Entity-aware Multi-task Learning for Query Understanding at Walmart

Zhiyuan Peng*
Santa Clara University
Santa Clara, USA
zpeng@scu.edu

Vachik Dave
Walmart Global Tech
Sunnyvale, USA
vachik.dave@walmart.com

Nicole McNabb
Walmart Global Tech
Sunnyvale, USA
nicole.mcnabb@walmart.com

Rahul Sharnagat
Walmart Global Tech
Sunnyvale, USA
rahul.sharnagat@walmart.com

Alessandro Magnani
Walmart Global Tech
Sunnyvale, USA
alessandro.magnani@walmart.com

Ciya Liao
Walmart Global Tech
Sunnyvale, USA
ciya.liao@walmart.com

Yi Fang
Santa Clara University
Santa Clara, USA
yfang@scu.edu

Sravanthi Rajanala†
Walmart Global Tech
Sunnyvale, USA
sravanthi.rajanala@walmart.com

ABSTRACT

Query Understanding (QU) is a fundamental process in E-commerce search engines by extracting the shopping intents of customers. It usually includes a set of different tasks such as named entity recognization and query classification. Traditional approaches often tackle each task separately by its own network, which leads to excessive workload for development and maintenance as well as increased latency and resource usage in large-scale E-commerce platforms. To tackle these challenges, this paper presents a multi-task learning approach to query understanding at Walmart. We experimented with several state-of-the-art multi-task learning architectures including MTDNN, MMoE, and PLE. Furthermore, we propose a novel large-scale entity-aware multi-task learning model (EAMT)[1] by retrieving entities from engagement data as query context to augment the query representation. To the best of our knowledge, there exists no prior work on multi-task learning for E-commerce query understanding. Comprehensive offline experiments are conducted on industry-scale datasets (up to 965M queries) to illustrate the effectiveness of our approach. The results from online experiments show substantial gains in key accuracy and latency metrics.

CCS CONCEPTS

• **Computing methodologies → Multi-task learning**; • **Information systems → Information retrieval query processing**.

*The work was done during the internship at Walmart Global Tech.
†Sravanthi Rajanala is the corresponding author.
[1]Our code can be found at https://github.com/zhiyuanpeng/KDD2023-EAMT

KEYWORDS

Multi-task learning, Semi-supervised learning, Query Understanding, E-commerce

ACM Reference Format:
Zhiyuan Peng, Vachik Dave, Nicole McNabb, Rahul Sharnagat, Alessandro Magnani, Ciya Liao, Yi Fang, and Sravanthi Rajanala. 2023. Entity-aware Multi-task Learning for Query Understanding at Walmart. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3580305.3599816

1 INTRODUCTION

Query Understanding (QU) is an essential component of E-commerce search engines, which aims to predict the customer intent given a search query. Queries are usually short texts [16] containing several words indicating what customers intend to buy. For instance, it is QU's job to understand that a customer who searches the query "Sony TV 64 black" wants to buy a TV (product type) with the brand "Sony", size "64 inches", and color "black".

QU usually consists of a set of different tasks and each one can capture one aspect of the customer's intent. The tasks may vary with different E-commerce platforms. In this paper, we focus on the following four tasks of QU on Walmart's online shopping platform:

- Product Type Classification (PT). PT is a sentence-level multi-label classification task with about 6K labels in total.
- Query Catalog Classification (QC). QC is the same as PT except for the number of labels. For QC, there are about 1k labels in total.
- Named Entity Recognition (NER). NER is to identify named entities in queries and classify them into predefined semantic categories, like color, size, etc. Specifically, we use IOB2 scheme [19] to indicate each entity's start and end positions. We treat NER as a token-level multi-class classification task with about 60 labels in total.
- Term Weighting (TW). TW is to distinguish whether or not each token in the query is extraneous, which is a token-level

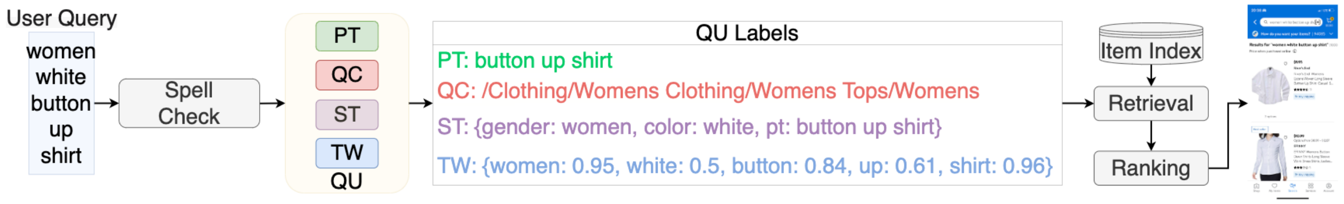

User Query
women white button up shirt
Spell Check
PT
QC
ST
TW
QU
QU Labels
PT: button up shirt
QC: /Clothing/Womens Clothing/Womens Tops/Womens
ST: {gender: women, color: white, pt: button up shirt}
TW: {women: 0.95, white: 0.5, button: 0.84, up: 0.61, shirt: 0.96}
Item Index
Retrieval
Ranking

Figure 1: An overview of QU in a retrieval system.

binary classification task. Instead of treating each token in the query equally, TW detects the extraneous tokens and excludes them when retrieving products.

An overview of QU on the Walmart e-commerce platform is shown in Figure 1. After correcting the input query, QU models predict the labels of the four QU tasks. The retrieval takes the QU labels as inputs and searches items from the item index. Customers will see a list of items ranked by ranking model.

Traditional approaches to QU tackle each task separately by its own network and pipeline, which leads to excessive workload for development and maintenance as well as increased latency and resource usage in large-scale E-commerce platforms. In this paper, we present a multi-task learning approach to query understanding at Walmart, which enables several key advantages for QU in large-scale E-commerce platforms. First of all, multi-task learning enables a single launch for a combination of all models for individual QU tasks. This will increase the engineering development and maintenance drastically. It also help measure the unified impact of model changes. For example, whenever we add a new feature it will be automatically incorporated to all predictions. Moreover, it saves the computational resources and memory footprint as all tasks share a significant portion of a single model. Last but not the least, we can leverage transfer learning between different QU tasks to improve overall performance and alleviate potential overfitting on a single task [27]. In query understanding, the training corpora of different tasks are all queries with different labels, which make it naturally fits the multi-task learning paradigm. Moreover, we resort to semi-supervised learning via retrieval augmentation, to learn from the additional information extracted from the relevant engagement data for prediction to achieve improved performance. Our main contributions can be summarized as follows:

- We introduce a large-scale multi-task deep learning approach to modeling multiple query understanding tasks. To the best of our knowledge, there is no prior work on multi-task learning for E-commerce query understanding.
- We compared and experimented with several state-of-the-art multi-task learning architectures including MTDNN, MMoE, and PLE, for query understanding.
- We propose a novel large-scale entity-aware multi-task learning model (EAMT) with a clean and effective engineering solution by retrieving entities from engagement data as query context to augment the query representation.
- Comprehensive offline experiments are conducted on industry-scale datasets (up to 965M queries) to illustrate the effectiveness of our approach. The results from online experiments show substantial gains in key accuracy and latency metrics.

