Neural Search: Learning Query and Product Representations in Fashion E-commerce

Sagnik Sarkar*
sagnik.sarkar@myntra.com
Myntra Designs Pvt. Ltd.
India

Lakshya Kumar*
lakshya.kumar@myntra.com
Myntra Designs Pvt. Ltd.
India

ABSTRACT

Typical e-commerce platforms contain millions of products in the catalog. Users visit these platforms and enter search queries to retrieve their desired products. Therefore, showing the relevant products at the top is essential for the success of e-commerce platforms. We approach this problem by learning low dimension representations for queries and product descriptions by leveraging user click-stream data as our main source of signal for product relevance. Starting from GRU-based architectures as our baseline model, we move towards a more advanced transformer-based architecture. This helps the model to learn contextual representations of queries and products to serve better search results and understand the user intent in an efficient manner. We perform experiments related to pre-training of the Transformer based RoBERTa model using a fashion corpus and fine-tuning it over the triplet loss. Our experiments on the product ranking task show that the RoBERTa model is able to give an improvement of **7.8% in Mean Reciprocal Rank(MRR)**, **15.8% in Mean Average Precision(MAP)** and **8.8% in Normalized Discounted Cumulative Gain(NDCG)**, thus outperforming our GRU based baselines. For the product retrieval task, RoBERTa model is able to outperform other two models with an improvement of **164.7% in Precision@50 and 145.3% in Recall@50**. In order to highlight the importance of pre-training RoBERTa for fashion domain, we qualitatively compare already pre-trained RoBERTa on standard datasets with our custom pre-trained RoBERTa over a fashion corpus for the query token prediction task. Finally, we also show a qualitative comparison between GRU and RoBERTa results for product retrieval task for some test queries. RoBERTa model can be utilized for improving the product search task and act as a good baseline that can be fine-tuned for various information retrieval tasks like query recommendations, query re-formulation, etc.

KEYWORDS

Language Model, Transformers, BERT, RoBERTa, Transfer Learning, Product Ranking and Retrieval, E-commerce Products, pre-training and fine-tuning, metric learning

---

*Both authors contributed equally to this research.

---

1 INTRODUCTION & MOTIVATION

Efficient product search in Fashion e-commerce is pivotal for its success. Customers come to the platform with various intents of searching for different fashion products and poor results may cause bad customer experience. Understanding user intent in the form of queries is essential to serve customers with better results. Classic methods of representation learning like LDA[1] and LSA[7] use unsupervised objectives to map the user query and product in the same vector space for semantic matching. Since the advent of the Word2Vec [24] model, researchers started exploring shallow neural methods for representing queries and products. With several advancements in Natural Language Processing, researchers started experimenting with deeper neural models like LSTM[13], GRU[4], Transformers[37] and BERT[6] along with its variants like RoBERTa[21]. For a better understanding of queries and products these deeper models are used to model long-range context and learning better context dependent embeddings. In fact, the Transformer based models have shown to outperform other models for understanding Natural Language by showing significant improvements over standard data sets like GLUE[38]. When we compare Myntra Fashion E-commerce search with web search engines, the queries that come to our platform are also based on natural language but, they majorly focus on particular products or product categories. Most of the queries are shorter in length which creates a challenge in terms of correctly understanding the user intent and serving relevant results. Some of the example queries are shown below:

- 'hrx by hrithik roshan jeans men'
- 'nike tracksuit men'
- 'w legging'
- 'red lehenga choli'

In this paper, we propose different neural architectures for understanding queries and products by learning their representation in a low-dimension space. These different models are trained on data that is generated by creating a Query-Product graph and Product-Product co-occurrence graph from Myntra's click-stream data. Our experiments show the comparison of these different models with respect to different ranking metrics on the product ranking task. The performance of these models with respect to the product retrieval task is also discussed. Among the proposed architectures, the transformer-based RoBERTa[21] model is able to achieve the

best performance. We show how we have pre-trained an in-house RoBERTa model over a Fashion Corpus that we created using various product descriptions, reviews, and search queries. We also show how this model is fine-tuned for triplet loss optimization. The main contributions of the paper are:

- Propose RoBERTa in the task of learning latent query and product representations.
- Show a quantitative comparison of RoBERTa and GRU based models on product ranking and retrieval tasks.
- Highlight the need for pre-training the RoBERTa model over Fashion Domain.
- Report findings of augmenting Product-Product data with Query-Product data while training different neural models.

The rest of the paper is organized as follows: In Section 2, we review previous works that deal with representing queries/products for tasks like query rewriting, query attribute extraction and product retrieval. In Section 3, we first present a brief overview of data preparation and then present different neural models to learn low-dimensional embeddings of query and product. In Section 4, we outline the experimental setup and define different model training strategies. In Section 5, we present the results and visualizations of our experiments by reporting the performance of proposed models over different tasks. Finally, we conclude the paper and discuss future work in Section 6.

2 RELATED WORK

Representing queries and products in the same space is useful for tasks like product retrieval, ranking, etc. These tasks are solved in various ways in the e-commerce setting. Traditionally Information Retrieval(IR) in the domain of e-commerce search has been based on exact keyword matching between search queries and product descriptions/titles like BM25[31]. But this suffers from the problem of "vocabulary gap". With reference to semantic product search in e-commerce, three types of approaches have gained prominence in the recent years, namely, a) Query rewriting, b) Query attribute extraction and c) Embedding-based retrieval.

2.1 Query rewriting

The problem of vocabulary gap is pronounced in the e-commerce setting where queries are in informal language whereas the product titles/descriptions are written in formal language. One approach of mitigating this is by re-writing the original query into a query which is semantically similar but has less "lexical chasm"[12]. The problem can be viewed as a translation task trained on clicked query-document pairs. [8] A recent deep learning approach is the "Learning to Rewrite"[12] framework which leverages the query-product bipartite graph built from click-stream data, to build a candidate query generation phase and a ranking phase to re-write a poorly performing query to a well performing query. Some practical applications of query re-writing in the e-commerce domain include [23], [35].

