Semantic Product Search for Matching Structured Product Catalogs in E-Commerce

Jason Ingyu Choi
Emory University
in.gyu.choi@emory.edu

Surya Kallumadi
Home Depot
surya_kallumadi@homedepot.com

Bhaskar Mitra
Microsoft
bhaskar.mitra@microsoft.com

Eugene Agichtein
Emory University
eugene.agichtein@emory.edu

Faizan Javed
Home Depot
faizan_javed@homedepot.com

ABSTRACT

Retrieving all semantically relevant products from the product catalog is an important problem in E-commerce. Compared to web documents, product catalogs are more structured and sparse due to multi-instance fields that encode heterogeneous aspects of products (e.g. brand name and product dimensions). In this paper, we propose a new semantic product search algorithm that learns to represent and aggregate multi-instance fields into a document representation using state of the art transformers as encoders. Our experiments investigate two aspects of the proposed approach: (1) effectiveness of field representations and structured matching; (2) effectiveness of adding lexical features to semantic search. After training our models using user click logs from a well-known E-commerce platform, we show that our results provide useful insights for improving product search. Lastly, we present a detailed error analysis to show which types of queries benefited the most by fielded representations and structured matching.

1 INTRODUCTION

E-commerce platforms and web search engines share similar goals, which is to display or recommend relevant items (e.g. web documents or products) for a search [9]. However, because product catalogs are more heterogeneous and structured compared to web documents, encoding such documents presents new challenges. For instance, products can have several structured and unstructured fields such as title, description and metadata. Each field can be further divided into multiple instances (e.g. long description, short description, dimensions, units), which vary significantly across product domains [18]. To be successful, models first need to understand the semantics of each field, and utilize fielded representations to perform structured matching between query and document.

Before displaying ranked products to users, typical e-commerce platforms undergo two phases: (1) candidate generation; (2) candidate re-ranking [9]. This work focuses on the candidate generation phase. Our goal is to retrieve all relevant products, which is equivalent to maximizing the recall. To accomplish this, we propose a structured matching module (SMM) that leverages multiple fielded representations to learn a structured matching function. Our method has two advantages. First, SMM utilizes a bottom-up approach to encode instances in each field and transforms these partial representations into an overall document vector. This is more effective when compared to encoding long, heterogeneous documents in one shot. Second, because the encoder is trained to learn query and document representations separately, production systems can generate candidates faster by pre-computing product embeddings.

For evaluation, we trained and validated our model using two data sources in the home-improvement domain: (1) internal user click logs; (2) product search relevance (PSR) dataset. The click logs dataset is sub-sampled from our private click logs. For reproducibility, we selected PSR dataset, which is in the same e-commerce domain but has human-annotated relevance labels. After evaluating our model on these two datasets, we show that incorporating SMM after pre-trained transformer improves the overall matching performance. Our contributions are:

- A new structured matching module (SMM) that extends Siamese transformer structures by incorporating fielded representations and lexical signals.
- A large-scale empirical evaluation, demonstrating promising performance of SMM for semantic product search in e-commerce.

Next, we review related work to place our contributions in context.

2 RELATED WORK

Traditional ranking approaches combined probabilistic signals and lexical features such as term frequency and document length to rank documents according to their relevance [11, 12]. However, the main challenges of lexical ranking models are on their coverage since these models cannot make any semantic connection. Distributional semantics [6] was introduced to solve this challenge by training a language model using latent features to obtain a vector representation for each word.

The earliest models were representation-based models that encode two texts into a fixed-size vector, and computes the similarities by taking cosine or dot product between two vectors [4, 5]. However, these methods were limited because the network did not consider any interaction between the two inputs. To address this problem, interaction-based models were introduced to capture more complex patterns between query and document matrix [16]. To combine interactions with lexical signals, hybrid models were proposed and validated the effectiveness of lexical matching when combined with semantic matching [2, 8]. Several approaches evaluated the potentials of incorporating structured information to a document ranking task [11, 18].

