Improving Text-based Similar Product Recommendation for Dynamic Product Advertising at Yahoo

Xiao Bai
Yahoo Research
San Jose, CA, USA
xbai@yahooinc.com

Lei Duan
Yahoo
San Jose, CA, USA
leiduan@yahooinc.com

Richard Tang
Yahoo
San Jose, CA, USA
rtang@yahooinc.com

Gaurav Batra
Yahoo
San Jose, CA, USA
gbatra@yahooinc.com

Ritesh Agrawal
Yahoo
San Jose, CA, USA
ritesh.agrawal21@gmail.com

ABSTRACT

Retrieving similar products is a critical functionality required by many e-commerce websites as well as dynamic product advertising systems. Retargeting and Prospecting are two major forms of dynamic product advertising. Typically, after a user interacts with a product on an advertiser website (e.g., Macy's), when the user later visits a website (e.g., yahoo.com) supported by a dynamic product advertising system, the same product may be shown to the user as a Retargeting product ad, while some similar products may be shown to the user as Prospecting product ads on the web page. Similar products can enrich users' ad experience based on users' intent on the Prospecting product ads through which the users interacted. These product ads can also serve as substitutes when Retargeting ad candidates are out of stock. However, it is challenging to retrieve similar products among billions of products in a product catalog efficiently. Deep Siamese models allow efficient retrieval but do not put enough emphasize on key product attributes. To improve the quality of the similar products, we propose to first use a Siamese Transformer-based model to retrieve similar products and then refine them with the attribute "product name" that indicates the type of a product (e.g., running shoes, engagement ring, etc.) for post filtering. We propose a novel product name generation model that fine tunes a pre-trained Transformer-based language model with a sequence to sequence objective. To the best of our knowledge, this is the first work using a generative approach for identifying product attributes. We introduce two applications of the proposed approach for the dynamic product advertising system of Yahoo for Retargeting and Prospecting respectively. Offline evaluation and online A/B testing shows that the proposed approach retrieves high quality similar products, leading to an increase of ad clicks and ad revenue.

CIKM '22, October 17–21, 2022, Atlanta, GA, USA

ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557129

CCS CONCEPTS

• **Information systems** → **Online advertising**; **Recommender systems**; *Online shopping*; • **Computing methodologies** → **Natural language generation**; *Neural networks.*

KEYWORDS

similar product recommendation; dynamic product advertising; semantic retrieval; product name generation

1 INTRODUCTION

Dynamic Product Ads is an emerging format of advertising that creates a personalized ad experience for e-commerce users. Advertisers (i.e., e-commerce websites such as Macy's, Walmart, etc.) simply need to register their desired product catalogs with an ad system. Dynamic ad creatives will be automatically created, which removes the burden of creating individual ads for each item they sell that a traditional advertising system would require. It helps advertisers to scale their ads to users who have expressed interests in their products. Retargeting and Prospecting are two major forms of dynamic product advertising. Typically, after a user interacts with a product on an advertiser website (e.g., Macy's), such as viewing a product, adding a product to the cart, or purchasing a product, when the user later visits a website (e.g., yahoo.com) or a mobile app (e.g., Facebook) supported by a dynamic product advertising system, the same product may be shown to the user as a Retargeting product ad. Figure 1 gives an example of a dynamic product ad appearing in Yahoo's news feed after a user viewed the product at macys.com. Propsecting, in the contract, leverages additional signals, such as users search history, to recommend product ads that users have not directly interacted with.

Similar product recommendation has been playing an important role in e-commerce websites to engage users and drive purchases [38, 42, 44]. It can also be leveraged by a dynamic product advertising system to improve users' ad experience and increase its ad revenue: (i) as shown in Figure 1, a product ad is usually not shown alone. A number of similar products are included in a carousel to allow users to browse and interact with multiple ads; (ii) if a Retargeting product is out of stock, instead of excluding it from ad serving, using similar products as substitutes allows users to continue seeing ads on the products they are interested in; (iii) showing similar products via Prospecting allows advertisers to reach users

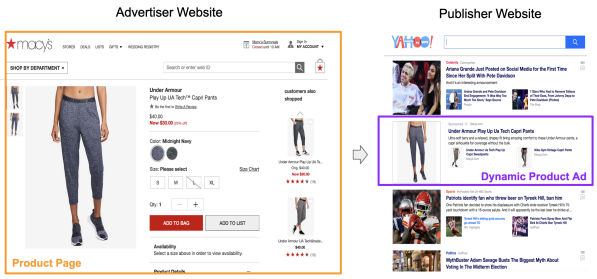

Figure 1: Example of a Dynamic Product Ad.

who may not visit their websites recently to explore their ads based on user interaction with other products.

Collaborative filtering [34] and other user behavior based approaches [6, 39] are widely used for product recommendation. Alternatively, content-based approaches [13, 28] aim to return products with similar textual description [28], image [42], or both [13]. In this work, we focus on textual description of products to make similar product recommendation for dynamic product advertising. This is because content similarity is essential to provide a consistent experience for users who have shown interests in certain products. Moreover, compared to using user behaviors, only using content allows caching the similar products and serving the same products of different users efficiently in the ad auctions. In the end, personalized ad experience will be ensured by the click prediction model [2] that decides the winning ad based on user behaviors. We leave image features for future exploration as well-designed text models can achieve relevant recommendation while saving the huge cost of image processing.

Text-based product retrieval has been extensively studied in the context of product search for e-commerce. The existing approaches mostly follow the general search process to first retrieve a large set of relevant candidates based on matches on terms and product attributes and then use a learning-to-rank model to return the most relevant products [15, 22]. With recent progress in neural information retrieval, more systems start using neural networks to learn query-to-product or product-to-product relevance end-to-end using click logs from product search engines [17, 29, 32, 35, 44]. Among these models, deep Siamese models [17, 32, 35] have attracted great attentions from the industry [29, 44] as they allow efficient retrieval of relevant products from large product catalogs. Product embeddings can be pre-computed and indexed to enable efficient approximate nearest neighbor search [20] using cosine or dot-product similarity.

However, there are a number of challenges when applying deep Siamese models for product recommendation in dynamic product advertising. First, existing models are mostly trained end-to-end for a specific application. As we have several application scenarios and each of them has its own constraints for similar products, training one embedding per application is not cost effective. Second, in an ad system, the recommended products are usually not shown together with the original product since only one ad can win an impression at a given slot on a web page. Therefore, user clicks on ads is not a strong signal that indicates the similarity between two products and cannot be used to train the models as in an e-commerce system. Third, the semantic embedding of products are usually created by considering product title/description as a sequence of tokens without explicitly giving emphasize on important product attribute related tokens such as the type of a product (e.g., running shoes, engagement ring, etc.). Our evaluation shows that without restricting the semantic retrieval to be within the same type of products, it may result in 26% more irrelevant products. Therefore, it is critical to correctly identify product types from product titles/descriptions for accurate post-retrieval filtering.

