System Design of Extreme Multi-label Query Classification using a Hybrid Model

Xianjing Liu*
eBay Search
xianjliu@ebay.com

Hejia Zhang†*
Princeton University
hejiaz@princeton.edu

Mingkuan Liu
eBay Search
mingkuan@gmail.com

Alan Lu
eBay Search
alalu@ebay.com

ABSTRACT

Extreme multi-label classification is a rapidly growing research area with many applications. In this paper we propose a system design of extreme multi-label text classification (XMTC) on query classification in the e-commerce domain. Search query classification is more challenging than conventional document classification because queries are usually very short and often ambiguous. We design a hybrid model that uses a deep neural network for long queries and uses a Naive Bayes model for short queries. We formulate and apply new data augmentation techniques and create new evaluation metrics that are more suitable for the extreme multi-label task in e-commerce. We also design end-to-end system level evaluation methods to address the challenge in human judgment due to the extremely large label space. We compare our deep neural network model with the state-of-the-art method on our real e-commerce data and observe about a 15% improvement in the F1 score. The end-to-end system evaluation results show that our new system improves query classification performance for a variety of query sets. In particular, for the torso and tail queries in e-commerce, we see 0.3% and 1.1% improvements in the NDCG score.

KEYWORDS

multi-label, extreme text classification, query classification, neural networks, CNN-RNN, deep learning, e-commerce

ACM Reference Format:
Xianjing Liu, Hejia Zhang, Mingkuan Liu, and Alan Lu. 2019. System Design of Extreme Multi-label Query Classification using a Hybrid Model. In *Proceedings of the SIGIR 2019 Workshop on eCommerce (SIGIR 2019 eCom)*, 8 pages.

1 INTRODUCTION

In e-commerce, there are many cases where we need to classify text to a large label space [1, 2, 3], such as classifying search queries, product listing titles, or product listing descriptions to merchandise categories. A typical merchandise category taxonomy is a multi-leveled tree. Among these classification tasks, query categorization is much more challenging than traditional document classification tasks. First, queries are usually very short. The average length of queries by eBay users is about four tokens. Second, many queries are ambiguous. For example, the query 'Nike' can be classified to many categories such as *men's shoes*, *men's clothing*, *women's shoes*, *kids' clothing*, *Golf*, and *Yoga*. 'Father's day gift' can be categorized to *home and living*, *electronic*, *art*, *clothing*. The relevant search results for such queries can span many leaf categories from multiple meta categories (categories at the root level). Third, the label set is very large when classifying the query to the leaf level of categories. There are, for example, over 20k leaf categories at the eBay U.S. site alone.

It is natural to model query categorization as a multi-class classification problem. The traditional binary or multi-class classification problems, where one and only one label belongs to each document, have been studied heavily in the literature [4, 5, 6]. However, we observe that, as mentioned above, many queries have more than one ground-truth label, so we consider query categorization as a multi-label problem. Multi-label classification is fundamentally different from binary and multi-class classifications. A multi-label classifier assigns the most relevant subset of labels to each sample while the label set in a multi-class classifier is treated as independent variables, and the dependencies among labels are not leveraged. That is, a multi-class classifier assumes that the class labels are mutually exclusive [7].

The query categorization in our task is also an extreme classification problem since the label set contains a tremendous number of labels. Extreme classification is a rapidly growing research area dealing with multi-class and multi-label problems with a very large set of labels [8, 9, 10, 11, 12]. Combining the properties of our query categorization, we treat it as an extreme multi-label (XML) classification problem.

In this paper, we propose a system design to tackle query categorization as an XML problem. We design a hybrid model where we combine deep neural network (DNN) model with a Naive Bayes model. In particular, the model handles short queries using a Naive Bayes model and handles long queries with DNN. The motivation for using a hybrid model is based on our experiment results, detailed in Section 4.3 Table4, where we found Naive Bayes model has better performance in the short queries, while the DNN model

*Both authors contributed equally to the paper, in alphabetical order of last names.
†This work is done when Hejia Zhang is intern at eBay.

In: J. Degenhardt, S. Kallumadi, U. Porwal, A. Trotman (eds.):
Proceedings of the SIGIR 2019 eCom workshop, July 2019, Paris, France, published at http://ceur-ws.org

outperforms the Naive Bayes model in long queries. We also apply new data augmentation techniques and create new evaluation metrics for XML problems that are more suitable in the e-commerce setting. We compare our system with the state-of-the-art method on real data in e-commerce and show that our system enhances the performance of query categorization in a few success measures.