2.2 Query attribute extraction

Typical search queries in e-commerce often include a collection of product attributes that are desired by a customer. One line of work for extracting these attributes from the query is by treating it as

a Named Entity Recognition(NER) problem. A weakly-supervised approach for doing this is the work by Guo et al [11] which applies a weakly supervised LDA algorithm, to identify four types of entities from commercial web search queries containing single named entities. A supervised learning approach is by Cowan et al. [5] in the domain of travel search queries, where a linear chain CRF is trained on a manually labeled NER dataset. Some NER based models in e-commerce search are [39] and [3]. The problem of attribute extraction from search queries can also be treated as a multi-label text categorization problem as demonstrated by Wu et al [41]

2.3 Embedding based retrieval

Classic Embedding Based Retrieval(EBR) methods such as LSA[7], LDA[1], BLTM and DPM[9] involve projecting the query and the document/product into a common embedding space thus capturing concept based similarity. These methods either suffer from being trained on an unsupervised objective which does not align well with the retrieval/ranking task or they have issues with scalability. In recent times, neural network based methods have gained popularity due to their ability of distributed representation learning and scalability. Mitra et. al.[25] provides a good survey of neural methods in IR. Guo et. al. [10] categorize deep neural network based methods for IR into 2 categories namely representation-focused models and interaction-focused models. Our approach falls in the category of representation-focused models since e-commerce product descriptions mostly satisfy the "Verbosity hypothesis" [10]. A seminal work in the space of representation-focused models is the DSSM model [16] which discriminatively trains a DNN by maximizing the conditional likelihood of the clicked documents given a query using the clickthrough data. New models such as DRMM [10], Duet [26] have been further developed to include traditional IR lexical matching signals. Few examples of applying EBR in e-commerce settings include JD.com[42] and in other search engines include Search engine EBR papers [15] [28].

In this work, we handle the problem of representation learning of queries and products in the same space by proposing different deep models. We report the performance of learned representations from these models on product ranking and retrieval tasks.

3 METHODOLOGIES

Myntra's click-stream data is our main source of information for getting products that are relevant to a given query. Using the click-stream data, we form a Query-Product bipartite graph with edge weights representing the number of sessions across which a product is clicked for a given query, as shown in Figure 1 (edge weights represented as w_{index}). From the click-stream data, we also form an un-directed Product-Product graph with edge weights representing the number of sessions where 2 products are clicked in the same session, as shown in Figure 2 (edge weights represented as w_{index}).

3.1 Data Preparation

The query-product co-occurrence is mainly driven by the search and ranking engine whereas the product-product co-occurrence is also influenced by the similar product recommendation engine
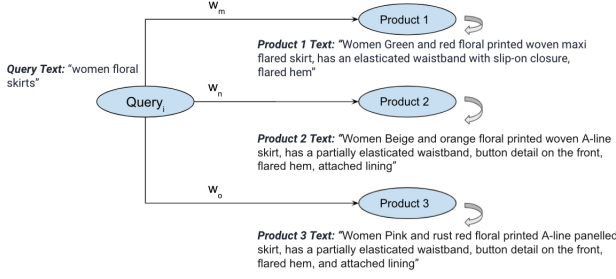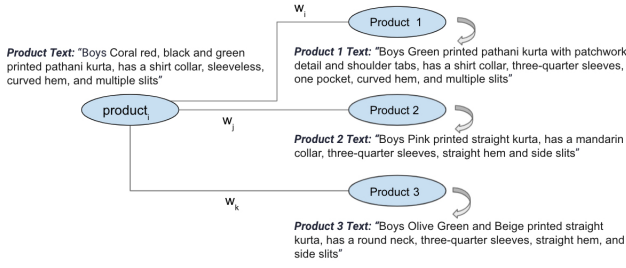
Figure 1: Query-Product Graph

Figure 2: Product-Product Graph

and thus it captures a richer source of information about product-product similarity. All products in Myntra are categorized into various ArticleType-Gender(ATG) groups e.g (AT:Dresses, G:Women), (AT:T-shirts, G:Men) etc. For the Product-Product graph, we only add edges between 2 products belonging to the same ATG. We create (query, positive-product) pairs by taking the query as the anchor and the top 100 products from the Query-Product graph (based on edge-weights) as the positive examples for the query. We observe that clicked results for some queries are limited to 1 ATG whereas for some queries the results span several ATGs. We mark queries spanning several ATGs as "unnamed queries" since they do not refer to a specific ATG. For example, the query "bags" would result in different types of bags such as *ladies handbags, laptop bags, trolleys* etc. We remove such queries from our dataset as they are already handled well by the current search engine. For queries spanning a single ATG, we observe that some queries are very broad in nature and almost all products in the relevant ATG may be relevant to the query for e.g: "men tshirts" (ATG: (T-shirts, Men)). Any query whose clicked results cover more than 30% of the products in the relevant ATG is marked as a "broad query" and the rest are marked as "narrow queries". As we use triplet loss in order to optimize different neural models, we follow 2 types of negative sampling strategies for mining negative examples to calculate the triplet loss. For "broad queries", we randomly sample a product from a different ATG (different from that of the positive product) whereas for "narrow queries" such as *"skinny fit jeans for men"*, half of the time, we randomly sample a product from the same ATG as that of the positive product and otherwise we sample from a different ATG. We also use the Product-Product graph to generate anchor-positive pairs of products. This is done by simulating a fixed number of

fixed length short random walks from each product node (anchor) and making all visited nodes its positive examples. The negative products are randomly sampled half of the time from the same ATG as the anchor and rest of the times from a different ATG. These product-product pairs are also augmented to the query-product pairs for training in one set of experiments. Using the mentioned approach, we finally obtain two sets of data, i.e., Query-Product and Product-Product data. We explain different neural models that we develop in order to learn the latent representation for queries and products in Myntra Fashion e-commerce. The basic entity for the first two neural architecture is GRU cell[4]. The third neural model is based on Transformer[37], i.e., RoBERTa[21] model which is first pre-trained on the Fashion corpus that we created manually and then fine-tuned by optimizing Triplet loss. We experiment with each of these different deep models in two settings, i.e., first by training them on Query-Product data and then by augmenting Query-Product data with Product-Product data for training.