To address these challenges, we propose to use a two-step approach that can on the one hand improve the quality of the similar products and on the other hand be customized to different application scenarios. The main contributions are summarized as follows:

- We adopt a Transformer-based Siamese network to train an embedding based semantic product retrieval model. We define an editorial guideline that enables fine-grain assessments to allow flexible definition of similarity in downstream applications through post filtering on desired product attributes. This model improves NDCG@5 of our production learning-to-rank model and a baseline deep Siamese model by 5.2% and 4.8% respectively.

- We propose to use the attribute "product name" as a filter to require similar products to be the same type of products, which allows improving the relevance of the similar products retrieved by the embedding based model. Offline evaluation shows a reduction of irrelevant products by 26% after applying the filter at a moderate cost in the number of products having similar product recommendation.

- We formulate the problem of identifying the attribute "product name" as a keyphrase generation task. We propose a novel approach that fine tunes a pre-trained Transformer-based language model with a sequence to sequence objective for product name generation. To the best of our knowledge, this is the first work using a generative approach for identifying product attributes. It improves the state-of-the-art sequence tagging approaches by a large margin (e.g., 13.5% higher F1 score on a set of random products) for generating product names.

- We present two applications of the proposed similar product recommendation approach in the dynamic product advertising system of Yahoo. We show through offline evaluation and online A/B tests that the proposed approach retrieves high quality similar products, which in turn increases the ad revenue by 2.4% and 1.1% for Retargeting and Prospecting respectively.

2 RELATED WORK

In this section, we summarize prior literature in a number of areas that are relevant to this work.

Product Search and Neural Information Retrieval. Our work is closely related to product search which mostly relies on textual information to retrieve relevant products. Santu et al. [22] discussed the challenges of product search and proposed a learning-to-rank model for product re-ranking. An similar approach was also proposed by Goswami et al. [15]. These approaches rely on

heavy feature engineering, especially on features related to product attributes. Recent progress in distributional representation motivates alternative ways for doing product search. Van Gysel et al. [36] projected query into latent vector space by averaging the term embeddings. Bianchi et al. [7] proposed Query2Prod2Vec that built query representation using the representations of the products it refers to encode relevant information beyond query itself. Alternatively, deep Siamese models such as DSSM [17], CLSM [35] and SentenceBERT [32] learns pair-wise similarity end-to-end and results in effective representations for similarity search at large-scale. Embedding-based search has been widely adopted in the industry such as Facebook [16], Amazon [44], and Walmart [29]. In our work, we leverage the SentenceBERT architecture [32] to build our product retrieval model.

Product Attribute Identification. Automatic product attribute identification is important for e-commerce websites to supplement their product catalog with missing attribute values. One approach is to treat each attribute value as a class and apply multi-class classification to identify the attribute value for a given product [14, 25]. This however cannot predict new attribute values. Another approach is to formulate attribute value extraction as a special case of named entity recognition and apply sequence tagging models, such as CRF [24], BiLSTM [43] and BiLSTM-CRF [18, 40, 43] to extract product attributes. However, such extractive approach cannot identify attribute values that do not appear as a continuous text span in the input, making it sub-optimal for the product name generation task we have. Instead, we formulate the product attribute identification problem as a keyphrase generation task and rely on a generative approach to also generate relevant product names that do not directly appear in the product title.

Keyphrase Generation. Keyphrase generation methods aim at predicting phrases relevant to the content of a document. Meng et al. [30] proposed the first generative model for keyphrase generation which were further improved by a number of following works [9, 10]. Yuan et al. [41] proposed to concatenate all the ground-truth keyphrases and train models to generate them as one output sequence. This approach has the flexibility of generating different number of keyphrases for different inputs so we adopt this approach in this work. Ahmad et al. [3] later proposed a model that extracts present keyphrases and generates absent keyphrases simultaneously. Keyphrase generation has previously been used for improving document ranking [31] and session-based product recommendation [27]. Different from the existing works, we leverage pre-trained language models for product name generation and use the generated product names as a filter to improve embedding-based product retrieval.

Pre-trained Language Models. Models based on the Transformer architecture [37], which uses a self-attention mechanism, have driven significant advances on a variety of natural language understanding and generation tasks. Some examples of recent models built on this architecture include BERT [11], RoBERTa [26], DistilBERT [33], and UNILM [5, 12]. These models are pre-trained on a large unsupervised document corpus, and can be fine-tuned on supervised downstream tasks. In this work, we train our product retrieval model and product name generation model by fine tuning DistilBERT and UNILM respectively.

3 SIMILAR PRODUCT RECOMMENDATION

Content-based similar product recommendation relies on content understanding techniques to identify key product attributes, which can either serve as features for product retrieval models or filters for post retrieval processing. In this section, we first describe our embedding-based similar product retrieval approach. We then propose a better product name generation approach to enable accurate post filtering on this critical product attribute.

3.1 System Overview

Given a product, we adopt an embedding-based approach to retrieve its similar products. As shown in Section 4.2, well-learnt product embeddings can achieve better ranking performance than a tree-based model that requires heavy feature engineering. An embedding-based approach also allows efficient retrieval of similar products.

Figure 2 shows the high-level architecture of our similar product retrieval system. Every product in the product catalog is represented as an embedding vector using a semantic retrieval model. For each product that needs to retrieve similar products (referred to as query product), the same model is applied to create its embedding vector. Nearest Neighbor Search is then applied to retrieve, for each query product, the N most similar products based on the cosine similarity between the embedding vectors. Note that N is usually set to a large number (i.e., in the order of hundreds or thousands) so that when applying post filtering for downstream applications, there are enough candidates to identify the application-specific top-k similar products. Depending on the specific application scenario, the definition of similarity may vary. Therefore, post filtering is applied at attribute level (e.g., advertiser, product name, gender, model, etc.) to only retain the top-k similar products whose attribute values match the corresponding values of the query product.

In this section, we focus on describing our model to generate product names while the identification of other product attributes remain the same. Since our embedding-based product retrieval model does not explicitly put emphasis on any product attribute, it may also retrieve related products (e.g., phone cases for phones). Filtering on product names ensures users are only advertised for the products they were explicitly interested in, which is important for advertisers' return on investment.

