf{W}_L^T\mathbf{z}_{L-1} + \mathbf{b}_L) \tag{3.33}$$

$$\hat{y}_{ui} = \sigma(\mathbf{h}^T\phi_L(\mathbf{z}_{L-1})) \tag{3.34}$$

Where $\mathbf{W}_x$, $\mathbf{b}_x$, and a_x denote the weight matrix, bias vector, and activation function for the x-th layers neural network.

Furthermore, I will use ReLu[4] for activation with the binary cross-entropy loss function. ReLu is chosen because of its proven resistance to saturation and being suitable for sparse data[25]. Furthermore, the sigmoid function suffers from saturation, and the tanh function is a rescaled version of sigmoid with lower saturation. For training MLP, mini-batch Adaptive Moment Estimation(ADAM)[36] is used due to its competence in tuning the learning rate. Moreover, I initialized the parameters using a random initialization from a Gaussian distribution (with a mean of 0 and a standard deviation of 0.01) until convergence. For network structure design, a standard solution is to adopt a tower model, where the bottom layer is the largest. Each successive layer has fewer neurons to learn more abstractive features of the data by using a small number of hidden units for higher layers [29]. I empirically implement the tower structure, halving the layer size for each successive upper layer. This model employs three hidden layers, and the factors of [8, 16, 32, 64] are tested for the number of neurons for the last layer.

3.3.3 Fusion of MLP and GMF

So far, I have developed two models, GMF, which uses a linear kernel to model the latent feature interactions and MLP, which uses a non-linear kernel to learn the interaction function from data. For adding more flexibility to the fused model, I allow GMF and MLP to learn separate embeddings and combine the two models by concatenating their last hidden layer. Consequently, NeuMF combines the linearity of MF and non-linearity of deep neural networks for modelling expertquestion latent structures.

$$\phi^{GMF} = \mathbf{p}_u^G \odot \mathbf{q}_i^G \tag{3.35}$$

$$\phi^{MLP} = a_L(\mathbf{W}_L^T(a_{L-1}(...a_2(\mathbf{W}_2^T \begin{bmatrix} \mathbf{p}_u^M \\ \mathbf{q}_i^M \end{bmatrix} + \mathbf{b}_2)...)) + \mathbf{b}_L) \tag{3.36}$$

$$\hat{y}_{ui} = \sigma(\mathbf{h}^T \begin{bmatrix} \phi^{GMF} \\ \phi^{MLP} \end{bmatrix}) \tag{3.37}$$

Where $\mathbf{p}_u^G$ and $\mathbf{p}_u^M$ are expert embeddings for GMF and MLP model, and $\mathbf{q}_i^G$ and $\mathbf{q}_i^M$ are for question embeddings. Finally, each model parameter can be calculated with standard back-propagation. The architecture of NeuMF is presented in figure 3.1.

Two approaches are tested for training NeumF, firstly, pre-training MLP and GMF separately and initializing parameters of NeuMF with the learned parameters, and secondly, training NeuMF without pre-training. For the pre-trained model, the NeuMF parameters will be initialized by pre-trained models of GMF and MLP. So, the weights of the two models are concatenated with:

$$\mathbf{h} \leftarrow \begin{bmatrix} \alpha \mathbf{h}^{GMF} \\ 1 - \alpha \mathbf{h}^{MLP} \end{bmatrix} \tag{3.38}$$

Where $\mathbf{h}^{GMF}$ and $\mathbf{h}^{MLP}$ denote the $\mathbf{h}$ vector of pre-trained GMF and MLP model and α is a hyper-parameter showing the trade-off between the two models. For training NeuMF, the Stochastic Gradient Descent(SGD) [60] is used for optimization because the pre-trained parameters utilize ADAM optimizer and have the momentum information implicitly, which mitigates the need for using momentum-based optimizers.

Training NeuMF without the pre-trained GMF and MLP models uses Adaptive Moment Estimation(ADAM)[36] for optimization with randomly initializing parameters from a Gaussian distribution (with a mean of 0 and a standard deviation of 0.01).

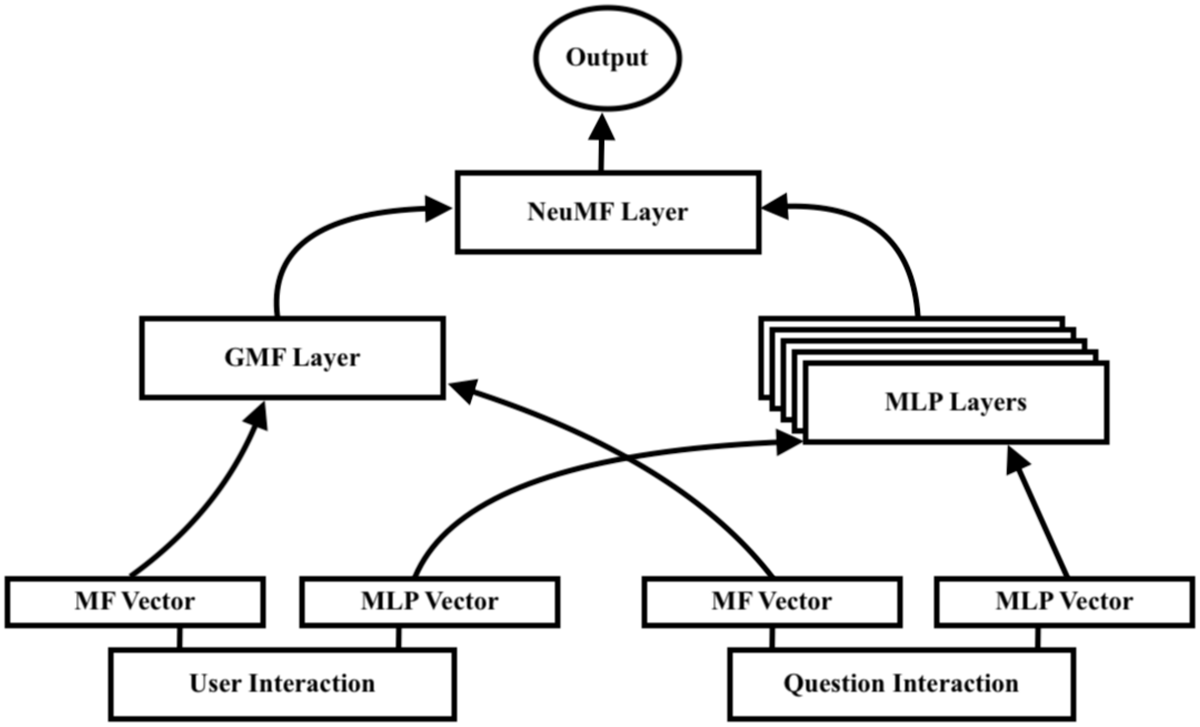


Figure 3.1: Neu MF architecture

3.4 Hierarchical Deep Expert Recommendation

In this section, we introduce our novel architecture to increase the effectiveness of the current expert recommendation systems. Our primary goal in creating a new model is to fill the research gap by leveraging hierarchical textual information and improve them to increase the overall accuracy of these recommendation systems.

In order to fill the research gap, we propose the Hierarchical Expert Recommendation (HER) model, which is a hierarchical attention-based network [80] that learns latent representations of questions and users' expertise for the expert recommendation. Specifically, HER first learn the latent user and question representations from their meta-data (e.g., question's vote counts, user's reputation scores, etc.) using a deep neural network.

The hierarchical part of this model enabled us to assign separate importance for sentences and words, which provided us valuable data to distinguish the keywords in key sentences. This method departs us from existing expert recommendation systems, which treat each question as a combination of words without considering the sentences. We also utilize an attention mechanism to enable us to capture these differences in both levels of hierarchies in the proposed model. This technique also helps us provide a clear explanation to the end-users by showing them the key aspect of the question, which our model predicts is related to them.

We will show a complete example of this feature in the results section.

The interaction between users and questions is much more complex than a simple rating mechanism. To overcome the shortcomings of machine learning and traditional recommendation techniques, we utilize a deep neural network to extract these complicated relations between both users and questions, even considering the sparsity of the current CQA platform datasets.

The question title and main body contain rich information about the questions themselves and provide important information about their answerers' expertise. For instance, a user who answered a Python data processing-related question is likely to have the expertise in performing data processing with Python. To extract the rich information from the questions, HER first initializes each question's text feature with Word2Vec embeddings [48] and feeds them to a hierarchical attention-based network [76] to learn an attended-representation, which considers the importance of the sentences and words that best represent the question for the expert recommendation task. Finally, we combine the hierarchically-attended question representation with the latent representations learned from the questions and users' meta-data to predict if a given question is likely to be answered by a given user.