2 RELATED WORK

2.1 Naive Bayes text classification model

Naive Bayes classifiers are used widely in text classification tasks. These classifiers belong to the probabilistic classification family. They are based on the Bayes theorem and assume that the features are mutually independent. Despite its oversimplified assumption on feature independence, Naive Bayes achieves competitive performance in many complex applications [13, 14, 15]. The paper in [16] extended the Naive Bayes classifier to utilize the structural characteristics of e-catalogs for e-commerce and achieved improved accuracy. The study of [17] proposed a semantic Naive Bayes classifier that incorporates the semantic feature of the document to improve the conventional Naive Bayes classifier. Because of the simplicity, effectiveness and excellent performance, Naive Bayes classifier has been widely used in the industry, including the e-commerce domain. Therefore, we use Naive Bayes classifier as one of the baselines, although it is not specifically designed for the XML classification. Naive Bayes performs very well with a small amount of training data [18] that most other classifiers would find insufficient, especially deep neural network. However, the deep neural network usually outperforms the Naive Bayes where a large amount of training data has been provided like in e-commerce.

2.2 Models for XML

2.2.1 Deep neural network models for XML. There are many existing methods for the XML text classification problem. Among those methods, the neural network models, such as FastText [19], CNN-Kim [20], Bow-CNN [21], and XML-CNN [7], constitute a big family. These methods design different neural network structures and directly map input text into the label space. XML-CNN is the state-of-the-art (SOTA) in the XML text classification task as shown in the paper [7]. It passes the document through convolutional filters and dynamic max-pooling layers to extract features and maps those features to the label space with two fully-connected layers. Despite the large amount of DNN methods in the XML text classification field, there are few DNN models designed for the XML query categorization problem. Since queries are much shorter than the samples of text datasets used in most of those DNN methods, it is not desirable to apply those methods directly on query datasets.

2.2.2 Other models for XML. There are also some other methods for the XML text classification problem. They can be roughly divided into two groups. The first group is target-embedding methods, such as SLEEC [22]. These methods project the label vectors to a low-dimensional space to deal with the sparsity issue of the label space. The second group is tree-based methods, such as FastXML [23]. These methods have a hierarchical structure similar to decision trees. The XML-CNN paper [7] also compares these methods to XML-CNN and shows that XML-CNN is the SOTA, so we will only compare our model with XML-CNN in our paper.

3 PROPOSED SYSTEM

Here we propose a system design for the extreme multi-label query classification with a hybrid model. The hybrid model is a combination of a DNN model and a Naive Bayes model where the Naive Bayes classifier is called when the query length is 1 (token), and the DNN model is used otherwise. We experimented with various cutoff query length thresholds, and a cutoff length of 1 gives the best overall performance. Our DNN model for queries, namely XML-Q, is a CNN-RNN model with differentiable F1 loss. It is an extension of the model XML_CNN [7] that is more suitable for the query classification.

We also propose three techniques for training data augmentation to address the skewness of the data to improve model performance. The standard definition of evaluation metric precision at k is flawed for the multi-label task when the total number of true label for the query is less than k, more details in Section 4.2.1. We therefore design a new evaluation metric PR@k for the multi-label classifier in Section 4.2.1.

The challenge of using human judgment to evaluate the XML classifiers directly is that the label space is too large for the judge to categorize the query to the entire label space according to the judgment guidelines. Thus, we propose an end-to-end evaluation in the system/application level. We implement the query classifier in the search system and design an offline evaluation pipeline using golden data and the side-by-side human judgment of search performance in 4.2.2 and Section 4.2.3. The end-to-end system performance such as search relevance and ranking quality is evaluated instead.

3.1 Hybrid Model

We propose a hybrid model where the Naive Bayes model is used when the query length is 1, and the XML-Q model is called when query length is larger than 1. The design is based on the comparison of these two classifiers in the evaluation results shown in Table 4. The Naive Bayes model has better performance for the head queries where the query length is mostly 1-2, while the XML-Q classifier is better for the torso and tail queries where query lengths are usually longer than the head queries. One possible explanation is that Naive Bayes is based on the joint probability of n-gram tokens. The probability would vanish as the length of query increases. On the other hand, the XML-Q model has higher capacity to learn the fine-grained patterns in long queries. Also, the large number of noisy training samples makes the XML-Q model robust by reducing the variance of the XML-Q model. An additional explanation is discussed below.

The current setting of cutoff query length for the hybrid model is 1. Other choices of cutoff length such as 2,3 and 4 have also been evaluated in our study, but the cutoff length of 1 gives the best performance. Note that cutoff length of 1 means the function of Naive Bayes classifier is similar to a look-up table, which gives a good performance when plenty of user click data is accumulated, and a high-quality look-up table is generated. Meanwhile, the performance of XML-Q is not ideal because of the limited input

information when the query length is 1. This analysis could also be a reason why Naive Bayes outperforms XML-Q for head queries with a length of 1.