Recently proposed text matching approaches adopt transformer-based encoders to benefit from their rich semantics. However, since

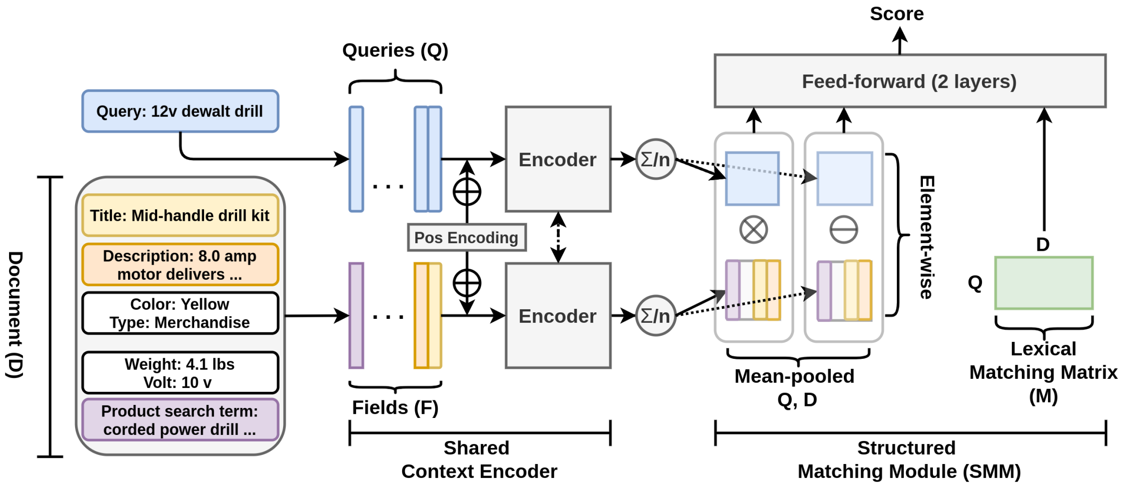
Score
Feed-forward (2 layers)
Queries (Q)
Query: 12v dewalt drill
Encoder
Σ/n
Pos Encoding
Encoder
Σ/n
Element-wise
Document (D)
Title: Mid-handle drill kit
Description: 8.0 amp motor delivers ...
Color: Yellow Type: Merchandise
Weight: 4.1 lbs Volt: 10 v
Product search term: corded power drill ...
Fields (F)
Shared Context Encoder
Mean-pooled Q, D
D
Q
Lexical Matching Matrix (M)
Structured Matching Module (SMM)

Figure 1: Overall architecture of our structured semantic matching model.

transformers use cross-encoder structures, there have been several attempts to decouple the two inputs to learn semantically meaningful vectors for scalable comparisons [3, 10, 17]. In our work, we maintain the Siamese structure to benefit from these advantages.

3 PROPOSED APPROACH

In this section, we formally explain the problem and present an overview of our proposed architecture, described in Figure 1. Our model is composed of two sub-modules: (1) context encoders; (2) structured matching modules.

Problem and product catalog description. Given a user query Q and product D, we define product fields $F = \{F_1 ... F_n\}$ and field instances $I = \{F_{1i} ... F_{nk}\}$. All products share the same fields as described in Table 1. Instances are attributes of each field that represent certain aspects of a product. For all products, the number of fields n is fixed to 7 while the number of instances k varies between fields and products. The last field titled Product search terms is collected from previously issued queries that resulted in a click on this product. We sampled these queries from the training data, and selected the top 10 unique queries based on frequency. In addition, we highlight that Title field does not have other instances except the product title.

Fields	Instances
Title	Product title
Description	Types of descriptions (short, long)
Product category	Category
Metadata	Additional descriptions (color, texture)
Brand	Brand information (brand name)
Numeric	Numeric values (height, width)
Product search terms	Top-10 queries from click logs

Table 1: Seven fields selected for representing products.

For this study, we represented each field F_i as a concatenated sequence of tokens from its instance group. Based on the input pairs $(Q, D = [F_1 ... F_n])$, the goal is to train a function f that maps this input into a probability score s, indicating the likelihood of relevance.

$$f(Q, D) \in [0, 1] \tag{1}$$

Context encoders. When encoding query and fields, we used a pre-trained DistilBERT [14], which is a distilled version of BERT that retains 97% of the original performance. We chose this model over the original BERT because DistilBERT required less GPU memory and has faster convergence. The weights of transformer blocks are initialized from a pre-trained model, and are tuned using our data. Our encoder used 6 hidden layers with 12 attention heads.

