Cold Start Thread Recommendation as Extreme Multi-label Classification

Kishaloy Halder
School of Computing
National University of Singapore
kishaloy@comp.nus.edu.sg

Lahari Poddar
School of Computing
National University of Singapore
lahari@comp.nus.edu.sg

Min-Yen Kan
School of Computing
National University of Singapore
kanmy@comp.nus.edu.sg

ABSTRACT

In public online discussion forums, the large user base and frequent posts can create challenges for recommending threads to users. Importantly, traditional recommender systems, based on collaborative filtering, are not capable of handling *never-seen-before* items (threads). We can view this task as a form of Extreme Multi-label Classification (XMLC), where for a newly-posted thread, we predict the set of users (labels) who will want to respond to it. Selecting a subset of users from the set of all users in the community poses significant challenges due to scalability, and sparsity. We propose a neural network architecture to solve this *new* thread recommendation task. Our architecture uses stacked bi-directional Gated Recurrent Units (GRU) for text encoding along with cluster sensitive attention for exploiting correlations among the large label space. Experimental evaluation with four datasets from different domains show that our model outperforms both the state-of-the-art recommendation systems as well as other XMLC approaches for this task in terms of MRR, Recall, and NDCG.

KEYWORDS

Recommendation System; Cold Start; Extreme Multi-Label Classification; Neural Network; Discussion Forum

ACM Reference Format:
Kishaloy Halder, Lahari Poddar, and Min-Yen Kan. 2018. Cold Start Thread Recommendation as Extreme Multi-label Classification. In *WWW '18 Companion: The 2018 Web Conference Companion, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3184558.3191659

1 INTRODUCTION

Online Forums have become an important social media platform across many domains such as health[1], education[2], technical question answering[3], e-commerce, government policy making and so on. Discussion in these forums are usually in the form of *threads*. One starts a thread by posting a question or asking others for opinions on a certain topic. Community members then participate in the thread by replying with their knowledge and opinions on the topic.

These forums are continuously growing as new threads are created frequently. While this enables users to ask questions to a large community, ensuring that the members find questions relevant to their interests, is key to getting them answered. This is a challenging matching problem, due to the huge number of threads, and the large number of active members in the community. Recommendation systems can help bridge this gap by suggesting users with relevant interests and expertise for a discussion thread.

We build a system that recommends incoming threads to relevant users for participation. Recommendation systems mostly use past interaction history of a user or item to solve the matching problem. Even though this strategy can model users, given the threads they responded to in the past, it will fail on new threads. They have no interaction history to facilitate predictions – this is a form of cold start. For such a thread, the system needs to use its textual content in order to find potentially interested users.

We view this cold start thread recommendation from a different perspective – as one of supervised eXtreme Multi-Label Classification (XMLC). XMLC has been applied for text classification in domains where a document can have multiple tags among several thousands of possible tags (e.g., Wikipedia page categorization or product categorization in e-commerce). Recently deep learning approaches have been proposed for this area for better text understanding and handling the large label space efficiently [18].

We propose a novel neural network architecture for this recommendation task. Inspired by the success of Recurrent Neural Networks (RNNs) on a range of natural language processing tasks, we apply stacked bidirectional RNN for encoding the raw textual content of a post. We consider the multi-label prediction task as multiple, individual binary classifications where the correlation among labels (i.e., users) is exploited by the model.

We hypothesize that users can be subdivided into clusters in a latent space depending on their interests. Users belonging to the same cluster are likely to have similar preferences and vice versa. In the literature, we find similar observations in different contexts around recommendation systems [32]. Inspired from this, we introduce a novel, cluster-sensitive attention (CSA) mechanism. It allows a post text to be encoded differently for different clusters using cluster-specific attention weights. This lets the network focus on parts of the text that might be more important for the set of clustered users while predicting their participation interest. Assuming similarity of preferences among users, and learning text encoding per cluster (as opposed to every individual user), helps us in addressing the scalability of the extreme multi-label task by reducing the parameter space. Additionally, it also helps in alleviating the sparsity issue, as the limited amount of evidence per user could easily lead to overfitting in such complex model architecture,

[1]https://www.healthboards.com/boards/
[2]https://www.coursera.org
[3]https://www.stackoverflow.com

This paper is published under the Creative Commons Attribution 4.0 International (CC BY 4.0) license. Authors reserve their rights to disseminate the work on their personal and corporate Web sites with the appropriate attribution.
WWW '18 Companion, April 23–27, 2018, Lyon, France
© 2018 IW3C2 (International World Wide Web Conference Committee), published under Creative Commons CC BY 4.0 License.
ACM ISBN 978-1-4503-5640-4/18/04.
https://doi.org/10.1145/3184558.3191659

otherwise. From our results over multiple datasets, we find that our CSA-based XMLC model outperforms standard content-based recommendation algorithms as well as state-of-the-art XMLC models significantly.

To the best of our knowledge, this is the first attempt at solving the cold start recommendation problem from the extreme multi-label classification perspective. To summarize, our contributions are the following:

- We formulate the well-known cold start recommendation problem as an Extreme Multi-Label Classification task.
- We propose a neural architecture using a novel cluster sensitive attention mechanism to cater to the varying interests of users.
- We show the effectiveness and generalization ability of our approach through a set of carefully-designed experiments, over multiple datasets. Additionally, we validate our problem formulation by comparing our model with traditional recommendation algorithms.