3.2 Naive Bayes model

Our Naive Bayes model is a production model that is generated using user clicked data. It is one of the baseline models in our study. For each query, the clicked product listings are logged, and the related categories for the listings are aggregated and grouped. Then a table that maps queries to categories and click counts is generated. The query has been further broken down to uni-gram and bi-gram tokens to generate a table that contains tokens, categories, and click counts. The noise in the table is reduced by removing rows where the click counts are below a threshold where the threshold is a hyperparemeter we tuned. The bag of words and multinomial Naive Bayes methods are applied in the model. The probability of the query given the category can be calculated as the production from the likelihoods of bi-grams or uni-grams in the query.

3.3 Deep Neural Network Model

Our DNN model for long queries is called XML-Q (XML for queries). It is based on the XML-CNN [7], which is designed for XML classification of the documents. In XML-Q, we made modifications to the XML-CNN model architecture and loss function due to the differences between query datasets and the document dataset used in [7].

3.3.1 Architecture. Our queries, with an average length of four tokens, as shown in Table 1, are much shorter compared with the document dataset which has an average length of hundreds of tokens. our queries contain far less information than document and need a richer feature extractor. While the XML-CNN uses only a convolutional layer to extract features from the documents, XML-Q has an extra recurrent units layer after the convolutional layer to extract more sequential information. This recurrent layer also replaces the functionality of the pooling layer in XML-CNN. In our CNN+RNN structure, the convolutional layer extracts n-gram information and the recurrent layer further extracts sequential information, thus it is a stronger feature extractor compared with CNN or RNN alone. This observation is consistent with our experiment results that the CNN+RNN structure has a better prediction performance.

The architecture of XML-Q is summarized in Figure 1. All of the hyper-parameters in the model architecture are tuned with a validation set. The input query is truncated or padded to 10 tokens where the last token is fixed to be <EOS>. Based on Table 1, 90% of our queries have length less than 6, so we are not losing much information in truncation. If the query is shorter than 10 tokens, <PAD> tokens are padded to the beginning of the query to comply with the preference of recurrent units. The tokens are then mapped to learn-able length-256 word embeddings. The embeddings are passed through a convolutional layer which is arranged similar to XML-CNN [7]. The convolutional layer consists of three groups of convolutional filters. The three groups have 512, 1024, 512 filters with filter size 1, 2, 3, respectively. A filter with filter size k actually has size $256 \times k$, meaning that the filter is convolved with word embedding of k tokens. The model has more filters with filter size 2, which implies the bi-gram information is more important in the query classification. The output from the convolutional layer then passes through a batch-normalization layer [24] and a ReLU layer and is fed into a layer with gated recurrent units (GRU) [25]. The GRU layer also consists of three groups of units with 512, 1024, 512 units, taking in the outputs from the three groups of filters from the last layer respectively. The output from the GRU layer then passes through a layer-normalization layer [26] and is concatenated. The concatenated length-2048 representation is then fed into two fully-connected layers where the number of units in the hidden layer is 80% of the number of units in the concatenated layer. The output from the last fully-connected layer is passed through a sigmoid function, and the output length is equals to the number of total categories.

3.3.2 Loss function. Another difference between the query dataset and document dataset in [7] is that the number of categories, or labels, per query is 1.5, which is much smaller than the number of labels per document. The total number of categories of queries is about $20,000$, which is similar to, or greater than the number of categories in document datasets. Therefore, each query, or sample, has a much sparser output space. This sparsity leads to an imbalanced classes issue. That is, the output tends to be all close to zero if no action is taken to alleviate this issue because the vast majority of the ground-truth labels are zero.

To solve this problem, instead of using the cross-entropy loss as in [7], we design a new soft F1 loss. According to the definition in [27], the F1 score in multi-label classification is calculated for each sample and is defined as the harmonic mean of precision and recall. The F1 score is a common metric for classification problem with imbalanced classes. It is desirable to optimize the F1 score directly in our case. However, the F1 score is not differentiable, thus it is hard to use it as a loss function in the DNN. Since the harmonic averaging operation itself is differentiable, we examine the precision and recall formula to identify the non-differentiable part.

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{TP}{\text{Number of positive labels in prediction}}$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{TP}{\text{Number of positive labels in ground-truth}}$$

, where TP is true positive, FP is false positive, and FN is false negative. We observe that thresholds are applied in deriving the predicted labels and TP, and the non-differentiability comes from this thresholding step. Therefore, we remove the thresholding and estimate TP and number of positive labels in prediction with a soft, differentiable formula defined as:

$$\text{soft TP} = \sum_{l=1}^{L} t_l \cdot \sigma(y_l)$$

$$\text{soft num. of positive predictions} = \|\sigma(y)\|_2^2$$

, where L is thenumber of categories, t is the ground-truth vector, y is the logits vector, σ is the sigmoid function, and subscript l means the l'th category. By replacing the TP and number of positive predictions with the soft version, the F1 score is differentiable, and we call it "soft F1 score". The soft F1 score is used as the cost function in XML-Q.