2 RELATED WORK

2.1 Query Understanding

Most of the researchers focus on a single QU task. [26] designs three unique loss functions to improve the performance of the basic Seq2Seq model to re-write the tail source queries to well-trained head queries while preserving the shopping intent. [3] introduces a model to identify the entities in queries from a knowledge base. A few works focus on more than one QU task. Relying on transfer learning, [2] presents a whole QU system with the ability to process another language with minimal annotation by experts, but the tasks are trained in sequence. [11] and [8] argue that queries are short texts with less contextual information than long texts, so they propose different methods to train better pre-trained language models (PLMs) for queries which can improve QU tasks. Much fewer works about multi-tasking learning in QU. Rao and Ture, et al. [17] develop a multi-task learning model based on LSTM to process the voice queries.

2.2 Multi-task Learning

Deep neural network based multitask learning is studied and applied in many academic areas like computer version and natural language processing, Web, and so on [27]. With the progress in sizeable masked language models (MLM) like BERT [10], multi-task neural network (MT-DNN) [13] was proposed to utilize the BERT as the shared structure among all tasks to learn expressive semantic embeddings that are fed into the following heads, usually small specific neural networks designed for different tasks. The shared BERT is trained on all the data, allowing knowledge to be transferred among different tasks, but the heads of different tasks are independently trained. In multi-task learning, not all the tasks update the parameters in the same direction, e.g., the tug-of-war phenomenon [7]. The multi-gated mixture of experts (MMoE) [6] explores to use the sparse network to learn the specialized information about individual tasks and the shared information among multiple tasks separately. Specifically, MMoE inserts a mixture of expert (MoE) [20] layer between BERT and the heads of tasks. Instead of using only one shared gate in the original MoE, MMoE has a gate for each task. The number of experts and the structure of experts are all newly introduced hyperparameters; in practice, it is a burden to search such optimal parameters. [14] proposes a large-scale online multi-task learning model based on MMoE for ads auction. Like MMoE, PLE [21] also explicitly separates shared and task-specific experts to mitigate the negative transfer among tasks. The difference is that PLE introduces multi-level experts and gating networks, each of which is an MMoE layer, and applies a progressive separation routine to extract deeper knowledge from lower-layer experts and gradually separate task-specific parameters in higher levels. Apart from the hyperparameters introduced by MMoE, PLE has the challenge of designing a progressive learning structure. MMoE and PLE learn the sparse network after the transformer-based models like BERT. Recently, Switch [5], MT-Switch [6], and MT-TaG [6] learn the sparse network within the transformer. MLTR [24] proposes a multi-task learning framework for product ranking on a large-scale e-commerce dataset. In this paper, we implement MT-DNN, MMoE, and PLE models and compare their performances on the e-commerce QU dataset.

2.3 Semi-supervised Learning

Many semi-supervised learning works use unlabeled data to help the supervised learning task. UDA [25] proposes to use consistency loss to constrain the model output the same labels for unlabeled augmented instance pairs. UDA can be easily extended to multi-task learning settings, however, for token-level tasks like NER, there will be fewer choices for data augmentation methods because the relative positions of tokens should keep the same when doing data augmentation. CL-KL [23] proposes to retrieve unlabeled instances to be context to help the NER task. Our entity-aware model also uses external memory to retrieve contexts for each query. However, the contexts are not the unlabeled queries but the entities extracted from the engagement data. Another difference in our work with CL-KL is that we use the listwise loss to make the model learn the relative importance of each retrieved entity.

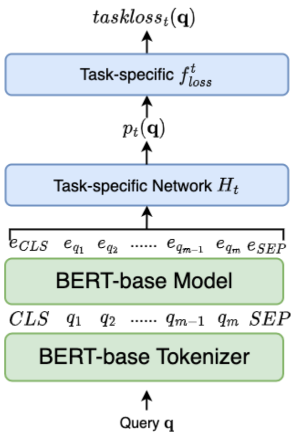
Different from all the works above, our proposed *EAMT* model can learn the relative importance of retrieved entities by calculating the listwise loss on the entities. The model can learn a better entity context for each query to help the overall performance of QU tasks. To the best of our knowledge, this is the first paper discussing how to successfully build an online large-scale multi-task learning for e-commerce query understanding. Also, our work can be easily extended to any number of tasks and other domains.

3 TASKS OF QUERY UNDERSTANDING

As introduced in Section 1, the query understanding process at Walmart focuses on four individual tasks: Product Type Classification (PT), Query Catalog Classification (QC), Named Entity Recognition (NER), and Term Weighting (TW). All the QU tasks adopt the same BERT-base model architecture as demonstrated in Figure 2, but they have different task-specific networks H_t and loss functions f_{loss}^t for the task t. A query $\mathbf{q}$ is tokenized into a list of token pieces q_1, q_2, ..., q_{m-1}, q_m. After concatenating the tokenized $\mathbf{q}$ with the start token CLS and end token SEP, a pre-trained BERT base model fine-tuned by Walmart queries encodes token pieces into embeddings $\mathbf{e}_{CLS}, \mathbf{e}_{q_1}, \mathbf{e}_{q_2}, ..., \mathbf{e}_{q_2}, \mathbf{e}_{q_m}, \mathbf{e}_{SEP}$ that are then fed into a followed task-specific network H_t to get the predictions $p_t(\mathbf{q})$. W_{H_t} denotes the parameters of H_t. We use the following prediction functions and loss functions for QU tasks.

