E-commerce Product Query Classification Using Implicit User's Feedback from Clicks

Yiu-Chang Lin
Rakuten Institute of Technology
Boston, Massachusetts - USA 02110
yiuchang.lin@rakuten.com

Ankur Datta
Rakuten Institute of Technology
Boston, Massachusetts - USA 02110
ankur.datta@rakuten.com

Giuseppe Di Fabbrizio*
Boston, Massachusetts - USA 02110
difabbrizio@gmail.com

***Abstract*—Query classification (QC) has been widely studied to understand users' search intent. For e-commerce search queries, users typically search for either a specific product or a category of products. In both cases, a query can be associated with a category label that belongs to a taxonomy tree describing the items in the catalog. However, product-related search queries are typically short, ambiguous, and continuously changing depending on seasonal trends and the introduction of new products over time. Traditional supervised approaches to e-commerce QC are not feasible due to the high cost of manual annotation and the high volume of traffic on e-commerce search engines. In this work, we introduce an unsupervised method to collect large amounts of query classification data using user's implicit click feedback. We obtain a large multi-label dataset containing 403,349 unique queries from 2,085 categories. We compare and contrast different state-of-the-art text classifiers and demonstrate that an ensemble of linear SVMs models achieves a micro-F1 score of 0.60 and 0.82 at leaf and top level, respectively.**

***Index Terms*—Query classification, Taxonomy categorization, Multi-label classifier**

I. Introduction

Query understanding (QU) is core to search engines to infer the precise intent expressed in users' queries and retrieve relevant content to improve user' satisfaction and e-commerce conversion rates. QU is a challenging task since queries are typically short, ambiguous, and domain-dependent. A first level of query understanding includes query classification (QC) defined as the task of classifying queries into single or multiple predefined target categories. QC can boost relevance in content search by predicting which category the query belongs to and passing the hypothesis to the search engine as ranking signal. Such capability is crucial, especially in the e-commerce domain where content search is mostly directed to specific products that are typically categorized into a taxonomy tree.

Recently, an empirical study of the queries and search behavior of users in e-commerce search has been done by analyzing the search log [14]. With k-Means clustering algorithm [9], the analysis results show that e-commerce queries can be categorized into five different categories, each with distinctive search behaviors. These five categories form a taxonomy and shed some light on how customized search technologies can be developed for each type of search queries to improve search engine quality.

One of the early work in this area is the ACM KDDCUP 2005 competition [11]. The task was to classify 800,000 search queries into 67 hierarchical target categories where each query can belong to more than one category at the same time. The challenge only provided a small set of 111 queries for training, each annotated with up to five of the 67 categories, and 800 queries randomly selected from the corpus and annotated by following the same annotation schema. The winning solution [12] combined with its later modification [13] built a bridging classifier on an intermediate taxonomy offline and adopted category selection for online classification. This competition revealed a number of challenges related to the task: 1) the lack of annotated data due to the cost of human annotations and the large scale of query data that makes coverage almost impossible; 2) the ambiguity of search queries due to multiple users' intents expressed by the same words.

To address the two challenges above, we introduce a general unsupervised data collection method that automatically generates QC data from four months of query logs obtained from Rakuten[1] using user's implicit feedback from clicks. We captured the queries ambiguity by assigning multi-label annotations to queries that are interpreted differently by users.

When considering a web-scale product catalog, queries are unlikely to cover the entire list of products in the catalog. Typically, the more popular products are retrieved and click-rates are unevenly distributed across the catalog taxonomy categories. To mitigate the data sparseness, we adopted a taxonomy tree pruning strategy by merging leaf nodes with a limited number of clicks with their parent node. In such a way, 403,349 queries are collected, labeled with 2,085 target categories in total where no human annotator is involved. Furthermore, we compare and contrast several traditional and state-of-the-art text classifiers, including logistic regression, SVMs [4], XGBoost [2], `fastText` [8] and attention-based CNNs [17].

II. Data Collection

In this section, we describe our QC data collection approach and product taxonomy tree pruning strategy in the e-commerce domain.

*Contribution done when working at Rakuten.