The similar product retrieval system is generic as different applications can leverage the same embedding-based retrieval process while implementing their application-specific filters. The resulting top-k similar products for each query product and each application is then loaded in a data store to serve ad calls at run time.

3.2 Embedding-based Product Retrieval

We adopt a Transformer-based Siamese network structure (also known as Sentence-BERT [32]) to build product embeddings and measure product similarities using the textual information about products (i.e., title/description). Given the title (or description in case title is not available) of a product, the Siamese network produces token embeddings using a Transformer-based encoder and uses an average pooling operation on the token embeddings to produce a fix sized product embedding. The cosine similarity between

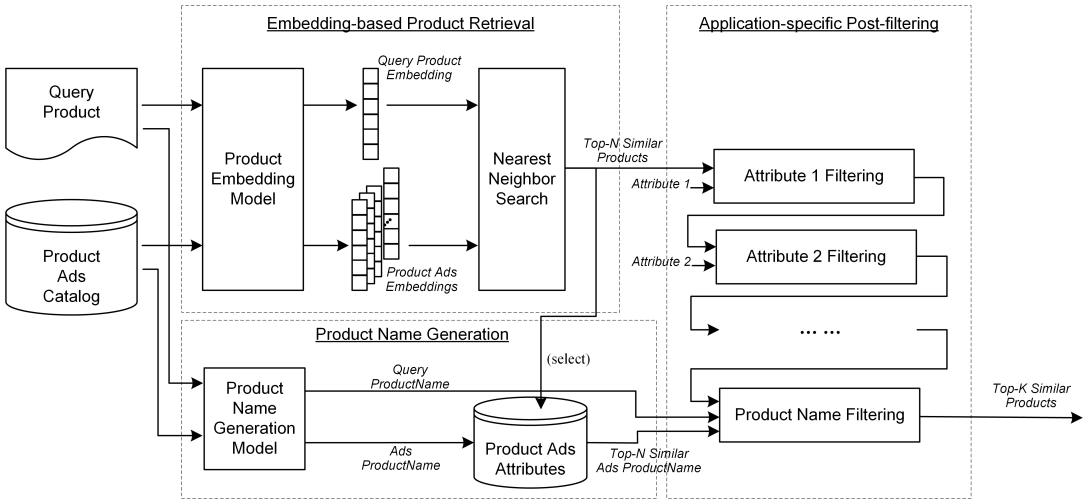

Figure 2: Architecture of our similar product retrieval system.

two product embeddings are used to optimize the mean-squared-error loss.

We fine tune the Siamese network with our in-house, product-to-product similarity dataset that is labeled by professional editors for similar product retrieval. We define a set of guidelines[1] to annotate each product pairs with one of the following label:

- *Excellent.* Two products are exactly the same or only slightly differ in size or color.
- *Good.* Two products are the same kinds of products (e.g., TV) but differ in characteristics like brand, model, material, etc, or have bigger difference in size (e.g., 70" vs. 50").
- *Fair.* Two products belong to the same category but differ in major characteristics (e.g., bike vs. electronic bike) or in usage (e.g., play tent vs. camping tent).
- *Bad.* Two products are different kinds of products (e.g., boots vs. sandals) or mismatch in gender or age groups.

As the same model is used to retrieve similar products for different downstream applications that may have different requirements on similarity, we consider Excellent/Good/Fair labels as positive (+1) and Bad label as negative (-1) while fine tuning the model with the mean-squared-error loss.

Once a model is trained, we apply it on all the products in the ad catalog to compute their embeddings. To reduce storage cost, we use random projection [21] to reduce the dimension of the resulting vectors. Specifically, a d-dimensional vector is projected to an l-dimensional vector ($l < d$) by multiplying it with a random matrix R of dimensions $d \times l$. We follow the approach of Achlioptas et al. [1] to generate the random matrix, which allows approximately preserving the pairwise distance between any two vectors.

The use of a Siamese network structure allows much faster retrieval of the N most similar product embeddings compared to using a default BERT architecture to evaluate the similarity of a pair of products. This makes the similar product retrieval against billions of products feasible. In this work, we use Locality Sensitive Hashing [19] to process batches of query products that come into our ad system periodically.

3.3 Product Name Generation

Product name is one of the most important attributes for post filtering to ensure products different from what users previous interacted with would not been advertised to them. We propose to fine tune a pre-trained Transformer-based language model with a sequence to sequence objective to generate product names from the textual content of a product. In this work, we consider product title (or description if the title is not available) and the Google Product Taxonomy[2] (GPT) category (either provided by advertisers or generated by our in-house category classifier) of a product as input to generate product names.

Our design choice is motivated by the following observations:

(i) Generating product names instead of extracting them allows identifying product names that do not appear as a contiguous text span in the input text. For example, the product name "*engagement ring*" does not appear as a contiguous text span in the product title "*Star K Heart Shape 8mm Created Sapphire Antique Vintage Style Solitaire Engagement Promise Ring*". With an extractive approach, e.g., a sequence tagging model, extracted product names would either be "*ring*" or "*engagement promise ring*". Considering that the product names are used for post filtering, a match on "*ring*" may allow other types of rings to be recommended as similar products while a mismatch on "*engagement promise ring*" may risk to filter out many similar products that are actually "*engagement ring*".

(ii) Using pre-trained Transformer-based models allows deep understanding of long text and maximal utilization of available information for product name generation. For example, given the product description "*A round brilliant-cut diamond is held with an x-prong trellis setting over fourteen-karat gold and pave-set side diamonds. The band is decorated with vintage style milgrain edges which instantly enhances the dazzling diamonds with a dynamic of textures*

[1]The description is indicative. The complete guidelines are enriched with representative examples covering different categories and corner cases.

[2]https://www.google.com/basepages/producttype/taxonomy.en-US.txt

and a captivating appearance fit for royalty. The smart design of this ring features a square shank known as a European shank to help keep the ring from spinning off center as you wear it.…", it is impossible for an extractive model to identify the correct product name "*engagement ring*" as it does not exist in the product description. Transformer-based language models allows understanding relative long text and generating product names that may not exit.