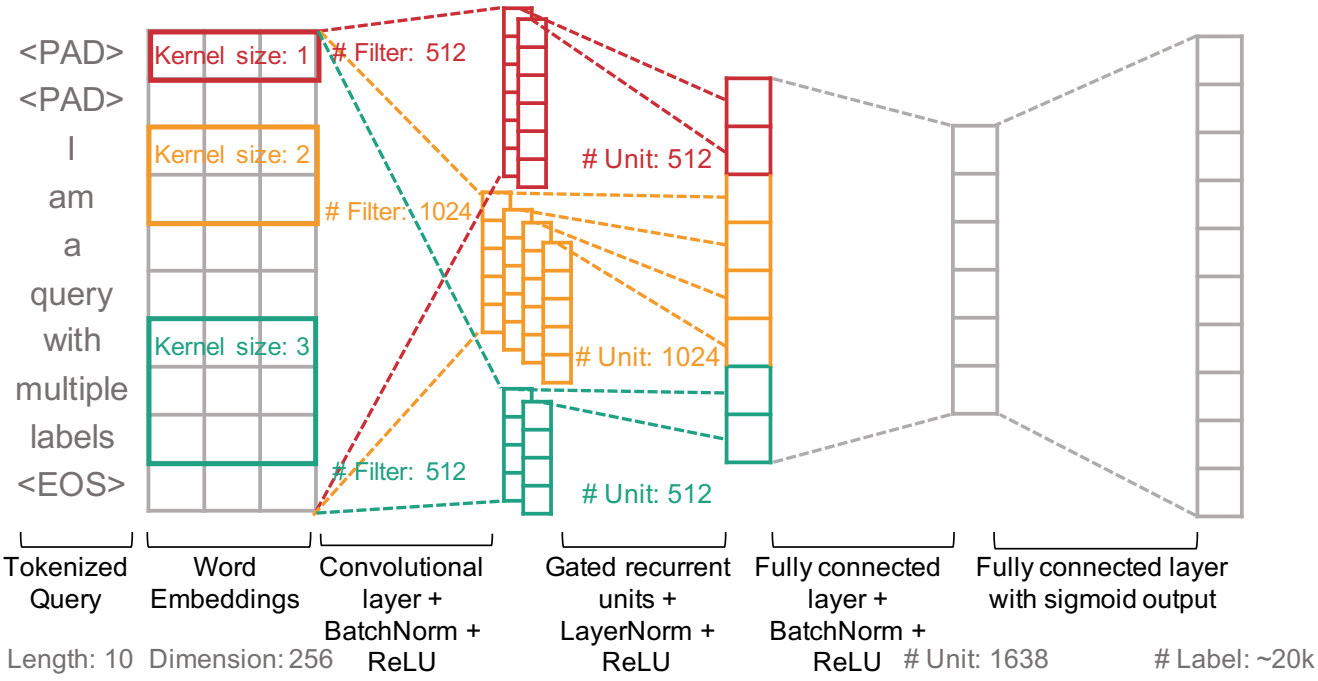

Figure 1: Architecture of deep neural network model XML-Q for long queries

3.3.3 dropout and learning rate decay. During the training phase, we applied two techniques to prevent over-fitting and accelerate the convergence. First, we use dropout [28] with a keep-rate of 0.8 in the fully-connected layers to prevent over-fitting. The second technique is the application of cosine decay with restarts to the learning rate as proposed in [29]. The learning rate starts from 0.001 and follows a cosine decay function with a period of 4 epochs. When the learning rate touches zero in 2 epochs, it restarts to 0.0008, and the cosine decay function has a period of 8 epochs. Every time it restarts, the learning rate is 0.8 of the previous restart, and the cosine function period is twice as long as the previous one. The learning rate scheduler is shown in Figure 2.

Figure 2: Learning rate scheduler of the DNN model

3.4 Data augmentation

The number of available training samples in e-commerce is usually very large, but we found that it is still helpful to augment the dataset to balance the data distribution in query classification. We collect the latest half-year data as the base dataset and collect another half-year data before the base dataset as the augmentation candidate set. The final training set consists of the full base set and part of the candidate set we selected. The final training set leads to a better performance in our experiment compared with the base set alone or base set plus full candidate set. We applied three augmentation techniques.