We summarize this section's contributions as follows:

- We propose a novel expert recommender system called HER, which recommends high-quality answerers to answer questions in a CQA platform.
- We conduct empirical analyses on the HER model and provide insights into the salient features that helped in recommending answerers for a given question. The salient feature analysis also improves the explainability of our proposed model.

3.4.1 Problem Mathematical Representation

Before we introduce our HER model, we first define the expert recommendation problem. Given a question Q, the objective of the expert recommendation task is to suggest Q to experts who can potentially provide correct answers. A question can be answered by an expert who has already answered a set of similar questions that reflect his/her expertise in the question domain.

To be specific, given a question Q and set of users, $U = \{u_1, u_2, ..., u_n\}$, our purpose is to find the ideal ranking list of n users $Z = \{z_1, z_2, ..., z_n\} \in U$ who can answer the question correctly and z_1 is potentially a better expert than the z_2 for answering this question. Then, we recommend this question to extracted experts following the calculated ranking in the Z. $f(Q, u)$ represents this mapping function where Q is a question, and u is a user and s is the ranking score.

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 CQA dataset. The 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. Specifically, the DNN component introduces two major hyper-parameters in our model:

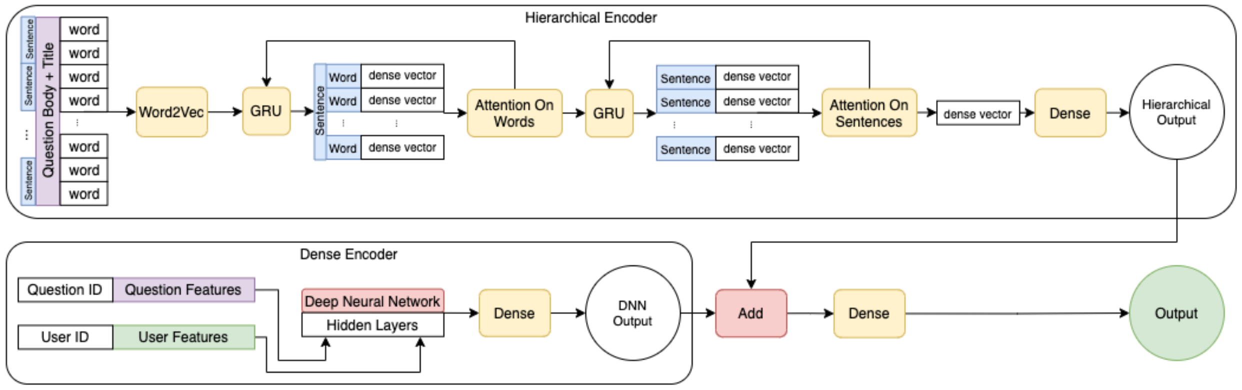


Figure 3.2: HER Architecture

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 [48] model that enables us to find semantically similar words by computing the cosine similarity between word vectors. The Gensim [56] library is used to train the Word2Vec model. We set the length of the vector to be 100, as suggested in [35]. We leave this vector trainable in the learning phase to have a fine-tuned vector based on our data. Now we have all the inputs we need for training HER.The architecture of HER is shown in Fig. 3.2.

As shown in Fig. 3.2., HER has two main components. The first component is a *Dense Enconder*, which utilizes a DNN to find a dense representation of the user and question meta-data and interactions. The second component the *Hierarchical Encoder*, which learns a dense representation of question texts. In the subsequent sub-sections, we will discuss the two components in greater details.

3.4.2 Dense Encoder

HER utilizes a DNN to extract the latent user and question features. The CQA data is often sparse, and extracting the interaction latent features between users and questions are impractical with machine learning methods. The factorization machine can be beneficial, but we have empirically checked that it is not able to outperform a set of hidden connected layers because of the sparsity issue in the dataset. The DNN contains a large set of hidden layers, and each has a specific number of units to help extract latent features. 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.

$$MLP(V, AFunc) = AFunc(WV + b) \tag{3.39}$$

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.

3.4.3 Hierarchical Attention Encoder

The most informative piece of data available on CQA platforms is the question text. The hierarchical attention networks are promised to extract multilevel information from texts. However, there was not any implementation of text feature extraction using hierarchical approaches in recommendation systems. Therefore, we implemented this hierarchical encoder to capture fine-grained information about each question and relate them to experts accurately.

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) [17] to extract the information of each word and give us a dense representation of words in each sentence. To emphasize the 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. Next, we add another Bidirectional GRU layer to perform the same operation on the sentences and force the model to learn the important sentences using another attention layer.

$$L_{w|s} = MLP(V_{w|s}, tanh) \tag{3.40}$$

$$A_{w|s} = \frac{exp(L^T_{w|s} E_{w|s})}{\sum_{W|S} exp(L^T_{w|s} E_{w|s})} \tag{3.41}$$

$$S_{w|s} = \sum_{W|S} A_{w|s} V_{w|s} \tag{3.42}$$

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. Next, we compute the word or sentence's attention value as the similarity of V with the embedding E to get a normalized attention value A after feeding it to a softmax function. Lastly, we measure the summary 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.

3.4.4 Representation Fusion and Classification

Finally, after learning two dense representations, we combine the learned representations to perform expert recommendation. Specifically, we add our two outputs together and connect them to a sigmoid activation function to predict the likelihood of a user answering the given question. Lastly, 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.

$$p(\gamma, \bar{\gamma} | \mathbf{U}, \mathbf{Q}, \theta) = \Pi_{(u,q)\in\gamma} \hat{y}_{uq} \Pi_{(u,q)\in\bar{\gamma}} (1 - \hat{y_{uq}}) \tag{3.43}$$

$$L = - \sum_{(u,q) \in \gamma \cup \bar{\gamma}} y_{uq} \log \hat{y}_{uq} + (1 - y_{uq}) \log (1 - \hat{y}_{uq}) \tag{3.44}$$

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, $\hat{\gamma}$ is negative samples and θ is our model parameters.

3.4.5 Implementation

In the Dense Encoder, we set the number of hidden layers to 1024, and for each layer, we have 1024 units. The dropout rate for the dropout layer is empirically set to 0.2. These hyperparameters are tuned based on the validation set. In the Hierarchical Encoder, we train a word2vec model on the question to learn the word embedding. We set 12 words as the maximum length of each sentence and a total of ten sentences in every question.

4 Results and Discussion

In the previous chapter, the concepts and the implementation of the machine learning methods for expert recommendation systems, neural expert recommendation, and hierarchical expert recommendation systems were explained.

In the first section of this chapter, we start with the performance analysis of different machine learning methods and the results of various algorithms, i.e. Naive Bayes, Random Forest, Logistic Regression and Support Vector Machines are discussed and compared. The main shortcomings of machine learning models are articulated using tailored experiments and metrics to these models. In a nutshell, the dimensionality of the dataset and lack of generalization in imbalanced datasets are the main reasons to avoid machine learning methods for this task.

In the second section, the performance of Generalized Matrix Factorization, Multi-layer Perceptron and hybrid method using these two methods are compared together. The section clearly shows the advantages that a deep neural network can bring to traditional recommendation systems. The last section compares the performance of our proposed Hierarchical Expert Recommendation and state-of-the-art expert recommendation systems.

4.1 Analyzing Machine Learning Methods for Expert Recommendation

4.1.1 Experimental Settings

Our pipeline for this section consists of data collection, feature engineering, and modelling phase. In the data collection phase, we retrieved the StackOverflow archive and choose the recent one year span. Further, we cleaned our textual data by removing all unnecessary characters and limiting question languages only to English characters. In the feature engineering phase, we extract the expert-question interaction matrix. First, we applied the TF-IDF method. Our corpus consists of 52,326 terms, which led to a gigantic sparse matrix. This large number of dimensions would be troublesome. So we limited our corpus to 100 most common terms. At this stage, we split our questions into 80% training set, which will be further used for cross-validation [37], and the remaining 20% as the test set. Further, we obtained experts TF-IDF terms as the average of question titles they had answered and compute the average score of these answers.

In the modelling phase, we designed models taking into cognizance the imbalanced class distribution (four negative samples per each positive sample) in our dataset. Random Forest hyperparameters were optimized using a randomized search cross-validation method [8], which executes a randomized search over parameters that samples each setting from a distribution over possible parameter values. We picked this method over the exhaustive search for its efficiency. We used f1-weighted scoring metric for optimization because of the importance of positive class; additionally, F1-Score considers class imbalance by calculating metrics for each class, and finding their average weighted score. For the Logistic Regression method, we examined both *SAG* and *LIBLINEAR* solvers due to their ability to handle high dimensional datasets. Furthermore, we adjusted the class weights inversely proportional to the class frequencies considering our unbalanced dataset. For the SVM model, we utilized the radial basis function kernel for its generalization and for the value of gamma, which specifies the impact of each data point, we analyzed both auto and scale. Auto specifies gamma as inversely proportional to the number of features and scale considers the variance of samples in addition to the number of features. Moreover, we adjusted the class weights inversely proportional to the class frequencies regarding the imbalanced class distribution in our unbalanced dataset.