(iii) Product categories often provide additional information and is helpful to disambiguate product titles or descriptions. This piece of information is important for products such as books and DVDs, where the product title is simply the book title or movie title. Without additional information, it is hard for a model to generate the right product name, which may later cause a DVD with a similar title being recommended as a similar product for a book. Product category is also very helpful to identify the core product from ambiguous product titles where multiple product names are mentioned. For example, it is easier for a model to generate "*laptop charger*" as the product name from the title "*Laptop AC Power Adapter Charger for HP Pavilion dv4-2145dx*" and the GPT category "*Electronics > Electronics Accessories > Power > Power Adapters & Chargers*" since the GPT category indicates that "*Power Adapters*" and "*Chargers*" are two separate names.

3.3.1 Problem Definition. Given a product ad, represented by its title (or description) and its corresponding Google Product Taxonomy category, the goal is to generate a set of t product names ($t \geq 1$), where each product name consists of a small number of words that describe a type of the product. Our definition of product name allows a core product word (e.g., t-shirt, ring, fireplace, etc.) along with at most one attribute that is not gender, age, color, brand, size, shape, material (e.g., *long-sleeve* t-shirt, *engagement* ring, *electric* fireplace, etc are valid product names). Attributes like gender and age are excluded from a product name as they are pre-defined attributes that can be directly combined with a product name in the post filtering when retrieving similar products. Allowing other attributes in a product name alleviates the needs of defining new attributes or dealing with new attribute values while new product ads are created consistently. Given that a product name may contain an attribute, a product ad may correspond to multiple product names. For example, for the product "*25 inch glass 3d stereo flame portable electric fireplace with adjustable indoor*", "*fireplace*", "*portable fireplace*", "*electric fireplace*" are all valid product names. Having multiple product names instead of one product name with multiple attributes gives more flexibility while retrieving similar products. Our product name generation model aims at generating all the valid product names given the title and GPT of a product.

3.3.2 Model Architecture. Following the unified pre-trained language models for natural language understanding and generation tasks [5, 12], our model uses the same architecture as BERT_{base}, takes both the product text (i.e., title and GPT) and the ground-truth product names as input, and relies on self-attention masks for the fine tuning.

The input is a sequence of words that is composed of two sentences as in a BERT model. Sentence A is a concatenation of the product title and the corresponding GPT category, separated by a special token "[SEP]". Sentence B is a concatenation of all the ground-truth product names, separated by semicolon ";". Following

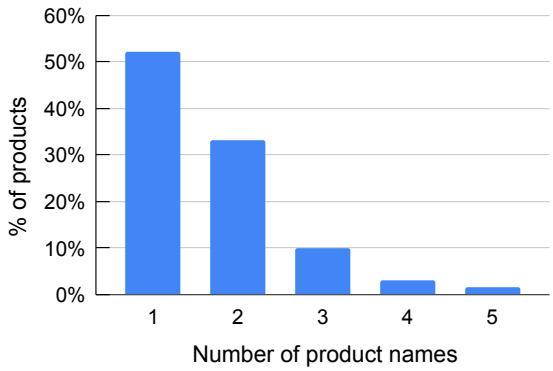

Figure 3: Distribution of the number of editor suggested ground-truth product names per product based on a random sample of our product ad catalog.

the *ONE2SEQ* schema in the keyphrase generation literature [41], we concatenate all the ground-truth product names into one sequence. As shown in Figure 3, 48% of our product ads are relevant to more than one product names, varying from 2 to 5. Using the same pre-defined number for all products to generate product names one by one risk to over generate wrong product names that would in turn hurt similar product retrieval. As in [5, 12], special tokens "[SOS]" (i.e., Start Of Sentence) and "[EOS]" (i.e., End Of Sentence) are added to separate the two sentences. During the product name generation, "[EOS]" also indicates the termination of the generation process. A complete input sequence is in the following format:

$$[SOS]\ title\ [SEP]\ GPT\ [EOS]\ pn_1; pn_2 ...; pn_t\ [EOS],$$

where pn_t denotes the t-th product name for a given product. The vector of each input token follows that of BERT [11] and is represented by the summation of its token embedding, position embedding, and segment embedding.

The input vectors are fed into an L-layer Transformer network. In each Transformer block, multiple self-attention heads are used to aggregate the output vectors of the previous layer. Compared to the standard self-attention, a self-attention mask M is applied to compute the self-attention function A for queries Q, keys K and values V of dimension d_k, as follows:

$$A = softmax\left(\frac{QK^T}{\sqrt{d_k}} + M\right)V,$$

where $M_{ij} = 0$ indicates input tokens x_i and x_j can attend to each other and $M_{ij} = -\infty$ indicates the attention between input tokens x_i and x_j is not allowed. In this work, the tokens in the input title and GPT can attend to each other from both directions. The tokens in the ground-truth product names can only attend to itself and the tokens on its left in the input sequence, which includes all the tokens in the title and GPT, as well as the tokens in the ground-truth product names that qualify.

3.3.3 Fine-tuning for Generation. Following [5], a pre-trained model is fine tuned by first adding a pseudo mask "[Pseudo]" for each tokens in the ground-truth product names (i.e., Sentence B in the input sequence), then applying the self-attention mask described

in Section 3.3.2 to predict the tokens in an autoregressive manner. The "[Pseudo]" token has the same position embedding as its corresponding token in the input sequence. A softmax function is applied to the "[Pseudo]" token's last layer of the hidden states for the predictions. The generation process ends when the token "[EOS]" is generated. The fine-tuning objective is to maximize the likelihood of the entire sequence of the ground-truth product names given the title and GPT. During decoding, we use beam search of size 1 to generate the target tokens one by one in a greedy manner.

4 EXPERIMENTAL EVALUATION

In this section, we report the evaluation of the product name generation model and the embedding-based product retrieval model respectively. We report the evaluation of the end-to-end similar product recommendation for two application scenarios in Section 5.

4.1 Evaluation of the Product Generation Model

4.1.1 Setup. We rely on product names suggested by professional editors to train the product generation model.

Datasets. We collected a dataset of 58K products using a random sample of our product ad catalog. Since the products across different advertisers are highly heterogeneous, we sampled 150 random products for each of the large advertisers. We refer to this dataset as **Random** products. In addition, we collected 3 advertiser-agnostic datasets of 5K random samples to represent easy products, difficult products and ambiguous products respectively: (i) **Easy** products: products for which our production product name extraction model can extract some product names; (ii) **Difficult** products: products for which our production product name extraction model fails to extract any product names (either because input text is too long or because confidence score of the extracted product name does not meet the expected threshold); (iii) **Ambiguous** products: products for which our production product name extraction model extracts multiple product names. All the datasets were evaluated by our in-house editorial team to suggested product names as described in Section 3.3.1. The training data consists of 55K random products, 2.5K difficult products and 2.5K ambiguous products. We use 3K random products, 5K easy-products, 2.5K difficult products and 2.5K ambiguous products as test datasets and report evaluation metrics for each of them.

