DeepCAT: Deep Category Representation for Query Understanding in E-commerce Search

Ali Ahmadvand
Emory University, USA
ali.ahmadvand@emory.edu

Surya Kallumadi
The Home Depot, USA
surya@ksu.edu

Faizan Javed
The Home Depot, USA
Faizan_Javed@homedepot.com

Eugene Agichtein
Emory University, USA
eugene.agichtein@emory.edu

Abstract

Mapping a search query to a set of relevant categories in the product taxonomy is a significant challenge in e-commerce search for two reasons: 1) Training data exhibits severe class imbalance problem due to biased click behavior, and 2) queries with little customer feedback (e.g., *tail* queries) are not well-represented in the training set, and cause difficulties for query understanding. To address these problems, we propose a deep learning model, DeepCAT, which learns joint word-category representations to enhance the query understanding process. We believe learning category interactions helps to improve the performance of category mapping on *minority* classes, *tail* and *torso* queries. DeepCAT contains a novel word-category representation model that trains the category representations based on word-category co-occurrences in the training set. The category representation is then leveraged to introduce a new loss function to estimate the category-category co-occurrences for refining joint word-category embeddings. To demonstrate our model's effectiveness on *minority* categories and *tail* queries, we conduct two sets of experiments. The results show that DeepCAT reaches a 10% improvement on *minority* classes and a 7.1% improvement on *tail* queries over a state-of-the-art label embedding model. Our findings suggest a promising direction for improving e-commerce search by semantic modeling of taxonomy hierarchies.

ACM Reference Format:
Ali Ahmadvand, Surya Kallumadi, Faizan Javed, and Eugene Agichtein. 2021. DeepCAT: Deep Category Representation for Query Understanding in E-commerce Search. In *Proceedings of ACM SIGIR Workshop on eCommerce (SIGIR eCom'21).* ACM, New York, NY, USA, 5 pages.

1 Introduction and Related Work

Query understanding is an essential step in developing advanced retrieval systems (e.g., e-commerce search engines) [5]. In an e-commerce setting, one aspect of query understanding is achieved by mapping a query to a set of relevant product categories [15]. For example, for the query " motion activated kitchen faucet", an e-commerce search engine should return products from relevant categories like *bath, plumbing, kitchen.* These categories match the customer's intent and provide signals for downstream tasks such as retrieval and ranking. In this paper, we develop a new model for query understanding in an e-commerce search engine, depicted in Fig. 1. Fig. 1 shows the query understanding procedure where a search query like "motion activated kitchen faucet" is mapped to a set of relevant product categories in a hierarchical product taxonomy.

Query understanding is a challenging task since: 1) queries are often short, vague, and suffer from the lack of textual evidence [6], 2) queries with similar textual information with slight variations such as "9 cu. ft. chest freezer in white" and "9 cu. ft. upright white freezer" belong to different categories. However, queries with no term overlap like "french door 32 inch. refrigerator" and "black fridge with glass panes" are semantically similar and may belong to related categories, 3) The severe data imbalance problem resulted from customer *bias* towards some specific products in general or in a particular time. Also, the product categories' correlations directly impact customer click behavior, where some of them received more click rates than usual, and others get fewer click rates, and 4) queries with low customer behavior feedback (e.g., *tail* and *torso*) are more challenging to classify as they have a high signal-to-noise ratio. Current neural models achieve a *softer representation* with *richer compositionality* of the queries compared to conventional term-based models [14].

There have been numerous studies in neural models for text representation in different levels, such as characters, subwords, words [4, 9, 12]. These models utilize distributed representation by transferring knowledge from other resources to enrich the query representation [2, 7]. However, they still have difficulty properly addressing challenges (3) and (4) for query understanding. To alleviate these problems, inspired by work in information networks [11], we propose a joint word-category (label) representation to provide both word and category embeddings. Then, category representations are leveraged to boost the model's efficiency on both *tail* queries and the *minority* classes.