Assume PT has N_{pt} labels and for each query $\mathbf{q}$, the true labels are $y_1, y_2, ..., y_{N_{pt}}$ each element of which is a binary indicator. The multi-label prediction of the task PT is:

$$p_{pt}(\mathbf{q}) = sigmoid(W_{H_{pt}}^T \cdot \mathbf{e}_{CLS}) \tag{1}$$



$taskloss_t(\mathbf{q})$
Task-specific f_{loss}^t
$p_t(\mathbf{q})$
Task-specific Network H_t
e_{CLS} e_{q_1} e_{q_2} ...... $e_{q_{m-1}}$ e_{q_m} e_{SEP}
BERT-base Model
CLS q_1 q_2 ...... q_{m-1} q_m SEP
BERT-base Tokenizer
Query $\mathbf{q}$

Figure 2: The architecture of each single QU task. All the tasks use the same green blocks and have different blue blocks.

where $p_{pt}(\mathbf{q}) = l_1, ..., l_{N_{pt}}$. l_n is the probability of query $\mathbf{q}$ have the nth label. We use the binary cross-entropy loss as the objective:

$$taskloss_{pt}(\mathbf{q}) = -\sum_{n=1}^{N_{pt}} [y_n \cdot \log l_n + (1 - y_n) \cdot \log(1 - l_n)] \tag{2}$$

QC has the same architecture as PT. Similarly, for QC, we have:

$$p_{qc}(\mathbf{q}) = sigmoid(W_{H_{qc}}^T \cdot \mathbf{e}_{CLS}) \tag{3}$$

$$taskloss_{qc}(\mathbf{q}) = -\sum_{n=1}^{N_{qc}} [y_n \cdot \log l_n + (1 - y_n) \cdot \log(1 - l_n)] \tag{4}$$

BERT tokenizer may tokenize each token into several small pieces. For NER, we assign each small piece a label by the rules: 1) O label if the original token has label O; 2) $I - original\_label$ if the original label is not O. Assume there are N_{ner} labels. Each token q_i in query $\mathbf{q}$ has only one label ranging from 1 to N_{ner}. The prediction of NER is:

$$p_{ner}(\mathbf{q}) = softmax(W_{H_{ner}}^T \cdot \mathbf{e}) \tag{5}$$

where $\mathbf{e} = \mathbf{e}_{q_1}, \mathbf{e}_{q_2}, ..., \mathbf{e}_{q_2}, \mathbf{e}_{q_m}$ and $p_{ner}(\mathbf{q}) \in \mathbb{R}^{m \times N_{ner}}$. $l_{i,j}$ is the element in $p_{ner}(\mathbf{q})$ with ith row and jth column. We use cross-entropy loss as the objective:

$$taskloss_{ner}(\mathbf{q}) = -\sum_{i=1}^{m} \sum_{j=1}^{N_{ner}} \mathbb{1}(q_i, j) \log l_{i,j} \tag{6}$$

where $\mathbb{1}(q_i, j)$ is the binary indicator (0 or 1) if the class label j is the correct classification for token q_i. CLS and SEP are masked when computing the loss.

The prediction of TW is:

$$p_{tw}(\mathbf{q}) = sigmoid(W_{H_{tw}}^T \cdot \mathbf{e}) \tag{7}$$

where $\mathbf{e} = \mathbf{e}_{q_1}, \mathbf{e}_{q_2}, ..., \mathbf{e}_{q_2}, \mathbf{e}_{q_m}$ and $p_{tw}(\mathbf{q}) = l_1, ..., l_m$. Assume y_i is the true label of ith token q_i that has prediction probability l_i. We use binary cross-entropy as the objective:

$$taskloss_{tw}(\mathbf{q}) = -\sum_{i=1}^{m} [y_i \cdot \log l_i + (1 - y_i) \cdot \log(1 - l_i)] \quad (8)$$

Only the first piece of the token will contribute to the loss for TW. *CLS* and *SEP* are masked when computing the loss.

4 MULTI-TASK LEARNING FOR QU

Our *EAMT* model is constructed on top of the baseline multi-task models by introducing an entity-aware component. This section will first introduce the baseline multi-task models (Section 4.1) and then present the proposed *EAMT* model (Section 4.2) with the entity retrieval model (Section 4.3).

4.1 Baseline Multi-task learning Models

In this paper, we compare three popular multi-task learning models: *MTDNN*, *MMoE*, and *PLE*, which can also serve as the building block for our proposed *EAMT* models. The differences between these three models are illustrated in Figure 3. For conciseness, we only draw two tasks *A* and *B*. The two tasks share the green structures. The blue structure is for task *A* and the red structure is for task *B*. *E* blocks are expert networks, and the circle *G* represents the gate network. During training, the green structure is always activated, but for each training step, only one of the red and blue structures is activated.

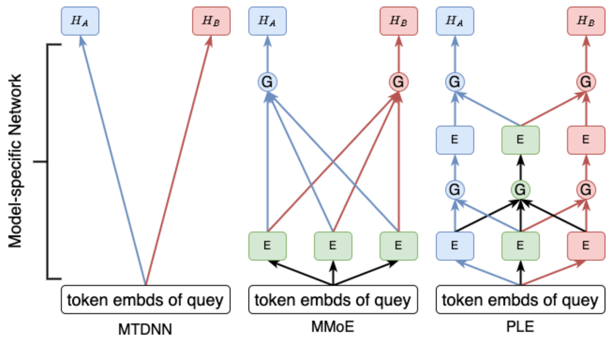


Figure 3: The architecture of the model-specific network.

4.2 Entity-aware Model Structure

Utilizing context information to help the large language models (LLM) like BERT has been well studied recently [4] [12]. For queries, the related entities in the engagement data, like brand, have semantic relationships with the PT, QC, and NER labels. Utilizing the entities as context can transfer knowledge to queries. The scores of entities indicating their importance also contain helpful information. To utilize the knowledge of entities, we propose the EAMT model to further improve the baseline multi-task model. EAMT retrieves entities as context while utilizing the scores of entities to regularize the model.

As shown in Figure 4, for the query "kung fu panda", we first call the entity retrieval model to retrieve the entities "isbn = kung fu panda & pt = books" and "film = kung fu panda & pt = movie". The two entities are concatenated by an unused token "[unused0]" as a context and then input the query and its context into the BERT-base model to obtain the embeddings. Model-specific network M_i takes the embeddings and outputs the embeddings with the same dimensionality. The head network H_t takes the embeddings from the previous step as input to compute the prediction $p_t^i(\mathbf{q})$. We average all the token embeddings in each entity as an entity representation and calculate the ListNet [1] loss on the entity representations.

More formally, for a baseline multi-task model i with model-specific network M_i, we use W_{M_i} to denote the parameters of M_i and W_{H_t} to represent the parameters of task-specific network H_t. For a query $\mathbf{q}$ that can be tokenized into a list of token pieces $q_1, q_2, ..., q_{m-1}, q_m$, we first call entity retrieval model to retrieve **entities** consisting of K entities each of which has a score indicating its order among the entities. Then, all the entities are shuffled, tokenized into token pieces, and concatenated by "[unused0]". All the entity token pieces $c_1, c_2, ..., c_{n-1}, c_n$ are concatenated as a query context $\mathbf{context_q}$ for qeury $\mathbf{q}$. Finally, query $\mathbf{q}$ and its context $\mathbf{context_q}$ are concatenated by "SEP" as input of BERT-base model to acquire the embeddings $\mathbf{e}_{\text{CLS}}, \mathbf{e}_{\mathbf{q}_1}, ..., \mathbf{e}_{\mathbf{q}_m}, \mathbf{e}_{\text{SEP}}, \mathbf{e}_{\mathbf{c}_1}, ..., \mathbf{e}_{\mathbf{c}_n}, \mathbf{e}_{\text{SEP}}$. $P_t^i(\mathbf{q})$ represents the task prediction of q for multi-task baseline model i's task t. The loss of task t is expressed as $taskloss_t^i$. The loss function of each task in the multi-task learning model is the same as that of the single task, which we have discussed in section 3. Here, we only show the prediction functions of each task as follows.