Baselines. We compare our model with the following baselines:

- **Production-CRF [24]**: Our in-house product name extraction model which is a Conditional Random Fields (CRF) based sequence tagging model. This model was previously trained with a different dataset.
- **BiLSTM-CRF [18]**: A pioneer and state-of-the-art sequence tagging model for named entity recognition tasks. This model can only extract product names that exist in the title and GPT category as a continuous text span so we use BIO tagging to annotate such suggested product names for training. Evaluation of the model is conducted using all the suggested product names.
- **SEG-NET [3]**: An state-of-the-art Transformer-based model that extracts present keyphrases (i.e, those exist in the input text) and generates absent keyphrases (i.e., those do not exist in the input text) simultaneously. We use the entire title (or description) and GPT category without the sentence selector of SEG-NET as their length rarely exceeds the limit. This model does not rely on pre-trained models.

Metrics. Following the literature in keyphrase generation [8], we use F1@M as the evaluation metric. Given a model, we compute the F1@M scores for each test sample by comparing the M predicted product names with all the ground-truth product names suggested by editors, and report the macro average. As F1 score requires an exact match between two product names, which is very strict, we also conduct an editorial evaluation of the generated product names and report the percentage of Good/Fair/Bad product names. A product name is labeled as Good if it describes the core product, Fair if it is still the core product but either contains more than necessary attributes or misses some important attribute that makes it too broad, Bad if it is different from the core product, and NJ if no product name is extracted or generated.

Implementation Details. We fine tune our model using the PyTorch package s2s-ft[3] that fine-tunes pre-trained Transformers for sequence-to-sequence language generation. We use the pre-trained pseudo-masked language model UNILMv2 [5] which was pre-trained for both autoencoding and partially autoregressive objectives. Our model is trained with a maximum source sequence (i.e., title and GPT) length of 128 tokens, and a target sequence (i.e., product names) length of 32 tokens. We used the Adam [23] optimizer. The learning rate was set to 1e-4, with linear warmup over the first 5000 steps and linear decay. Models are trained with batch size 24 for 20 epochs. Experiments were conducted using one NVidia V100 GPU with 32GB of RAM and an Intel Xeon 2.6GHz CPU with a total of 8 cores and 64GB of RAM.

4.1.2 Results. We compare our model, referred to as **PNG** (Product Name Generator), with the sequence tagging and keyphrase generation baselines. We also report some ablation results to justify our design choices.

Generation vs. Extraction. Table 1 shows the F1@M score for our model and the baselines on the four test datasets. We observe that compared to the sequence tagging baseline BiLSTM-CRF, our model PNG achieves significantly higher F1@M score. This is partially because among the ground truth product names, 27% of them do not appear in the title and GPT as a continuous span and thus could not be extracted by BiLSTM-CRF. SEG-NET performs poorly compared to BiLSTM-CRF and PNG. This may be caused by the limited size of training data as the model has 54M parameters but there are only 60K training samples. Using a pre-trained language model allows leveraging existing knowledge and thus leads to much more accurate prediction of the ground truth product names. The larger difference for difficult and ambiguous products further demonstrates the power of our generation approach based on the pre-trained model.

Table 2 reports the editorial evaluation of the product names generated by our model. We only compare our model with the Production-CRF model due to the high cost of human evaluation. We observe that although our model cannot generate all the ground truth product names, the percentage of Bad product names is low

[3] https://github.com/microsoft/unilm/tree/master/s2s-ft

Table 1: F1@M of different models for extracting and/or generating editor suggested ground-truth product names.

Model	Random	Easy	Difficult	Ambiguous
Production-CRF [24]	0.381	0.503	0.000	0.357
BiLSTM-CRF [18]	0.602	0.616	0.375	0.558
SEG-NET [3]	0.436	0.432	0.392	0.416
PNG (Our model)	0.683	0.628	0.583	0.659

for the Random, Easy, and Difficult datasets. The percentage of Bad is higher on the Ambiguous dataset but it is much lower than the production model.

Impact of Product Category. Table 3 compares our model for using and not using GPT as part of the input in addition to product title or description. We observe that using GPT consistently improves the generated product names. The improvement varies between 1.1% and 2.2%, and is higher for more difficult datasets.

Impact of Pre-trained Models. We use UNILMv2 as the pre-trained model and fine tune it for the product name generation task. Table 4 compares the F1@M scores of models using BERT_{base} or RoBERTa_{base} as the pre-trained model. We observe that using UNILMv2 leads to slightly better F1@M than using BERT. This may stem from the fact that UNILMv2 is pre-trained with both masked language model and sequence to sequence language model objectives. RoBERTa_{base} has comparable performance on the Random test dataset which is similar to the training dataset, but does not generalize well on other test datasets.

4.2 Evaluation of the Product Retrieval Model

4.2.1 Setup. We fine tune a pre-trained distilled BERT model using the Siamese network with a human annotated product similarity dataset.

Datasets. We collected 500K product-product similarity data labeled by professional editors and split them into training (90%) and development (10%) datasets. The query products were randomly sampled from users' search/click/purchase history, and each of them have 5 random products matched by our production product retrieval system. We built 3 test datasets that represent the **Head**, **Torso** and **Tail** query products based on the frequency they appear in users' historical behaviors. Each dataset contains 2K randomly sample query products along with 5 random products matched by our production retrieval system.

Baselines. We compare our model, referred to as **EPR** (Embedding-based Product Retrieval), with the following two baselines:

- **Production-GBDT**: This is the product ranking model we used in our production system for other applications. It is a Gradient Boosted Decision Tree (GBDT) model trained with binary features indicating whether there is a match on selected product attributes as well as textual similarity features similar to those discussed in [4].
- **CLSM [35]**: This is a convolution neural network that learns a low-dimensional semantic vector representations of word sequence for information retrieval tasks with cosine similarity. We trained this model with our product-product similarity dataset for 128 dimension vectors as described in the original paper [35].

Metrics. We report NDCG@5 as there are 5 retrieved products per query product in the test datasets.

Implementation Details. We fine tuned a distilled BERT-base model that was previously trained with 500K training samples from MS MARCO's passage retrieval dataset[4] for cosine similarity, using the Open Source implementation of Sentence Transformers[5]. Models were trained with batch size 16 for at most 10 epochs with early stopping using the development dataset. The embedding size is reduced from 768 to 128 using a random production [1] for computing the similarity. Experiments were conducted using one NVidia V100 GPU with 32GB of RAM and one Intel Xeon 2.6GHz CPU, with a total of 8 cores and 64GB of RAM.