Tang et al. [10] introduce the idea of heterogeneous text network embedding to model the word and label interactions. Guoyin et al. [13] expand the concept to extract the relative spatial information among consecutive terms with their associated labels. Although these models leveraged the joint word-label interactions, they still lose the knowledge in label-label correlations. Extracting category (label) co-occurrence information is essential for query understanding, where product categories inherent this correlation

SIGIR eCom'21, July 15, 2021, Virtual Event, Montreal, Canada

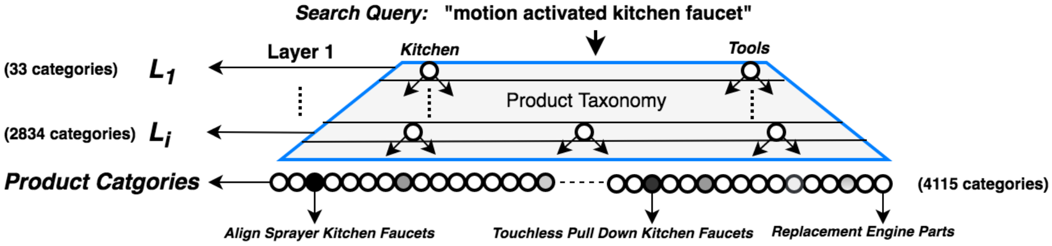

Figure 1: Query understanding procedure.

during taxonomy formation. Product categories are not mutually exclusive and are semantically related to each other. This correlation between product categories impacts the customer click behavior, which utilizes as supervision signals in dataset generation. Thus, category co-occurrences can be used to improve the quality of *minority* classes and *tail* queries, where there is less customer feedback available. To this end, we consider the product categories as an undirected *homogeneous* graph, where the edges represent category correlations.

In this paper, we introduce a data-driven approach named DeepCAT for query understanding. Our model consists of a pipeline of deep learning models that utilize both word-category and category-category interactions. In summary, our contributions are: (1) proposing a novel deep learning model for joint word-category representation, and (2) introducing a new loss function to incorporate pairwise category information into the query understanding process.

2 DeepCAT: Model and Implementation

In this section, we present our DeepCAT model. First, we provide a high-level overview of the model architecture and then describe the model implementation's details. Then , we describe *query representation* and *word-category representation*, and *category-category representation* models, followed by our new loss function to incorporate category co-occurrences.

Model Overview: The DeepCAT network architecture is illustrated in Figure 2. DeepCAT consists of three main components: (a) query representation, (b) joint-word-category representation, and (c) category-category representation. Any state-of-the-art deep network could be used to develop the query representation (Query2Vector Network). We deploy a CNN-based model, which consists of convolutional layers followed by highway layers [14], to add more non-linearity to the model and improve the model capacity by allowing information flow in the network. Recurrent [3] or transformer [12] neural models could also be used as an alternative for Query2Vector network. However, due to their high latency time during inference compared to feed-forward neural models, we decided to choose the convolutional neural network-based models for query representation.

We leverage the word-category co-occurrence concepts for joint-word-category representation, which computes using a cosine similarity between query words and their associated categories. Then, a multi-head self-attention deploys to generate the contribution of each word to each specific product category. These attention scores utilize to modify the word's contribution in the query modeling. Finally, category and query representations are concatenated to create the final query representation. A sigmoid cross-entropy is deployed to compute the loss values for this multi-label problem. For category-category representation, first, we extract the experimental category co-occurrence matrix CM from the training set. Next, it normalized using the *Cosine* normalization method. In each training step, the CM is estimated using the category representations, and the loss values are propagated through the network using matrix approximation [8].