To obtain the sentence embeddings, we used the mean pooling strategy that takes the mean of hidden states for each sequence position from the final transformer block. This method has shown to be effective compared to directly using the vector for [CLS] token or max-pooling, and is similar to extracting bag-of-words representations where words have interacted with others through multiple self-attention layers [10, 17]. These pooled vectors are stacked to form query and document matrices $\hat{Q}$ and $\hat{D}$.

Structured matching modules (SMM). The goal of SMM is to extract matching signals from $\hat{Q}$ and $\hat{D}$. Based on previous literature, we applied element-wise multiplication and subtraction to these matrices to generate features [10, 18]. We emphasize that because $\hat{D}$ is a stack of field vectors, element-wise operations allow different field vectors to interact with query vectors. This is equivalent to performing pairwise comparisons between query and field in a latent space. In addition, we generated a binary matrix M from the query and document tokens to encode lexical matches. These three outputs are concatenated as following:

$$[|\hat{Q} - \hat{D}|; \hat{Q} \circ \hat{D}; M] \tag{2}$$

These outputs are then fed to two layers of non-linear transformations. We used ReLU activation and dropout between these layers. Binary cross-entropy was minimized to predict relevant (1) and non-relevant (0) labels using Adam with 1e-4 learning rate and 16 batch size. Learning rate was warmed-up over the first 10% of our training data, and linearly decayed with 0.01 decay rate. 0.1 dropout probability was used on all transformer layers to improve regularization. We used 0.5 dropout for our feed-forward layers.

4 EXPERIMENTAL SETUP

In this section, we present an overview of our data collection process, followed by statistics of our training, validation and two test datasets. The labels of the first test dataset were generated from clicks while the second test dataset was manually annotated from three human workers.

Training dataset from internal click logs. For this study, we used the subset of click logs from a popular E-commerce platform. For each query, we collected the top 100 (product, clicks) pairs, which are ranked by a production system. We define each entry in our dataset as (query, product, clicks) triple. In total, there are 11,650,964 entries, 1,675,630 unique queries and 3,372,715 unique products. Since our goal is to retrieve all relevant products and not necessarily rank them, we defined a click threshold r to distinguish positive and negative pairs. After manual evaluation, $r = 5$ was used to convert click values into a binary label. We filtered out queries that do not contain any relevant product, has a length smaller than 3 characters, or contain numeric values only. For products, we filtered out items that contain too few attributes or do not contain important attributes such as Title and Description. Lastly, we reserved 5,000 unique queries for a validation set and another 5,000 unique queries for a test set.

	Training	Validation	Test
Entry	11,650,964	227,276	219,728
Unique query	258,666	5,000	5,000
Relevant	51.8%	50.7%	52.2%
Not relevant	48.2%	49.3%	47.8%
Unique products	384,506		

Table 2: Click logs training, validation and test statistics.

Human-annotated test set from Kaggle. In addition to the click logs dataset, we used a publicly available E-commerce dataset titled Product Search Relevance (PSR) dataset, which was released in 2016 as a Kaggle competition by an e-commerce site in home-improvement domain[1]. The goal is to perform a more robust evaluation since labels from the first dataset are heuristically generated from clicks. Instead, the labels of PSR dataset is obtained from three human workers where 1 indicates irrelevant, 2 as partially relevant and 3 as perfect match. Each (Q, D) pair was given to at least three workers, and the final scores were averaged. We observed that this dataset lacks negative samples since 83.9% of the pairs are labeled at least partially relevant (>=2). After rounding off the decimal values, we reduced the labels into three discrete labels $\in [1, 2, 3]$. $r = 2.5$ was used to convert labels into binary labels.

In addition to these ground-truth labels, we recruited one domain expert and asked to annotate 1,000 randomly sampled queries into 6 classes of Brand/Collection, Color/Finish, Unit, Material, Model, and Typo for error analysis. Each class represents whether the query terms contain important keywords that identify specific classes. Given the nature of multi-intent queries, it is possible to have multiple classes (e.g. Black Samsung TV) per each sample. Our annotated results show that 17.4%, 4.7%, 21.8%, 5.7%, 3.7%, and 11.7% queries (with duplicates) belong to each class respectively.