2 BACKGROUND

We first describe the cold start problem commonly found in recommendation systems, and thereafter describe the approaches for extreme multi-label classification. We then conclude this section by connecting these two parts in a formal problem statement.

2.1 Cold Start Recommendation Problem

The two primary elements in a recommendation scenario are users and items. The user–item interaction forms a bipartite graph (Figure 1a) where a directed edge from a user to an item represents that the user has interacted in some way with the item (e.g 'like', 'comment', 'retweet' etc). The corresponding interaction matrix is shown in Figure 1b. In the widely used latent factor models, the user and item are represented in a low-dimensional (D) space - user is denoted by a latent vector $\mathbf{u}_i \in \mathbb{R}^D$, and item by $\mathbf{v}_j \in \mathbb{R}^D$. The prediction r_{ij} is formed by an inner product of these two vectors,

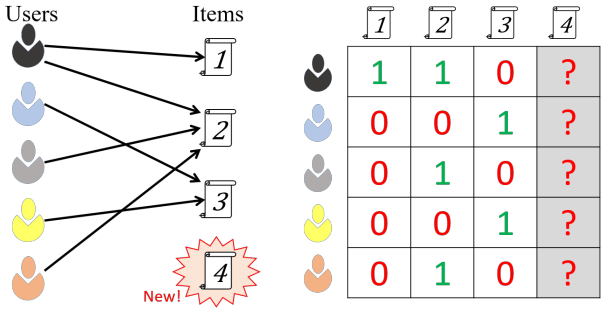
$$r_{ij} = \mathbf{u}_i^T \mathbf{v}_j$$

In non-negative matrix factorization (NMF) based approaches, the latent vectors are initialized randomly and can be learned using a regularized squared error loss in terms of $\mathbf{u}_i$ and $\mathbf{v}_j$, where $i \in \{1, \cdots, U\}$ and $j \in \{1, \cdots, V\}$; U, and V are the number of users and items respectively.

Let's consider the dynamics when a newly-created item '4' is introduced. In the interaction matrix, since the column for item '4' is entirely unobserved, it is also referred as *out-of-matrix* item recommendation [26, 27]. As no ground truth value of r_{ij} for $j = 4$ is available, the model will *not* be able to learn the correct representation of $\mathbf{v}_{j=4}$, giving rise to cold start problem. This is a significant limitation of an NMF based recommender system in a forum context where new threads are posted quite frequently needing user participation.

2.2 Extreme Multi-label Classification

Extreme multi-label classification (XMLC) refers to the task of assigning each item its most relevant subset of labels from an extremely large collection of class labels. The fundamental difference



(a) Interaction Graph. (b) Interaction Matrix.

Figure 1: Illustration of the cold start problem. (a) Edges represent user interactions. Item 4 has no interaction. (b) In interaction matrix: '1' ⇒ interaction, '0' ⇒ no interaction.

between multi-label classification and traditional binary or multi-class classification task is that in multi-class classification only one among the possible labels applies to an item, whereas in multi-label classification the labels can be correlated with each other or have a subsuming relationship, and multiple labels can apply for an item (e.g., 'politics' and 'White House' for news articles, 'electronics', 'Samsung' and 'smartphone' for products, 'Eiffel tower' and 'vacation 2017' for an image).

In this setting, an instance can be considered as a pair $(\mathbf{x}, \mathbf{y})$ where $\mathbf{x}$ is the feature vector for an item, and $\mathbf{y}$ is the label vector i.e., $\mathbf{y} \in \{0, 1\}^L$, L is the number of labels. Given n such training instances, a classifier is trained which can predict the label vector for an unseen test item. Since the label space L can be extremely large, it suffers from scalability, and sparsity issues. Properly exploiting the correlation among labels can help in alleviating them.

Problem Statement: We approach the cold-start thread recommendation problem from an XMLC task perspective, where given a *new* thread, using only its textual features we try to predict the set of interested users. We formalize the problem statement as,

Given a piece of text $t \in T$, find a mapping $f : T \rightarrow \{0, 1\}^U$ where T is the set of all items. f would give us a probability score for each of the U labels given t,

$$f(t) = P(r_i = 1|t)$$

where $i \in \{1, \cdots, U\}$, and r_i is the label corresponding to i^{th} user.

3 PROPOSED METHOD

We propose a neural network architecture (Figure 2) to predict the subset of users interested in a new thread from the extremely large set of users in the forum community. As a newly-created thread has only a single post, we use the terms *thread* and *post* interchangeably.

3.1 Text Encoding

The network takes as input a post text p consisting of a sequence of words $(w_1, w_2, \ldots, w_n)$.