Query Representation (Query2Vector Network): Suppose there is a search query dataset $D = \{Q, C\}$, where Q is a set of search queries and C is candidate product categories. Each query consists of a sequence of words $q = [w_1; w_2; \dots; w_n]$ of size $n = 10$, and is represented as $q_w^{|n| \times V}$. Also, C is mapped to embedding spaces of $\mathbf{C}^{|C| \times V}$. The word and category embeddings are initialized with Word2Vec and random embedding of size $|V| = 100$, respectively. For the query representation, any complex deep learning model could be used. Our implementation of Query2Network uses a 3-layer CNN model, where it receives the word embeddings and produces the query representation. $cnn(q_w)$, goes through a *highway* layer [14]. A highway layer combines a ReLU function for a non-linear projection, followed by a sigmoid function for smoothing the projection of each convolutional layer, $highway(q_w) = relu\Big(sigmoid\big(cnn\big)\Big)$.

Word-Category Representation: To train category representation, first, in each training step, we form a word-category co-occurrence matrix. The index (i, j) of this matrix indicates the co-occurrence of word i and associated category j of the query. To estimate this matrix during the training, we need a dot-product between word representations of query $(n \times V)$ with the category representations $(|C| \times V)$. The output is of size $(n \times |C|)$, where n, $|C|$, and $|V|$ indicate the query length, number of categories, and embedding size, respectively. After estimating the word-category co-occurrence matrix, we need to extract each word's contribution in the query to all product categories. We deploy a self-attention mechanism with $n = 10$ different heads to compute the scores. We use ten heads since we consider each query at most includes ten words. Finally, an attention matrix of size $(n \times |C|)$ creates $A_{wc} = Self\_Attention\big(l2\_norm(q_w) \odot l2\_norm(C)\big)$, where the value at (i, j) represents the contribution of word i to category j. The output goes through a max-pooling layer to form the attention weights. The attention weights multiples to the word vectors to generate the weighted word embeddings $R_{wc} = q_w \odot A_{wc}$. A multi-head self-attention mechanism applies to q_w. Multi-head self-attention contains several linear projections of a single scaled dot-product function that are parallelly implemented $head_i = SoftMax\Big(\frac{q_w K^T}{\sqrt{d_k}}\Big)\, V$.

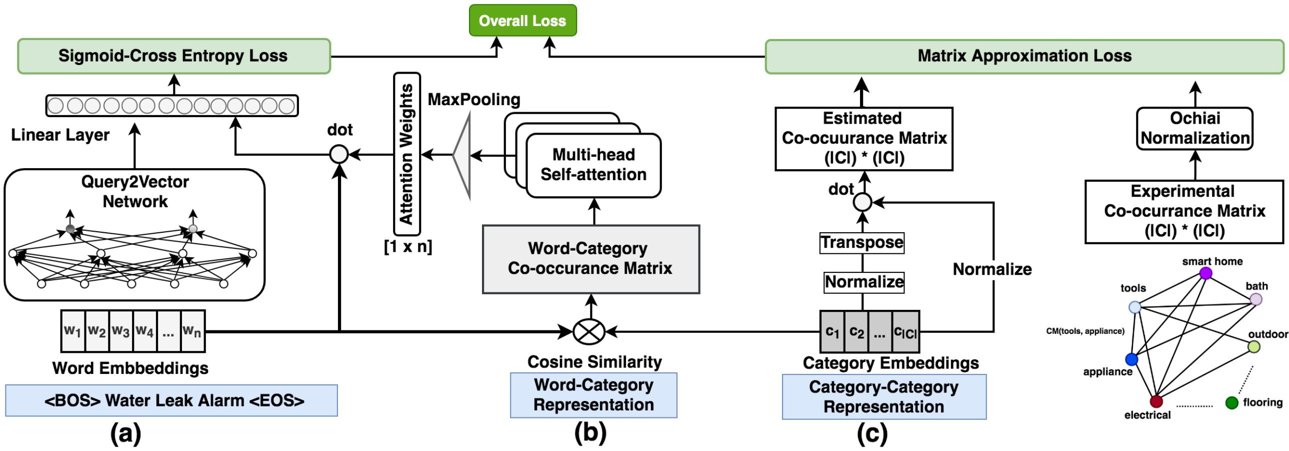

Figure 2: DeepCAT architecture, (a) query representation (b) word-category representation (c) category-category representation.

