Implicit Query Parsing for Product Search

Chen Luo
cheluo@amazon.com
Amazon Search
USA

Rahul Goutam
rgoutam@amazon.com
Amazon Search
USA

Haiyang Zhang
hhaiz@amazon.com
Amazon Search
USA

Chao Zhang
chaozhang@gatech.edu
Gatech
USA

Yangqiu Song
yqsong@cse.ust.hk
HKUST
Hong Kong

Bing Yin
alexbyin@amazon.com
Amazon Search
USA

ABSTRACT

Query Parsing aims to extract product attributes, such as color, brand, and product type, from search queries. These attributes play a crucial role in search engines for tasks such as matching, ranking, and recommendation. There are two types of attributes: explicit attributes that are mentioned explicitly in the search query, and implicit attributes that are mentioned implicitly. Existing works on query parsing do not differentiate between explicit query parsing and implicit query parsing, which limits their performance in product search engines. In this work, we demonstrate the critical importance of implicit attributes in real-world product search engines. We then present our solution for implicit query parsing at an e-commerce product search engine, which is a unified framework combining recent advancements in knowledge graph technologies and customer behavior analysis. We demonstrate the effectiveness of our proposal through offline experiments on search log data. We also show how to use the framework on an e-commerce search engine to improve customers' shopping experiences.

CCS CONCEPTS

• **Information systems** → **Information retrieval query processing**.

KEYWORDS

Query Understanding, Product Search, Implicit Parsing

ACM Reference Format:
Chen Luo, Rahul Goutam, Haiyang Zhang, Chao Zhang, Yangqiu Song, and Bing Yin. 2023. Implicit Query Parsing for Product Search. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23), July 23–27, 2023, Taipei, Taiwan.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3539618.3591858

1 INTRODUCTION

Product Search Engines are a crucial component of online shopping websites, such as Amazon, eBay, etc. Unlike in-store shopping, online customers rely on product search to find what they need. According to a recent study on an e-commerce search engine, over 90% of customers base their shopping choices on the results of the product search engine. In other words, people start their online shopping journey with the product search engine.

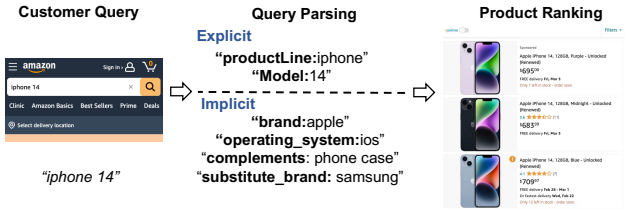

Figure 1: Query Parsing is the entry point of modern product search engine. The explicit and implicit attributes for product search queries are critical for product ranking and recommendation.

Query Understanding (QU) is the most critical component of modern product search engines, serving as the entry point for search (Fig. 1). QU models extract attributes from the customers' query, such as color, brand, etc., and the search engine relies on these attributes for ranking, recommendation, etc. For example, query intent description from query parsing can be used to improve exploratory search, and QU features can be combined with behavior features for product ranking [5].

Existing query parsing models primarily focus on extracting explicit attributes from search queries. In our previous work [6], we used weakly supervised data to build an industry-scale explicit query parsing framework for an e-commerce search. However, customer queries in product search engines are usually short, containing only three to four words on average. Customers trust the search engine to understand their implicit intents and tend to omit important concepts such as brand or author if they are implied by other elements in their query. This makes it harder for the search engine, as only 1% of customers type in "apple" when searching for "iPhone" or "MacBook Air." The lack of information in the query limits the power of the product search engine.