Baseline models and metrics. We chose baselines models in two groups: (1) lexical baselines; (2) neural baselines. For lexical baselines, we will report the performance of BM25 and BM25F rankers, which are tuned on our validation set. Please note that we are not using top@k retrieved results from lexical models to do re-ranking task, but the scores from these models are directly computed as a final matching score for test samples. To index the documents, we used an open-source indexing software titled Terrier [7], and indexed using 7 pre-defined fields from Section 3. Standard pre-processing steps such as removing stopwords and stemming are applied before indexing.

For neural baselines, we experimented with an interaction-based Arc-II model and a hybrid model Duet [4, 8]. These two models are trained in a pairwise setting to minimize rank hinge loss. To evaluate matching performance, we chose NDCG, MAP and MRR with $k \in [1, 5]$, since these metrics capture how accurately our model retrieves the correct items and their respective positions [13]. The positions are ranked by the output score from our model. All of these models including ours are implemented in PyTorch framework [1, 15], and hyperparemeters are tuned using the validation set.

5 EMPIRICAL RESULTS AND DISCUSSION

We report the comparison of our method against other baselines, followed by feature ablation and error analysis.

Results, ablation study and insights. Table 4 shows that our model outperformed all lexical and neural baselines, showing the effectiveness of combining transformers and SMM. Compared to duet, our model achieved 2.20% improved NDCG@5, 0.93% improved MAP and 4.14% improved MRR respectively. For non-transformer models, duet outperformed all other baselines, validating the effectiveness of leveraging distributed and lexical representations.

Models	NDCG@1	NDCG@5	MAP	MRR
BM25	0.280	0.384	0.419	0.462
BM25F	0.287	0.384	0.421	0.466
ArcII	0.285	0.380	0.412	0.465
Duet	0.301	0.408	0.428	0.482
Ours	**0.309***	**0.417***	**0.432**	**0.502***

Table 4: Performance comparison of our proposed model on PSR test dataset. "*" indicates statistical significance of improvement based on two-tailed Student's t-test with $p < 0.05$, compared to Duet model.

To measure the gains from FMM, we conducted an ablation study by training a pre-trained DistilBERT (DB) without fielded representations. For this model, documents are encoded as one long text, thus removing any structured matching advantage. According to Table 5, after adding FMM, we observed statistically significant improvements on MAP and MRR on both test set. Interestingly, we noticed the improvements on NDCG@1 was very small. We hypothesize that FMM does not contribute much to obvious cases but

[1] https://www.kaggle.com/c/home-depot-product-search-relevance

Query labels	NDCG@1		NDCG@5		MAP		MRR	
	DB	**Ours**	**DB**	**Ours**	**DB**	**Ours**	**DB**	**Ours**
Brand/Collection	**0.276**	0.263	**0.418**	0.404	**0.427**	0.416	**0.479**	0.474
Color/Finish	**0.311**	0.278	**0.429**	0.394	**0.443**	0.412	**0.504**	0.460
Unit	0.264	**0.302**	0.374	**0.390**	0.391	**0.407**	0.449	**0.482**
Material	0.247	**0.376**	0.416	**0.438**	0.447	**0.470**	0.484	**0.591**
Model	0.257	**0.283**	0.364	**0.383**	**0.410**	0.404	0.451	**0.460**
Typo	0.312	**0.334**	0.423	**0.428**	0.454	**0.462**	0.521	**0.541**
All others	0.306	**0.317**	0.422	**0.433**	0.442	**0.452**	0.504	**0.517**

Table 3: Error analysis of different types of queries for DistilBERT (DB) and our model with FMM.

more to harder cases with heterogeneous instances. For significance testing, we used two-tailed Student's t-test with $p < 0.05$.