Where $\odot$ indicates a dot-product. Finally, R_{q_w} and R_{wc} go through a linear layer to form R, the final joint word-category representation.

Category-Category representation: A co-occurrence matrix creates on training data to model the category-category interactions. In this matrix each element (i, j) represents the co-occurrence frequency between label-pair of (c_i, c_j) in the training set. Finally, category-category co-occurrence matrix has the size of $|C| \times |C|$. Then, the final matrix is calculated by applying a matrix normalization. We deployed *Cosine* normalization to normalize the CM, where the values on the main diagonal are one. Moreover, the experimental category-category CM is computed using category co-occurrences in the training set. To estimate the normalized matrix, *Cosine* similarity is used between category representations.

Joint Word-Category Loss: A sigmoid cross-entropy loss function $\mathcal{L}_{pc}$ uses for final product category classification. Sigmoid cross-entropy applies since, in sigmoid, the loss computed for every output s_i is not affected by other component values. $\mathcal{L}_{pc} = -\sum_{c=1}^{|C|} t_c \log(Sigmoid(s_c))$. Where s_c represents the predictions and t_c indicates the targets.

Category-Category Loss: The estimation error is calculated based on a matrix approximation loss [8], $\mathcal{L}_{CM} = \frac{1}{mn} \sum_{i,j \in C} log(1+ exp(C\hat{M}_{ij} \odot CM_{ij}))$.

The Overall Loss: To compute the overall loss, a weighted average of $\mathcal{L}_W$ and $\mathcal{L}_{CM}$ is computed as $\mathcal{L}_{overall} = \lambda_1 \mathcal{L}_{CM} + \lambda_2 \mathcal{L}_W$.

3 Experimental Evaluation

This section describes dataset overview, experimental design, parameter setting, metrics, baseline models, and evaluation.

Dataset Overview: Similar to [15], we utilize customer behavior feedback (e.g., click rate) to obtain the category labels associated with each search query. We collect two weeks of search log to create both training and test sets, where the first week is used to create the training set and the second week for the test set. The training set contains more than 11M search queries. We used 25% of the training data for validation. To generate the test set, we map queries into three different buckets using a simple query frequency. Queries with only one occurrence experimental period are considered as *tail* queries; the ones between 2 and 100 impressions are counted as *torso*, and the rest as *head* queries. Then, to fairly evaluate the models' performance, stratified sampling [1] is used to generate the test set, where we randomly select 2000 different queries from each bucket to create the test set.

DeepCAT Experimental Design: We designed two different experiments to evaluate DeepCAT. In the first experiment, we assess the DeepCAT capability in mapping an input query to the first level in the taxonomy hierarchies, *L1*, with 33 different classes. The *L1* level contains the most abstract product categories (e.g., "appliances", "tools", and "flooring"). This experiment is mainly outlined to estimate the performance of *minority* classes. The *minority* classes include the categories that contain a fairly small number of samples in the training set due to customer click behavior and category overlaps or correlations. The second experiment evaluates DeepCAT on actual product categories in the last layer of taxonomy *Product Categories* with 4115 distinct categories (e.g., "replacement engine parts", "wood adirondack chair", and "window evaporative coolers").

Parameter Setting: We used an Adam optimizer with a learning rate of $\eta = 0.001$, a mini-batch of size 64 for training, and embedding of size 100 for both word and category. The dropout rate of 0.5 is applied at the fully-connected and ReLU layers to prevent the model from overfitting.

Evaluation Metrics: Following the conventions of the search literature to evaluate DeepCAT, we reported the overall Macro- and Micro- averaged *F1*, *P@K*, *R@K*, *F1@K* and *MAP@K* on the top-*K* results. Also, query understanding is a multi-label problem; we reported precision and recall since a practical solution must cover broader possible correct categories while simultaneously keeping precision as high as possible [15].

Methods Compared: We summarize the **multi-label** classification methods compared in the experimental results.