We first embed each word in a lower-dimensional space so that a post is now represented as a sequence of word vectors

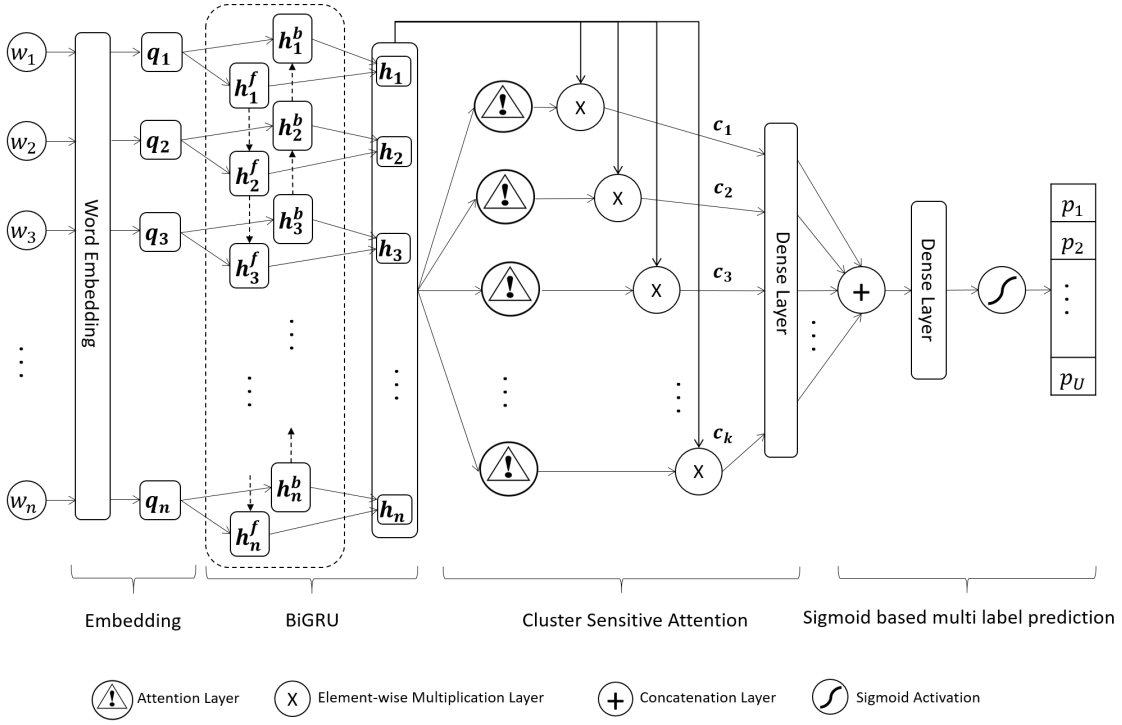


Figure 2: Overall model architecture.

$\{\mathbf{q_1}, \mathbf{q_2}, \cdots, \mathbf{q_n}\}$ where $\mathbf{q_i} \in \mathbb{R}^d$. We initialize the word vectors using pre-trained GloVe embeddings [20] but tune it during training to capture domain specific semantics.

The post is then encoded using bi-directional RNNs. The input to the bi-directional RNN is the embedded word sequence of a post $\{\mathbf{q_1}, \mathbf{q_2}, \cdots, \mathbf{q_n}\}$ and the output is a sequence of vectors $\mathbf{h}^p = \{\mathbf{h_1}, \mathbf{h_2}, \cdots, \mathbf{h_n}\}$ where $\mathbf{h_i} \in \mathbb{R}^g$ denotes the encoded representation of the post.

An RNN reads the sequence of word vectors $\{\mathbf{q_1}, \mathbf{q_2}, \cdots, \mathbf{q_n}\}$ from left to right in the forward pass and creates a sequence of hidden states $\{\mathbf{h_1^f}, \mathbf{h_2^f}, \cdots, \mathbf{h_n^f}\}$, where $\mathbf{h_i^f}$ is computed as:

$$\mathbf{h_i^f} = RNN(\mathbf{q_i}, \mathbf{h_{i-1}^f}) \tag{1}$$

where RNN is a function. Due to vanishing (and conversely, exploding) gradients, the basic RNN cannot learn long-distance temporal dependencies with gradient-based optimization [4]. To deal with this, extensions to the basic RNN have been proposed that incorporate a memory unit to remember long term dependencies. We use one such variant named Gated Recurrent Unit (GRU) [7] instead of the basic RNN in our model.

In the backward pass, a GRU reads the input sequence in reverse order and returns a sequence of hidden states $\{\mathbf{h_n^b}, \mathbf{h_{n-1}^b}, \cdots, \mathbf{h_1^b}\}$. The forward and backward hidden states are then concatenated to create the encoded hidden state of a word $\mathbf{h_i} = [\mathbf{h_i^f}; \mathbf{h_i^b}]$ considering all its surrounding words.

We use a stack of such bi-directional GRUs where the output of a GRU layer is fed as input to the GRU at next level. This increases the expressive power of the network by capturing higher-level feature interactions between different words. The output sequence from the final bi-directional GRU layer is the representation of the post text $\mathbf{h}^p$. In our experiments, we have used a stack of two bi-directional GRUs. We also experimented with adding more layers, but that did not lead to much improvements in our results.

3.2 Cluster Sensitive Attention

> *I have been recommended to undergo tracheotomy and put in a PEG. I am wondering how many days I'll have to stay in the hospital? Will I have a hard time adjusting afterwards? Does the hose need to be connected while transferring? Will the equipments take up a lot of room? How do you call for help? I am unable to talk or move. What type of tube would you suggest? I have been a member of the ALS community for some time now. It is nice to read the way some people think and face ALS, it gives me courage.*

The above is an illustrative post (synthetically modified for anonymity) in an ALS forum. The patient is about to undergo a surgical procedure (tracheotomy) and has queries regarding the procedure, recovery time and the after effects. Furthermore, since the procedure creates a hole in the neck to provide an air passage to the windpipe, it disrupts the normal eating and speaking abilities of a person. The patient therefore has additional questions regarding the best feeding tubes and ways of communicating with others. Given its complexity and detailed information need, individual users are unlikely to be able to answer all parts of it. Instead, we envision that users with different backgrounds and experience

could address specific parts; e.g., someone having experience with a PEG (Percutaneous Endoscopic Gastrostomy) could answer the queries regarding it, while someone else could help clear the user's concerns regarding the procedure and recovery. Put succinctly, different users may be interested in disparate parts of a (new) post.