$$p_{pt}^i(\mathbf{q}) = sigmoid(W_{H_{pt}}^T \cdot (M_i^T \cdot \mathbf{e}_{CLS})) \quad (9)$$

$$p_{qc}^i(\mathbf{q}) = sigmoid(W_{H_{qc}}^T \cdot (M_i^T \cdot \mathbf{e}_{CLS})) \quad (10)$$

$$p_{ner}^i(\mathbf{q}) = softmax(W_{H_{ner}}^T \cdot (M_i^T \cdot \mathbf{e})) \quad (11)$$

$$p_{tw}^i(\mathbf{q}) = sigmoid(W_{H_{tw}}^T \cdot (M_i^T \cdot \mathbf{e})) \quad (12)$$

where $\mathbf{e} = \mathbf{e}_{q_1}, \mathbf{e}_{q_2}, ..., \mathbf{e}_{q_2}, \mathbf{e}_{q_m}$.

Each entity retrieved from the entity retrieval model has a score that indicates its relative importance compared to other retrieved entities. A ranking loss is utilized to make the model learn which entity is more important, so the attention networks [22] in the BERT-base model will pay more attention to the crucial entities, and better query embeddings are then learned. Suppose we retrieve K entities, we first extract the embeddings of tokens belonging to each entity and average the embeddings of all the tokens within the entity as the entity representation $\mathbf{ent}_k$. Dense network D_k with parameters W_{D_k} compress $\mathbf{ent}_k$ into a score s_k. We use listwise loss ListNet [1] to calculate $rankloss(\mathbf{entities})$ as follow:

$$rankloss(\mathbf{entities}) = \sum_{k=1}^{K} -softmax(y_k) \log softmax(s_k) \quad (13)$$

where y_k is the true score of the k^{th} entity and $softmax$ function is:

$$softmax(x_k) = \frac{e^{x_k}}{\sum_{k=1}^{K} e^{x_K}} \quad (14)$$

At last, the loss of task t and baseline multi-task learning model i is computed as:

$$loss_t^i = taskloss_t^i(\mathbf{q}) + rankloss(\textbf{entities}) \tag{15}$$

EAMT can be constructed on the three popular multi-task learning models: *MTDNN*, *MMoE*, and *PLE*, introduced in Section 4.1. The three proposed *EAMT* models $EMAT_{mtdnn}$, $EMAT_{mmoe}$, and $EMAT_{ple}$ only differ in the model-specific network M_i which is the red block in Figure 4. $EMAT_{mtdnn}$ has no model-specific network which means the token embeddings of a query are directly passed into a followed task-specific network. EA_{mmoe} has shared experts but independent gate networks. Instead of sharing all the experts in $EAMT_{mmoe}$, $EAMT_{ple}$ has both shared expert networks and task-specific expert networks. The expert network in *PLE* has multi-layers to learn the features progressively.

4.3 Entity Retrieval Models

We developed three entity retrieval models: the entity nearest neighbor (ENT-NN) model, the engagement nearest neighbor (ENG-NN) model, and the exact match (EXACT) model. The three models share a similar structure shown in Figure 5. Faiss indexer stores the normalized embeddings of targets encoded by the sentence transformer model all-MiniLM-L6-v2 [18] and their corresponding entities are stored in the entity memory. Each entity is a dictionary with keys like entity represents the entity type, name stores the entity's value, and pt stores a list of product types related to the entity. For conciseness, we only draw one product type for key pt in Figure 5, but in practice, each pt has at most ten product types concatenated by ";". When a query $\mathbf{q}$ comes into the entity retrieval model, it will be first encoded and normalized into an embedding $\mathbf{e}_q$ by the same all-MiniLM-L6-v2, and then we use Faiss [9] to conduct the nearest neighbor (NN) search to obtain the ids of the top k most similar targets. Finally, we locate the entities in the entity memory by the ids and retrieve the entities. ENT-NN and ENG-NN differ in the Faiss indexers and entity memories.

ENT-NN. We collected about 7M entities which include brands, book titles, author names, music names, and movie names by the algorithm. We directly encoded and normalized the 7M entities into embeddings and built the "IndexFlatIP" Fasiss index on them. For query $\mathbf{q}$ with M tokens, we first broke $\mathbf{q}$ into $M(M+1)/2$ grams. The tokens in a gram are concatenated into a sub-query $\mathbf{q}_{sub}$, which has m tokens. To retrieve the entities for $\mathbf{q}$, we first used the longest $\mathbf{q}_{sub}$ to do NN search, and the Faiss will return ids. Each one has a score of S_{faiss}, which is the Cosine similarity score between the normalized embedding of q_{sub} and the retrieved normalized entity embedding. The retrieved ids with the scores bigger than 0.99 are valid. Short $\mathbf{q}_{sub}$ will introduce severe noise, and thus S_{faiss} is punished by m/M. The entity score S_{ent-nn} is computed as:

$$S_{ent-nn} = S_{faiss} \times (m/M) \tag{16}$$

The advantage of ENT-NN is that we can add new entities to the Faiss indexer to handle the unseen queries or queries with fewer engagement data, while the disadvantage is the noisy entities introduced by the entity collecting algorithm.

ENG-NN. We collected 2 years of engagement data with each instance representing a pair of a query $\mathbf{q}_{eng}$ and its engagement entities. Each entity is brought by customers ord times. We used threshold $ord > 1$ to filter the $\mathbf{q}_{eng}$ and obtained about 20M different query-entities pairs. Instead of directly encoding the entities, ENG-NN encodes $\mathbf{q}_{eng}$ and builds the "PCA64, IVF16384_HNSW32, Flat" Faiss index for fast approximate NN search. The entity memory stores the corresponding entities, and each entry in it has a list of entity dictionaries. Different from ENT-NN's entity dictionary, ENG-NN's entity dictionary has one more key ord. "PCA64, IVF16384_HNSW32, Flat" cannot guarantee the retrieved $\mathbf{q}_{eng}$s are in the descending order of $Cosine(\mathbf{q}, \mathbf{q}_{eng})$. After retrieving a list of $\mathbf{q}_{eng}$s, we calculated the entity score S_{eng-nn} as:

$$S_{eng-nn} = Cosine(\mathbf{q}, \mathbf{q}_{eng}) \tag{17}$$

The average entity retrieval time for one query of QU-3.75M dataset is about 1ms.

EXACT. We directly applied ENG-NN's engagement queries as a dictionary to conduct an exact match to find the entities for queries. Compared with ENG-NN, EXACT retrieves entities with a higher precision but a lower recall. EXACT is utilized as a baseline to be compared with ENT-NN and ENG-NN.