3.2 Gated Recurrent Units(GRU)

The GRU has gating units like LSTM[13] that control the flow of information inside the unit. The GRU is different from LSTM as it does not have separate memory cells. The GRU cell mainly consists of two gates, i.e., Reset Gate and Update Gate. These gates are described below[1]:

- **Reset Gate**: Helps to control how much of the previous state information will be remembered.
- **Update Gate**: Controls the amount of previous information to throw away and what new information to add based on the current input to the GRU unit. This gate acts similar to the forget and input gate present in LSTM cell unit.

Formally, for a given time step t, assume that the input minibatch $X_t \in \mathbb{R}^{nxd}$ with number of examples as n and the number of inputs as d. Also, the hidden state of the previous time step is $H_{t-1} \in \mathbb{R}^{nxh}$ with number of hidden units as h. Then, the update gate $Z_t \in \mathbb{R}^{nxh}$ and reset gate $R_t \in \mathbb{R}^{nxh}$ are obtained as follows:

$$R_t = \sigma(X_t W_{xr} + H_{t-1} W_{hr} + b_r) \tag{1}$$

$$Z_t = \sigma(X_t W_{xz} + H_{t-1} W_{hz} + b_z) \tag{2}$$

where $W_{xr}, W_{xz} \in \mathbb{R}^{dxh}$ and $W_{hr}, W_{hz} \in \mathbb{R}^{hxh}$ are weights and $b_r, b_z \in \mathbb{R}^{1xh}$ are biases. The candidate hidden state $\tilde{H}_t \in \mathbb{R}^{nxh}$ at time step t,

$$\tilde{H}_t = \tanh(X_t W_{xh} + (R_t \odot H_{t-1}) W_{hh} + b_h) \tag{3}$$

where $W_{xh} \in \mathbb{R}^{dxh}$ and $W_{hh} \in \mathbb{R}^{hxh}$ are weights and $b_h \in \mathbb{R}^{1xh}$ is bias. The equation that leads to hidden state at time step t is given as,

$$H_t = Z_t \odot H_{t-1} + (1 - Z_t) \odot \tilde{H}_t \tag{4}$$

GRU has fewer tensor operations as compared to LSTM and they are faster to train as well, so we choose GRU cell unit in order to build first two neural architectures. Below we describe bi-directional GRU based neural model, where first neural model has only single layer, i.e., one forward GRU and one backward GRU in order to read the textual sentence. The second neural model is more deep as

---

[1]The definition of GRU cell and equations are taken from https://d2l.ai/chapter$_r$ecurrent − modern/gru.html

it consists of two layers, i.e., two forward GRU and two backward GRU.
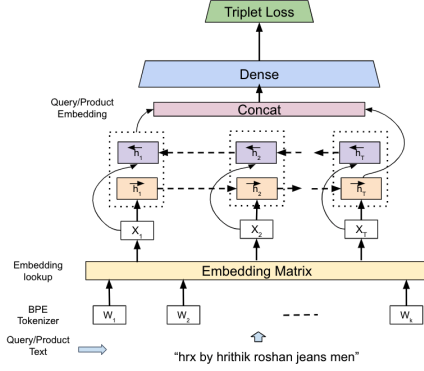
3.3 BiGRU: Single Layer

Figure 3: BiGRU:Single Layer Architecture

Single Layer BiGRU based model is shown in Figure 3. It consists of one forward and one backward GRU layer in order to learn the query/product representations in latent space. We train a BPE[32] tokenizer over the Fashion corpus[2] and use it in all the proposed neural models for tokenizing the input. The input to the model can be a query text or product description. As shown in the architecture, the last forward and backward GRU hidden states are concatenated and passed to a fully connected layer and finally the model is trained to optimize for the Triplet Loss function. The mathematical formulation of the Triplet Loss function is given as:

$$Loss = \sum_{i=1}^{N} [d(f_i^a, f_i^p) - d(f_i^a, f_i^n) + \alpha]_+ \tag{5}$$

where f_i^a, f_i^p, f_i^n correspond to the embeddings of query/product, positive product(clicked product) and negative products(as mentioned in Section 3.1) obtained from the model respectively. And d denotes the distance metric which is cosine distance in our model optimization. $[]_+$ denotes the function which will be non-zero if the value inside is positive and zero otherwise. The model optimization will try to bring the query/product embedding closer to positive product, i.e., clicked product and it will push away the random negative products from the query/product embedding. We have used this loss function in all our neural models in order to do the model training and then obtained the embeddings from the trained model for doing the evaluation over test dataset.

3.4 BiGRU: Multi Layer

In order to increase the model capacity, we have introduced another BiGRU layer and the same is shown in Figure 4. This architecture contains two forward GRU Layers and two backward GRU layers. The BPE tokenizer that is shown is same as described in Section 3.3. This model is almost similar to single layer BiGRU architecture proposed in Figure 3 except for one more layer that gives the model
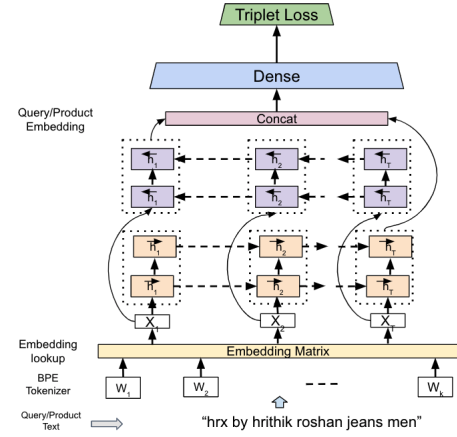

Figure 4: BiGRU:Multi Layer Architecture

more flexibility to learn better query/product representations that result in improved metrics as discussed in Section 5. This model is also trained by optimizing for Triplet Loss function which is given in Equation 5. Each of the proposed neural model is capable to take query text or product text as input. In Section 4, we will describe how we train each of the proposed model by giving two types of data, i.e., Query-Product data(obtained from Query-Product graph) and Product-Product data(which is obtained from Product-Product co-occurence graph).