[1]https://www.rakuten.com/

A. Clickstream Data

A *session* S_i (or *visit*) is a chronological sequence of recorded user's actions including *search*, *click*, *add-to-cart*, *purchase*, and so on. A user may interact with the search engine m times in a single S_i. For each query q_j $(1 \leq j \leq m)$, where a user may retrieve n_j products $\{p_{j1}, p_{j2}, ...p_{jn_j}\}$, we collect all the query and product pairs (q_j, p_{jk}) where $1 \leq k \leq n_j$ that have been selected by mouse clicks, added into the shopping cart, or purchased. If the time interval between two actions in a single session is more than 30 minutes, we divide them into two separate sessions. After performing the above collecting process across all the sessions, we obtain a set of query and product pairs along with their cumulative click frequencies.

The next step is to leverage the click information to associate product category labels to each query. A simple, yet effective way, is to assign the query category label to the taxonomy category associated to the selected product. This is a reasonable approximation under the assumption that the user's behavior is coherent with the selection of the retrieved product.

B. Product Taxonomy

A product taxonomy is a tree-based hierarchical representation where each node is mapped into a product category from a product catalog. In a typical e-commerce setting, merchants are responsible to match their products with an existing category defined as leaf in the taxonomy tree. The sequence of nodes from the tree root to the leaf represents the semantic labels of a product and it is often referred as *attribute-based breadcrumbs* [7]. With the taxonomy tree, a 1-to-1 mapping is conducted to transform each product to its corresponding path, i.e., label. As a result, previous (query, product) pairs are converted to (query, category/label) pairs. As mentioned before, each unique query may have multiple labels.

While e-commerce catalogs collect a wide number of products classified by a fine-grain taxonomy tree, the actual product retrieval process is driven by user's needs and does not necessarily reflect the initial taxonomy design. In this scenario, nodes such as *Sports* $\rightarrow$ *Cycling* $\rightarrow$ *Bike Accessories* $\rightarrow$ *Child Seats* and *Sports* $\rightarrow$ *Cycling* $\rightarrow$ *Bike Accessories* $\rightarrow$ *Bike Fenders* are unlikely to be visited by users due to small number of items categorized in the taxonomy leaf. Therefore, it is intuitive to merge these two nodes with their parent nodes, *Sports* $\rightarrow$ *Cycling* $\rightarrow$ *Bike Accessories*. In this way, products belonging to the removed leaf still get a chance to be categorized in the parent node as more general category.

Fig. 1 shows a small part of the entire catalog taxonomy where the number below each node is the sum of click frequencies for all the queries associated to that node. Due to the popularity of certain product categories, the node frequency is unevenly distributed. To re-balance the tree, we recursively merge less popular nodes with their parents if the summation of cumulative click frequencies of a node is less than a certain threshold. In practice, the number of categories reduces from 5,896 to 2,085 when frequency threshold is set to 50.

We validated this hypothesis by manually reviewing 600 queries with multi-label category pairs and observed that 24.8% were partially incorrect (i.e., included a correct subsequence of breadcrumb nodes) or controversial (i.e., the matching category was consistently wrong in the catalog). For instance: laptops like "*lenovo t 420 thinkpad 12 GB*" are consistently mapped to *Computers* $\rightarrow$ *Computer Systems* $\rightarrow$ *Mobile* while the rest correctly map the query to a valid set of categories. We also observed, that depending on the level of specificity for the query, the same query could be categorized with more than one taxonomy label.

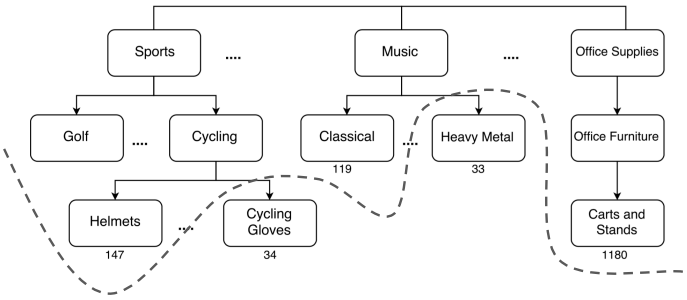

Fig. 1: A part of the entire product taxonomy. The number below each node is the sum of click frequencies of all the queries associated to that node. The dotted line shows the pruning boundaries based on a threshold of 50.