5 EXPERIMENTAL SETUP

5.1 Datasets

We conducted our offline experiments on two large-scale QU datasets. The QU-3.75M dataset contains 3.75 million instances which are randomly sampled from the QU-965M dataset, which has about 965 million instances. The statistics of the datasets is shown in Table 1, and all the numbers in the table are approximate numbers. Among the four tasks, only NER's data is labeled by experts with domain knowledge. Other datasets are collected from engagement data and labeled by algorithms. We first conducted experiments on the QU-3.75M dataset to search optimal hyperparameters and model structures. The model performing best on the QU-3.75M dataset is picked up for training on the QU-965M dataset.

Table 1: Statistics of the QU datasets

Task	QU-3.75M			QU-965M		
	Train	Dev	Test	Train	Dev	Test
PT	1M	0.1M	0.1M	45.8M	0.46M	0.2M
QC	1M	0.1M	0.1M	44.7M	0.45M	0.2M
NER	0.14M	3.5K	10.5K	0.14M	3.5K	10.5K
TW	1M	0.1M	0.1M	3.2M	0.69M	0.69M

5.2 Training Details

We utilized two heuristics to mitigate the uneven learning problem of multi-task learning models. First, for different tasks, we selected different batch sizes to make each task have a similar number of batches in each training epoch. This heuristic can obtain a nearly balanced number of batches on QU-3.75M as PT, QC, and TW already have the same number of instances and we can select a smaller batch size for NER to make it have a similar number of batches. On the QU-965M dataset, PT, QC, and TW have much more instances than NER. We oversampled NER 10 times. In Section 6.6, we study the effect of oversampling. In addition, after each epoch,

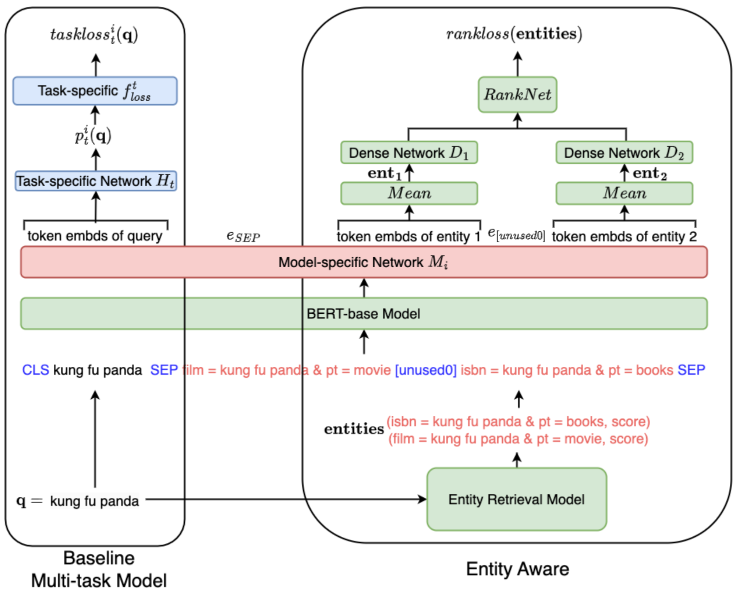

$taskloss_t^i(\mathbf{q})$
Task-specific f_{loss}^t
$p_t^i(\mathbf{q})$
Task-specific Network H_t
token embds of query
e_{SEP}
$rankloss(\mathbf{entities})$
$RankNet$
Dense Network D_1
Dense Network D_2
$\mathbf{ent}_1$
$\mathbf{ent}_2$
$Mean$
$Mean$
token embds of entity 1
$e_{[unused0]}$
token embds of entity 2
Model-specific Network M_i
BERT-base Model
CLS kung fu panda SEP film = kung fu panda & pt = movie [unused0] isbn = kung fu panda & pt = books SEP
$\mathbf{entities}$ (isbn = kung fu panda & pt = books, score) (film = kung fu panda & pt = movie, score)
$\mathbf{q} =$ kung fu panda
Entity Retrieval Model
Baseline Multi-task Model
Entity Aware

Figure 4: The architecture of the proposed *EAMT* model. For conciseness, we only draw two entities. All the tasks share the green blocks. Model-specific network M_i is different for different baseline multi-task models. The task-specific network H_t is unique for each task. The dataset of each task is split into mini-batches and merged together. All the mini-batches are iterated for training and at each training step, only one task is activated.

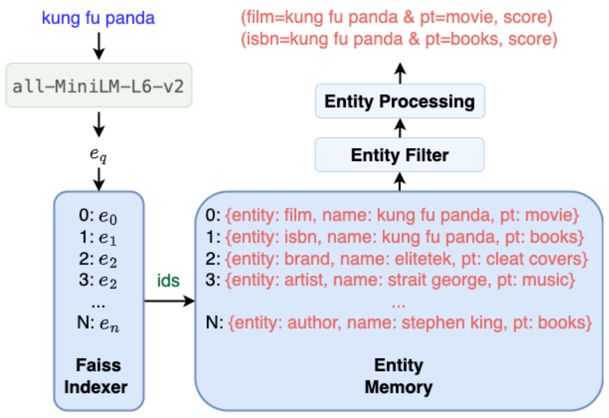

kung fu panda
all-MiniLM-L6-v2
e_q
0: e_0
1: e_1
2: e_2
3: e_2
...
N: e_n
Faiss Indexer
ids
(film=kung fu panda & pt=movie, score)
(isbn=kung fu panda & pt=books, score)
Entity Processing
Entity Filter
0: {entity: film, name: kung fu panda, pt: movie}
1: {entity: isbn, name: kung fu panda, pt: books}
2: {entity: brand, name: elitetek, pt: cleat covers}
3: {entity: artist, name: strait george, pt: music}
...
N: {entity: author, name: stephen king, pt: books}
Entity Memory

Figure 5: The architecture of entity retrieval model

we evaluated each task to obtain the F1 score and we calculate the geometric mean of all the F1 scores as:

$$Val\ F1 = \left(\prod_{t=1}^{4} F_t\right)^{\frac{1}{4}} \tag{18}$$

where F_t is the validation F1 score of task t. We track $Val\ F1$ score during training, and will stop the training if we find there is no better $Val\ F1$ score in recent ten epochs. The hyperparameter values used in the experiments are shown in Table 2.