Implicit attributes, in contrast to explicit attributes, are attributes that are not explicitly mentioned but can be inferred from the query. For example, the query "iPhone 14" has the explicit attributes of product line "iPhone" and model number "14," but also has implicit attributes such as brand "Apple" and operating system "iOS." According to our experiments on an e-commerce search, 84% of product search queries contain high-confidence implicit information. However, customers typically do not mention this information in their

SIGIR '23, July 23–27, 2023, Taipei, Taiwan

ACM ISBN 978-1-4503-9408-6/23/07.
https://doi.org/10.1145/3539618.3591858

Example	Freq.
macbook air m2	56.9
iphone	93.1
macbook	28.7
apple macbook air m2	1
apple macbook	1
apple iphone	1

Attribute	Implicit
Brand	*54.8%*
Color	*94.1%*
Author	*92.2%*
Genre	*99.1%*

Table 1: The left table compares the relative frequency of brand queries versus non-brand queries. The right table compares the percentage of queries that have implicit attributes.

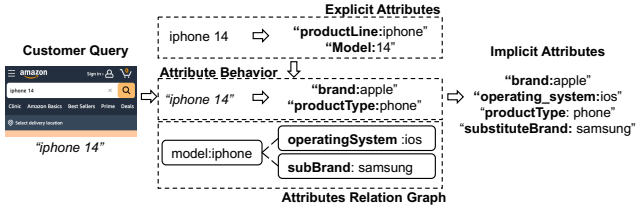

Figure 2: The query implicit attribute parsing framework at an e-commerce search.

search query keywords. Noticing that, in most e-commerce systems, such as Amazon, we have a predefined list of product attributes ontology that shows which attributes belong to which product type.

In this work, we would like to share our year-long effort to build an industrial solution for implicit query parsing at an e-commerce search. Our solution is a unified framework that combines recent advances in knowledge graph and customer behavior analysis. We show that implicit attributes are critical for query understanding. Our experiments on offline data demonstrate the effectiveness of our proposed methods, and we show how to deploy and use the framework in an e-commerce search to improve shopping experiences.

2 BACKGROUND

2.1 Implicit Attribute

Implicit query attributes are the attributes that belong to a query, but are not explicitly mentioned in the query keywords. For example, the query 'iPhone 14' has the explicit attributes of product line 'iphone' and model number '14'. However, this query also has implicit attributes such as the brand 'apple' and operating system 'ios' as shown in Figure 1.

In Table 1, we analyzed the query frequency from one day of an e-commerce search logs in August 2022. We calculated the frequency of queries that contain information about the product line, such as "macbook" or "iphone." We also calculated the query frequency of queries with both product line and brand tokens. We observed that the keywords without brand tokens are 30 to 90 times more frequent than the queries with a brand token. This observation suggests that customers do not type all the necessary information in their search queries and trust the search engine to understand their implicit intents. As a result, customers tend to omit important concepts like brand, if they are implied by other elements in the query. This omission may seem natural to humans, but it makes the task of the search engine more challenging [4].

Furthermore, we also analyzed the behavior data. We calculated the queries that do not mention an attribute in the query but whose actions (such as clicks or purchases) are directed towards one particular attribute, as detailed in section 3.2. We selected four attributes for this analysis: brand, color, author, and genre. Table 1 shows that more than 54% of the queries have a high confidence brand, even though they do not contain a brand token. For attributes like color, author, or genre, this number is even higher, exceeding 90%. Implicit attributes are critical for understanding customer invent for product search engine. As a result, in this work, we introduce a solution in a real product search engine for implicit attribute parsing. Our solution is a unified framework that combines the recent advance in knowledge graph and customer behavior analysis.

2.2 Problem Formulation

Given a query q with tokens $t_0, t_1, .., t_k$, where k is the number of tokens in the query. Let $A = a_0, a_1, .., a_n$ be the universal set of attributes, where n is the total number of attributes in the product catalog. The problem of explicit query parsing is to determine the corresponding attribute for each token, such that: $P_e(t_i) = a_i$, where P_e is a function that maps token t_i to its corresponding attribute a_i, and $a_i \in A$. In contrast, implicit query parsing aims to determine the values for every attribute in A, whether the attribute is explicitly mentioned in the query or not. Thus, we seek to find the implicit attribute parsing function P_i for q: $P_i(q) = v_0, v_1, ..., v_n$, where $v_i \in V_{a_i}$ represents a value of the attribute a_i. V_{a_i} is the set of possible attribute values for a_i. For example, if a_i is the attribute *color*, then $v_i \in V_{a_i}$ would be the set of colors.