3.5 RoBERTa model

RoBERTa[21] model which is a variant of BERT[6] is used for learning Fashion language using Fashion Corpus that is described in Section 4. The BERT model optimizes over two auxiliary pre-training tasks:

- **Mask Language Model (MLM)**: Randomly masking 15% of the tokens in each sequence and predicting the masked tokens
- **Next Sentence Prediction (NSP)**: Randomly sampling sentence pairs and predicting whether the latter sentence is the next sentence of the former

BERT based representations try to learn the context around a word and is able to better capture its meaning syntactically and semantically. For our case, we directly use RoBERTa model as it only optimises for MLM auxiliary task which is sufficient for efficient pre-training as shown in [21]. In experiments we use byte-level BPE [32] tokenization[3] for encoding sentences present in Fashion corpus. We use perplexity[2] score for evaluating the RoBERTa language model. Perplexity is defined as the exponentiated average log-likelihood of a sequence. If we have a tokenized sequence $X = (x_0, x_1, x_2, ..., x_t)$ then perplexity of X is,

$$PPL(X) = \exp\{\frac{-1}{t} \sum_{i}^{t} \log P_\theta(x_i|x_{<i})\} \tag{6}$$

where $\log P_\theta(x_i|x_{<i})$ is the log-likelihood of the ith token conditioned on the preceding tokens $x_{<i}$ according to RoBERTa model,

---

[2]Consists of queries, product titles, descriptions and reviews etc.

[3]The tokenizer is same across both the GRU and RoBERTa based models.

where θ indicates the model parameters. Generally, lower is the perplexity, better is the language model. After pre-training, the model would have learnt the syntactic and semantic aspects of tokens present in sentences in the Fashion corpus. With the help of self-attention it learns the context in which different tokens appear and tries to predict masked tokens based on the left and right context. Masking helps the RoBERTa model to use both the left and the right context without facing the problem of data leakage and the model learns contextual low dimension representation of tokens. The pre-training setup of RoBERTa model is shown in Figure 5. Again the BPE tokenizer used is same as mentioned in other neural approaches described in Sections 3.3 and 3.4.
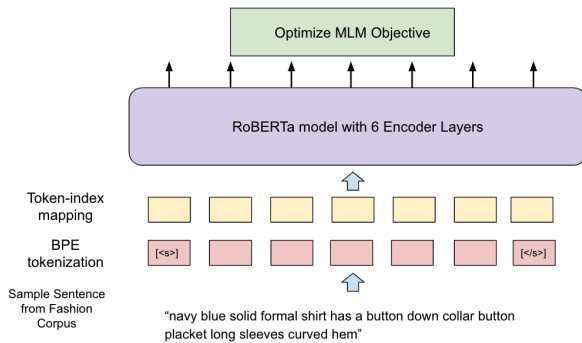

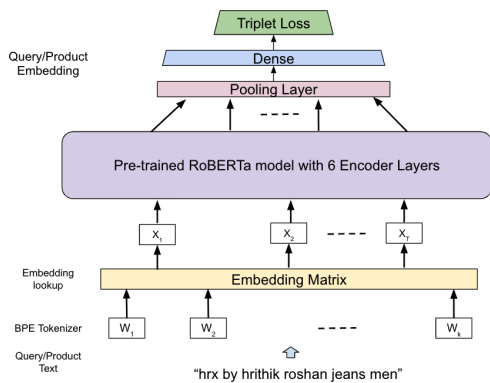**Figure 5: RoBERTa Pre-training setup using Fashion Corpus**

Figure 6: Fine-Tuning RoBERTa model with Triplet Loss

After Pre-training the RoBERTa model over Fashion corpus, we fine tune it to optimize for the Triplet loss function. The architecture for fine-tuning the pre-trained RoBERTa model is shown in Figure 6. Similar to other BiGRU based neural models, we fine-tune it over Query-Product and Product-Product data as described in Section 4. A query text or product description can be given as input to this model after tokenizing through BPE tokenizer and then the model will generate the embeddings corresponding to different tokens. A pooling layer is applied to the embeddings to generate a single embedding which acts as a latent representation of the input. Again the Triplet Loss is optimized using this model in order to

reduce the distance between query/product and clicked product and increase the distance between query/product and random negatively sampled products.

The three neural approaches mentioned are compared by computing different metrics like Mean Reciprocal Rank(MRR), Mean Average Precision(MAP) and Normalized Discounted Cumulative Gain(NDCG) for product ranking task. We also evaluate the proposed models for product retrieval task by calculating precision@K and recall@K metrics. The quantitative analysis is explained in Section 5.

4 EXPERIMENTAL SETUP

4.1 Dataset Description

For all our experiments we use 60 days of click-stream data. Spark[33] framework is used to prepare the Query-Product and Product-Product graphs. The resulting data contains approximately **140k** "unique"(in terms of exact matching) queries and **950k** products. We randomly split the queries into train and test sets in the ratio of 85:15 resulting in around **119k** train queries and **21k** test queries. We only use queries in the train set to form the query-product pairs. The Product-Product graph is also formed from click-stream events in the same time-span as the Query-Product graph. For generating product-product pairs for data augmentation, we simulate 5 random walks of length 5 per product node in the product co-occurrence graph and also remove repeating nodes from the random walk. We use the igraph[17] library for simulating the random walks.

4.2 Model Training

We train different models proposed in Section 3 in different data settings. Each of the mentioned neural model is trained using only Query-Product data as well as a larger data consisting of Query-Product and Product-Product data. We call the models trained with the larger data as models with Augmented Data as shown in Table 1 and 2. We will first describe the settings and the framework used to train GRU based neural models and then explain the pre-training and fine-tuning of the RoBERTa model. For all the models, we train the Byte-Pair Encoding(BPE) tokenizer over the whole dataset consisting of product descriptions, queries and product reviews. This dataset is called as Fashion Corpus. This tokenizer is used in order to tokenize the model input. The vocabulary size of the tokenizer is kept as **30K**. The Query-Product training data consists of query along with one clicked product that acts as positive and a randomly sampled product that acts as negative. The negative product is sampled from the same ATG or from a different ATG based on "narrow" or "broad" query type as mentioned in Section 3.1. For example: If the query is *'women kurtas'*, the positive product is the one which is clicked by the user and the negative product is sampled from different ATG because it is a broad query. In each of the neural models, the triplet loss optimization is performed by taking these positive and negative examples. For Product-Product data, we have an anchor product and other positive product which co-occurred with the anchor product in the random walk over Product-Product Graph. For this data, the negative product is sampled half of the time from the same ATG and half of the time from a different ATG. The model architectures proposed in Section 3 are trained first on Query-Product data and then by augmenting Product-Product data

to see the impact of adding the second data on model performance. The results with respect to different metrics are explained in Section 5.