The first technique is used to alleviate the skewness of labels, or categories, in query classification. Some popular categories have lots of data while some others have far less. For example, in our dataset, the maximum number of queries for a category is 364771 while the minimum number is 1, and the average number of queries for each category is 1377, as shown in Table 1. We, therefore, added data from the 1500 categories with the lowest frequencies in the candidate set to the final set.

	mean	90% tile
Query length	3.9	6.0
#labels for each query	1.5	2.0
#search-requests for each query	1377.1	2420.0

Table 1: query set statistics

The second technique is to augment selected queries that contain brand names, such as Dior, Haier, or Nike. When such queries are short and ambiguous, the predicted categories are sometimes dominated by the information in the brand name. For example, the ground-truth category of the query "Dior Gaudron" is "pottery &

glass" where "Gaudron" is a series of products of Dior for dinnerware. However, since the majority of queries with brand "Dior" are clothes or cosmetics and "Gaudron" contains only very ambiguous information to the model, the model tends to classify "Dior Gaudron" into clothes and cosmetics. Therefore, for queries with brand names, if the category has low frequencies within the brand, repeat the queries 50 times and add to the final set. We repeat the queries 50 times as a result of tuning with the validation set. More specifically, for each brand, we collect all queries with this brand name and count the number of queries for each category, and then we sort the categories according to the query frequencies, and take half of the categories with the lowest query frequency. The queries in those categories with that specific brand are then selected from the candidate set and added to the final set.

The third technique is used to catch the residuals of our model. We first train XML-Q with the base set and record the misclassification rate of each category on the base training set. We then add the samples of 1000 categories with worst performance and which have not yet added to the final set.

4 EXPERIMENT

4.1 Dataset collection and preprocessing

In the e-commerce domain, there are millions of buyer engagement data (click/add2cart/purchase) generated in search log systems every day. Our data set consists of 275 million <query, category> pairs and the click-through rate (CTR) for each pair. The query is normalized by removing certain special characters. For each query, the clicked product listings and the click/impression count are aggregated for one year and grouped by the category. We eliminated those <query, category> pairs where aggregated click counts are less than 3 to reduce the noise in the data while retaining as many categories as possible. In our dataset, the number of total categories is about 20,000, covering the entire label space.

4.2 Evaluation

The evaluation in our study includes a model level evaluation and a system level evaluation. We propose a system level evaluation because the number of total categories is too large for human judges to evaluate the quality of the query classifier directly. In our study, the query categorization is a component in the e-commerce search system. We implement the baseline or the target query models in the search system and compare whether there is any improvement for the search system with the target model and how much improvement it brings.

We design two approaches for the system level evaluations. The first approach is the offline golden data evaluation, and the second approach is the human judgment of end-to-end search relevance.

4.2.1 model level evaluation. The evaluation of a multi-label classifier is more challenging than the evaluation of a multi-class classifier. In the multi-label setting, both the ground-truth and predicted labels for a testing sample could be a subset of the label set. Hence, the prediction can be entirely correct, partially correct or entirely incorrect. There are three ways to evaluate a multi-label classifier. First, one could evaluate partitions, which measures how far the classifier predictions are from the ground-truth labels; second, one can evaluate ranking, which evaluates if the labels are ranked in order of human judged relevance; third, one can use the label hierarchy, which evaluates the effectiveness of the system to take into account the hierarchical structure of the labels [30, 31].

In our study, all the categories are at the leaf level, and the hierarchical structure of the labels can be ignored, so we evaluated both the partitions and the rankings. The partitions measure the partial correctness. We used the definition of precision, recall, and F1 proposed by [27]. The precision is the ratio of the correctly predicted true labels to the total number of true labels, averaged over all samples. The recall is the ratio of correctly predicted true labels to the total number of predicted true labels, averaged over all samples. F1 is the harmonic mean of precision and recall.

$$\text{Precision} = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \bigcap Z_i|}{|Z_i|}$$

$$\text{Recall} = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \bigcap Z_i|}{|Y_i|}$$

$$\text{F1} = \frac{1}{n}\sum_{i=1}^{n}\frac{2\times|Y_i \bigcap Z_i|}{|Y_i| + |Z_i|}$$

, where n is the number of instances in evaluation set. Y_i are the true labels for the instance x_i, and Z_i are the predicted labels for the instance x_i. Both Y_i and Z_i $\in \{0,1\}$

In the e-commerce domain, the order of the predicted categories for the query matters. The more relevant categories with higher CTR should rank higher than the less relevant categories with lower CTR and the categories that are not relevant. Rank-based evaluation metrics, such as precision at top k (P@k), have been widely used in the multi-label tasks [7, 23, 10, 22]. P@k is calculated for each query and then averaged over the whole evaluation set. For each query, the P@k is defined as:

$$\text{p@k} = \frac{1}{k}\sum y_l$$

, where k could be 1,3,10 and y_l is the ground-truth labels among the top k predicted labels for each query.

The standard definition of P@k is flawed when the total number of ground-truth labels is less than k. For example, assume the query has two true labels only, and the query classifier predicts both of them right. Then the expected precision at top k should be 1. However, according to the definition, the P@3 will be 2/3. Furthermore, based on the definition, P@k is not a comparable metric between datasets when the average number of labels of queries various a lot.