3 METHOD

In this section, we describe our framework for implicit query parsing. Figure 2 illustrates the overall framework of our solution. The input to the system is the customer query "iPhone 14," and the output is a list of implicit attributes: 'brand: Apple', 'operating system: iOS', 'accessory: phone case', and 'substitute brand: Samsung'.

There are three components in this solution: (1) An explicit query parsing model that extracts the attributes mentioned explicitly in the search query. (2) An attribute behavior inference model that uses query behaviors to determine the attributes with high click/purchase frequency in history. (3) A knowledge graph-based model that infers implicit attributes from the explicit attributes in an attribute graph. In the following sections, we will elaborate on each of these components in detail.

3.1 Explicit Attribute Parsing

Query Attribute Parsing is the task of extracting product attributes from search queries. For instance, given the query "nike shoes," the Query Attribute Parsing model identifies "nike" as a brand token and "shoes" as a product type. To accomplish this, we have designed a Transformer-based Query Parsing Model.

We utilized human-labeled data for training the model. The data comprises of 12 different languages and over 600,000 queries, along with their ground truth tokens. We employed a multilingual

Transformer Model to handle the language-agnostic input and used the Transformer embeddings as the input for the final classification layer. Subsequently, each token is assigned a classification label to represent the attribute it belongs to. For further details regarding our explicit attribute parsing framework, please refer to [6].

3.2 Aggregation Actions for Attribute Parsing

One assumption in our query parsing approach is that if a significant majority of customer actions are directed towards a particular attribute, then that attribute is likely to be relevant to the query. For example, in the query "iphone 14," after aggregating one year of query to product clicks data, we found that 95% of clicks were on products with the brand "apple." This indicates that "apple" is one of the implicit attributes for "iphone 14."

To formally predict the relevance of each attribute value to a given query, we use three customer actions: clicks, purchases, and add to cart. We aggregate the number of these actions for each query and corresponding product over a year's worth of data. Let $A = a_0, a_1, \ldots, a_n$ be the universal set of attributes, where n is the number of attributes in the product catalog, and let $v_i \in V_{a_i}$ be a predefined attribute-value pair for a given query q. We define the confidence of the attribute value v_i for the query q as:

$$F(q, v_i) = \frac{\sum_{c \in C} w_c N_c(q, p^{v_i})}{N_I^a(q, p^{v_i}) + \lambda_q}, \tag{1}$$

where C is the set of actions taken for the query-to-corresponding-product relationship. In this work, we only consider three actions, so $C = \{click, add, purchase\}$. $N_c(q, p^{v_i})$ is the number of actions c taken for the product that contains the attribute a with a value v_i, and $N_I^a(q, p^{v_i})$ denotes the number of impressions for the query-to-product relationship. An impression refers to a customer seeing the product in the search results. The parameter λ_q controls the smoothing of the confidence score and is typically set to a small positive value. In production, we set a threshold of 0.9 for the confidence score. If the confidence score is greater than 0.9, we return the result in production.

In Equation 1, w_c denotes the weights assigned to each action in C. In this work, we assume a linear relationship between customer actions and query attribute relation. So we build a linear model to estimate the w_c for each action, and optimize the following objective function: $\min_{w_c} \sum_c (p_r - (w_c a_r))^2$, where $c \in \{click, add, purchase\}$ is the set of customer actions, a_r are the action rate of each query product pair in the data, p_r denotes the purchase rate, which calculated by number of purchase devided by the number of impression of that product in a time range. We use 1 year of time range to collect this training data. The estimation result is $c_{click} = 1.05$, $c_{add} = 6.86$, and $c_{purchase} = 4.51$.