In the evaluation phase, we first reduce the problem from a ranking expert to a classification problem for these methods. In this case, if these models could solve them effectively, we would consider using them to solve our main task, which is an expert recommendation. In order to find the strengths and weaknesses of each model, we analyze our models based on various metrics. We consider that our positive class — the correct pairs of questions and users who already answered those questions — is more critical to our primary purpose for this reduced problem. The reason is that finding an expert who can answer a question is more valuable than finding one who can not answer that question. We choose specific metrics for evaluation to draw a solid conclusion out of this part of the experiment. These metrics are as follows:

- We computed sensitivity to find the performance of our models on predicting the positive class. The sensitivity or recall refers to the proportion of samples that are actually positive and yield a positive result.

$$\text{Sensitivity} = \frac{\text{True Positive Class Instances}}{\text{All Positive Class Instances}} \tag{4.1}$$

- Balanced accuracy is computed as the accuracy of samples weighted based on the inverse occurrence of its true class. Accuracy would obtain inflated results for predicting the negative class and would not be an appropriate metric for our problem. This metric is based on (1) sensitivity (true positive rate) and (2) specificity (true negative rate).

$$\text{Balanced Accuracy} = \frac{\text{Sensitivity} + \text{Specificity}}{2} \tag{4.2}$$

- We computed precision, which measures the ability of the classifiers not to classify negative samples as positive and calculate recall to classify all positive samples correctly. This metric is the fraction of positive class instances among all the outcomes.

$$\text{Precision} = \frac{\text{Positive Class Retrieved Instances}}{\text{All Retrieved Instances}} \tag{4.3}$$

- We estimated the weighted F1-Score due to our imbalance dataset, which specifies the harmonic mean between precision and recall for each class and finds the average weight based on the number of true samples.

$$\text{F1-Score } = 2 \cdot \frac{\text{Precision } \cdot \text{ Recall}}{\text{Precision } + \text{ Recall}} \tag{4.4}$$

4.1.2 Results

Figure 4.1 presents a comparison between the performance of our models. Logistic regression has the best performance in predicting the experts who will answer the question (SAG solver handles our sparse data appropriately). In contrast, SVM and Random Forest have the best accuracy and precision-recall score, indicating the ability of the models in discriminating classes and predicting both classes correctly. Interestingly Random Forest and Naive Bayes have the best precision, recall and F1-Score. However, the sensitivity of Random Forest and Naive Bayes shows that these models are biased over negative class by predicting negative for most of the test set. Thus, the F1-Score, precision and recall are affected by predicting our unbalanced negative class and do not measure the performance of our models appropriately. Logistic Regression results indicate the model's capability in learning the positive class with the lowest ability to learn the negative class correctly. Therefore, this model is useful for finding the experts who will answer the question but might also offer experts who would not answer the question. SVM results show the model's acceptable performance in inferring both positive class and negative class. This model has inflated F1-Score but acts the best with having acceptable sensitivity along with the highest average precision-recall score. This model performs better than other models in recommending an expert who would answer the question and not recommending an expert who would not answer the question.

Expert recommendation in SO is arduous because of the complex structure of the network, such as various topics that also have semantic relation to each other. For instance, the loop concept is repetitive in many languages. However, if an expert has answered a loop question in Python language, there is no guarantee that this user would also answer loop related questions in Go language. Thus, the programming language of the answered question is more critical than the topic of the question. Furthermore, in the current state of SO, numerous experts are knowledgeable in one area but didn't have the chance to answer any questions on this topic, which makes prior knowledge about experts misleading. Consequently, designing a model that can consider all these aspects is challenging.

The imbalanced class label in our dataset impacts F1-Score, and sensitivity only considers the positive class. Therefore, the average precision-recall score is a more reliable metric, accounting for both negative and positive classes with a focus on the negative class. Naive Bayes and Random Forest are profoundly impacted by our imbalanced dataset. Random Forest has the best capability in working with a high dimensional dataset, but decision trees are biased toward negative class and lead to the biased forest over the majority class. Logistic Regression and SVM overcome this overfitting by regularization.

The main problem with our dataset is its dimensionality, where the small number of TF-IDF terms

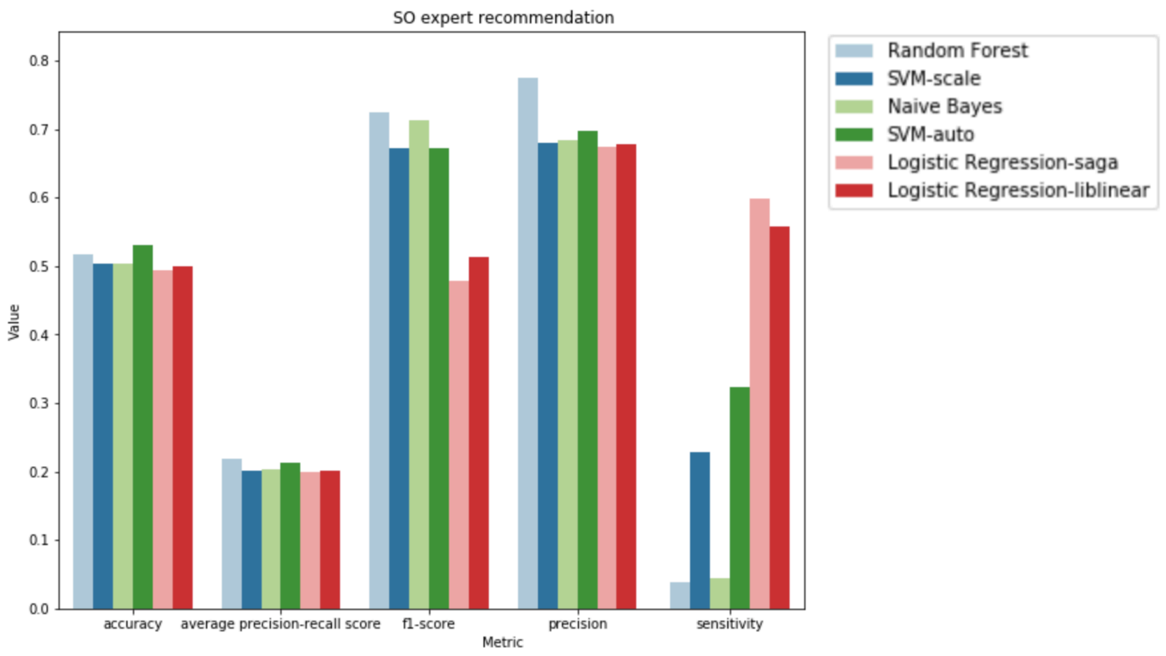


Figure 4.1: Machine Learning Methods' Results

result in different subjects overlapping with each other, and the vast number of terms leads to the *curse of dimensionality problem*[7]. Furthermore, we compared our models using many metrics and recognize the best metric for our problem.

These results suggest that using machine learning algorithms could not solve this problem effectively. The reason is that the real dataset is extremely imbalanced. Millions of users did not answer a particular question, and only a handful of them might answer that, and finally, only one of those got accepted as the correct or best answer to that question. As the results suggest, this attribute of the data will impact all of these methods to some degree. As the Balanced Accuracy is around 50% and Average Precision and Recall is near 20%, these methods are on par entirely with the distribution of the dataset and are biased toward that. However, the methods that use regularization achieved the best results, and it is a powerful sign to prove that deep learning can handle these tasks more effectively. Furthermore, to overcome the dimensionality problem, we need to consider deep learning methods in addition to other parts of our final model. The next step in our study is creating a hybrid method utilizing traditional recommendation systems and a simple deep neural network.

4.2 Neural Expert Recommendation

4.2.1 Experimental Settings

To evaluate the performance of the expert recommender, for each question in the test set, 100 negative experts are randomly sampled, and the model ranks all users, including the user who has truly answered the question

Table 4.1: Best Performance of Each NCF model

Model	NDCG@10	HR@10
GMF	0.0518	0.1095
MLP	0.0586	0.1219
NueMF	0.0586	0.1216
MF	0.0585	0.1215

based on prediction. For instance, in the labelling of our data, the expert who answered the question has label 1 and others are zero, so the model should rank it higher than others. Afterwards, we applied rank-based metrics such as Hit Ratio(HR) and Normalized Discounted Cumulative Gain (NDCG)[30] to evaluate the expert recommender. These metrics are calculated for the top 10 rank list. HR indicates whether the test expert is present in the top 10, and NDCG [30] considers the position of the hit by measuring higher scores for the top ranks.

We used this method and metrics so that we can compare it to NeuMF [31] result. Then, we calculated both metrics for each test user and reported the average score. We conducted this experiment for 20 iterations and chose the model with the best NDCG as the best iteration. Moreover, we explored the performance of NeuMF with both pre-trained and unpre-trained GMF and MLP models.