4.2.2 Results. Table 5 shows the NDCG@5 of our model and the two baseline models on the test datasets of head, torso and tail query products respectively. We observe that our model fine-tuned using the Siamese Transformer architecture achieves higher NDCG@5 for all datasets. The difference is more significant for tail products. We also observe that using 128 dimension vectors instead of 768 dimension vectors has little impact on NDCG@5 scores (0.05% difference for tail products) while significantly reduces the storage and computation cost for similar product retrieval. In addition, our embedding based approach allows efficient approximate nearest neighbor search using Locality Sensitive Hashing when retrieving the top-k similar products while GBDT can only be used for re-ranking as it requires computing pair-wise features.

5 SIMILAR PRODUCT RECOMMENDATION AND ONLINE EVALUATION

Our embedding-based product retrieval model and product name generation model are developed to improve similar product recommendation for dynamic product advertising at Yahoo. As described in Section 3.1, the product names are used as a filter to improve the quality of the similar products. In this section, we first discuss the impact of applying the filter of product names. We then describe two application scenarios of these models and show how they drive better user experience and ad revenue.

5.1 Impact of Product Name Filtering

We propose in this work to use the product attribute "product name" as a post filter to improve embedding-based semantic product retrieval. Table 6 shows how this filter impacts the amount of query products that can get at least one similar product (referred to as *coverage*) and the quality of the similar products (according to labels assigned by human annotators). We compare the top-5 most similar products returned by the proposed product retrieval model using Locality Sensitive Hashing (EPR), after filtering products using the product names produced by our production CRF product name extraction model (EPR+Production) and the new product name generation model (EPR+PNG) respectively, for a random sample of 1000 query products. The retrieval is done without restriction on campaign type or advertisers. We observe that applying the product name filter using our PNG model increases the percentage of Excellent and Good products and reduces the percentage of Bad

[4]https://www.sbert.net/docs/pretrained-models/msmarco-v3.html
[5]https://www.sbert.net

Table 2: Editorial evaluation of the model extracted or generated product names.

Dataset	Random		Easy		Difficult		Ambiguous	
Label	Production (CRF)	PNG	Production (CRF)	PNG	Production (CRF)	PNG	Production (CRF)	PNG
Good	42.8%	58.8%	55.6%	62.2%	0.0%	63.6%	36.9%	40.0%
Fair	27.4%	39.4%	34.3%	34.1%	0.0%	32.9%	28.3%	46.7%
Bad	9.2%	1.8%	10.1%	3.7%	0.0%	3.5%	34.7%	13.3%
NJ	20.5%	0.0%	0.0%	0.0%	100.0%	0.0%	0.0%	0.0%

Table 3: Impact of product category (GPT) as part of the input text on our model for the metric F1@M.

Model	Random	Easy	Difficult	Ambiguous
PNG with GPT	0.683	0.628	0.583	0.659
PNG w/o GPT	0.673	0.621	0.573	0.645

Table 4: Impact of pre-trained language models on our model for the metric F1@M.

Pre-traind Model	Random	Easy	Difficult	Ambiguous
UNILMv2 (Our model)	0.683	0.628	0.583	0.659
BERT_{base}	0.672	0.629	0.572	0.654
RoBERTa_{base}	0.676	0.552	0.498	0.594

Table 5: NDCG@5 of different product retrieval models.

Model	Embedding Size	Head	Torso	Tail
Production (GBDT)	NA	0.8782	0.7713	0.7366
CLSM [35]	128	0.8756	0.7754	0.7388
EPR (Our model)	128	0.8837	0.7843	0.7746
EPR (Our model)	768	0.8836	0.7845	0.7750

Table 6: Impact of product name filtering on coverage and quality of similar products.

Approach	Coverage	Excellent	Good	Fair	Bad
EPR	87.8%	7.4%	85.0%	2.4%	5.3%
EPR+Production	53.4%	8.8%	86.9%	1.3%	3.1%
EPR+PNG	81.0%	8.1%	86.0%	2.0%	3.9%

products by 26%. This gain in quality is at the cost of retrieving similar products for fewer query products. Yet, using our PNG model significantly improves the coverage by 52% compared to using the Production CRF model. As we will see in next two sections, our model represents a good trade-off between coverage and quality, and thus leads to higher CTR and revenue.

5.2 Similar Products as Out-of-stock Substitutes for Retargeting

Retargeting is the primary way of showing dynamic product ads to users in the industry. The key idea behind Retargeting is that after a user clicks a product (or adds a product into their shopping cart) on an advertiser website (e.g., Macy's), when the user later visits a website (e.g., yahoo.com) supported by an dynamic product advertising system, the same product is shown to the user as a Retargeting product ad on the web page. However, at the time a user returns to the web site, some products that they previously clicked may become out of stock. To avoid hurting user experience by showing unavailable products, an ad system usually validates the availability of the products a user previously clicked and only allows the products that are currently in stock to participate in the ad auction. This significantly reduces the number of Retargeting product ads that are eligible for an ad auction, and thus negatively impacts ad revenue. This may also create an unsatisfactory experience for users who liked the product that became out of stock without any alternative recommendations. Our analysis shows that about 7% of products previously clicked by users can be out of stock at the time of ad serving. Depending on advertisers, the ratio of out of stock products can be as high as 40%. Therefore, using a similar product to replace the out of stock product for Retargeting allows more eligible products to participate in an ad auction, which is important for advertisers to for users to explore relevant ads, for advertisers to retain advertising opportunities, and for ad systems to increase revenue.

To generate the substitute products for out of stock products, we first use our product ad catalog to identify out of stock products among all the products eligible for Retargeting during a time window based on signals received from advertisers' websites. We generate product embedding and product names for each of them using our models described in Section 3. We also compute product embeddings and product names for the products in the active Retargeting campaigns. We then apply Locality Sensitive Hashing on the product embeddings to retrieve for each out of stock product, top-1000 similar Retargeting products. We limit the retrieval to only return products with cosine similarity larger than 0.8. In addition to the default post filtering on product attributes, we apply the filter on product names so that only products with at least one common product name with the out of stock products are kept. We also apply an application specific filter on advertiser such that only the products belonging to the same advertiser as the out of stock product can be a substitute. Finally, the product with the highest similarity is used as the substitute of the out of stock product. The out of stock product to substitute product mapping is loaded into an in-memory cache for serving ad calls at run time.