This motivates us to build a component in our network that can help focus on parts of a post for different users. To achieve this, we need an *attention* mechanism that can give different weights to words of the post and generate an encoded text representation using the weighted words, thus focusing on important parts.

Given the encoded text representation of a post p, as $\mathbf{h}^p = \{\mathbf{h_1}, \mathbf{h_2}, \cdots, \mathbf{h_n}\}$ from the bi-directional GRU component, the attention mechanism [2, 17] weights each of the hidden states of the words i.e. $\mathbf{h_i}$. For each $\mathbf{h_i}$, we compute a weight a_i for its corresponding word w_i and get an attention vector $\mathbf{a} = \{a_1, a_2, \cdots, a_n\}$ as:

$$a_i = \frac{\exp(e_i)}{\sum_{j=1}^{n} \exp(e_j)}, \text{ where} \tag{2}$$

$$e_i = tanh(\mathbf{W_i}\mathbf{h_i} + b_i) \tag{3}$$

where $\mathbf{W_i}$ is a weight matrix of dimension $1 \times g$, and b_i is the bias term. The text representation with attention is then computed as:

$$\mathbf{c} = \sum_{i}^{n} a_i \mathbf{h_i} \tag{4}$$

Note that a single attention layer is insufficient, since the attention weights($\mathbf{a}$) should not be general but should instead be dependent on different users' interests. Naïvely, to achieve per-user attention, we need U such attentions. This will significantly expand the number of parameters to be estimated to an extremely large value ($U \times n \times g$), which is infeasible to train due to the scalability issue. Additionally, in most datasets not enough data-points are available for all users, to reliably learn the individual attention vectors.

We assume that, since the forums are topical, the users can be softly clustered in a finite number of clusters depending on their interests. The number of clusters k, would be much smaller than U (i.e. $k \ll U$). Therefore, instead of learning U different attention vectors, we only need to learn k such vectors. This reduces the parameter space hugely. We call this as cluster sensitive attention mechanism. From the same hidden text representation $\mathbf{h}^p$, we learn k different attention weight vectors $\mathbf{a^1}, \mathbf{a^2}, \cdots, \mathbf{a^k}$. Thereafter, by using the different attention weights on $\mathbf{h}^p$, we get a cluster sensitive encoding of the post text p ($\mathbf{C}^p = \mathbf{c_1}, \mathbf{c_2}, \cdots, \mathbf{c_k}$).

3.3 Multi-label Prediction

For a post p, we concatenate the k text encodings and feed through a fully connected layer with U output neurons. For each of the output neuron (corresponding to each user), the fully connected layer learns the weights for its k inputs (corresponding to the different text encodings).

$$\mathbf{z}_p = tanh(\mathbf{W}.\mathbf{C}^p + \mathbf{b}) \tag{5}$$

where $\mathbf{W}$ and $\mathbf{b}$ are weight and bias matrices respectively and $tanh$ is an element-wise non-linear activation function. The output of this feed-forward layer $\mathbf{z}_p \in \mathbb{R}^U$ is then passed through a *sigmoid* activation function to scale each of its element value in the range [0,1]. The model is trained using binary cross-entropy as the loss function which is defined as,

$$\mathcal{L} = -\frac{1}{T}\sum_{i=1}^{T}\sum_{j=1}^{U}\left(y_{ij}.\log(\sigma(z_{ij})) + (1 - y_{ij})\log(1 - \sigma(z_{ij}))\right) \tag{6}$$

where σ denotes the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$, z_{ij} is j^{th} element in $\mathbf{z}_i$, and y_{ij} is the ground truth value for j^{th} user (label) and i^{th} post. Our network is end-to-end trainable and is optimized with Adam optimizer [13].

4 EXPERIMENTS

To evaluate the generalization ability of our model, we experimented with multiple datasets from different domains involving users, and some form of textual items in a recommendation scenario. We also present a comparison with some of the well-known content based recommendation systems, as well as the state-of-the-art XMLC approaches to show its effectiveness.

4.1 Dataset

We used the following datasets in our experiments.

- **[1-3] Health Forum:** a popular online health discussion forum website where users can post a thread asking something related to their disease. Other relevant users reply in the threads to share their experiences with it. The website consists of subforums for different diseases. We used three subforum datasets i.e., 'Epilepsy,' 'ALS', and 'Fibromyalgia' for the experiments. We removed threads which have replies from lesser than 4 users or greater than 100 to get rid of extremely off-topic or survey threads.
- **[4] Stackoverflow:** is a CQA website for programming related questions. We obtained a data-dump from kaggle[4]. We have used all the questions posted during 2008 – 2010 to form the dataset. We have removed all the code snippets (encapsulated within the tags '<code></code>') from the question texts.

The dataset statistics are presented in Table 1. We observe that the number of labels (i.e., users) is quite large in the stackoverflow dataset. This leads to extremely high sparsity (99.99%) as well. We will describe in Section 4.5 how this affects the recommendation accuracy compared to the others.