4.2.2 Baseline

We used the Matrix Factorization method [40] as the baseline for our model, which is the de-facto approach to latent factor model-based recommendation. This model performance highly relies on the choice of user-interaction function. This method uses direct factorization of the incomplete matrix into low-rank U and V, with an L1 sparsity penalty on the elements of U and an L2 penalty on the elements of V to complete the matrix. We solved the factorization by gradient descent and set the learning rate to 0.001 so that small steps are taken towards the minimum without jumping over it. Furthermore, the number of latent factors is set to default 10, which is recommended by the literature [40]. In this project, NeuMF aims to enhance this model by generalizing this user-interaction function using deep learning in the expert recommendation domain.

4.2.3 Results

Figures 4.2 and 4.3 demonstrate the performance of the NCF framework in SO. MLP has the best NDCG and HR between the 5 models. Table 4.1 indicates the best performance of these models. MLP has better performance than MF as the baseline. However, results indicated that GMF is prone to overfitting with the lowest metrics and NueMF with pre-trained parameter initialization has the same performance as MLP. Consequently, these results exhibited that the latent factors have different weights, and expert and question latent are linearly related to each other.

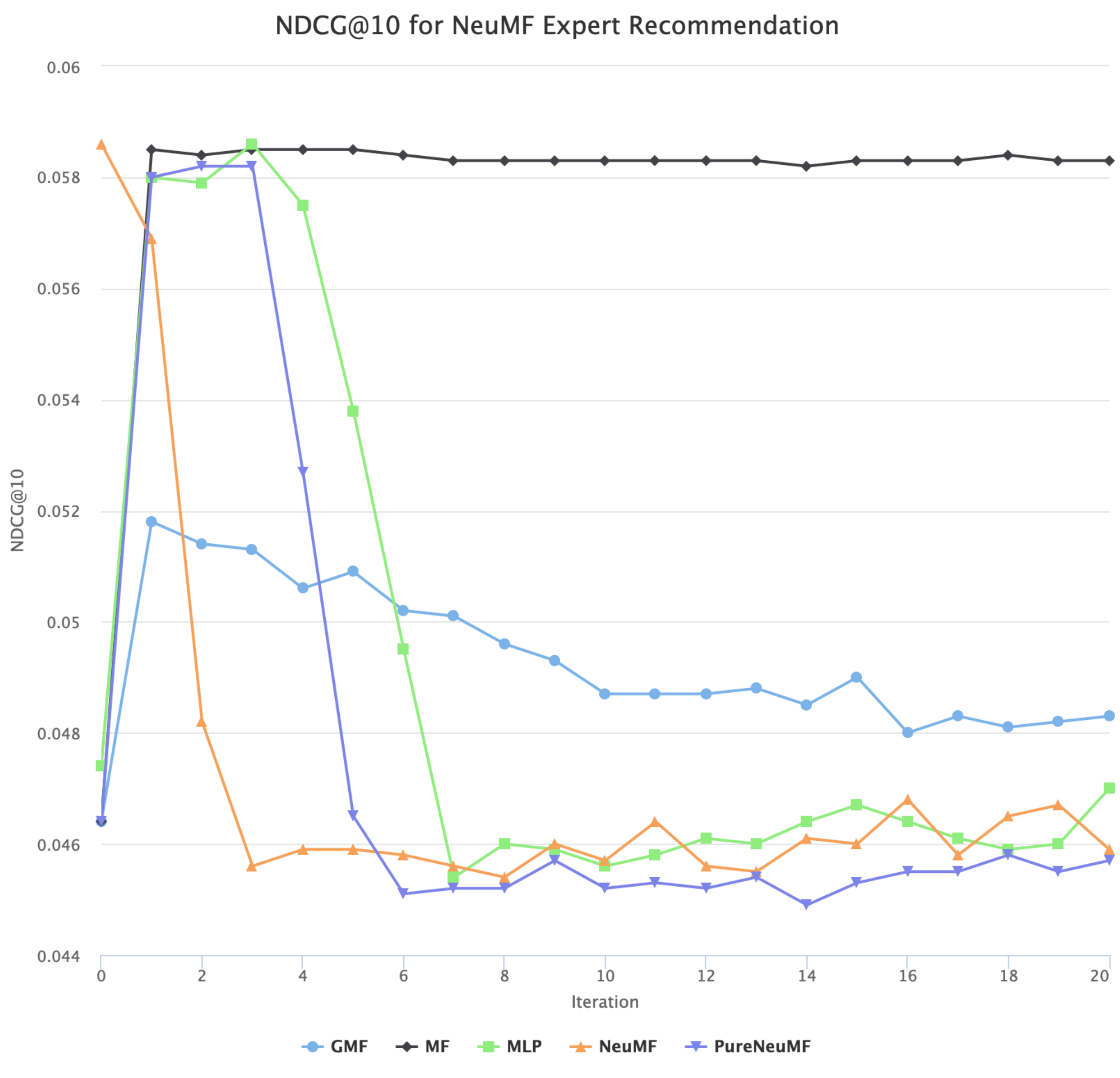


Figure 4.2: Recommendation NDCG Performance of NCF framework in SO

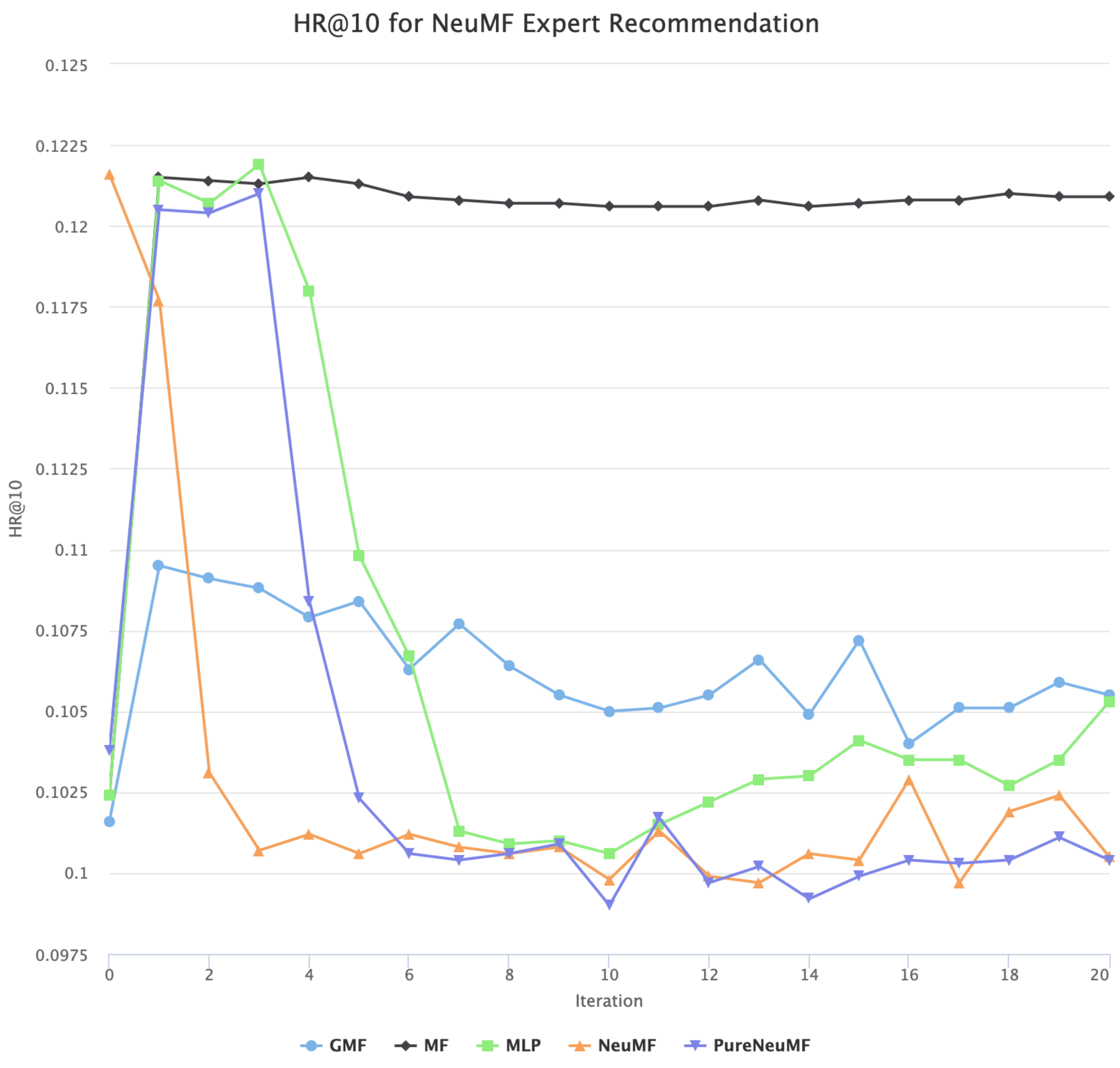


Figure 4.3: Recommendation HR Performance of NCF framework in SO

The results indicated that the linear inner product is sufficient for learning the expert-question interaction data, but these latent features have distinctive weights. NueMF had better performance than MF. Each of MLP and GMF models took four hours to converge using 4 GPUs, NueMF with pre-trained parameters converged in 1 day, and NueMF without pre-trained parameters completed execution in 3 days. MF only needs 5 minutes to factorize the interaction matrix. Therefore, the computational overhead of NCF models is substantial without improving the performance dramatically. [19] found that many deep learning models bring tiny advancement in performance with thoroughly investigating the best hyper-parameters and compared their models to the normal baseline without tweaking the hyper-parameters for the best performance. In [19], the authors investigated the NCF framework and realized that this method didn't have better performance than eALS[32], the state-of-the-art machine learning recommender. In this section, our results show the same outcome as [19], where NueMF improves MF by only 0.1%.