*4.2.1 **GRU based Neural Models**.* The single layer BiGRU model is trained in Pytorch[29] Deep Learning framework. The Embedding layer of the model is $E \in \mathbb{R}^{|V| x D}$, where $|V|$ is vocabulory size which is 30K and D is the input token embedding dimension which is **100**. The hidden unit size of GRU cell is also **100**. There is a dense layer after the BiGRU as shown in Figure 3. After the forward and backward GRU cells read the tokenized input, their last hidden states are concatenated and given as input to dense layer with $W_{Dense} \in \mathbb{R}^{200 x 100}$. While training the model, we pass query text as well as product description for both the positive product and negatively sampled product one after another and generate their embeddings. Finally, the Triplet loss optimization is performed over these embeddings to minimze the loss. The multi-layer BiGRU neural model which is shown in Figure 4 is trained in a similar fashion. The number of BiGRU layers in this model are 2, i.e., two forward GRU and two backward GRU cells. After taking the tokenized input, forward GRU reads it and finally generates the hidden state from second GRU cell. In a similar way, the second GRU cell of backward GRU produce the final state. These two hidden states are concatenated and passed to the dense layer as shown in the architecture. All the other hyperparameters for this model are kept same except from two BiGRU layers instead of one. GRU based neural models uses Adam[19] optimizer in order to update the parameters of the model. All the GRU based models are trained for **50 epochs**.

*4.2.2 **RoBERTa based Neural Models**.* In order to model the problem of query/product representation learning, we first train the RoBERTa model from scratch to learn the different syntactic and semantic aspects of words appearing in the context of Fashion. We have prepared a Fashion corpus of size $\sim 4.5GB$ consisting of product titles, descriptions, product reviews and queries. This corpus is used to pre-train the RoBERTa language model and then the pre-trained RoBERTa model is fine-tuned for the Triplet loss optimization.

*   **Pre-training over Fashion Corpus**. For RoBERTa model pre-training, the architecture is shown in Figure 5. The model is trained to optimize '**Masked Language Modelling**' objective as mentioned in 3.5 for **2 epochs** with a cumulative training time of **2.5 days** and per-gpu training batch size of **8**. The multi-gpu pre-training is done using **Pytorch**[30] framework and **HuggingFace** library[40] based implementation of RoBERTa model with 2 **Tesla V100 GPUs**. The architecture of RoBERTa model that is used is '**DistilRoBERTa- base**' from HuggingFace having **6 encoder layers, 12 attention heads per layer and 82 million parameters** and this model is called as *RoBERTaForMaskedLM*. The hidden embedding dimension of the RoBERTa model is **768** and the position embedding is also of the same dimension. The model accepts the tokenized input of maximum length **512**. The model uses **AdamW** [20][22] optimizer in order to update the parameters during training. The perplexity of the RoBERTa model over evaluation dataset is obtained as **3.5**. In Section 5, we show how the RoBERTa model pre-trained over Fashion corpus is able to capture the context and

predict the masked words with relevant words as compared to already pre-trained RoBERTa model that is pre-trained on standard datasets.

*   **Fine-tuning RoBERTa model**. The fine tuning architecture of the RoBERTa model is shown in Figure 6. After pre-training the RoBERTa model from scratch on Fashion corpus it is fine-tuned for the triplet loss optimization. In order to prevent deviation of model parameters while fine-tuning, we use slanted triangular learning rate[14] strategy which first linearly increases the learning rate and then linearly decays it as per defined update rule. In order to perform this fine-tuning, we use scheduler present in the HuggingFace library. The optimizer for updating the parameters of this model is same as mentioned in pre-training of the RoBERTa model. The architecture for this model is same as the pre-trained RoBERTa model except that there is one RoBERTa pooling layer present at the top of the model. The class name of the model is called *RoBERTaModel* in the HuggingFace. When we initialize the *RoBERTaModel* with the pre-trained model, all the weights get initialized with the pre-trained weights. Given the tokenized input text, this model will output a embedding of dimension **768** which is then passed to a dense layer of dimension:$W_{Dense} \in \mathbb{R}^{768 x 100}$. Finally, we optimize this model for the Triplet loss function in order to fine-tune the model in two different data settings, i.e., training on Query-Product data and training by augmenting Product-Product data. This model is fine-tuned for only **1 epoch**.

We report the performance of different models for ranking and retrieval tasks in Section 5. Among different models, RoBERTa model act as a good baseline for serving the relevant products over fashion e-commerce given the user query.

5 RESULTS & VISUALIZATION

In order to evaluate the embeddings obtained from different models proposed in Section 3, we choose two tasks: Product Ranking and Product Retrieval. For these two tasks, we report different metrics.

5.1 Product Ranking Task

In order to compare different neural models with respect to a downstream task, we use query-clicked product ranking task. Table 1 show different ranking metrics computed for different models. In order to compute Mean Reciprocal Rank(MRR) for all the models with respect to clicked product, we have created a test dataset containing one clicked product and 20 negative products for every query, so the ratio of positive to negative is 1:20. All the models are evaluated on this test dataset. Among the different neural baselines, the single layer BiGRU with augmented data is able to give **38.8%** MRR. BiGRU with 2 layers trained over only Query-Product data give a MRR score of **44.5%**. The fine-tuned RoBERTa model give a MRR score of **48%** which is trained with only Query-Product data and a MRR of **40%** with augmenting Product-Product data. For Multi-layer GRU, we observe a performance drop in the MRR but an increase in MAP and NDCG with augmented data. For the RoBERTa model, we observe a decline in all the ranking metrics with augmented data. We suspect that the drop is due to fine-tuning only for 1 epoch even after augmenting more data. The RoBERTa model is able to outperform both the single layer GRU and multi layer GRU with just 1 epoch of fine-tuning as compared to 50 epochs for the other