4.2 Metrics

In our setting, the label set is huge with very high sparsity. Therefore we do not use overall accuracy as our evaluation metric and only aim to evaluate the positive instances i.e. the users who actually participated in a thread. To ensure participation, the ranking quality of the recommended list of users should be evaluated and commonly used metrics for such evaluation include the Mean Reciprocal Rank, precision at top M, Normalized Discounted Cumulated Gains at top M, and Recall at top M. Even though *precision at top M* is usually used for evaluating XMLC methods, it is not appropriate in our case. This is due to the fact that the labels are implicit user feedback. A

[4] https://www.kaggle.com/stackoverflow/stacksample/data

Table 1: Dataset Statistics

Dataset	#users	#threads		Avg #word in thread		Avg #user per thread		Sparsity
		train	test	train	test	train	test	
1. Epilepsy	1506	1644	412	147	168	7.39	9.29	99.49%
2. ALS	3182	6466	1617	148	135	9.85	9.75	99.69%
3. Fibromyalgia	5669	8576	2144	203	233	9.02	9.14	99.84%
4. Stackoverflow	69,631	20,137	5035	93	99	6.81	7.29	99.99%

negative instance could imply that the user is actually not interested in the thread but could also imply that the user had not seen it (and could have been interested in).

We use the following three metrics to evaluate the recommendation quality of the competing methods

- **Mean Reciprocal Rank (MRR)** indicates the position of the first relevant user in the ranked list. This measures the ability of a system in identifying an interested user at the top of the ranking. Let r_t be the rank of the highest ranking relevant user for a test thread t. MRR is just the reciprocal rank, averaged over all threads in test set, n:

$$MRR = \frac{1}{n}\sum_{t=1}^{n}\frac{1}{r_t}$$

- **Recall@M** considers how many top-M users actually interacted with the thread (higher is better). Recall for the entire system is computed as the average recall value for all threads in test data.
- **Normalized Discounted Cumulative Gain (NDCG@M)** is well suited for evaluation of recommendation system, as it rewards relevant results ranked higher in the returned list more heavily than those ranked lower. NDCG@M for a thread t is computed as:

$$NDCG_t = Z_t \sum_{j=1}^{M}\frac{2^{r(j)}-1}{log(1+j)}$$

where Z_i is a normalization constant calculated so that a perfect ordering would obtain NDCG of 1; and each $r(j)$ is an integer relevance level (for our case, $r(j) = 1$ and $r(j) = 0$ for relevant and irrelevant recommendations, respectively) of result returned at the rank $j \in \{1, \cdots, k\}$. Then, for each M value, $NDCG_t$ is averaged over all (n) threads in the test set to get the overall NDCG@M.

In our evaluation, we experiment with $M = \{5, 10, 30, 50, 100\}$ to determine the quality of recommendation at different thresholds of the ranked list.

4.3 Baselines

We compare our model with the following competing methods:

CVAE [16] : was proposed to tackle cold start problem using a Bayesian generative model. It reportedly outperforms many state-of-the-art recommendation systems by considering both rating and textual content using deep learning.

CTR [27] : takes LDA [6] discovered topic distributions as input along with the user-item interaction matrix. This has proven to be a very solid baseline for cold start problem and we use it as a representative of traditional recommendation algorithms.

CNN-Kim [12]: constructs a document vector with its constituent word embeddings, and then convolutional filters are applied to this feature maps. The features pass through a max-over-time pooling layer to construct the document representation. For prediction, the document representation is fed to a fully-connected layer with L softmax outputs, corresponding to the L labels.

XML-CNN [18]: introduces some advancements over CNN-Kim. It adopts a dynamic max pooling scheme, a bottleneck layer and a loss function more suitable for multi-label prediction. It has reportedly outperformed many traditional XMLC models over several datasets.

BiGRU-2: is a baseline implemented by us which uses a stack of two Bidirectional GRU layers for text representation. This is essentially equivalent to our model without the CSA component.

4.4 Experimental Settings

Pre-processing for CTR is done as per the recommendations in the paper. We remove all the stopwords and compute tf-idf scores for all the words in all the documents in the training set and retain the top 8000 words to form the vocabulary. Thereafter LDA is run with 100 topics and LDA discovered document-, and word-topic distributions are provided to CTR. For CVAE we used the implementation provided by the authors[5].

For the CNN based models (CNN-Kim and XML-CNN), we used rectified linear units as activation functions, and one-dimensional convolutional filters with window sizes of 2, 4, 8. The number of feature maps for each convolutional filter was 128. For XML-CNN the dropout rate was $p = 0.5$, and hidden units of the bottleneck layer was 512 as suggested by the authors [18].

For the baseline BiGRU-2 and the proposed model, we set the number of neurons for the GRUs to 128, and number of clusters (k) to 100. A dropout layer with 0.3 dropout rate is used after the fully connected layer. To deal with the highly imbalanced class distribution, we use normalized class weights to weigh the sparse positive training examples more. All the deep learning models are implemented using Keras library[6] with Theano[7] as the backend.

[5]https://github.com/eelxpeng/CollaborativeVAE
[6]https://keras.io/
[7]https://github.com/Theano/Theano

4.5 Results

Table 2, 3, and 4 show the performance of different methods on the four datasets in terms of MRR, Recall@M, and NDCG@M respectively.