- **TF-IDF + SVM:** One-Vs-Rest SVM with a linear kernel.
- **FastText:** Text classification method by Facebook [2].
- **XML-CNN:** Extreme multi-label text classification [9].
- **LEAM:** Word-label representation model [13].
- **DeepCAT:** The proposed word-label representation.

Method	**Leaf Nodes (Product Categories)**									
	P@1	R@1	F1@1	P@3	R@3	F1@3	P@5	R@5	F1@5	MAP@5
TF*IDF BOW	0.783	0.259	0.356	0.617	0.478	0.538	0.514	0.594	0.551	0.623
FastText [2]	0.856	0.2001	0.324	0.634	0.444	0.522	0.504	0.557	0.542	0.666
XML-CNN [9]	0.875	0.314	0.463	0.683	0.549	0.609	0.568	0.666	0.613	0.703
LEAM [13]	0.862	0.302	0.447	0.676	0.531	0.595	0.566	0.651	0.606	0.697
DeepCAT	**0.888***	**0.325***	**0.475***	**0.690**	**0.560***	**0.619***	**0.576**	**0.680***	**0.624***	**0.717***

Table 1: Performances on *Product Categories* with about 4200 categories. "*" indicates statistically significant improvements $p < 0.05$.

Method	**First Layer (L1)**		
	Macro-F1	Micro-F1	MAP@3
TF*IDF BOW	0.466	0.669	0.669
FastText [2]	0.496	0.686	0.653
XML-CNN [9]	0.511	0.706	0.694
LEAM [13]	0.521	0.709	0.701
DeepCAT	**0.540***	**0.720***	**0.710***

Table 2: Performances on *L1* with 33 categories. "*" indicates statistically significant improvements $p < 0.05$.

3.1 Results and Discussion

Table. 2 and 1 summarizes the performance of different state-of-the-art models on curated datasets described in section 3. The results show that DeepCAT significantly improves Macro- and Micro-average *F1*, and *MAP@3* by (3.6%, 1.5%, and 1.2%) over LEAM, as the best model among deep networks, on *L1* level. As a results, an average improvements of (6%, 2.8%, and 4%) on Macro- and Micro-averaged *F1*, and *MAP@3* over state-of-the-art deep learning models. For *product categories*, DeepCAT outperforms LEAM by (6.2%, 4%, 3%, and 3%) on *F1@1, F1@3, F1@5,* and *MAP@5*, respectively.

Results on Minority Classes: Table. 2 indicates that Macro-averaged *F1* improves by 2% over Micro-averaged *F1*, which shows a higher impact on the minority classes. This impact is more noticeable on 8-button minority classes, where the Macro-averaged *F1* for the for XML-CNN and LEAM are 0.41.01%, 42.90%. At the same time, this number jumps to 47.16% for DeepCAT, which shows more than 12% and 10% relative improvements, respectively.

Results on Traffic Buckets: Table. 3 shows the performance of the models described in section. 3 across three main buckets of *tail, torso,* and *head.*

Method	**FastText**	**LEAM**	**XML-CNN**	**DeepCAT**
Head	0.508	0.563	0.560	**0.565 (+0.0%)**
Torso	0.584	0.646	0.648	**0.682 (+5.3%)**
Tail	0.381	0.337	0.373	**0.401 (+7.1%)**

Table 3: *F1@3* results on *head, torso,* and *tail* buckets.

The results show that DeepCAT significantly outperforms the other models on both *tail* and *torso* buckets, while it reaches competitive results to XML-CNN and LEAM on "head" bucket. According to higher traffic on both *tail* and *torso* queries, the overall performance of DeepCAT is significantly higher compared to the other models. The *F1@3* is lower on *head* compared to *torso* queries due to a significantly higher number of correct (relevant) categories, which causes a higher *P@3* and a significantly lower *R@3.*

Ablation Analysis: DeepCAT is a complex model that consists of several components. We performed a comprehensive ablation study to evaluate each component's impact on the overall performance of DeepCAT. Table. 4 reports the contribution of each component on performance. The results illustrate that utilizing the category representation describe in section. 2 provides a (5.1%, 3.2%) improvement on Macro- and Micro-averaged *F1*, respectively. Moreover, using $\mathcal{L}_{CM}$ improves Macro-averaged *F1* by (2.8%, 1.3%), respectively.