C. Data Characteristics

We collect 403,349 queries labeled over 2,085 categories and partitioned the obtained corpus into training and test sets with a 3:1 split. Each query may be tagged with more than one category and the average number of labels per query in the training set is 1.4. Being one of the challenges of QC, it is not surprising that 48% of the queries has less than or equal to 2 words and 93% of them has less than or equal to 5 words. Fig. 2 shows the percentage distribution of queries with different length in training set.

The product taxonomy has 36 categories at level one. The maximum depth of the tree is 7 and the average depth is 3.37. As shown in Fig. 3 (left), queries are unevenly distributed among these 36 top level categories. 23% of the queries fall in the largest category (*Computers*) while only less than 0.01% of the queries fall in the smallest one (*Hair Jewelry*).

III. Methods

Given a web query, query classification can be modeled as a text classification task where the objective is to find a category or top n categories. To address this task, we experiment with several text classifiers including logistic regression, SVMs, Gradient Boosting Trees (GBTs[2]), `fastText`[3], and attention-based CNNs [17].

[2]https://github.com/dmlc/xgboost

[3]https://github.com/facebookresearch/fastText

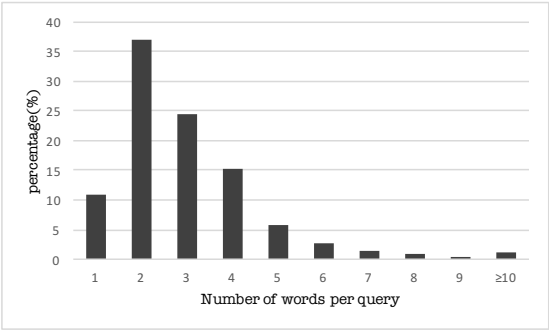

Fig. 2: Percentage of queries with different length in the training set.

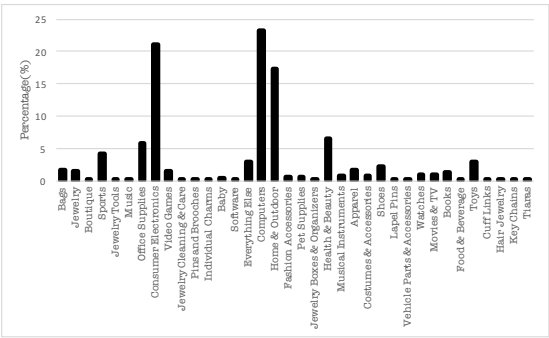

Fig. 3: Percentage of queries among 36 level-1 categories in the training set.

A. GBTs

In the predictive learning problem, the goal is to find an approximation $F(\mathbf{x})$ of the function $F^*(\mathbf{x})$ that maps a set of input $\mathbf{x} = \{x_1, ..., x_n\}$ to a label $\mathbf{y}$, and, at the same time, minimizes a specified loss function using N training data $\{x_i, y_i\}_{i=1}^{N}$ [6]. Gradient Boosted Trees, GBTs, seeks to minimize the following loss function:

$$\mathcal{L} = E_y \left[L\left(y, F\left(\mathbf{x}\right)\right) | \mathbf{x} \right] \tag{1}$$

where $F(x)$ can be any function, such as a decision tree. Following the numerical optimization paradigm and additive strategy, the optimal solution can be expressed as:

$$F^*(\mathbf{x}) = \sum_{m=0}^{M} f_m\left(\mathbf{x}, \mathbf{a}, \mathbf{w}\right) \tag{2}$$

where $f_0\left(\mathbf{x}, \mathbf{a}, \mathbf{w}\right)$ is the initial guess and $\sum_{m=1}^{M} f_m\left(\mathbf{x}, \mathbf{a}, \mathbf{w}\right)$ are the incremental boosts on $\mathbf{x}$, while $\mathbf{a}$ and $\mathbf{w}$ can be viewed as split points of predictors and boosting weight on the leaf nodes, respectively.