Firstly, we note that all the XMLC models outperform the widely used off-the-shelf recommendation algorithms comfortably in most cases. However the same does not hold true for the off-the-shelf text classifier as CNN-Kim's scores are not always better. This empirical proof works as a validation of our approach of posing cold start recommendation problem as an XMLC task.

Moreover, we observe that our model outperforms the baselines consistently in all datasets. We achieve a relative performance gain of $4.5\% - 21.7\%$ (depending on the dataset) in terms of MRR compared to current state-of-the-art for XMLC i.e., XML-CNN.

We find the performance of the models to be consistent in terms of both Recall, and nDCG@M. From the NDCG scores we conclude that our model is able to correctly identify interested users and places them near the top of the list in most cases. For $M = 100$, we achieve a relative performance gain of $7.79\% - 16.19\%$ in terms of NDCG compared to XML-CNN. We observe similar trends in case of recall, with a relative performance gain of $3.23\% - 15.39\%$. We would like to mention that in our setting, the recall values at larger M values are equally important as the lower ones – quite unlike the traditional case, where a recommended list of items are presented to every user. Since it is infeasible for a user to look through more than the first $5 - 10$ items, the objective is to have better recall, and NDCG scores for small M (e.g., $5 - 10$). However for a new item, we are trying to identify the set of interested users who would be notified individually. Typically the recommendation engines try to notify as many interested users as possible to ensure sufficient user-engagement. For this reason, we argue that our model would be more appropriate as it consistently achieves higher recall, and NDCG scores for large M values compared to the state-of-the-art for XMLC. Although CVAE uses both rating and textual content, we observe that it struggles to provide accurate recommendation in our scenario. It was reported to outperform other methods when it has seen the test item *at least* once [16]. However in our case, the test item is *never* seen during training – we believe this makes it challenging for CVAE to perform well.

In absolute terms, the performance of all the competing methods degrade drastically in case of the stackoverflow dataset because of extremely high sparsity (99.99%) and huge label space ($\sim 70K$). However relatively speaking, our model fairs well compared to the others with better (in most cases) or very close scores (in few cases) in terms of all the metrics.

Ablation Study: The choice of baselines allows us to do two ablation studies. Firstly, we observe that BiGRU encoding of text works much better compared to XML-CNN which uses CNN to encode the text. We believe that the long sequential nature of posts is better captured with a recurrent network rather than fixed length convolution filters. Finally, recall that the BiGRU-2 is primarily our model without the CSA component. This allows to us to an ablation study between our model variants with/without it. We observe that the attention mechanism achieves a relative performance improvement of upto 6.33% over the BiGRU-2 model in MRR, 3.40% in Recall@100, and 4.67% in NDCG@100 respectively. Moreover, the attention mechanism consistently scores better than BiGRU-2 for larger values of M. This study quantitatively validates the hypothesis of having the CSA component in our model.

5 RELATED WORK

Recommendation System: Collaborative Filtering (CF) based approaches have been the most popular recommendation systems in the past decade. CF based approaches [14, 15, 19, 22, 29] suffer when the data is sparse (i.e. not much interaction history available for a user or item) and do not work for cold start (i.e. no interaction history available for a new item or user).

In order to make recommendations for a new item, in absence of any interaction history, the recommender system needs to make use of additional information such as item content or metadata. Similar to our setting, the authors in [23] tackle the problem of recommending incoming news articles for users to comment. However, they do not use the whole article content but only use the tags associated with a document. Collaborative Topic Regression (CTR) [27] proposes an elegant method of using the textual content for recommendation. It is a probabilistic graphical model that integrates a topic model, latent Dirichlet allocation (LDA) [6] for modeling contents of a document, and uses the LDA-discovered topics while doing the regression later with probabilistic matrix factorization (PMF) [19]. Afterwards in [9] the authors extend CTR by incorporating explicitly mentioned user interests in order to handle cold start recommendation for new users as well.

Some recent works [16, 24, 26, 28–30] have explored deep learning models for recommendation based on item content. In [24] the authors use CNNs to model the acoustic signals present in a music video in order to predict the latent factors to be used by a CF model to make recommendation. Similar to CTR, Collaborative Deep Learning (CDL) has been proposed that uses stacked denoising autoencoders (SDAE) [25] for representation learning of the textual content, and collaborative filtering for the rating matrix. To eliminate the bag-of-words assumption of CDL, Collaborative Recurrent Autoencoder (CRAE) [29] is proposed to model the sequence information in item content. Instead of using a denoising autoencoder, CVAE [16] uses a Bayesian generative approach for the content representation and reportedly outperforms the other methods. Recently, for cold-start recommendation, dropout is applied to input mini-batches, for training Deep Neural Networks to generalize for a missing input [26].

Extreme Multi-label Classification: Embedding based approaches have proved to be popular for handling the extreme multi-label learning problem by reducing the effective number of labels. Generally, they assume that the label matrix is low-rank, and project label vectors into a lower dimensional subspace. Hence, instead of predicting the original high-dimensional label vector for each instance, they reliably train for prediction of embedded label vectors, and then employ a decompression algorithm to map the embedded label vectors back to the original label space. Various compression and decompression techniques have been proposed in the literature to achieve this [3, 5, 8, 10, 11, 33].

Table 2: Comparison of Mean Reciprocal Rank (MRR) of different methods for the four datasets.