models. As RoBERTa model is already pre-trained over Fashion corpus, it is able to achieve a higher MRR score thereby leveraging transfer learning. In order to further compare these different models, we compute other ranking metrics like MAP and NDCG. We prepare another smaller test dataset from Query-Product graph containing positive(clicked) products and 3 random negative products per positive product with respect to each query. In this smaller dataset, the queries are a subset of bigger dataset that we use for computing MRR metric. For every query in this smaller dataset, we rank the set of positive and negative products using the embeddings obtained from the trained models. For example, if for a given query there are n_1 positive(clicked) products and $n_2 = 3n_1$ negative products, then we first obtain the embedding for query text. Then we obtain the product embedding for each of the $n_1 + n_2$ products using their product description from the trained models. All the $n_1 + n_2$ products are ranked using cosine-similarity between query embedding and product embedding. After generating the predicted ranking, we compare it with the ideal ranking where all the positive(clicked) product should present before all the negative products(non-clicked). With respect to this dataset, the RoBERTa model fine-tuned with only Query-Product data outperforms other models with a MAP of **70.9%** and NDCG of **81.1%**.

Approach	MRR	MAP	NDCG
GRU:Single Layer	29.36%	59.7%	73.3%
GRU:Single Layer with Augmented Data	38.8%	60.9%	74.2%
GRU:Multi Layer	44.5%	58.9%	72.7%
GRU:Multi Layer with Augmented Data	41.5%	61.2%	74.5%
RoBERTa model	**48%**	**70.9%**	**81.1%**
RoBERTa model with Augmented Data	40%	63.6%	76.1%

Table 1: Evaluation for product ranking task

5.2 Product Retrieval Task

The query and product embeddings learned using different models can be used to retrieve the products given the query embedding. A comparison of all the models with respect to product retrieval task is shown in Table 2. For product retrieval task, we use the same dataset that we use to calculate MAP and NDCG and filtered only those queries that belong to 1 ATG. In order to assign the ATG to queries, we look at the clicked product for the queries and assign the ATG of the clicked products to the queries.

For retrieving the products based on query embedding, we construct six[4] different Annoy[34] index over the product embeddings obtained from different models. For every query in the test dataset, we obtain the query embedding from the model and then refer the Annoy index to fetch top 50 products. Along with each query, we also have a ground truth set of clicked products obtained from Query-Product graph that is used to compute the Precision@50 and Recall@50 metrics. Among the different models, RoBERTa model outperform all other models with a Precision@50 of **4.5%** and Recall@50 of **21.1%**.

In Figure 7 and 8, we show comparison of retrieved product results[5] for some test queries between GRU and RoBERTa models. For these test queries, we will first pass them through the model to obtain their embeddings. These embeddings are then used to retrieve the products from the product index created using product

---

[4]which corresponds to 6 different models trained as shown in Table 1 and Table 2.
[5]Due to the space limitation, we show less number of retrieved results

Approach	Precision@50 %	Recall@50 %
GRU:Single Layer	1.4%	7.5%
GRU:Single Layer with Augmented Data	1.7%	8.6%
GRU:Multi Layer	1.4%	7.4%
GRU:Multi Layer with Augmented Data	1.2%	6.1%
RoBERTa model	**4.5%**	**21.1%**
RoBERTa model with Augmented Data	2.7%	11.1%

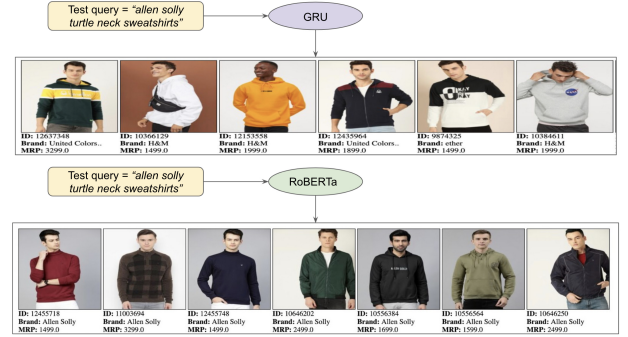**Table 2: Evaluation for product retrieval task**

Figure 7: Comparison of retrieved product results for test query: *'allen solly turtle neck sweatshirts'*

Figure 8: Comparison of retrieved product results for test query: *'nike running shoes men'*

embeddings obtained from GRU and RoBERTa separately.

Figure 7 show the results for the test query: *'allen solly turtle neck sweatshirts'*. The retrieved products from RoBERTa model clearly indicate the intent of the query, i.e., **'allen solly'**, **'sweatshirt'** with **'turtle neck'**, still some of the results have different neck. Whereas GRU return the products that are **'sweatshirt'** but from different brands and neck pattern.

Figure 8 show the results for the test query: *'nike running shoes men'*, and the retrieved products from RoBERTa clearly captures the intent of the query by showing the men running shoes from the **brand:Nike**. Whereas the retrieved products from GRU model are **'running shoes'** but from other brands.