B. fastText

`fastText` is an open source library for efficient learning of word representations and sentence classification. It has been shown to have classification accuracy on par with deep learning classifiers, such as the character level convolutional model (char-CNN) [18], the character based convolutional recurrent network (char-CRNN) [16] and very deep convolutional network (VDCNN) [3] with significant speed-up in both training and testing phase.

Each input sentence $x_1 x_2 ... x_N$ is represented as N bag of n-gram features to capture some partial information about the local word order. The features are embedded and averaged to form the hidden variable. A SofMax function is then applied to compute the probability distribution over the target classes.

C. Attention-Based CNNs

Attention mechanism allows a neural network text classification model to "attend" to different parts of the input. In other words, each input unit, which could be a character or a word, is of different importance from the model's perspective according to the learned attention weights. Hierarchical Attention Networks [17], which has two levels of attention mechanism, is currently the state-of-the-art classifier for document classification. Its two-level architecture is capable of learning deep word representation (word encoder) and sentence representation (sentence encoder), respectively.

Different from Hierarchical Attention Networks, we implement an Attentional CNN model that was shown to achieve best accuracy in a large-scale product title classification task [15]. Instead of employing two-level attention structure, we first use an attention module as a context encoder to learn the representation of input sentence. Secondly, the output of the previous module then serves as the context vector in a second attention module that is followed by a fully connected layer and softmax layer to predict the output class.

IV. Experiments and Results

In the query classification experiments, we considered two different settings: single-label and multi-label classification. For the former, we removed from the data sets the queries with more than one label to eliminate the ambiguity. This lowered the training set to 234,157 queries and testing set to 78,053 queries. The number of labels also reduces to 1,406. For the latter, as mentioned previously, the training set is comprised of 302,511 queries and the test set of 100,838 queries with 2,085 targeted categories.

For logistic regression, SVMs and XGBoost, several features are extracted both at the tokenized word level and character level. We use counts of word uni-gram and bi-gram together with TF-IDF of word uni-gram and bi-gram at word level. For character-level, we extract from uni-gram to 4-grams along with their frequency counts. No feature engineering is required for `fastText` and Attentional CNN where the input are raw text. The initial word representation for Attentional CNN is obtained by training `word2vec`[4] on the catalog product titles of clicked product associated with all the queries in the training data. Hyper-parameter tuning is conducted by using 5-fold cross validation on training set.

[4]https://www.tensorflow.org/tutorials/word2vec

A. Single-Label QC

Table I shows the best micro-F1 score for each text classifier at different prediction levels. Note that only one time leaf level prediction is performed per classifier. The evaluation at different level is based on taking into consideration only top k levels from the entire breadcrumb path.

Both multi-class LR and SVM are trained in 1-vs.-all manner to avoid $O\left(n^2\right)$ model complexity where n is number of classes. For logistic regression, the best micro-F1 score is achieved when regularization strength $C = 1.0$ with L_2 penalty. For SVMs, linear kernel with squared hinge loss and penalty parameter $C = 0.1$ has 0.58 leaf level micro-F1 and 0.78 level-1 micro-F1, which is the highest among all classifiers. Additionally, they are significantly higher compared to other algorithms using the Friedman test [5] with a p-value < 0.0001. XGBoost is trained with softmax objective function and allowed to grow to a maximum depth of 1,000. Initial learning rate is set to 0.3 and the number of boosting rounds is 50.

For `fastText`, the size of word vectors is set to 200 and learning rate is assigned a value of 0.1. The maximum length of word n-grams is 2 and minimal number of word occurrences is 1. The micro-F1 score converges after 200 epochs. For attention-based CNNs, we use an initial word and character embedding with dimension of 300. Filter sizes are set to 1, 2, 3 and 2, 3, 4 for the word and character convolutional layer, respectively. The size of the first hidden layer is 2,400, which is the value of the number of filter sizes times number of filters, 400. A dropout probability of 0.5 is applied to this layer. The second hidden layer, followed by softmax, is of size 1500 and cross entropy loss is used for optimizing the parameters.