We propose a modified metric called PR@k as:

$$\text{PR@k} = \frac{1}{min(k, tl)}\sum y_l$$

, where tl is the number of true labels in the top k. In the definition of PR@k, the sum of true positives is divided by the minimum of k and the number of true labels tl, instead of k, to address the problem

when k is larger than tl. In this new definition, the precision at k and the recall at k will be the same, so we name it as PR@k.

4.2.2 offline golden data evaluation pipeline. Human judgment is time- and money-consuming. Thus, we built the offline golden data evaluation pipeline and evaluated the system performance and the search relevance with the pipeline before the human judgment. The pipeline utilizes human judgment results accumulated from the past as the golden dataset.

The offline golden data evaluation pipeline can be treated as a mini search system. The inventory is a mixture of all product listings from the golden dataset and 10% of product listings from the production inventory. The golden data contains 500k human judgment data in the format of query, the top 3 returned product listings, and the human labeled relevance score for each <query, item> pair. The relevance score is either 0 or 1, where 0 means not relevant, and 1 means relevant. Based on the user search frequency, the queries can be divided into head, torso and tail queries. We define that head queries are those count for the top 30% of the total impressions, and torso queries are those with the top 30-60% impressions, and tail queries are those with the rest 40% impressions. The performance of a search engine is usually quite different in head, torso and tail queries. Most of the search engines perform very well in the head query. The performance in the tail query set distinguishes the good search engines from the others. To better evaluate the search system, we generated head, torso, and tail queries based on the demand frequency from the latest search log joined with the queries in the golden data. We sampled 10k queries from each intersection as the final evaluation set for head, torso and tail queries. For each query in the evaluation set, the top N product listings can be predicted with the golden data evaluation pipeline, and the precision, recall, accuracy, and AUC [32, 33] can be calculated by comparing the system prediction to the golden data label.

4.2.3 search relevance human judgment. If the result of the target model is better than the baseline from the offline golden data evaluation pipeline, the next step is to conduct human judgment on end-to-end search relevance. The search system is set up with baseline or control models and with the full inventory. The evaluation query set includes 4500 queries, 1/3 for each of the head, torso, and tail query segments. The judges are asked to judge the top 5 product listings for each query to four levels of relevance based on the judgment guideline: *Excellent*, *Good*, *Acceptable* and *off-topic*.

Three judgments were collected for each query to reduce bias. Meanwhile, we designed a side by side group comparison. Two groups of results were shown on the one page for each judgment, one group from the baseline, and another group from the target.

NDCG is a score defined to measure the search ranking. We modified the definition of NDCG to represent both the ranking quality and relevance.

$$\text{NDCG}^* = \frac{DCG}{IDCG}$$

$$\text{DCG} = \sum_{i=1}^{N} \frac{2^{rel_i}}{log_2(i+1)}$$

$$\text{IDCG} = \sum_{i=1}^{N} \frac{2^{prefect_i}}{log_2(i+1)}$$

where *N=min(5,#totalReturns)*, the NDCG* is calculated based on the top up to 5 returned product listings for each query. *rel_i* is the relevance score of the product listing at position i by a human judge. We modified the definition of *IDCG* such that all the top 5 product listings should be *Excellent* for a perfect search system, so `prefect_i` in the *IDCG* equation is *Excellent* score. The NDCG score is calculated for each query and averaged over all the evaluation set.

4.3 Results

The XML-CNN model [7] has achieved the-state-of-the-art performance and beats the other 7 most representative multi-label models such as FastXML [23], SLEEC [22], Bow-CNN [21], and FastText [19] on 6 benchmark datasets. Thus, among all the multi-label models, we will compare our model to XML-CNN only. Another baseline is the Naive Bayes model which was the production model in our system. In the experiments, we evaluated: 1) the performance of our deep learning model XML-Q compared to the XML-CNN model; 2) the comparison of the Naive Bayes model, XML-Q model, and hybrid model in the offline golden data evaluation pipeline; 3) the comparison of the hybrid model and Naive Bayes model in search relevance human judgments.

Our XML-Q model is an extension of XML-CNN by adding a few components that are more suitable for e-commerce query classification. These components include differentiable F1 loss, additional RNN layer, and cosine learning rate decay. As shown in Table 2, by replacing the original binary cross-entropy loss with the differentiable F1 loss, we see 22.95% improvement in the F1 score, 41.43% improvement in the precision, 17.43% drop in the recall, and 6.74%, 2.80%, and -0.22% changes in the PR@1, 3, and 10, respectively. The contributions of adding additional RNN layer and using cosine learning rate decay to the F1 score are 0.69% and 2.58%, and contribution to the PR@10 is 1.38% and 1.54% respectively.