Table 2: Description of Hyperparameters

Hyperparameters	QU-3.75M	QU-965M

Batch Size	PT=1024
	QC=1024
	NER=256
	TW=1024
		PT=4096
		QC=4096
		NER=512
		TW=512

Max Length	32	32
Learning Rate	$5e-5$	$5e-5$
DDP	Yes	Yes
Optimizer	AdamW	AdamW
Early Stop	10	10
Max epochs	30	50
GPU	8 A100s	16 A100s

5.3 Evaluation Metrics

For PT and QC, we report micro F1, because both of the tasks are at the query level. For NER, we use seqeval [15] to calculate the entity-level micro precision, micro recall, and micro F1. We report the precision, recall, and F1 for TW for positive labels. For overall performance, we report both micro F1 and macro F1 that are calculated as:

$$Micro\ F1 = \sum_{t=1}^{4} F1_t \times n_t / (\sum_{t=1}^{4} n_t) \tag{19}$$

$$Macro\ F1 = \sum_{t=1}^{4} F1_t / 4 \tag{20}$$

where, $F1_t$ and n_t are the F1 score and the number of test instance for task t respectively. Due to Walmart's confidentiality policy, relative improvements over baseline models varying with different experiments instead of absolute values are presented. F1 score ranges from 0 to 1; for brevity, we time F1 score by 100 and round it to two decimal places.

6 EXPERIMENTAL RESULTS

6.1 EAMT vs Baseline Models

By adding the entity-aware scheme, we built our $EAMT_{mtdnn}$ model based on MTDNN. MTDNN and $EAMT_{mtdnn}$ models are compared with baseline single-task models (Section 3) in Table 3. On QU-3.75M, both MTDNN and $EAMT_{mtdnn}$ gain relative improvements on PT, QC, and overall tasks, while $EAMT_{mtdnn}$ performs better than MTDNN. On NER and TW, the two models do not show apparent improvements, but still, $EAMT_{mtdnn}$ is better than MTDNN. This proves that the entity-aware scheme can improve the performance of baseline MTDNN further. We studied the training graphs of QU tasks and found TW is easy to converge, and after that, the performance drops, while PT, QC, and NER are much slower to converge. It is very challenging for the multi-task model to pick up a good checkpoint that works best for all tasks. NER has a much smaller number of instances than that of PT, QC, and TW, and during the training, the multi-task models receive fewer NER batches and the parameters may skew to other tasks with more instances or batches resulting in the underfitting of the NER task.

On QU-965M, apart from the similar patterns we have elaborated, $EAMT_{mtdnn}$ obtains 0.98 improvements on NER. This is because we oversample NER ten times so that NER has a closed number of batches as that of PT, QC, and TW, and the oversampling mitigates the imbalanced learning issue. We did not do oversampling in other experiments and the oversampling is studied in section 6.6. $EAMT_{mtdnn}$ consistently obtaining better overall performance than that of MTDNN and single task models on both QU-3.75M and QU-965M datasets demonstrates the effectiveness of our proposed EAMT model.

Table 3: The relative F1 improvements of MTDNN and $EAMT_{mtdnn}$ over single task models on QU-3.75M and QU-965M, respectively. $entity\_num = 3$ **for** $EAMT_{mtdnn}$.

Task	QU-3.75M		QU-965M	
	MTDNN	$EAMT_{mtdnn}$	MTDNN	$EAMT_{mtdnn}$
PT	0.81	1.7	1.41	1.94
QC	1.04	2.27	1.85	2.6
NER	-0.27	-0.18	-0.05	0.89
TW	-0.33	0.05	-0.13	-0.27
Micro	0.48	1.29	0.51	0.66
Macro	0.32	0.96	0.77	1.29

6.2 Impact of Entity-aware Component

We justify that the entity-aware component does help the baseline multi-task learning models. As shown in Table 4, all the EAMT models yield improvements on all four tasks except that $EAMT_{mmoe}$ and $EAMT_{ple}$ acquire a lightly worse result on TW. For overall performance, all EAMT models produce better results and $EAMT_{mtdnn}$ obtains the most significant overall improvements.

Table 4: Relative F1 improvements of EAMT models over their corresponding baseline multi-task models on QU-3.75M. $entity\_num = 3$ **for all EAMT models.**

Model	PT	QC	NER	TW	ALL	
					Micro	Macro
$EAMT_{mtdnn}$	0.89	1.23	0.09	0.38	0.81	0.65
$EAMT_{mmoe}$	1.14	1.19	0.19	-0.14	0.71	0.59
$EAMT_{ple}$	0.59	1.15	0.16	-0.04	0.55	0.47

6.3 Impact of Ranking Loss

We studied the effectiveness of ranking loss and the results are shown in Figure 5. For 3 entities and 5 entities, with ListNet loss, $EAMT_{mtdnn}$ get better results. For 2 entities, ListNet loss slightly decreases the performance of $EAMT_{mtdnn}$. Fewer entities mean the entities are more clean or related because we only take the top entity_num entities and the less need to use ListNet to distinguish which entity is more important.

Table 5: Relative F1 improvements of $EAMT_{mtdnn}$ with ListNet loss over $EAMT_{mtdnn}$ without ListNet loss on different entity_num. Results are reported on QU-3.75M.

entity_num	ALL	
	Micro	Macro
2	-0.01	-0.03
3	0.18	0.09
5	0.10	0.11

6.4 Impact of the Number of Entities

Retrieving more entities in EAMT will likely bring more valuable information and noise as well. Also, more entities may slow the training and increase the latency. We studied the influence of different numbers of entities retrived by the $EAMT_{mtdnn}$ model. The results are shown in Table 6. We did not do experiments with entity numbers more than 5 due to the much longer training time and the BERT's limitation of the input length. For all $entity\_num$, we notice the improvements over MTDNN and $EAMT_{mtdnn}$ obtains the best results when $entity\_num = 3$. We fix $entity\_num = 3$ when training the $EAMT_{mtdnn}$ model on QU-965M.

Table 6: Relative F1 improvements of $EAMT_{mtdnn}$ with different entity_num over MTDNN on QU-3.75M. ListNet loss is utilized.

entity_num	PT	QC	NER	TW	ALL	
					Micro	Macro
2	0.76	1.08	0.00	0.19	0.65	0.51
3	0.89	1.23	0.09	0.38	0.81	0.65
5	0.74	1.32	0.29	0.18	0.73	0.63

6.5 Impact of Different Filter Threshold

For ENT-NN, the score of each retrieved entity S_{ent-nn} is calculated as equation 16. An entity with a small S_{ent-nn} will be short compared with the search query because $S_{faiss} > 0.99$. Short entities will likely introduce noises; for instance, both "arm and hammer" and "hammer" are all entities, and if the search query contains "arm and hammer", we want to retrieve "arm and hammer" instead of "hammer". For ENG-NN, S_{eng-nn} is computed as equation 17. A bigger S_{eng-nn} means the retrieved entity belongs to an engagement query that is very similar to the search query. Conversely, an entity with a small S_{eng-nn} may likely introduce noises. We studied the influence of different thresholds on the quality of entities retrieved from NN models to choose optimal thresholds for NN models. To measure the quality of retrieved entities, we train $EAMT_{mtdnn}$ model on QU-3.75M utilizing the entities from NN models, and the higher the test F1 score is, the better quality of entities will be.