These findings give us a clear image of what we need to design to outperform the existing recommendation systems. Leaning heavily on small dimensional feature sets can not hugely improve the result of the deep networks, and in these situations, machine learning algorithms can manage the data computationally more cost effectively. On the other hand, machine learning algorithms are unsuitable for complex and multi-dimensional data like textual data. We already know that deep neural networks can slightly improve traditional recommendation methods. Also, we know that processing complex data such as texts is viable using deep neural networks. We saw the opportunity to fuse these two techniques to improve the accuracy of the ERS further. The result of the fusion method, also called Hierarchical Expert Recommendation (HER), is presented in the next section.

4.3 Hierarchical Expert Recomendation

4.3.1 Data Analysis

The dataset and the preprocessing has been described in previous sections. In this section, we perform an analysis of the dataset to better understand this data and help us evaluate the results more profoundly. Fig. 4.4 shows a histogram that plots the number of users against the number of answers. From Fig. 4.4, we note that most of the users have answered between 15 to 50 questions, and fewer users have answered more than 100 questions.

We also examine the types of questions answered by the expert users in our dataset. Fig. 4.5 shows the top 10 most used tags in questions answered by the expert users. The top 10 most used tags in the dataset appeared in more than 50% of the questions. We decided to choose the top 50 tags which covers more than 76% of the questions. Interestingly, we notice that close to 60% of the questions have used the top 10 most used tags and that the top two tags, Python and Javascript, made up more than 25% of the questions.

After retrieving the dataset, we extract the relevant user and question meta-data that are used as input features in the dense encoder of our HER model. Specifically, we extracted the following meta-data:

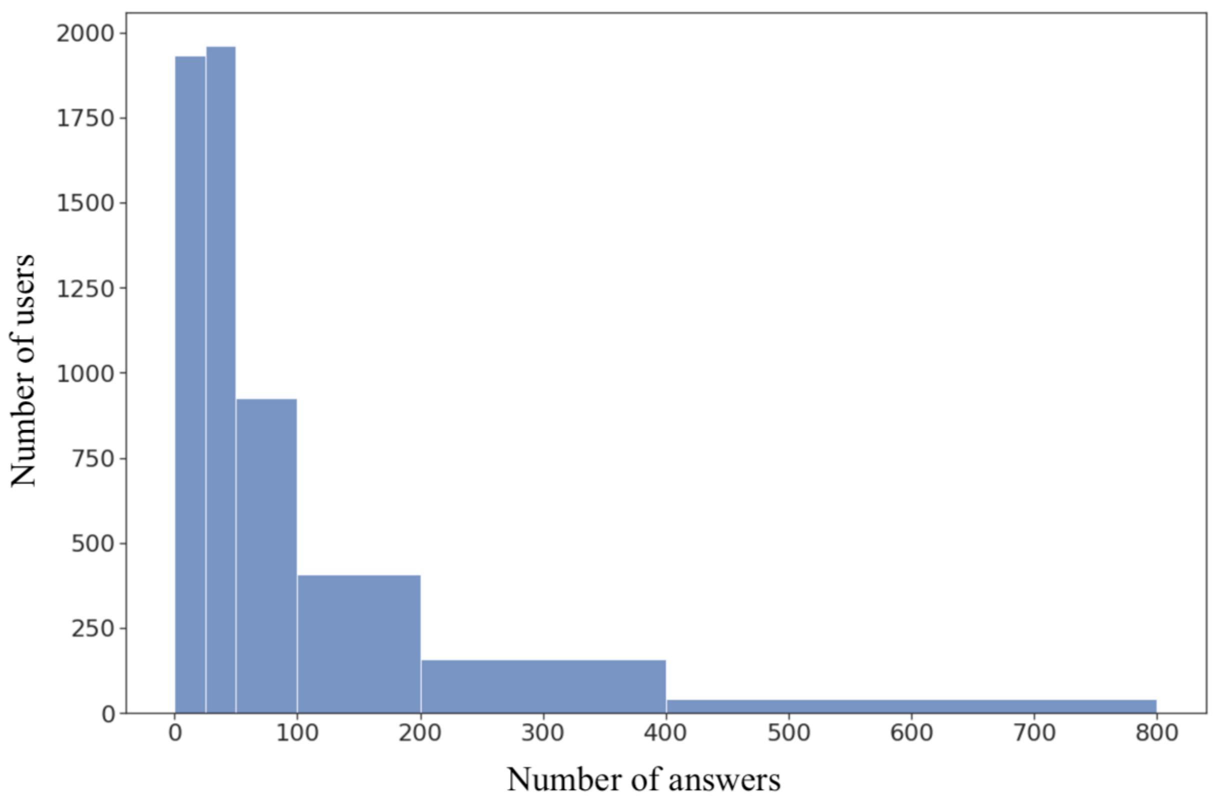


Figure 4.4: User-answer distribution.

- Question features
  - Vote: Voting is how the community indicates which questions and answers are most useful and appropriate, which can be either affirmative or negative(i.e., upvote and downvote).
  - Tags: Binary indicator of the top 50 most used tags. For instance, if a question has "Python" and "Pandas" tags, the corresponding binary indicator for the two tags will be '1', while the other tags' binary indicators will be '0'.
  - Vote Difference: The difference between question's number of upvotes and downvotes. Note that this feature could be a positive or negative value.
  - View Count: Number of times this question has been viewed.
- User features
  - Views: Number of times this user has been viewed on Stack Overflow.
  - Upvote: Number of affirmative votes the user has obtained in previous questions and answers.
  - Downvote: Number of negative votes the user has obtained in previous questions and answers.
  - Reputation: a user is awarded 10 reputation points for receiving an upvote on an answer given to a question and 10 points for an upvote for his or her answer and question.

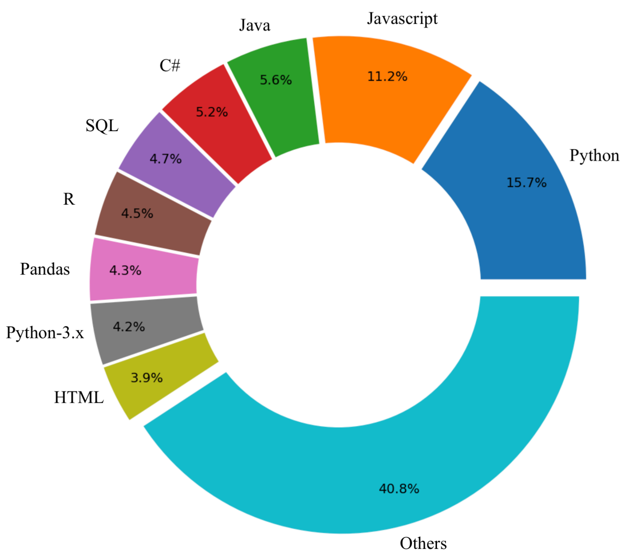


Figure 4.5: Type of questions answered by users

Table 4.2: Distribution of User-Answered Question Interactions

# Dataset	# Users	# Questions	Sparsity
Main	5,448	392,798	99.98%
Python	2,211	54,471	99.95%
JavaScript	2,315	35,160	99.95%

The hierarchical encoder in HER takes in the question's textual features (i.e., title and body) as input. To pre-process the questions' textual features, we first lower-case all characters in the question text features. Next, we fix the length of sentences by limiting the number of words in each sentence and padding with white space the last sentences to equal length. Finally, we create equal-length questions by limiting the number of sentences in each question. The limits are set empirically by computing the average length of sentences and questions. Specifically, we choose 12 words as the maximum length of each sentence and limited the number of sentences in every question to 10.