Method	**Macro-F1**	**Micro-F1**
Word Rep.	0.500	0.689
Joint Word-Category Rep.	0.526 (+5.0%)	0.711 (+3.1%)
Joint Word-Category Rep. + $\mathcal{L}_{CM}$	**0.540 (+2.9%)**	**0.720 (+1.3%)**

Table 4: Ablation analysis results.

Summary: Our experimental results show the robust performance of DeepCAT compared to state-of-the-art models. For *minority* classes, *tail,* and *torso* queries, we observed 10%, 7%, and 5.3% relative improvements, respectively. We also report the performance on the last layer (leaf nodes) of product taxonomy consisting of 4115 categories. The results show that DeepCAT achieves (6.2%, 4%, 3%, and 3%) increase on *F1@1, F1@3, F1@5,* and *MAP@5*, respectively. In ablation analysis, we show that the improvements come from all three components of DeepCAT. The joint word-category representation improves the query representation by 5%, and the loss function can further improve it by 2.9%.

4 Conclusions

We introduced a deep learning model, DeepCAT, for query understanding in e-commerce search. DeepCAT contains a new joint word-category representation component in which category representations are learned using word-category co-occurrences. Then, we proposed a novel loss function utilizing category representations to model category-category co-occurrences. Our comprehensive experiments showed that using category representation significantly improved the results, particularly on minority classes and *tail* queries. DeepCAT achieved a 10% improvement on *minority* classes and a 7.1% increase on *tail* queries over a state-of-the-art label embedding model.

Acknowledgements We gratefully acknowledge the financial and computing support from The Home Depot Search & NLP team.

References

[1] B. Babcock, S. Chaudhuri, and G. Das. Dynamic sample selection for approximate query processing. In *SIGMOD*, pages 539–550, 2003.

[2] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[3] J. Chen, Y. Hu, J. Liu, Y. Xiao, and H. Jiang. Deep short text classification with knowledge powered attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6252–6259, 2019.

[4] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun. Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781*, 2016.

[5] W. B. Croft, M. Bendersky, H. Li, and G. Xu. Query representation and understanding workshop. In *SIGIR Forum*, volume 44, pages 48–53, 2010.

[6] J.-W. Ha, H. Pyo, and J. Kim. Large-scale item categorization in e-commerce using multiple recurrent neural networks. In *SIGKDD*, pages 107–115. ACM, 2016.

[7] Y. Kim. Convolutional neural networks for sentence classification. In *EMNLP*, 2014.

[8] D. Li, C. Miao, S. Chu, J. Mallen, T. Yoshioka, and P. Srivastava. Stable matrix approximation for top-n recommendation on implicit feedback data. In *Proceedings of the 51st Hawaii International Conference on System Sciences*, 2018.

[9] J. Liu, W.-C. Chang, Y. Wu, and Y. Yang. Deep learning for extreme multi-label text classification. In *proceedings of SIGIR*, pages 115–124, 2017.

[10] J. Tang, M. Qu, and Q. Mei. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1165–1174, 2015.

[11] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *WWW*, pages 1067–1077, 2015.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[13] G. Wang, C. Li, W. Wang, Y. Zhang, D. Shen, X. Zhang, R. Henao, and L. Carin. Joint embedding of words and labels for text classification. *arXiv preprint arXiv:1805.04174*, 2018.

[14] H. Zhang, X. Song, C. Xiong, C. Rosset, P. N. Bennett, N. Craswell, and S. Tiwary. Generic intent representation in web search. In *SIGIR*, pages 65–74. ACM, 2019.

[15] J. Zhao, H. Chen, and D. Yin. A dynamic product-aware learning model for e-commerce query intent understanding. In *CIKM*, pages 1843–1852, 2019.