	δ**P**	δ**R**	δ**F1**	δ**PR1**	δ**PR3**	δ**PR10**
F1 Loss	41.43	-17.43	22.95	6.74	2.80	-0.22
Add RNN	1.51	0.03	0.69	1.24	1.22	1.38
Cos Decay	2.39	2.72	2.58	2.15	1.86	1.54

Table 2: The contributions of new components in XML-Q to the model performance in precision, recall, F1, PR@k (k=1,3,10) (in %).

Since differentiable F1 loss has a big impact on the performance of query categorization, for easy comparison, we show the performance of XML-CNN with differentiable F1 loss (XML-CNN*) and

XML-Q model in Table 3. From the XML-CNN* model to XML-Q model, the total improvement is about 11.2% in precision, 18.6% in recall, 15.0% in F1 score, and 11.8%, 10.5%, 8.9% in PR@1, 3, 10, respectively. Note that data augmentation also contributes to total improvement.

Model	P	R	F1	PR1	PR3	PR10
XML-CNN*	61.01	55.09	57.90	64.79	71.28	78.89
XML-Q	72.16	73.68	72.91	76.57	81.75	87.75

Table 3: Comparison of XML-CNN* and XML-Q in precision, recall, F1, PR@k (k=1,3,10) (in %).

Table 4 shows the comparison of XML-Q and the Naive Bayes model in the top and the comparison of the hybrid model and the Naive Bayes model in the bottom using the offline golden data evaluation pipeline. The top of the table illustrates that XML-Q model has a better performance in both torso and tail query sets and the Naive Bayes model outperforms the XML-Q model in the head query set. This mixed result is one of the reasons why we propose the hybrid model that combines the Naive Bayes model and XML-Q. It is not easy to classify whether a query is the head query or not. We could build a dictionary that contains all the head queries based on user behavior date and maintain it. Alternatively, to our observation, most of the head queries are shorter than torso and tail queries, so it is also straightforward to build a hybrid model based on the query length. We tested different cutoff query lengths from 1 to 4, and the hybrid model with a cutoff length 1 gives the best performance. As shown in the bottom of the table, the hybrid model outperforms the Naive Bayes model among all head, torso, and tail query sets.

	QSet	δ Acc	δ P	δR	δF1	δAUC
XML_Q-NBayes	head	-0.3	-0.2	-0.8	-0.8	-0.1
	torso	0.2	0.2	0.5	0.5	0.1
	tail	0.4	0.0	1.7	1.5	0.1
Hybrid-NBayes	head	0.3	0.3	1.1	1.0	0.2
	torso	0.5	0.0	1.9	1.6	0.1
	tail	0.8	0.3	2.6	2.2	0.3

Table 4: Comparison of XML_Q and Naive Bayes model (top) and Hybrid model and Naive Bayes model (bottom) in accuracy, precision, recall, F1 and AUC (in %) for head, torso and tail query set using Offline golden data evaluation pipeline.

The search relevance human judgment result in Table 5 shows that from Naive Bayes model to the hybrid model, the NDCG* score improves by 0.3% for torso queries and 1.1% for tail queries. The improvement of the head queries is neutral. One reason is that a large portion of the head queries is of length 1, so the hybrid and Naive Bayes are the same model in this case. Another reason is that the head queries are relatively easy to classify based on the human judgment results, so the predictions of both the Naive Bayes model and XML-Q are correct. The average performance improvement of NDCG* for the full query sets is 0.4%.

QuerySet	NDCG improvement
head	0.0
torso	0.3
tail	1.1
mix	0.4

Table 5: Comparison of hybrid model and Naive Bayes model in NDCG* (in %) for head, torso and tail query sets from search relevance human judgment results.

5 CONCLUSION

This paper presents the system design of extreme multi-label query classification as an application in the e-commerce search system. We propose a hybrid model that combines the Naive Bayes classifier and a deep neural network model, XML-Q, for the query classification based on the length of queries. The Naive Bayes model is used for the queries with one word, and the deep neural network model is used for the rest of the queries. The deep learning model is an extension of the state-of-the-art XML-CNN model. A few components and adjustments have been made to make the model more suitable for query classification, such as the differentiable F1 loss, additional RNN layers, and the cosine learning rate decay. Three new data augmentation techniques have been applied to the training data which significantly improve the model performance. New evaluation metric PR@k is designed to address the problem in the standard precision@k metrics when k is larger than the total true labels of the query. Since the label set is huge for the query classification in this paper, it is almost impossible to do direct human judgment on such large label space. Therefore, we propose end-to-end system level evaluations. The evaluation results show that the hybrid model enhances the performance of query classification for different query sets, especially the torso and tail query sets.

REFERENCES

[1] Jung-Woo Ha, Hyuna Pyo, and Jeonghee Kim. 2016. Large-scale item categorization in e-commerce using multiple recurrent neural networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 107–115.