4.3.2 Experimental Settings

The first step in the project is preparing the data. We used the Stack Overflow dataset between 2018 and 2019. As shown in the figure 3.2, the HER needs two different feature sets. Question texts consist of question text without stopwords and the code parts. We then make them all normalized by creating all non-capital characters, fixed-length sentences by limiting the number of words in each sentence, and padding the last sentences to make them equal in length and finally create equal length questions limiting the number of sentences in each question. For finding the limit for each part, we found the average length of sentences and the average length of questions beforehand. Then we choose them as the starting hyperparameter for our model, and then using the validation set, we optimize them to have the best possible results. The second part is question and user features, both of them have identification part and feature part. We encode these IDs to one-hot encoding and normalize the static features and tags with scalers.

Baselines. We benchmark HER against the commonly-used recommendation methods: Matrix Factorization (MF), Deep Factorization Machine (DeepFM) [28], and Product-Based Neural Network (PNN) [53]. DeepFM utilizes Factorization Machine (FM) [57] to extract latent features for questions and users based on their interaction and features.

Training and Testing Set. In our experiments, we adopt an 80-20 split, where for each dataset, 80% of the questions are used for training with the remaining 20% used for testing. Our recommendation task's goal is to recommend a more relevant user (expert) to the specific question. Therefore, we only care about the presence of the expert in our recommended user list. 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 if our model can rank the expert higher than the others.

Evaluation Metrics. We adopt evaluation metrics that are commonly used in recommendation studies. Specifically, we adopted the following evaluation metrics:

- *Mean Reciprocal Rank (MRR)*, is a statistic ranking correctness measure of possible responses to the number of queries ordered by their predicted correctness value.
- *Normalized Discounted Cumulative Gain (NDCG)*, is a measure of ranking quality. In information retrieval, it is often used to measure the effectiveness of retrieved results. Using a graded relevance scale of items in a system result set, DCG measures the usefulness or gain of an item based on its position in the result list. The gain is accumulated from the top of the result list to the bottom, with the gain of each result discounted at lower ranks. To make this measurement comparable and independent of the context of our system, we divide the DCG by the ideal DCG to have a number between 0 and 1 and call it NDCG.
- *Hit Rate (HR)*, is the percentage of relevant item presence in the retrieved item list.

4.3.3 Results

Table 4.3 shows the experiment results. We observe that HER outperforms all the baselines. We note the poor performance of MF, which highlights the challenge of adopting collaborative filtering approach for expert recommendation task. We also observe that DeepFM was able to outperform PNN. However, HER has achieved better results on the expert recommendation task, demonstrating the model's ability to learn a better representations of the questions and user expertise.

One of the most informative features used in expert recommendation is the question tags. This raises some concerns about the effectiveness of the proposed model because tags are directly used by Dense Encoder and it can bias the model toward the the dominant tags. In order to eliminate this concern, we decide to conduct

Table 4.3: Main Data-set Results

Model	HR@1	HR@5	HR@10	HR@50	NDCG@5	NDCG@10	NDCG@50	MRR@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**

Table 4.4: Python Data-set Results

Model	HR@1	HR@5	HR@10	HR@50	NDCG@5	NDCG@10	NDCG@50	MRR@100
MF	0.0132	0.0638	0.1206	0.5683	0.0377	0.0559	0.1497	0.0481
DeepFM	0.0267	0.1025	0.1729	0.6248	0.0649	0.0874	0.1822	0.0628
PNN	0.0291	0.0995	0.1629	0.6280	0.0642	0.0845	0.1831	0.0626
HER	**0.0400**	**0.1150**	**0.1698**	**0.6395**	**0.0779**	**0.0954**	**0.1936**	**0.0676**

two different experiments using only one tag to check the effectiveness of our final model in comparison to the baselines. To analyze the effectiveness of the hierarchical encoder, which learns question representation from text instead of tags, we conducted further experiments that evaluate the model's performance when limiting the importance of question tag features.

Specifically, we created two separate sub-sample of the main dataset used in our experiments, where each sub-sampled dataset only contains questions of a specific tag. We selected the top two used tags in our dataset, namely, "Python" and "Javascript". The distributions of the user-answered question interactions for the two sub-sampled datasets are shown in Table 4.2.

Tables 4.4 and 4.5 present the expert recommendation results on the two sub-sampled datasets. We observed a significant decrease in the performance of all models. Nevertheless, HER still outperforms the baselines in both sub-samples, demonstrating the hierarchical encoder's effectiveness in learning better question representation from the rich question textual information.

Table 4.5: JavaScript Data-set Results

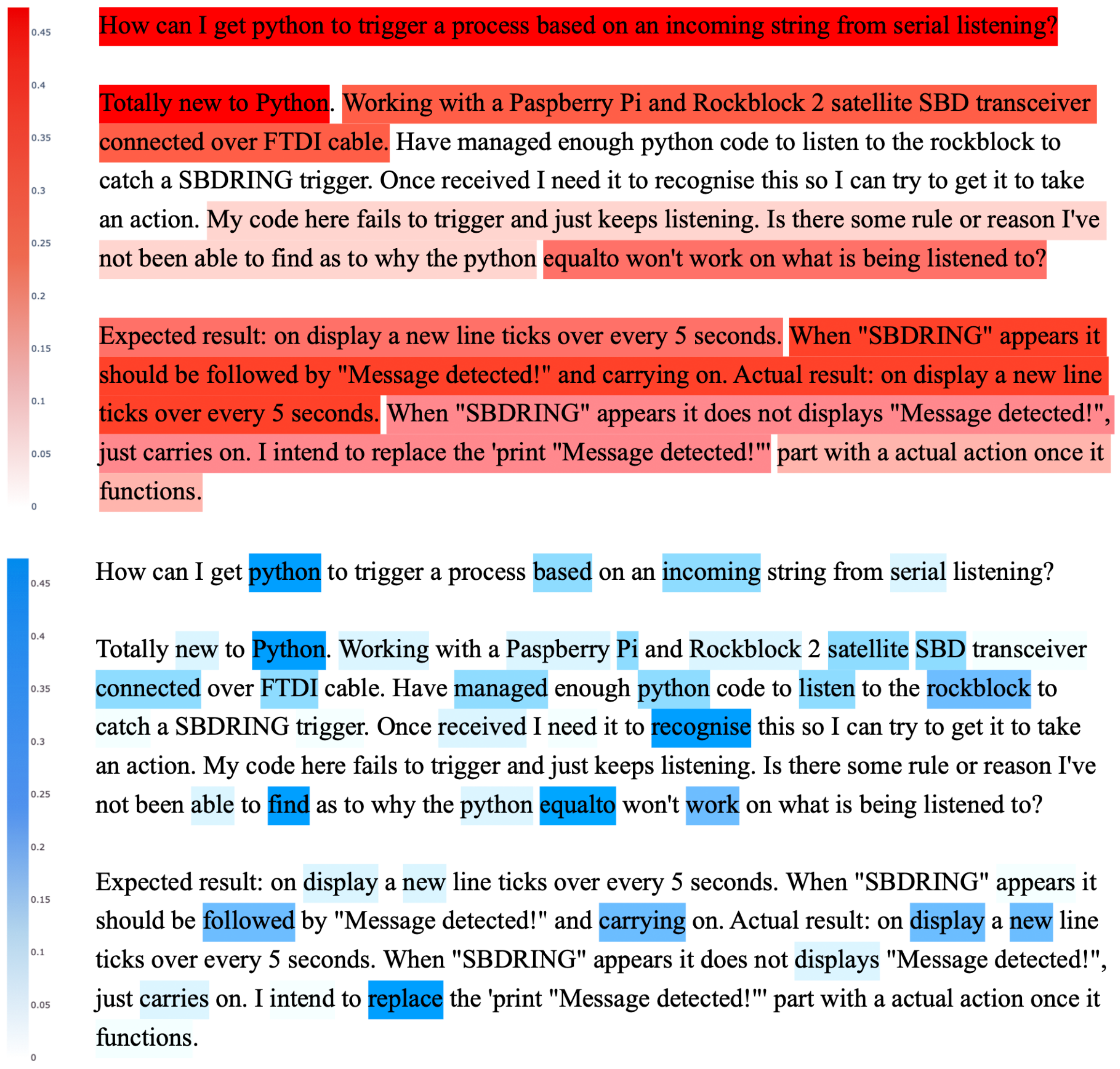
Model	HR@1	HR@5	HR@10	HR@50	NDCG@5	NDCG@10	NDCG@50	MRR@100
MF	0.0071	0.0491	0.1032	0.5429	0.0274	0.0445	0.1380	0.0432
DeepFM	0.0338	0.1067	0.1814	0.5989	0.0698	0.0939	0.1824	0.0654
PNN	0.0352	0.1097	0.1846	0.5827	0.0725	0.0964	0.1807	0.0659
HER	**0.0386**	**0.1228**	**0.1976**	**0.6043**	**0.0808**	**0.1048**	**0.1900**	**0.0696**

4.3.4 Explaining Expert Recommendation

The last problem that we had with our architecture was explainability. One of the disadvantages of using deep learning is that we usually do not have any explanation about how the model works and, in our case, why our model suggests a user answer a specific question. To overcome this problem, we decided to extract the Hierarchical Encoder weights and apply them to each sentence and words to find their importance in order to create explainable descriptions for our recommendations.