The results are shown in Figure 6 where EXAT is the exact match model that have no threshold parameters, so we draw the EXACT's results as a horizontal line. ENT-NN's performance is worse than EXACT because the collecting entity algorithm introduces noises and currently, we don't have a method to remove all the noise entities automatically. For instance, for query "waterproof iphone xs max case", the retrieved top one entity is "brand=iphone 13 pro max phone cases" and obviously "iphone 13 pro max phone cases" is not a brand name. Improving the threshold can reduce the probability of retrieving short and incomplete entities like "arm and hammer" and "hammer" example above, but can't mitigate the influence of fake entities that are not brand, movie, etc.

Compared with ENT-NN, ENG-NN's entities are much cleaner because the entities are at least brought one time by the customers who search the engagement query. The higher threshold is, the more similar the search query and the matched engagement query are, and the cleaner entities are. From Figure 6, the relative improvement of $EAMT_{mtdnn}$ with ENG-NN over MTDNN improves along with the increasing of threshold, which works as expected. The EXACT method can only retrieve entities for about 20% of the queries in QU-3.75M as the exact matching criterion is very strict, and any variation of the queries is unaccepted even though they look very similar. In contrast, the ENG-NN searching entities in embedding space can tolerate specific variants of the query and retrieve entities for more queries. When set threshold as 0.99, ENG-NN can retrieve at least one entity for about 33% of the queries in QU-3.75M. Finally, ENG-NN is chosen as the default entity retriever.

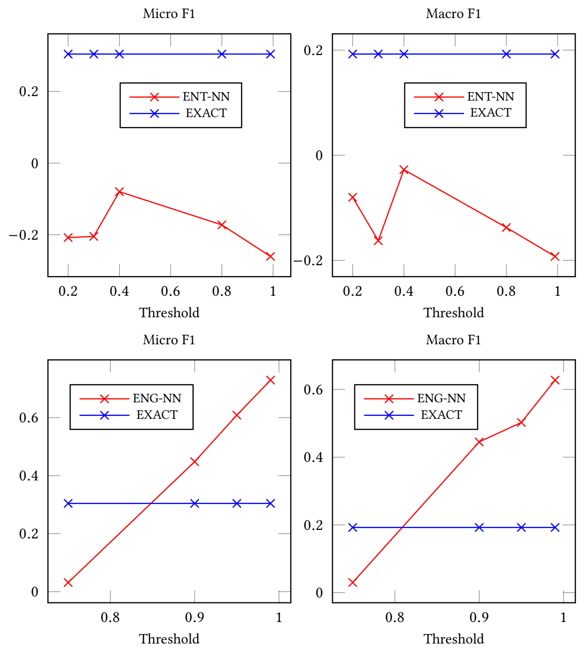


Figure 6: Relative F1 improvements of $EAMT_{mtdnn}$ with different entity retrieval models using different filter thresholds over MTDNN model on QU-3.75M. $entity\_num = 5$ **is default for all experiments.**

6.6 Impact of Oversampling

On QU-965M, the number of NER instances is much smaller than that of other tasks. Even if we choose a very small batch size for NER, NER will still has a much smaller number of batches compared to other tasks. The uneven number of batches will make the multi-task learning models learn specific tasks faster and cause an imbalanced learning issue. On QU-965M, we studied the effectiveness of oversampling and the results are shown in Table 7. We only oversample NER ten times and keep the data of other tasks unchanged. Without oversampling, $EAMT_{mtdnn}$ performs worse on NER while $EAMT_{mtdnn}$ yields a much better NER result after oversampling NER.

Table 7: Ablation study of oversampling NER dataset on QU-965M. The results are the relative F1 improvements of $EAMT_{mtdnn}$ model over single task models.

Oversample	PT	QC	NER	TW	ALL	
					Micro	Macro
No	1.59	2.22	-4.02	-0.92	0.07	-0.28
Yes	1.94	2.60	0.89	-0.27	0.66	1.29

6.7 Study of Baseline Multi-task Models

Compared to MTDNN, MMoE and PLE introduce new hyperparameters like the number of experts x and the number of layers of

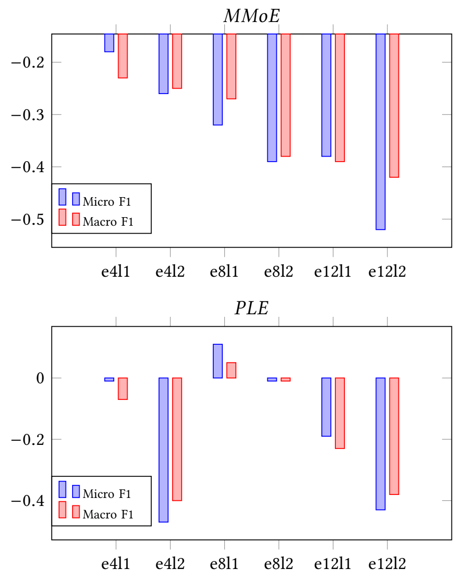


Figure 7: Relative F1 improvements of *MMoE* and *PLE* with different parameters over single task model on QU-3.75M.

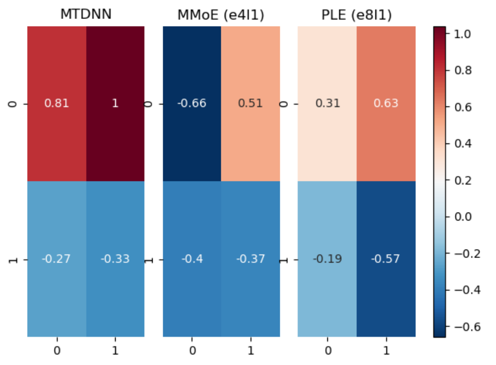


Figure 8: The relative F1 improvements of MTDNN, MMoE, and PLE over a single task. The coordinates of PT, QC, NER, and TW are (0,0), (0,1), (1,0), and (1,1), respectively.

each expert y. We first conducted experiments to select the best parameters x and y for MMoE and MTDNN, respectively. MMoE with $exly$ represents that it has x experts shared by all tasks. PLE with $exly$ denotes that the PLE model has x experts for each task and there are another x experts shared by all tasks. For MMoE and PLE, $exly$ indicates that each expert consists of y dense layers. Every expert in both PLE and MMoE does not change the input dimension.

The relative improvements of MMoE and PLE models with different $exly$ parameters over the single task model on QU-3.75M can be found in Figure 7. For MMoE, with a more complicated structure introduced, larger x and y, the performance decreases. For PLE, e8l1 is the best. The relative improvements of MTDNN, MMoE (e4l1), and PLE (e8l1) over a single task are shown in Figure 8. MTDNN enjoys the biggest improvement in the PT and QC tasks among the three models. For all three models, they obtain worse results on NER and TW. At last, we choose the MMoE (e4l1) and PLE (e8l1) to build our EAMT models on top of them.