However, in practice, we have encountered two limitations with this method: (1) We can only obtain results for queries with sufficient behavior data. For new queries or queries without behavior information, we cannot calculate Equation (1). (2) This method is biased towards the search results and customer behavior, as biases always exist in ranking systems [5]. For example, if Amazon.com only sells products from one brand for a particular query, then the attribute value will be biased towards that brand. To overcome these limitations, we combine the behavior-based method with a knowledge graph-based approach.

3.3 Product KG for Attribute Parsing

Knowledge graphs are heavily used by search engines today [1]. In this work, we generate implicit attributes based on both explicit attributes and product knowledge. For example, given the query "iphone 14" with the explicit attribute "iphone," we can check the knowledge graph and determine that the corresponding attribute, the operating system, is IOS.

In this work, the product knowledge graph is an attribute relation graph, as depicted in Fig. 2. We construct this graph by learning from two data sources in an e-commerce plateform: (1) product attribute data and (2) query attribute data. The product attribute data is extracted from the catalog, which contains information about all the products and their corresponding attributes. Sellers upload their products and input the corresponding attributes into our service, which we then automatically correct to ensure data quality [7]. The query attribute data is collected from customer search queries and their corresponding explicit attributes. We use the parsing model introduced in Section 3.1 to obtain each query's attributes.

There are thousands of attributes in real product search services, and hundreds of millions of queries are processed every day. The sheer size of this data makes processing it prohibitive. However, the attributes from different product types don't overlap much with each other. For example, the attribute "operating system" only appears in products such as phones or laptops. To address this, we split the data by product type based on a product knowledge graph [1].

To determine the relationships between attributes, we employ a Graph Neural Network (GNN) model, specifically the unsupervised GNN model proposed by Lu et al. [2]. This model generates embeddings for each attribute by considering the graph structure and relationships between the attributes. Given the attribute relation graph $G = (V, E)$, where V is the set of attribute values (e.g. color:red, brand:nike, etc) and E is the set of edges (e.g. brand:apple and productType:phone will have an edge). We build a GNN model $f : \mathcal{G} \rightarrow \mathcal{Y}$ to map the graph $\mathcal{G}$ to an output $\mathcal{Y}$. The objective function of our GNN model is:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} L(f(x^{(i)}; \theta), y^{(i)}) + \lambda R(\theta), \tag{2}$$

where θ are the parameters of the neural network, m is the number of training examples, $x^{(i)}$ is the input for the ith example, $y^{(i)}$ is the corresponding target output, f is the neural network function, L is the loss function, and R is a regularization term that penalizes large parameter values to prevent overfitting. The parameter λ controls the strength of the regularization term. In our experiment, we choose these parameters follow the ones in [2]. The GNN model is trained on large-scale attribute graph data, with the objective of predicting relationships between attributes. These relationships are derived from the co-occurrence of attributes in the same product or query, and we used more than 100 million products and 200 million queries in the an e-commerce catalog for training.

Once we get the attribute embeddings from GNN, we then use these embeddings to construct relationships between attributes. If the cosine similarity between the embeddings of two attributes is greater than 0.5, we add an edge between them. During online

inference, we use the attribute relation graph to obtain implicit attributes by performing a graph random walk through the graph. Given the explicit query attributes and behavior-implicit attribute, we combine the results of the knowledge graph-based implicit attribute and the behavior-based implicit attribute to produce the final output, as shown in Fig. 2.

4 OFFLINE EXPERIMENT

Dataset: We sampled approximately 45 million queries from an e-commerce search logs as query set. We sample these queries from English language countries and the time range of the sampled queries expanded from one year.