TABLE I: Best micro-F1 score of multi-class single-label LR (logistic regression), SVMs, XGBoost, fastText and Attentional CNN classifier at different levels.

level	LR	SVM	XGBoost	fastText	ACNN
1	0.74	0.78	0.66	0.68	0.71
2	0.64	0.68	0.55	0.52	0.61
3	0.58	0.63	0.49	0.46	0.55
4	0.55	0.59	0.46	0.42	0.51
5	0.55	0.59	0.45	0.41	0.50
leaf	**0.54**	**0.58**	**0.44**	**0.41**	**0.49**

Linear SVMs have the highest micro-F1 score in single-label QC setting. One possible reason could be related to the large feature space (727,405 dimensions), which is three times larger than the number of training instances (234,157 queries), but also sparse. In such scenario, a linear kernel would be enough to achieve high accuracy and there may not be need to map data to a higher dimensional space. This make linear SVMs an ideal classifier for query classification. On the other hand, although logistic regression is robust to small noise in the data (via L_2 regularization), it is not suitable for handling large number of features. Other than the feature sparsity and its high dimensionality, the highly imbalanced nature of QC data, as shown in Fig. 3 (right), might explain the weak performance of XGBoost. Deep learning based models, e.g., `fastText` and Attentional CNN, do not achieve high micro-f1 score as described in other tasks, such as document classification and sentiment analysis, just to name a few, where the input to the model are usually longer text compared to queries. This fundamental challenge severely limits the power of neural networks in the QC domain. [5]

Intuitively, for text classification task, the longer the input text is, the more information a classifier can learn from. As a consequence, to understand how query length affects the performance of a text classifier, we further investigate the micro-F1 score of the best classifier from Table I, i.e., SVMs, by filtering out queries whose length are less than a specific threshold. The first row in Table II indicates the value of filtering threshold. A threshold = 1 means all the queries with length less than or equal to 1 are discarded from the data set. micro-F1 score of every level unsurprisingly increases as we raise the threshold, which verifies the challenge of short query length in query classification.

TABLE II: micro-F1 for multi-class single-label SVMs classifier (1 v.s.all) at different levels and query length thresholds.

query length threshold	0	1	2	3	4
# of queries	78,053	69,243	43,227	23,892	11,198
level	micro F1 score				
1	0.78	0.80	0.82	0.85	0.87
2	0.68	0.70	0.73	0.76	0.78
3	0.63	0.65	0.68	0.71	0.74
4	0.59	0.62	0.64	0.67	0.69
5	0.59	0.62	0.64	0.66	0.69
leaf	**0.58**	**0.60**	**0.62**	**0.64**	**0.67**

B. Multi-label QC

Compared to the single-label query classification setting, multi-label classification is more challenging yet closer to real world applications. Instead of predicting 1 out of n categories for each query, the model is expected to predict l out of n categories where $1 \leq l \leq n$ and l varies from query to query. Based on the result from previous single-label experiments, we choose the best classifier, i.e., linear SVMs, as the main classifier and report its micro-precision, micro-recall and micro-F1 in Table III. To tackle multi-label query classification problem, in practice, each class is treated as an independent category. Therefore, n 1-vs.-all classifiers are trained where n is the number of classes and parameter tuning is conducted within each class. This method achieves 0.70 micro-F1 at top level and 0.51 micro-F1 at leaf level.

Theoretically, a multi-label classifier may predict zero number of label [6], which causes harm to recall. This explains why micro-recall is much lower than micro-precision in the left column of Table III. To mitigate the low recall issue, when no label is predicted, we fall back to the single-label model trained in section IV-A and adopt its prediction outcome. By

[5]In [10], the average length of most dataset lies between 10 and 23. The complexity of classification experiments is much less owing to only a few classes in each instance instead of thousands in our case.

[6]This happens when all the predicted probabilities of n classifiers are below the decision threshold.