6.8 Online Experiments

As a preliminary step, we have deployed the baseline $MTDNN$ model trained on QU-965M dataset to the production system and conducted a series of experiments. The first of these was a query-item relevance test utilizing historical queries. We gained **2.37%** NDCG improvement over the existing system. We then proceeded to an online interleaving test for top-ranked items. This led to **4.82%** ATC (add to cart) lift over the current production. Finally, we conducted both the A/B test and the reverse A/B test that yielded statistical significance lifts over the current production for different metrics such as: **0.51%** lift for GMV (gross merchandise value), **0.65%** lift on $ORDER$, **1.08%** lift for $UNITS$ of items sold, and **0.65%** lift on ATC.

As shown in Table 8, we also noticed over **25%** latency improvement for 90 percentile (p90) of search traffic, while noticing over 2% average response time reduction over the existing production system. At the same time, the requirement of GPU machines is reduced to $1/4^{th}$, because we replace 4 GPU-required tasks with a single model.

Table 8: Online latency improvement over the current production

Avg.	p90	p95	p99
2.10%	26.22%	24.96%	18.66%

7 CONCLUSION AND FUTURE WORK

In this paper, we propose a multi-task learning approach to QU tasks and conduct a comprehensive set of experiments. The MTDNN model obtains the best performance on Walmart QU tasks among three popular multi-task learning models. Online test results support that MTDNN can improve overall performance, reducing latency and resource usage compared to single tasks. To further improve the performance, we propose a novel entity-aware scheme that retrieves entities from engagement data as query context to augment the query representation. The offline test demonstrates the effectiveness of our EAMT model. Our model can be easily extended to other QU tasks as long as the inputs are queries. In future work, we will finalize the online test of the $EAMT_{mtdnn}$ model and explore to mitigate the imbalanced learning issue and reduce negative transfer in multi-task learning.

ACKNOWLEDGMENTS

The authors would like to thank Behnoush Abdollahi, Mossaab Bagdoui, Satya Chembolu, Lin Gong, and Sandip Sinha for their help in this work.

REFERENCES

[1] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*. 129–136.

[2] Ramji Chandrasekaran, Harsh Nilesh Pathak, and Tae Yano. 2020. Deep neural query understanding system at eXpedia group. In *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 1476–1484.

[3] Arash Dargahi Nobari, Arian Askari, Faegheh Hasibi, and Mahmood Neshati. 2018. Query understanding via entity attribute identification. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 1759–1762.

[4] Timo I Denk and Ana Peleteiro Ramallo. 2020. Contextual BERT: Conditioning the language model using a global state. *arXiv preprint arXiv:2010.15778* (2020).

[5] William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *J. Mach. Learn. Res* 23 (2021), 1–40.

[6] Shashank Gupta, Subhabrata Mukherjee, Krishan Subudhi, Eduardo Gonzalez, Damien Jose, Ahmed H Awadallah, and Jianfeng Gao. 2022. Sparsely activated mixture-of-experts are robust multi-task learners. *arXiv preprint arXiv:2204.07689* (2022).

[7] Raia Hadsell, Dushyant Rao, Andrei A Rusu, and Razvan Pascanu. 2020. Embracing change: Continual learning in deep neural networks. *Trends in cognitive sciences* 24, 12 (2020), 1028–1040.

[8] Haoming Jiang, Tianyu Cao, Zheng Li, Chen Luo, Xianfeng Tang, Qingyu Yin, Danqing Zhang, Rahul Goutam, and Bing Yin. 2022. Short Text Pre-training with Extended Token Classification for E-commerce Query Understanding. *arXiv preprint arXiv:2210.03915* (2022).

[9] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2019), 535–547.

[10] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT*. 4171–4186.

[11] Juanhui Li, Yao Ma, Wei Zeng, Suqi Cheng, Jiliang Tang, Shuaiqiang Wang, and Dawei Yin. 2022. Graph Enhanced BERT for Query Understanding. *arXiv preprint arXiv:2204.06522* (2022).

[12] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *Comput. Surveys* 55, 9 (2023), 1–35.

[13] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-Task Deep Neural Networks for Natural Language Understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 4487–4496.

[14] Ning Ma, Mustafa Ispir, Yuan Li, Yongpeng Yang, Zhe Chen, Derek Zhiyuan Cheng, Lan Nie, and Kishor Barman. 2022. An Online Multi-task Learning Framework for Google Feed Ads Auction Models. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3477–3485.

[15] Hiroki Nakayama. 2018. seqeval: A Python framework for sequence labeling evaluation. https://github.com/chakki-works/seqeval Software available from https://github.com/chakki-works/seqeval.

[16] Zhiyuan Peng, Behnoush Abdollahi, Min Xie, and Yi Fang. 2021. Multi-label classification of short texts with label correlated recurrent neural networks. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*. 119–122.

[17] Jinfeng Rao, Ferhan Ture, and Jimmy Lin. 2018. Multi-task learning with neural networks for voice query understanding on an entertainment platform. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 636–645.

[18] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. https://arxiv.org/abs/1908.10084

[19] Erik Tjong Kim Sang and Jorn Veenstra. 1999. Representing Text Chunks. In *Ninth Conference of the European Chapter of the Association for Computational Linguistics*. 173–179.

[20] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538* (2017).

[21] Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. 2020. Progressive layered extraction (ple): A novel multi-task learning (mtl) model for personalized recommendations. In *Proceedings of the 14th ACM Conference on Recommender Systems*. 269–278.

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[23] Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2021. Improving named entity recognition by external context retrieving and cooperative learning. *arXiv preprint arXiv:2105.03654* (2021).

[24] Xuyang Wu, Alessandro Magnani, Suthee Chaidaroon, Ajit Puthenputhussery, Ciya Liao, and Yi Fang. 2022. A Multi-task Learning Framework for Product Ranking with BERT. In *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, Frédérique Laforest, Raphaël Troncy, Elena Simperl, Deepak Agarwal, Aristides Gionis, Ivan Herman, and Lionel Médini (Eds.). ACM, 493–501. https://doi.org/10.1145/3485447.3511977

[25] Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. *Advances in neural information processing systems* 33 (2020), 6256–6268.

[26] Mengxiao Zhang, Yongning Wu, Raif Rustamov, Hongyu Zhu, Haoran Shi, Yuqi Wu, Lei Tang, Zuohua Zhang, and Chu Wang. 2022. Advancing query rewriting in e-commerce via shopping intent learning. In *SIGIR 2022 Workshop on eCommerce*. https://www.amazon.science/publications/advancing-query-rewriting-in-e-commerce-via-shopping-intent-learning

[27] Yu Zhang and Qiang Yang. 2021. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering* (2021).