[2] Vivek Gupta, Harish Karnick, Ashendra Bansal, and Pradhuman Jhala. 2016. Product classification in e-commerce using distributional semantics. *arXiv preprint arXiv:1606.06083*.

[3] Mangi Kang, Jaelim Ahn, and Kichun Lee. 2018. Opinion mining using ensemble text hidden markov models for text classification. *Expert Systems with Applications*, 94, 218–227.

[4] Zuxuan Wu, Yu-Gang Jiang, Xi Wang, Hao Ye, and Xiangyang Xue. 2016. Multi-stream multi-class fusion of deep networks for video classification. In *Proceedings of the 24th ACM international conference on Multimedia*. ACM, 791–800.

[5] Husheng Guo and Wenjian Wang. 2015. An active learning-based svm multi-class classification model. *Pattern recognition*, 48, 5, 1577–1597.

[6] Venkataraman Santhanam, Vlad I Morariu, David Harwood, and Larry S Davis. 2016. A non-parametric approach to extending generic binary classifiers for multi-classification. *Pattern Recognition*, 58, 149–158.

[7] Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 2017. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 115–124.

[8] Himanshu Jain, Yashoteja Prabhu, and Manik Varma. 2016. Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 935–944.

[9] Ian En-Hsu Yen, Xiangru Huang, Pradeep Ravikumar, Kai Zhong, and Inderjit Dhillon. 2016. Pd-sparse: a primal and dual sparse approach to extreme multiclass and multilabel classification. In *International Conference on Machine Learning*, 3069–3077.

[10] 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 *Proceedings of the 22nd international conference on World Wide Web*. ACM, 13–24.

[11] Wei Bi and James Kwok. 2013. Efficient multi-label classification with many labels. In *International Conference on Machine Learning*, 405–413.

[12] Himanshu Jain, Venkatesh Balasubramanian, Bhanu Chunduri, and Manik Varma. 2019. Slice: scalable linear extreme classifiers trained on 100 million labels for related searches. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. ACM, 528–536.

[13] Adel Hamdan Mohammad, Tariq AlwadaâĂŹn, and Omar Al-Momani. 2018. Arabic text categorization using support vector machine, naïve bayes and neural network. *GSTF Journal on Computing (JoC)*, 5, 1.

[14] Wei Chen, Xusheng Yan, Zhou Zhao, Haoyuan Hong, Dieu Tien Bui, and Biswajeet Pradhan. 2019. Spatial prediction of landslide susceptibility using data mining-based kernel logistic regression, naive bayes and rbfnetwork models for the long county area (china). *Bulletin of Engineering Geology and the Environment*, 78, 1, 247–266.

[15] S Vijayarani and S Dhayanand. 2015. Liver disease prediction using svm and naïve bayes algorithms. *International Journal of Science, Engineering and Technology Research (IJSETR)*, 4, 4, 816–820.

[16] Young-gon Kim, Taehee Lee, Jonghoon Chun, and Sanggoo Lee. 2006. Modified naïve bayes classifier for e-catalog classification. In *International Workshop on Data Engineering Issues in E-Commerce and Services*. Springer, 246–257.

[17] How Jing, Yu Tsao, Kuan-Yu Chen, and Hsin-Min Wang. 2013. Semantic naïve bayes classifier for document classification. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, 1117–1123.

[18] Andrew Y Ng and Michael I Jordan. 2002. On discriminative vs. generative classifiers: a comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, 841–848.

[19] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

[20] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

[21] Rie Johnson and Tong Zhang. 2014. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*.

[22] Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. 2015. Sparse local embeddings for extreme multi-label classification. In *Advances in neural information processing systems*, 730–738.

[23] Yashoteja Prabhu and Manik Varma. 2014. Fastxml: a fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 263–272.

[24] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

[25] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

[26] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

[27] Shantanu Godbole and Sunita Sarawagi. 2004. Discriminative methods for multi-labeled classification. In *Pacific-Asia conference on knowledge discovery and data mining*. Springer, 22–30.

[28] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15, 1, 1929–1958.

[29] Ilya Loshchilov and Frank Hutter. 2016. Sgdr: stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*.

[30] Mohammad S Sorower. 2010. A literature survey on algorithms for multi-label learning. *Oregon State University, Corvallis*, 18, 1–25.

[31] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2009. Mining multi-label data. In *Data mining and knowledge discovery handbook*. Springer, 667–685.

[32] Jin Huang and Charles X Ling. 2005. Using auc and accuracy in evaluating learning algorithms. *IEEE Transactions on knowledge and Data Engineering*, 17, 3, 299–310.

[33] Charles X Ling, Jin Huang, Harry Zhang, et al. 2003. Auc: a statistically consistent and more discriminating measure than accuracy. In *Ijcai*. Volume 3, 519–524.