Figure 9 show the visualization for token prediction task with respect to different queries. As we see from the visualizations, after pre-training the RoBERTa model over Fashion corpus, it has a good understanding of the fashion text and generate valid predictions for masked tokens in the queries. However, the already pre-trained RoBERTa model does not generate good predictions and fails to
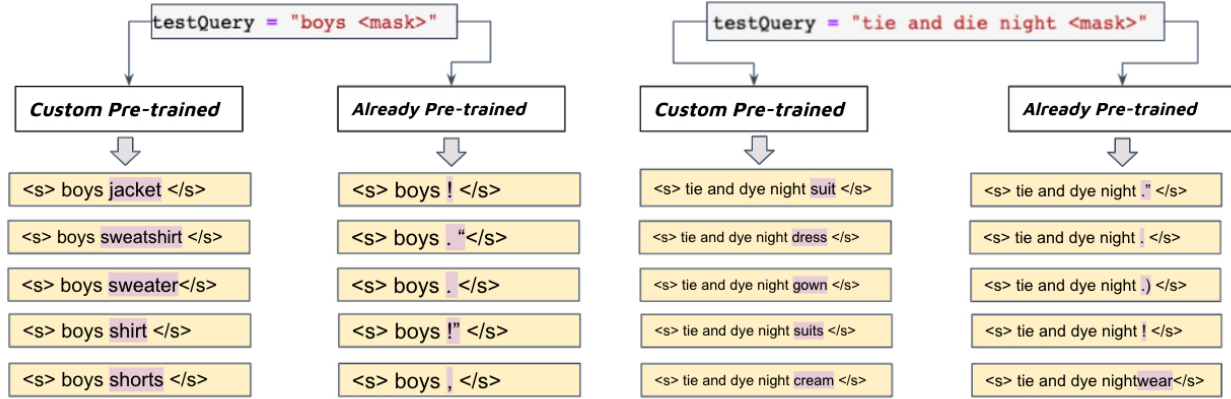
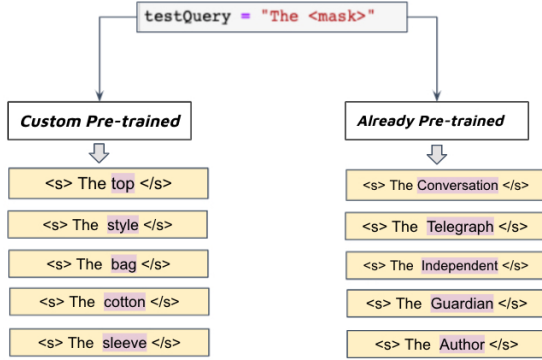**Figure 9: Custom pre-trained vs Already pre-trained RoBERTa for Query Completion**

Figure 10: Pre-training impact with respect to the word *'The'*

understand the context in some cases. For the test query: *'boys <mask>'*, the custom pre-trained RoBERTa model(pre-trained over the Fashion Corpus) give valid predictions for the <mask> token like **'jacket', 'sweatshirt', 'sweater', 'shirt', 'shorts' etc**. On the other hand the already pre-trained RoBERTa model give garbage predictions in the form of different characters like **'!', '.'** etc. For one more example test query: **'tie and dye night <mask>'**, the custom pre-trained model give valid predictions like **'suit', 'dress', 'gown', 'cream'** etc. But already pre-trained RoBERTa model again give random tokens as predictions. In order to further assess these two RoBERTa model, we give the test query as **'The <mask>'** as shown in Figure 10. The custom pre-trained model give predictions that are coherent with respect to Fashion like **'top', 'style', 'bag', 'sleeve'** etc. The already pre-trained model give predictions that are coherent with respect to english corpus over which it is pre-trained like **'Conversation', 'Telegraph', 'Author'** etc. The already pre-trained model is pre-trained over BookCorpus(16GB)[43], CC-NEWs(76GB[27]), OpenWebText(38GB)[18], Stories(31GB)[36] etc which explains the incoherent predictions from this model with respect to Fashion e-commerce. We next highlight the important

conclusions and also the future work that can be done to improve ranking and retrieval.

6 CONCLUSION & FUTURE WORK

In this paper we proposed different neural models to learn the representation of queries and products in the latent space. These low dimension query and product representations can be used to solve various problems in the fashion e-commerce domain. We proposed single and multi-layer GRU based baseline models that can be optimized for Triplet Loss using our click-stream data. We also showed how a RoBERTa model after pre-training on Fashion corpus can be again fine-tuned for Triplet loss optimization. Each of the proposed model is trained in two different settings, i.e., first with only Query-Product Data and then augmenting Query-Product data with Product-Product data. In order to compare these models and the representations learned by them, we take the product ranking and retrieval task. Our experiments showed that RoBERTa model that is fine-tuned using only Query-Product data outperformed other proposed models with an MRR of **48%**, MAP of **70.9%** and NDCG of **81.1%** for product ranking task. This model also gave a precision@50 of **4.5%** and Recall@50 of **21.1%** for product retrieval task. The embeddings learned can also be used to train downstream models for ranking optimization. RoBERTa model acts as a strong baseline which can be directly fine-tuned for different ranking tasks and also leverage the transfer learning due to pre-training over Fashion corpus. Directly fine-tuning the RoBERTa model for ranking optimization is an interesting future work. It will be interesting to compare different transformer based architectures for different ranking and relevance tasks.

REFERENCES

[1] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *J. Mach. Learn. Res.* 3, null (March 2003), 993–1022.
[2] Stanley F Chen, Douglas Beeferman, and Roni Rosenfeld. 2018. Evaluation Metrics For Language Models. https://doi.org/10.1184/R1/6605324.v1
[3] Xiang Cheng, Mitchell Bowden, Bhushan Bhange, Priyanka Goyal, Thomas Packer, and Faizan Javed. 2020. An End-to-End Solution for Named Entity Recognition in eCommerce Search.

[4] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR* abs/1412.3555 (2014). arXiv:1412.3555 http://arxiv.org/abs/1412.3555

[5] Brooke Cowan, Sven Zethelius, Brittany Luk, Teodora Baras, Prachi Ukarde, and Daodao Zhang. 2015. Named Entity Recognition in Travel-Related Search Queries. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence* (Austin, Texas) *(AAAI'15)*. AAAI Press, 3935–3941.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805

[7] S. T. Dumais, G. W. Furnas, T. K. Landauer, S. Deerwester, and R. Harshman. 1988. Using Latent Semantic Analysis to Improve Access to Textual Information. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Washington, D.C., USA) *(CHI '88)*. Association for Computing Machinery, New York, NY, USA, 281–285. https://doi.org/10.1145/57167.57214

[8] Jianfeng Gao, Xiaodong He, and Jian-yun Nie. 2010. Clickthrough-Based Translation Models for Web Search: from Word Models to Phrase Models. 1139–1148. https://doi.org/10.1145/1871437.1871582

[9] Jianfeng Gao, Kristina Toutanova, and Wen-tau Yih. 2011. Clickthrough-Based Latent Semantic Models for Web Search. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Beijing, China) *(SIGIR '11)*. Association for Computing Machinery, New York, NY, USA, 675–684. https://doi.org/10.1145/2009916.2010007