To analyze how our HER model is able to extract important features from the textual information to perform expert recommendation, we examine the most salient sections of a single question's latent representations. We adopt the saliency score defined by [42] to measure the saliency of input features. The score indicates how sensitive a model is to the changes in the embedding input, i.e., in our case, how much a specific word or sentence contributes to the final recommendation decision. Fig. 4.6 the visualized computed saliency scores at sentence level (red) and word level (blue). Note that the more salient a sentence in the question, the darker its shade of red. Similarly, the more salient a word in a sentence, the darker its shade of blue.

As shown in Fig. 4.6, we noted that the first sentence is observed to be most salient. In our model, the first sentence is the title of the question. Unsurprisingly, users who ask questions would try to summarize the question in its title before describing the problem in greater detail in the question body. We didn't observe any specific patterns in highlighting the most salient words in a sentence. However, we note that descriptive words such as the question domain expertise are highlighted (e.g., "Python"). The visualized saliency scores demonstrated the effectiveness of HER's hierarchical encoder in modeling the rich content in the question textual descriptions.



How can I get python to trigger a process based on an incoming string from serial listening?

Totally new to Python. Working with a Paspberry Pi and Rockblock 2 satellite SBD transceiver connected over FTDI cable. Have managed enough python code to listen to the rockblock to catch a SBDRING trigger. Once received I need it to recognise this so I can try to get it to take an action. My code here fails to trigger and just keeps listening. Is there some rule or reason I've not been able to find as to why the python equalto won't work on what is being listened to?

Expected result: on display a new line ticks over every 5 seconds. When "SBDRING" appears it should be followed by "Message detected!" and carrying on. Actual result: on display a new line ticks over every 5 seconds. When "SBDRING" appears it does not displays "Message detected!", just carries on. I intend to replace the 'print "Message detected!"' part with a actual action once it functions.

How can I get python to trigger a process based on an incoming string from serial listening?

Totally new to Python. Working with a Paspberry Pi and Rockblock 2 satellite SBD transceiver connected over FTDI cable. Have managed enough python code to listen to the rockblock to catch a SBDRING trigger. Once received I need it to recognise this so I can try to get it to take an action. My code here fails to trigger and just keeps listening. Is there some rule or reason I've not been able to find as to why the python equalto won't work on what is being listened to?

Expected result: on display a new line ticks over every 5 seconds. When "SBDRING" appears it should be followed by "Message detected!" and carrying on. Actual result: on display a new line ticks over every 5 seconds. When "SBDRING" appears it does not displays "Message detected!", just carries on. I intend to replace the 'print "Message detected!"' part with a actual action once it functions.

Figure 4.6: Visualized saliency scores on a sampled question.
The more salient a sentence in the question, the darker its shade of red. Similarly, the more salient a word in a sentence, the darker its shade of blue.

5 Conclusion and Future Directions

At first, we proposed a machine learning pipeline for expert recommendation in StackOverflow and designed features that are representative of users and questions. We bench Naive Bayes, Support Vector Machine, Logistic Regression, and Random Forest methods on the 2019 SO dataset. SVM model performs the best in recommending an expert who would answer the question and not recommending an expert who would not answer the question, and Logistic Regression has the best performance in finding the experts who will answer the question. We examined these methods using textual features from question titles, and experts answered question titles that result in a close performance compared to the prior work on the earlier version of the dataset.

Secondly, we investigated the application of the collaborative filtering NCF framework in SO platform. The results projected that latent features were not equally important, which further indicated the need to utilize content-based filtering recommenders in SO to consider textual features of questions and experts' field of interest. Moreover, attention-based neural networks [69] is a probable venue for the future of expert recommenders due to its capacity to identify the critical latent features in the interaction matrix.

At last, we proposed the Hierarchical Expert Recommendation (HER) model, a deep learning recommender system that recommends experts to answer a given question in the CQA platform. Specifically, HER utilizes hierarchical attention-based neural networks to learn a better representation of the questions and ultimately models the users' expertise through the user-question interactions. We conducted extensive experiments on a large real-world Stack Overflow dataset and benchmarked HER against the state-of-the-art baselines. The results from our extensive experiments shown that HER outperforms the state-of-the-art baselines in recommending experts to answer questions in the Stack Overflow.

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 that 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.

For future work, we aim to investigate the applicability of dimension reduction methods such as Principal Component Analysis in our problem. Besides, we can design a mixture of both SVM and Logistic Regression methods, where the result of Logistic Regression probability is also fed into the SVM model. Furthermore, we can design a model to recommend based on tags and their respective order and compare its performance to our current results. We would also need to perform more ablation studies to understand the effectiveness of different components in the 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 decide if we should recommend a particular question to the user.

The other aspect of the project that we plan to work on is more diverse in scope. We want to investigate how fair is our recommendation system. Giving all opportunities to only a group of users as experts can negatively impact community growth. We should give all users a fair chance to become experts in the algorithms. The other aspect would be how diverse our recommendations are. If we only tap into one of the expertise of an expert, it can ruin this expert experience of this service, and we miss the opportunity to answer some questions based on the other knowledge of that expert.

There are other ways to improve upon our proposed model. For instance, we can use other multi-domain datasets like Github and Stack Overflow, which enables us to use Github information of a user to gather more data about user expertise and improve our recommendation. The other way can be creating a graph of questions and creating a knowledge graph first to have a more profound understanding of how questions are connected and how experts are correlated to help us find better candidates to answer a specific question.

References

[1] Lada A. Adamic, Jun Zhang, Eytan Bakshy, and Mark S. Ackerman. Knowledge sharing and yahoo answers. *Proceeding of the 17th international conference on World Wide Web - WWW 08*, page 665674, May 2008.

[2] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749, 2005.

[3] Muhammad Aljukhadar, Sylvain Senecal, and Charles-Etienne Daoust. Information overload and usage of recommendations. In *Proceedings of the ACM RecSys 2010 workshop on User-Centric Evaluation of Recommender Systems and Their Interfaces (UCERSTI). CEUR-WS. org*, 2010.

[4] Ashton Anderson, Daniel Huttenlocher, Jon Kleinberg, and Jure Leskovec. Discovering value from community activity on focused question answering sites. *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 12*, page 850858, Aug 2012.

[5] Muhammad Asaduzzaman, Ahmed Shah Mashiyat, Chanchal K Roy, and Kevin A Schneider. Answering questions about unanswered questions of stack overflow. In *2013 10th Working Conference on Mining Software Repositories (MSR)*, pages 97–100. IEEE, 2013.

[6] Ana Belén Barragáns-Martínez, Enrique Costa-Montenegro, Juan C Burguillo, Marta Rey-López, Fernando A Mikic-Fonte, and Ana Peleteiro. A hybrid content-based and item-based collaborative filtering approach to recommend tv programs enhanced with singular value decomposition. *Information Sciences*, 180(22):4290–4311, 2010.

[7] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[8] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13:281–305, 2012.

[9] Vasudev Bhat, Adheesh Gokhale, Ravi Jadhav, Jagat Pudipeddi, and Leman Akoglu. Min(e)d your tags: Analysis of question response time in stackoverflow. *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014)*, page 328335, 2014.

[10] Jesus Bobadilla, Fernando Ortega, Antonio Hernando, and Javier Alcalá. Improving collaborative filtering recommender system results and performance using genetic algorithms. *Knowledge-based systems*, 24(8):1310–1316, 2011.

[11] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. Recommender systems survey. *Knowledge-based systems*, 46:109–132, 2013.

[12] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. *Proceedings of the fifth annual workshop on Computational learning theory - COLT 92*, 1992.

[13] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth and Brooks, Monterey, CA, 1984.

[14] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[15] Oscar Celma. Music recommendation. In *Music recommendation and discovery*, pages 43–85. Springer, 2010.

[16] Tong Chen, Hongzhi Yin, Hongxu Chen, Rui Yan, Quoc Viet Hung Nguyen, and Xue Li. Air: Attentional intention-aware recommender systems. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 304–315, 2019.

[17] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. 2014.

[18] Morakot Choetkiertikul, Daniel Avery, Hoa Khanh Dam, Truyen Tran, and Aditya Ghose. Who will answer my question on stack overflow? *2015 24th Australasian Software Engineering Conference*, page 155164, Sep 2015.