Dataset	Methods					
	CVAE	CTR	CNN-Kim	XML-CNN	BiGRU-2	Our Model
1.Epilepsy	0.159	0.443	0.536	0.551	0.631	**0.671**
2.ALS	0.201	0.275	0.270	0.293	0.297	**0.306**
3.Fibromyalgia	0.304	0.435	0.669	0.668	0.740	**0.773**
4.Stackoverflow	0.003	0.032	0.025	0.029	0.047	**0.050**

Table 3: Comparison of Recall@M of different methods across four datasets

Dataset	Metric	Method					
		CVAE	CTR	CNN-Kim	XML-CNN	BiGRU-2	Our Model
1. Epilepsy	recall@5	3.69	17.46	17.23	**22.76**	22.64	22.65
	recall@10	7.22	27.67	22.93	**34.67**	29.22	29.26
	recall@30	21.14	43.83	44.63	49.08	50.99	**51.21**
	recall@50	29.62	50.86	52.45	53.69	59.47	**59.80**
	recall@100	42.44	59.93	65.77	63.67	68.23	**69.37**
2. ALS	recall@5	4.17	7.05	6.19	6.51	7.63	**9.23**
	recall@10	7.07	12.08	10.09	11.44	**14.65**	13.89
	recall@30	17.04	25.00	22.15	23.56	30.18	**31.84**
	recall@50	24.07	32.46	31.27	30.61	36.32	**36.55**
	recall@100	35.77	44.14	43.82	43.14	48.14	**49.78**
3. Fibromyalgia	recall@5	8.24	14.58	23.01	22.11	25.63	**25.97**
	recall@10	14.93	27.18	34.77	33.88	35.18	**37.38**
	recall@30	32.83	54.39	58.04	61.83	62.39	**63.06**
	recall@50	42.43	63.91	67.83	68.92	69.17	**72.04**
	recall@100	55.02	72.31	76.37	75.74	77.98	**78.19**
4. Stackoverflow	recall@5	0.02	0.59	0.46	0.51	0.66	**0.86**
	recall@10	0.06	1.14	0.73	0.97	1.15	**1.30**
	recall@30	0.16	2.73	1.84	2.42	**2.94**	2.80
	recall@50	0.31	4.02	2.74	3.43	4.03	**4.11**
	recall@100	0.69	**6.36**	4.43	5.35	6.09	6.33

Table 4: Comparison of NDCG@M of different methods across four datasets

Dataset	Metric	Method					
		CVAE	CTR	CNN-Kim	XML-CNN	BiGRU-2	Our Model
1. Epilepsy	NDCG@5	3.80	19.44	19.52	25.80	27.80	**29.52**
	NDCG@10	5.95	25.80	24.26	33.08	31.96	**33.72**
	NDCG@30	12.26	33.38	34.10	39.91	41.78	**43.91**
	NDCG@50	15.38	36.02	38.49	41.70	45.14	**47.01**
	NDCG@100	19.50	38.97	42.18	44.88	47.93	**50.17**
2. ALS	NDCG@5	5.28	8.59	8.02	8.21	9.16	**10.24**
	NDCG@10	7.26	11.94	10.62	11.41	**13.71**	13.42
	NDCG@30	12.12	18.18	16.38	17.40	21.05	**22.49**
	NDCG@50	14.90	21.08	19.88	20.13	23.54	**23.86**
	NDCG@100	18.90	25.00	24.14	24.39	27.53	**28.34**
3. Fibromyalgia	NDCG@5	10.29	17.27	28.97	28.57	32.38	**33.71**
	NDCG@10	14.72	25.43	33.67	36.32	38.44	**41.05**
	NDCG@30	23.53	38.82	48.23	50.19	51.09	**54.03**
	NDCG@50	27.29	42.50	52.04	52.98	54.53	**57.36**
	NDCG@100	31.46	45.36	54.95	55.32	57.52	**59.63**
4. Stackoverflow	NDCG@5	0.02	0.64	0.54	0.59	1.01	**1.22**
	NDCG@10	0.04	0.98	0.70	0.87	1.31	**1.48**
	NDCG@30	0.09	1.68	1.19	1.52	2.09	**2.12**
	NDCG@50	0.14	2.14	1.51	1.88	2.47	**2.59**
	NDCG@100	0.26	2.86	2.03	2.46	3.11	**3.27**

In order to avoid the loss of information during the compression phase of the embedding based approaches, tree-based methods have been proposed that try to partition the label space similar to a decision tree. It recursively partitions the huge label space in subtrees until only a few labels are left at each leaf node. A base classifier at each leaf node then focuses on only the active labels in the node. The LPSR [31] method focuses on learning a hierarchy over a base classifier or ranker starting with a base multi-label classifier for the entire label set - this becomes computationally expensive to train if a discriminative classifier (e.g. SVM) is used. Instead of using a base classifier, MLRF [1] uses an ensemble of randomized trees with a modified Gini index for partitioning the nodes. In FastXML [21] an NDCG-based objective is used at each node of the hierarchy for optimization.

Despite the success of deep learning in many fields, it has not been explored much for XMLC tasks. Recently, a CNN based approach (XML-CNN [18]) has been proposed, which uses convolutional layers for text representation and a feed forward layer acting as a bottleneck layer for scalability. This has been shown to outperform both embedding based and tree based approaches for XMLC, therefore we chose this method for comparison in our experiments.