Table 7 shows the quality of the top-1 similar product for a random sample of out of stock products. We observe that 20.9% substitute products are the same as the out of stock products and 69.4% are similar alternatives. Only 5.2% are irrelevant products. Among the Bad substitutes, 3.5% are caused by incorrect product names but this is much lower than our production CRF product name extraction model would have as shown in Table 2.

To assess the value of the substitute products, we evaluate them in an online A/B test. The control bucket blocks out of stock products from ad auctions. The test bucket replaces an out of stock

Table 7: Quality evaluation of top-1 similar product as substitutes of out of stock products for Retargeting.

Label	Excellent	Good	Fair	Bad
Label distribution	20.9%	69.4%	3.0%	5.2%

Table 8: Online A/B test results for similar products as out of stock substitutes for Retargeting. Relative change of test bucket w.r.t. control bucket is reported.

Impression	CPM	CTR
+3.8%	+2.4%	+0.69%

product with the most similar product as described above. We ran the buckets with 5% live traffic for one week and summarize the key metrics in Table 8. As out of stock products participate in the ad auction through their substitutes, we observe 3.8% more impressions in the test bucket, and 0.69% higher CTR (Click-Through-Rate) which implies the substitute ads are meaningful for users. We also observe 2.4% higher CPM (Cost-Per-Mille impressions) which means higher ad revenue. Therefore, this new feature was launched in production for our dynamic product advertising system.

5.3 Similar Products as Recommendation for Prospecting

Prospecting, also known as Broad Audience, is another important way of approaching users with dynamic product ads. Different from Retargeting where users are typically shown the same products they previously interacted with in the advertiser websites, with Prospecting, users may also see products that they did not directly interacted with, but expressed their interests in other means implicitly. An important goal for advertisers to do Prospecting is to introduce their brands to potential new users. Therefore, recommending products that are similar to the products users are explicitly interested in from different advertisers becomes an appealing solution to enhance Prospecting. Our current system mainly relies on a collaborative filtering approach to recommend similar products. We explore the content based approach for recommending similar products.

We aim at generating recommendations for every product eligible for Retargeting. The recommended products are retrieved from active Prospecting campaigns and are served as Prospecting ads. To this end, we generate product embedding and product names for all the Retargeting candidate products during a time window and use Locality Sensitive Hashing to retrieve the top-1000 similar products that are active in Prospecting campaigns. We limit the retrieval to only return products with cosine similarity larger than 0.8. In addition to the default post filtering on product attributes, we apply the filter on product names so that only products with at least one common product name with the original Retargeting product are kept. Moreover, we apply an application-specific filter on product title so that the same product (title) from a different advertiser will not be shown as a direct competitor of the original Retargeting product. Finally, three products with the highest similarity scores are used as the recommendation for each Retargeting product, and are loaded into an in-memory cache for serving ad calls at run time.

Table 9: Quality evaluation of top-3 similar products as recommendation for Prospecting.

Label	Excellent	Good	Fair	Bad
Label distribution	5.0%	86.3%	1.7%	7.0%

Table 10: Online A/B test results for similar product recommendation for Prospecting. Relative change of test bucket w.r.t. control bucket is reported.

Impression	CPM	CTR
+2.2%	+1.1%	+0.0%

Table 9 shows the quality of the top-3 similar products for a random sample of Retargeting products. Since we explicitly filter out the products with same titles, there are only 5.0% similar products are labeled as Excellent, i.e, being the same products with slight difference in size or color. The majority of the similar products are Good alternatives. 7% of the similar products are labeled as Bad where 3.1% are caused by incorrectly generated product names.

To assess the impact of these similar products for Prospecting, we evaluate them in an online A/B test. The control bucket uses existing algorithms for identifying candidate products for Prospecting. The test bucket adds 3 similarity products for each Retargeting candidate as Prospecting candidates to participate in the ad auction. We ran the buckets with 10% live traffic for two weeks and summarize the key metrics in Table 10. We observe 2.2% more impressions in the test bucket with 1.1% higher CPM, which indicates higher ad revenue. The increase in impressions and revenue are driven by the additional Prospecting ads which also leads to a traffic shift from Retargeting to Prospecting. Considering that the industry trend is to reduce tracking of users' historical behaviors for better privacy protection, exploring similar products across different advertisers as Prospecting in addition to Retargeting that purely relies on user tracking is beneficial for our ad system to provide personalized advertising experience for users.

6 CONCLUSION

In this work, we studied the problem of text-based similar product recommendation for dynamic product advertising. We proposed to use a generative model to identify product names from product text and use them as a filter to further improve the output of our embedding based product retrieval model. This approach has been launched into the dynamic product advertising system of Yahoo to generate substitutes for out of stock Retargeting products and is under evaluation to discover similar products for Prospecting. As part of the future work, we plan to evaluate this approach in an online A/B test to serve similar products in an ad carousel, which is expected to help our system to explore new products more efficiently.

ACKNOWLEDGMENTS

We would like to thank all the anonymous editors from the Yahoo editorial team, led by Manju Hari Hara Prasad, Anup Patnaik and Karen Wu, to evaluate the datasets used in this work. We would also like to thank Evgeny Segal, Anna Farberman and Omer Duvdevany for performing online A/B tests in the production system.

REFERENCES

[1] Dimitris Achlioptas. 2001. Database-Friendly Random Projections. In *Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS '01)*. 274–281.

[2] Michal Aharon, Natalie Aizenberg, Edward Bortnikov, Ronny Lempel, Roi Adadi, Tomer Benyamini, Liron Levin, Ran Roth, and Ohad Serfaty. 2013. OFF-Set: One-Pass Factorization of Feature Sets for Online Recommendation in Persistent Cold Start Settings. In *Proceedings of the 7th ACM Conference on Recommender Systems (RecSys '13)*. 375–378.

[3] Wasi Ahmad, Xiao Bai, Soomin Lee, and Kai-Wei Chang. 2021. Select, Extract and Generate: Neural Keyphrase Generation with Layer-wise Coverage Attention. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL'20)*. 1389–1404.

[4] Luca Aiello, Ioannis Arapakis, Ricardo Baeza-Yates, Xiao Bai, Nicola Barbieri, Amin Mantrach, and Fabrizio Silvestri. 2016. The Role of Relevance in Sponsored Search. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM '16)*. 185–194.