Error Analysis. To more understand where FMM helps and fails, we conducted an error analysis to see trade-offs in metrics after adding FMM. Table 3 shows queries containing field-specific terms are harder than general queries since retrieval performances on All others class are higher than those of other classes. Among fields, matching numerical units are shown to be the most difficult task based on NDCG. This is true because numbers are usually filtered out before training, and without understanding the conversions between different units, it becomes very challenging for models to match query without sufficient text features. After adding FMM, we observed several improvements over various types of queries. There is a decrease in performance for Brand/Collection and Color/Finish types, but our model performed better on all other labels. Interestingly, our proposed model improved on Typo class, showing the benefits of subword lexical matching. Similarly for Units field, knowing the occurrences of unit matches benefited the overall matching performance. To conclude, we claim that our proposed FMM modules reduce the general errors by providing extra evidence from query to field relationships.

Models	Test	NDCG@1	NDCG@5	MAP	MRR
DB	**PSR**	0.304	0.411	0.428	0.488
Ours	**PSR**	**0.309**	**0.417**	**0.437^{*}**	**0.502^{*}**
DB	**CL**	0.682	0.696	0.647	0.785
Ours	**CL**	**0.682**	**0.712^{*}**	**0.705^{*}**	**0.807^{*}**

Table 5: Ablation study of our proposed model against DistilBERT (DB) baseline after removing FMM on both product search relevance (PSR) dataset and click logs (CL) dataset.

Conclusions. We proposed a novel and effective method for matching queries to structured product descriptions for e-commerce search and recommendation. We adopt a state of the art transformers in Siamese architecture to avoid jointly encoding query and documents for improved scalability. Multiple fielded representations are encoded to first form document matrix, and matched against query vectors to extract heterogeneous matching signals. After evaluating our method on two e-commerce dataset, we showed promising directions of representing documents into multiple vectors both rough ablation study and error analysis. Overall, our results provide useful insights into the benefits and limitations of the proposed method, which could further benefit improvements to e-commerce matching, search, and recommendation.

REFERENCES

[1] J. Guo, Y. Fan, X. Ji, and X. Cheng. Matchzoo: A learning, practicing, and developing system for neural text matching. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1297–1300, 2019.

[2] C. V. Gysel, M. De Rijke, and E. Kanoulas. Neural vector spaces for unsupervised information retrieval. *ACM Transactions on Information Systems (TOIS)*, 36(4):1–25, 2018.

[3] S. Hofstätter, M. Zlabinger, and A. Hanbury. Tu wien@ trec deep learning'19–simple contextualization for re-ranking. *arXiv preprint arXiv:1912.01385*, 2019.

[4] B. Hu, Z. Lu, H. Li, and Q. Chen. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*, pages 2042–2050, 2014.

[5] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2333–2338, 2013.

[6] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196, 2014.

[7] C. Macdonald, R. McCreadie, R. L. Santos, and I. Ounis. From puppy to maturity: Experiences in developing terrier. *Proc. of OSIR at SIGIR*, pages 60–63, 2012.

[8] B. Mitra, F. Diaz, and N. Craswell. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1291–1299, 2017.

[9] P. Nigam, Y. Song, V. Mohan, V. Lakshman, W. Ding, A. Shingavi, C. H. Teo, H. Gu, and B. Yin. Semantic product search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2876–2885, 2019.

[10] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3973–3983, 2019.

[11] S. Robertson, H. Zaragoza, and M. Taylor. Simple bm25 extension to multiple weighted fields. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 42–49, 2004.

[12] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR'94*, pages 232–241. Springer, 1994.

[13] M. Sanderson and J. Zobel. Information retrieval system evaluation: effort, sensitivity, and reliability. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 162–169, 2005.

[14] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[15] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

[16] C. Xiong, Z. Dai, J. Callan, Z. Liu, and R. Power. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR conference on research and development in information retrieval*, pages 55–64, 2017.

[17] Y. Yang, S. Yuan, D. Cer, S.-y. Kong, N. Constant, P. Pilar, H. Ge, Y.-H. Sung, B. Strope, and R. Kurzweil. Learning semantic textual similarity from conversations. In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 164–174, 2018.

[18] H. Zamani, B. Mitra, X. Song, N. Craswell, and S. Tiwary. Neural ranking models with multiple document fields. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 700–708, 2018.