6 CONCLUSION

We have addressed cold start thread recommendation in online forums, which is an important task to ensure user engagement. We recommend newly-posted threads to interested users in the community for participation. Mainstream recommendation systems cannot use collaborative filtering to address this phenomenon as there is no interaction history for such items.

We have applied an alternative approach utilizing extreme multi-label classification. In particular, we proposed a novel neural network architecture consisting of stacked bi-directional GRUs for text encoding, coupled with cluster-sensitive attention to address scalability, and sparsity.

Specifically, leveraging our insight that sets of users display different levels of interest within a long post text, the cluster-sensitive attention incorporates user interests by learning multiple attention layers for attending to different parts of a text. This cluster-sensitive attention layer also helps us in addressing the sparsity issues usually associated with extreme multi-label classification approaches, by exploiting the correlation between users within clusters. Thorough experimental evaluation show that the proposed model outperforms existing content based recommendation systems, deep learning based text classification systems, as well as state-of-the-art multi-label classification approaches.

In the future, we plan to model how interests of community members change over time. Not all users will remain interested in the same topic over a long course of time as their experiences and expertise change. Also, we encourage the research community to try our approach in other domains where recommending new items to interested users is the priority such as news articles, tweet recommendation, social media news feed generation and so on.

REFERENCES

[1] Rahul Agrawal, Archit Gupta, Yashoteja Prabhu, and Manik Varma. 2013. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *Proc. of WWW*. ACM, 13–24.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[3] Krishnakumar Balasubramanian and Guy Lebanon. 2012. The landmark selection method for multiple output prediction. In *Proc. of ICML*. Omnipress, 283–290.

[4] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks* 5, 2 (1994), 157–166.

[5] Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. 2015. Sparse local embeddings for extreme multi-label classification. In *Proc. of NIPS*. 730–738.

[6] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3, Jan (2003), 993–1022.

[7] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *Proc. of NIPS Deep Learning and Representation Learning Workshop* (2014).

[8] Moustapha M Cisse, Nicolas Usunier, Thierry Artieres, and Patrick Gallinari. 2013. Robust bloom filters for large multilabel classification tasks. In *Proc. of NIPS*. 1851–1859.

[9] Kishaloy Halder, Min-Yen Kan, and Kazunari Sugiyama. 2017. Health Forum Thread Recommendation Using an Interest Aware Topic Model. In *Proc. of CIKM*. ACM, 1589–1598.

[10] Daniel J Hsu, Sham M Kakade, John Langford, and Tong Zhang. 2009. Multi-label prediction via compressed sensing. In *Proc. of NIPS*. 772–780.

[11] Ashish Kapoor, Raajay Viswanathan, and Prateek Jain. 2012. Multilabel classification using bayesian compressed sensing. In *Proc. of NIPS*. 2645–2653.

[12] Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proc. of EMNLP*.

[13] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014).

[14] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proc. of SIGKDD*. ACM, 426–434.

[15] Daniel D Lee and H Sebastian Seung. 2001. Algorithms for non-negative matrix factorization. In *Proc. of NIPS*. 556–562.

[16] Xiaopeng Li and James She. 2017. Collaborative variational autoencoder for recommender systems. In *Proc. of SIGKDD*. ACM, 305–314.

[17] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *ICLR* (2017).

[18] Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 2017. Deep Learning for Extreme Multi-label Text Classification. In *Proc. of SIGIR*. ACM, 115–124.

[19] Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic matrix factorization. In *Proc. of NIPS*. 1257–1264.

[20] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation.. In *Proc. of EMNLP*.

[21] Yashoteja Prabhu and Manik Varma. 2014. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proc. of SIGKDD*. ACM, 263–272.

[22] Ruslan Salakhutdinov and Andriy Mnih. 2008. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proc. of ICML*. 880–887.

[23] Erez Shmueli, Amit Kagian, Yehuda Koren, and Ronny Lempel. 2012. Care to Comment?: Recommendations for Commenting on News Stories. In *Proc. of WWW*. ACM, 429–438.

[24] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *Proc. of NIPS*. 2643–2651.

[25] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research* 11, Dec (2010), 3371–3408.

[26] Maksims Volkovs, Guangwei Yu, and Tomi Poutanen. 2017. DropoutNet: Addressing Cold Start in Recommender Systems. In *Proc. of NIPS*. 4964–4973.

[27] Chong Wang and David M Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *Proc of SIGKDD*. ACM, 448–456.

[28] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative Deep Learning for Recommender Systems. In *Proc. of SIGKDD*. ACM, 1235–1244.

[29] Hao Wang, SHI Xingjian, and Dit-Yan Yeung. 2016. Collaborative recurrent autoencoder: recommend while learning to fill in the blanks. In *Proc. of NIPS*. 415–423.

[30] Xinxi Wang and Ye Wang. 2014. Improving Content-based and Hybrid Music Recommendation Using Deep Learning. In *Proceedings of the 22Nd ACM International Conference on Multimedia*. 627–636.

[31] Jason Weston, Ameesh Makadia, and Hector Yee. 2013. Label partitioning for sublinear ranking. In *Proc. of ICML*. 181–189.

[32] Bin Xu, Jiajun Bu, Chun Chen, and Deng Cai. 2012. An Exploration of Improving Collaborative Recommender Systems via User-item Subgroups. In *Proc. of WWW*. ACM, 21–30.

[33] Yi Zhang and Jeff Schneider. 2011. Multi-label output codes using canonical correlation analysis. In *Proc. of AISTATS*. 873–882.