[19] Maurizio Ferrari Dacrema, Simone Boglio, Paolo Cremonesi, and Dietmar Jannach. A troubling analysis of reproducibility and progress in recommender systems research, 2019.

[20] David Van Dijk, Manos Tsagkias, and Maarten De Rijke. Early detection of topical expertise in community question answering. *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR 15*, page 995998, Apr 2015.

[21] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings of the 24th International Conference on World Wide Web*, WWW 15, page 278288, Republic and Canton of Geneva, CHE, 2015. International World Wide Web Conferences Steering Committee.

[22] Martin J Eppler and Jeanne Mengis. The concept of information overload-a review of literature from organization science, accounting, marketing, mis, and related disciplines (2004). *Kommunikationsmanagement im Wandel*, pages 271–305, 2008.

[23] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9, 2008.

[24] Mouzhi Ge, Francesco Ricci, and David Massimo. Health-aware food recommender system. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 333–334, 2015.

[25] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudk, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[26] Carlos A Gomez-Uribe and Neil Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*, 6(4):1–19, 2015.

[27] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: A factorization-machine based neural network for ctr prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, page 17251731. AAAI Press, 2017.

[28] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: A factorization-machine based neural network for ctr prediction. 2017.

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[30] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. Trirank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM 15, page 16611670, New York, NY, USA, 2015. Association for Computing Machinery.

[31] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, WWW 17, page 173182, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee.

[32] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. Fast matrix factorization for online recommendation with implicit feedback, 2017.

[33] Begnaud Francis Hildebrand. *Introduction to Numerical Analysis: 2nd Edition.* Dover Publications, Inc., USA, 1987.

[34] Amirabbas Jalali and Roy Ka-Wei Lee. Hierarchical expert recommendation on community question answering platforms. *Proceedings of the Canadian Conference on Artificial Intelligence*, 6 2021. https://caiac.pubpub.org/pub/cmo7bqwy.

[35] Yoon 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*, pages 1746–1751, 2014.

[36] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[37] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International joint Conference on artificial intelligence*, volume 14, pages 1137–1145. Citeseer, 1995.

[38] Yehuda Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD 08, page 426434, New York, NY, USA, 2008. Association for Computing Machinery.

[39] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009.

[40] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):3037, August 2009.

[41] Long T. Le and Chirag Shah. Retrieving rising stars in focused community question-answering. *Intelligent Information and Database Systems Lecture Notes in Computer Science*, page 2536, Mar 2016.

[42] Guanbin Li and Yizhou Yu. Visual saliency detection based on multiscale deep cnn features. *IEEE Transactions on Image Processing*, 25(11):5012–5024, 2016.

[43] Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Math. Program.*, 45(13):503528, August 1989.

[44] Duen-Ren Liu, Yu-Hsuan Chen, Wei-Chen Kao, and Hsiu-Wen Wang. Integrating expert profile, reputation and link analysis for expert finding in question-answering websites. *Information Processing & Management*, 49(1):312329, 2013.

[45] Jansen J Bernard Liu Zhe. Predicting potential responders in social q&a based on non-qa features. *CHI'14 Extended Abstracts on Human Factors in Computing Systems*, pages 2131–2136, 2014.

[46] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. *Recommender systems handbook*, pages 73–105, 2011.

[47] P. Mccullagh and J. A. Nelder. An outline of generalized linear models. *Generalized Linear Models*, page 2147, 1989.

[48] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[49] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119. Curran Associates, Inc., 2013.

[50] Sara Mumtaz, Carlos Rodriguez, and Boualem Benatallah. Expert2vec: Experts representation in community question answering for question routing. *Advanced Information Systems Engineering*, page 213229, 2019.

[51] Aditya Pal and Joseph A. Konstan. Expert identification in community question answering. *Proceedings of the 19th ACM international conference on Information and knowledge management - CIKM 10*, page 15051508, 2010.

[52] Martin F Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[53] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. Product-based neural networks for user response prediction. 2016.

[54] Yanru Qu, Bohui Fang, Weinan Zhang, Ruiming Tang, Minzhe Niu, Huifeng Guo, Yong Yu, and Xiuqiang He. Product-based neural networks for user response prediction over multi-field categorical data. volume 37, New York, NY, USA, October 2018. Association for Computing Machinery.

[55] J. R. Quinlan. Induction of decision trees. *MACH. LEARN*, 1:81–106, 1986.

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

[57] Steffen Rendle. Factorization machines. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, page 9951000, USA, 2010. IEEE Computer Society.

[58] Fatemeh Riahi, Zainab Zolaktaf, Mahdi Shafiei, and Evangelos Milios. Finding expert users in community question answering. *Proceedings of the 21st international conference companion on World Wide Web - WWW 12 Companion*, 2012.

[59] Francesco Ricci, Lior Rokach, and Bracha Shapira. Recommender systems: introduction and challenges. In *Recommender systems handbook*, pages 1–34. Springer, 2015.

[60] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[61] Hiemstra Djoerd Zaragoza Hugo Rode Henning, Serdyukov Pavel. Entity ranking on graphs: Studies on expert finding. *CTIT Technical Report Series*, 2007.

[62] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2016.

[63] Gerard Salton and Michael J. McGill. *Introduction to modern information retrieval*. McGraw-Hill Intern., 1987.

[64] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW 01, page 285295, New York, NY, USA, 2001. Association for Computing Machinery.

[65] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Math. Program.*, 162(12):83112, March 2017.

[66] Florian Strub and Jérémie Mary. Collaborative Filtering with Stacked Denoising AutoEncoders and Sparse Inputs. In *NIPS Workshop on Machine Learning for eCommerce*, Montreal, Canada, December 2015.

[67] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009, 2009.

[68] Aaron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2643–2651. Curran Associates, Inc., 2013.

[69] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[70] Paula Cristina Vaz, David Martins de Matos, Bruno Martins, and Pavel Calado. Improving a hybrid literary book recommendation system through author ranking. In *Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries*, pages 387–388, 2012.

[71] Tvn Venkatesan, T Ramkumar, and Dr. K. Saravanan. Mining big data: Towards a machine learning framework based on collaborative filtering. volume 10, pages 69–77, 01 2017.

[72] JP. Vert, K. Tsuda, and B. Schölkopf. *A Primer on Kernel Methods*, pages 35–70. MIT Press, Cambridge, MA, USA, 2004.

[73] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems, 2014.

[74] Xianzhi Wang, Chaoran Huang, Lina Yao, Boualem Benatallah, and Manqing Dong. A survey on expert recommendation in community question answering. *Journal of Computer Science and Technology*, 33(4):625653, 2018.

[75] Xinxi Wang and Ye Wang. Improving content-based and hybrid music recommendation using deep learning. In *Proceedings of the 22nd ACM International Conference on Multimedia*, MM 14, page 627636, New York, NY, USA, 2014. Association for Computing Machinery.

[76] Chuhan Wu, Fangzhao Wu, Junxin Liu, and Yongfeng 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)*, pages 1818–1826, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[77] Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, WSDM 16, page 153162, New York, NY, USA, 2016. Association for Computing Machinery.

[78] Baoguo Yang and Suresh Manandhar. Tag-based expert recommendation in community question answering. In *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014)*, pages 960–963, 2014.

[79] Liu Yang, Minghui Qiu, Swapna Gottipati, Feida Zhu, Jing Jiang, Huiping Sun, and Zhong Chen. Cqarank: Jointly model topics and expertise in community question answering. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, CIKM '13, page 99108, New York, NY, USA, 2013. Association for Computing Machinery.

[80] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard 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*, pages 1480–1489, San Diego, California, June 2016. Association for Computational Linguistics.

[81] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD 16, page 353362, New York, NY, USA, 2016. Association for Computing Machinery.

[82] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1):1–38, 2019.

[83] Tong Zhao, Naiwen Bian, Chunping Li, and Mengya Li. Topic-level expert modeling in community question answering. *Proceedings of the 2013 SIAM International Conference on Data Mining*, 2013.

[84] He Tingting Wu Wensheng Zhou Guangyou, Zhao Jun. An empirical study of topic-sensitive probabilistic model for expert finding in question answer communities. *Knowledge-Based Systems*, pages 136–145, 2014.

[85] Liu Kang Zhao Jun Zhou Guangyou, Lai Siwei. Topic-sensitive probabilistic model for expert finding in question answer communities. *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1662–1666, 2012.

[86] Cai Deng He Xiaofei Yueting Zhuang Zhou Zhao, Qifan Yang. Expert finding for community-based question answering via ranking metric network learning. *IJCAI International Joint Conference on Artificial Intelligence*, pages 3000–3006, 2016.

[87] Qiannan Zhu, Xiaofei Zhou, Zeliang Song, Jianlong Tan, and Li Guo. Dan: Deep attention neural network for news recommendation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):5973–5980, Jul. 2019.