[10] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A Deep Relevance Matching Model for Ad-Hoc Retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management* (Indianapolis, Indiana, USA) *(CIKM '16)*. Association for Computing Machinery, New York, NY, USA, 55–64. https://doi.org/10.1145/2983323.2983769

[11] Jiafeng Guo, Gu Xu, Xueqi Cheng, and Hang Li. 2009. Named Entity Recognition in Query. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Boston, MA, USA) *(SIGIR '09)*. Association for Computing Machinery, New York, NY, USA, 267–274. https://doi.org/10.1145/1571941.1571989

[12] Yunlong He, Jiliang Tang, Hua Ouyang, Changsung Kang, Dawei Yin, and Yi Chang. 2016. Learning to Rewrite Queries. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management* (Indianapolis, Indiana, USA) *(CIKM '16)*. Association for Computing Machinery, New York, NY, USA, 1443–1452. https://doi.org/10.1145/2983323.2983835

[13] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[14] Jeremy Howard and Sebastian Ruder. 2018. Fine-tuned Language Models for Text Classification. *CoRR* abs/1801.06146 (2018). arXiv:1801.06146 http://arxiv.org/abs/1801.06146

[15] Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-Based Retrieval in Facebook Search. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery Data Mining* (Virtual Event, CA, USA) *(KDD '20)*. Association for Computing Machinery, New York, NY, USA, 2553–2561. https://doi.org/10.1145/3394486.3403305

[16] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning Deep Structured Semantic Models for Web Search Using Clickthrough Data. In *Proceedings of the 22nd ACM International Conference on Information Knowledge Management* (San Francisco, California, USA) *(CIKM '13)*. Association for Computing Machinery, New York, NY, USA, 2333–2338. https://doi.org/10.1145/2505515.2505665

[17] igraph. 2021. igraph. (2021). https://igraph.org/

[18] David Bourginl Joshua Peterson, Stephan Meylan. 2019. openwebtext. (2019). https://github.com/jcpeterson/openwebtext

[19] Diederik Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations* (12 2014).

[20] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. http://arxiv.org/abs/1412.6980 cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[21] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR* abs/1907.11692 (2019). arXiv:1907.11692 http://arxiv.org/abs/1907.11692

[22] Ilya Loshchilov and Frank Hutter. 2017. Decoupled Weight Decay Regularization. arXiv:1711.05101 [cs.LG]

[23] Subhadeep Maji, Rohan Kumar, Manish Bansal, Kalyani Roy, Mohit Kumar, and Pawan Goyal. 2019. Addressing Vocabulary Gap in E-Commerce Search. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Paris, France) *(SIGIR'19)*. Association for Computing Machinery, New York, NY, USA, 1073–1076. https://doi.org/10.1145/3331184.3331323

[24] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. (2013).

[25] Bhaskar Mitra and Nick Craswell. 2018. An Introduction to Neural Information Retrieval. *Foundations and Trends® in Information Retrieval* 13, 1 (December 2018), 1–126. https://www.microsoft.com/en-us/research/publication/introduction-neural-information-retrieval/

[26] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to Match Using Local and Distributed Representations of Text for Web Search. In *Proceedings of the 26th International Conference on World Wide Web* (Perth, Australia) *(WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1291–1299. https://doi.org/10.1145/3038912.3052579

[27] Sebastian Nagel. 2016. News Dataset. (2016). https://commoncrawl.org/2016/10/news-dataset-available/

[28] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep Sentence Embedding Using Long Short-Term Memory Networks: Analysis and Application to Information Retrieval. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.* 24, 4 (April 2016), 694–707. https://doi.org/10.1109/TASLP.2016.2520371

[29] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[30] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[31] Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.* 3, 4 (April 2009), 333–389. https://doi.org/10.1561/1500000019

[32] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural Machine Translation of Rare Words with Subword Units. *CoRR* abs/1508.07909 (2015). arXiv:1508.07909 http://arxiv.org/abs/1508.07909

[33] Apache Spark. 2021. Spark. (2021). https://spark.apache.org/

[34] Spotify. 2020. Implicit. (2020). https://github.com/spotify/annoy

[35] Zehong Tan, Canran Xu, Mengjie Jiang, Hua Yang, and X. Wu. 2017. Query Rewrite for Null and Low Search Results in eCommerce. In *eCOM@SIGIR*.

[36] Trieu H. Trinh and Quoc V. Le. 2018. A Simple Method for Commonsense Reasoning. *CoRR* abs/1806.02847 (2018). arXiv:1806.02847 http://arxiv.org/abs/1806.02847

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *CoRR* abs/1706.03762 (2017). arXiv:1706.03762 http://arxiv.org/abs/1706.03762

[38] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Association for Computational Linguistics, Brussels, Belgium, 353–355. https://doi.org/10.18653/v1/W18-5446

[39] Musen Wen, Deepak Kumar Vasthimal, Alan Lu, Tian Wang, and Aimin Guo. 2019. Building Large-Scale Deep Learning System for Entity Recognition in E-Commerce Search. In *Proceedings of the 6th IEEE/ACM International Conference on Big Data Computing, Applications and Technologies* (Auckland, New Zealand) *(BDCAT '19)*. Association for Computing Machinery, New York, NY, USA, 149–154. https://doi.org/10.1145/3365109.3368765

[40] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. arXiv:1910.03771 [cs.CL]

[41] Chao-Yuan Wu, Amr Ahmed, Gowtham Ramani Kumar, and Ritendra Datta. 2017. Predicting Latent Structured Intents from Shopping Queries. In *Proceedings of the 26th International Conference on World Wide Web* (Perth, Australia) *(WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1133–1141. https://doi.org/10.1145/3038912.3052704

[42] Han Zhang, Songlin Wang, Kang Zhang, Zhiling Tang, Yunjiang Jiang, Yun Xiao, Weipeng Yan, and Wen-Yun Yang. 2020. Towards Personalized and Semantic Retrieval: An End-to-End Solution for E-Commerce Search via Embedding

Learning. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, China) *(SIGIR '20)*. Association for Computing Machinery, New York, NY, USA, 2407–2416. https://doi.org/10.1145/3397271.3401446

[43] Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books. In *arXiv preprint arXiv:1506.06724*.