TABLE III: micro-Precision, micro-Recall and micro-F1 score of multi-label SVMs classifier and ensemble SVMs classifier (multi-label plus single label) at different tree depth levels.

	multi-label SVMs only			multi-label + single label SVM		
level	P	R	F1	P	R	F1
1	0.84	0.61	0.70	0.84	0.81	0.82
2	0.73	0.52	0.61	0.74	0.69	0.71
3	0.67	0.48	0.56	0.68	0.62	0.65
4	0.63	0.45	0.52	0.64	0.58	0.61
5	0.62	0.44	0.52	0.63	0.57	0.60
leaf	**0.62**	**0.44**	**0.51**	**0.63**	**0.57**	**0.60**

simply ensembling both multi-label and single-label models, micro-recall is largely increased because the aforementioned zero label scenario no longer exists. As shown in the right column of Table III, for top level, micro-Recall improves from 0.61 to 0.81 ($+33\%$) and micro-F1 from 0.70 to 0.82 ($+17\%$) accordingly. For leaf level, micro-Recall has a $+30\%$ gain from 0.44 to 0.57 and micro-F1 has a $+18\%$ gain from 0.51 to 0.60.

V. Conclusion and Future Work

Lack of training/testing data due to costly human annotation along with short and ambiguous queries are the two main challenges in query classification. This paper introduces an unsupervised way to collect a large amount of data from users' implicit click feedback in a e-commerce domain, resulting in 403,349 queries and 2,085 categories. Furthermore, we compare several state-of-the art text classifiers on two different settings, single-label and multi-label. Experimental results show that linear SVMs are suitable for QC while deep learning models suffer due to the short nature of queries in this task, which limits neural network's abilities to learn deep representations. An interesting future research direction is to incorporate query expansion methods to enrich short queries or take context information into account [1]. Another direction is to integrate the query classification module into online commercial search engine and conduct A/B testing to measure quantitatively how it affects ranking metric such as NDCG@k of real word product search. Moreover, the authors also look forward to making this dataset publicly available and providing the linear SVMs classifier as a baseline model.

References

[1] Huanhuan Cao, Derek Hao Hu, Dou Shen, Daxin Jiang, Jian-Tao Sun, Enhong Chen, and Qiang Yang. Context-aware query classification. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 3–10. ACM, 2009.

[2] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.

[3] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. Very deep convolutional networks for natural language processing. *arXiv preprint arXiv:1606.01781*, 2016.

[4] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, September 1995.

[5] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.

[6] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[7] Keith Instone. Location, path, and attribute breadcrumbs. In *The 3rd Annual Information Architecture Summit*, 2002.

[8] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[9] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(7):881–892, July 2002.

[10] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics.

[11] Ying Li, Zijian Zheng, and Honghua Kathy Dai. Kdd cup-2005 report: facing a great challenge. *ACM SIGKDD Explorations Newsletter*, 7(2):91–99, 2005.

[12] Dou Shen, Rong Pan, Jian-Tao Sun, Jeffrey Junfeng Pan, Kangheng Wu, Jie Yin, and Qiang Yang. Q 2 C@ UST: our winning solution to query classification in kddcup 2005. *ACM SIGKDD Explorations Newsletter*, 7(2):100–110, 2005.

[13] Dou Shen, Jian-Tao Sun, Qiang Yang, and Zheng Chen. Building bridges for web query classification. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 131–138. ACM, 2006.

[14] Parikshit Sondhi, Mohit Sharma, Pranam Kolari, and ChengXiang Zhai. A taxonomy of queries for e-commerce search. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, pages 1245–1248, New York, NY, USA, 2018. ACM.

[15] Yandi Xia, Aaron Levine, Pradipto Das, Giuseppe Di Fabbrizio, Keiji Shinzato, and Ankur Datta. Large-scale categorization of japanese product titles using neural attention models. In *The European Chapter of the Association for Computational Linguistics (EACL 2017)*, 2017.

[16] Yijun Xiao and Kyunghyun Cho. Efficient character-level document classification by combining convolution and recurrent layers. *arXiv preprint arXiv:1602.00367*, 2016.

[17] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of NAACL-HLT*, pages 1480–1489, 2016.

[18] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.