[5] Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Songhao Piao, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2020. UNILMv2: Pseudo-Masked Language Models for Unified Language Model Pre-Training. In *Proceedings of the 37th International Conference on Machine Learning (ICML'20)*. 642–652.

[6] Rahul Bhagat, Srevatsan Muralidharan, Alex Lobzhanidze, and Shankar Vishwanath. 2018. Buy It Again: Modeling Repeat Purchase Recommendations. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '18)*. 62–70.

[7] Federico Bianchi, Jacopo Tagliabue, and Bingqing Yu. 2021. Query2Prod2Vec: Grounded Word Embeddings for eCommerce. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Papers*. 154–162.

[8] Hou Pong Chan, Wang Chen, Lu Wang, and Irwin King. 2019. Neural Keyphrase Generation via Reinforcement Learning with Adaptive Rewards. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL'19)*. 2163–2174.

[9] Jun Chen, Xiaoming Zhang, Yu Wu, Zhao Yan, and Zhoujun Li. 2018. Keyphrase Generation with Correlation Constraints. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP'18)*. 4057–4066.

[10] Wang Chen, Yifan Gao, Jiani Zhang, Irwin King, and Michael R. Lyu. 2019. Title-Guided Encoding for Keyphrase Generation. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI'19)*. 6268–6275.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 4171–4186.

[12] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified Language Model Pre-training for Natural Language Understanding and Generation. In *Advances in Neural Information Processing Systems*, Vol. 32.

[13] Dehong Gao, Linbo Jin, Ben Chen, Minghui Qiu, Peng Li, Yi Wei, Yi Hu, and Hao Wang. 2020. FashionBERT: Text and Image Matching with Adaptive Loss for Cross-Modal Retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2251–2260.

[14] Rayid Ghani, Katharina Probst, Yan Liu, Marko Krema, and Andrew Fano. 2006. Text Mining for Product Attribute Extraction. *SIGKDD Explor. Newsl.* 8, 1 (2006), 41–48.

[15] Anjan Goswami, Chengxiang Zhai, and Prasant Mohapatra. 2018. Learning to Rank and Discover for E-Commerce Search. In *Machine Learning and Data Mining in Pattern Recognition*. 331–346.

[16] 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 (KDD '20)*. 2553–2561.

[17] 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 (CIKM '13)*. 2333–2338.

[18] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging. arXiv:1907.11692

[19] Omid Jafari, Preeti Maurya, Parth Nagarkar, Khandker Mushfiqul Islam, and Chidambaram Crushev. 2021. A Survey on Locality Sensitive Hashing Algorithms and their Applications. *CoRR* abs/2102.08942 (2021).

[20] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-Scale Similarity Search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2021), 535–547.

[21] William Johnson and Joram Lindenstrauss. 1984. Extensions of Lipschitz maps into a Hilbert space. *Contemp. Math.* 26 (01 1984), 189–206.

[22] Shubhra Kanti Karmaker Santu, Parikshit Sondhi, and ChengXiang Zhai. 2017. On Application of Learning to Rank for E-Commerce Search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '17)*. 475–484.

[23] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR'15)*.

[24] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML '01)*. 282–289.

[25] Xiao Ling and Daniel Weld. 2021. Fine-Grained Entity Recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI '21)*. 94–100.

[26] 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. arXiv:cs.CL/1907.11692

[27] Yuanxing Liu, Zhaochun Ren, Wei-Nan Zhang, Wanxiang Che, Ting Liu, and Dawei Yin. 2020. Keywords Generation Improves E-Commerce Session-Based Recommendation. In *Proceedings of The Web Conference 2020 (WWW '20)*. 1604–1614.

[28] Szymon Łukasik, Andrzej Michałowski, Piotr A. Kowalski, and Amir H. Gandomi. 2021. Text-Based Product Matching with Incomplete and Inconsistent Items Descriptions. In *Computational Science – ICCS 2021*. 92–103.

[29] Alessandro Magnani, Feng Liu, Min Xie, and Somnath Banerjee. 2019. Neural Product Retrieval at Walmart.Com. In *Companion Proceedings of The 2019 World Wide Web Conference (WWW '19)*. 367–372.

[30] Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep Keyphrase Generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL'21)*. 582–592.

[31] Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document Ranking with a Pretrained Sequence-to-Sequence Model. In *Findings of the Association for Computational Linguistics: EMNLP 2020 (EMNLP'20)*. 708–718.

[32] 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 (EMNLP'19)*. 3982–3992.

[33] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In *5th Workshop on Energy Efficient Machine Learning and Cognitive Computing @ NeurIPS 2019*.

[34] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-Based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web (WWW '01)*. 285–295.

[35] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM '14)*. 101–110.

[36] Christophe Van Gysel, Maarten de Rijke, and Evangelos Kanoulas. 2016. Learning Latent Vector Spaces for Product Search. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM '16)*. 165–174.

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30*. 5998–6008.

[38] Jian Wang, Badrul Sarwar, and Neel Sundaresan. 2011. Utilizing Related Products for Post-Purchase Recommendation in e-Commerce. In *Proceedings of the Fifth ACM Conference on Recommender Systems (RecSys '11)*. 329–332.

[39] Chen Wu and Ming Yan. 2017. Session-aware Information Embedding for E-commerce Product Recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM '17)*. 2379–2382.

[40] Huimin Xu, Wenting Wang, Xin Mao, Xinyu Jiang, and Man Lan. 2019. Scaling up Open Tagging from Tens to Thousands: Comprehension Empowered Attribute Value Extraction from Product Title. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL'19)*. 5214–5223.

[41] Xingdi Yuan, Tong Wang, Rui Meng, Khushboo Thaker, Peter Brusilovsky, Daqing He, and Adam Trischler. 2020. One Size Does Not Fit All: Generating and Evaluating Variable Number of Keyphrases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL'20)*. 7961–7975.

[42] Yanhao Zhang, Pan Pan, Yun Zheng, Kang Zhao, Yingya Zhang, Xiaofeng Ren, and Rong Jin. 2018. Visual Search at Alibaba. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '18)*. 993–1001.

[43] Guineng Zheng, Subhabrata Mukherjee, Xin Luna Dong, and Feifei Li. 2018. OpenTag: Open Attribute Value Extraction from Product Profiles. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '18)*. 1049–1058.

[44] Zhen Zuo, Lixi Wang, Michinari Momma, Wenbo Wang, Yikai Ni, Jianfeng Lin, and Yi Sun. 2020. A flexible large-scale similar product identification system in e-commerce. In *KDD 2020 Workshop on Industrial Recommendation*.