Following the evaluation strategies outlined in [3], three sub-evaluation datasets were sampled from three different buckets: Normal Queries (NQ), the high-frequency queries; Hard Queries (HQ), queries sampled from the middle tercile based on frequency; Long tail queries (LTQ), queries have the lowest frequency of the tercile. These queries were randomly selected from the search logs in one month. Each of the three sets contains 500 queries. The results for each query were obtained from various methods, and a group of highly trained human judges was used to assign a binary relevance grade (correct or wrong) to each returned attribute with respect to the original query's intent. This grade was used to calculate the performance metrics of each method.

Comparising Methods: We choose two baseline methods: **KG-IA**, the knowledge graph based implicit attribute. We infer the implicit attribute based on the explicit attribute and the attribute relations extracted in KG. **Behavir-IA:** We use the behavior based method to extract the implicit attributes. In the experiment, we extracted the following attributes: author, brand, product type, color, and gender.

Metrics: Two commonly used metrics were adopted for offline evaluation: Precision and Coverage. Precision is the percentage of returned implicit attributes that are correct. Coverage denotes the number of queries that contain the implicit attributes returned by the method. To compute precision, we utilized human judgments of relevance. To calculate the coverage, we used real e-commerce search traffic to see how many queries have implicit attributes inferred.

Overall Performance: The results for all methods under the two metrics are presented in Table 2. The proposed solution performed the best compared to other methods across all three datasets. Specifically, the proposed solution showed a relative performance gain of 11.7% in Precision and 16.1% in Coverage over the best baselines, on average across the three datasets. In summary, the proposed solution shows strong performance in extracting implicit attributes, making it a compelling solution for query understanding services in product search engines.

5 SYSTEM DEPLOYMENT AND IMPACT

We tested the system within a product search engine for product navigation and related query recommendation and proven to provide better shopping experience (Fig. 3).

Implicit Attribute for Search Navigation We tested the implicit attribute extraction framework to provide navigational attribute choices. In product search engine, navigation is one of the critical components. Navigation provides navigational choices

Data	Metrics	KG-IA	B-IA	Proposed
NQ	Precision	0.85±.01	0.83±.00	1.0±.00
	Coverage	0.32±.04	0.62±.04	1.0±.00
HQ	Precision	0.84±.01	0.82±.03	1.0±.00
	Coverage	0.28±.09	0.53±.05	1.0±.00
LTQ	Precision	0.84±.03	0.73±.06	1.0±.00
	Coverage	0.20±.04	0.50±.03	1.0±.00

Table 2: Offline Experiment Results. We report the relative values for all the metrics due to the privacy concern on the data.

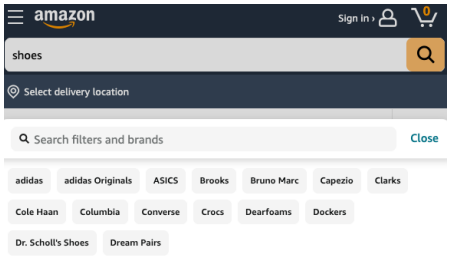

Figure 3: We use implicit attributes to suggest navigation options for our customers. Customers are able to choose from different attribute suggestions to narrow down their search and engage in exploratory shopping.

when customers shop for broad queries. For example, when a customer shops for 'shoes', we provide brand navigation, color navigation. In this work, we use the proposed framework to generate implicit attributes for the broad queries as the navigation choices. We ran an online A/B experiment on the product engine to test the impact on the user experience.

In the online experiment, users in the treatment group saw the attribute navigation choices generated by the proposed solution, while the control group saw the standard search results. We measured business metrics such as revenue and purchased units, and observed a big increase in both revenue and purchased units ($p < 0.05$). Our system did a better job in providing more relevant results and significantly improved several business metrics.

Implicit Attribute for Related Brand We also tested the implicit attribute framework for related search choice generation. Implicit attributes provide additional attributes that are not mentioned in the search query. In this experience, we added the implicit attribute to the query keywords and generated a new search query.

We ran an online A/B experiment on the an e-commerce search engine to test the impact on the user experience. In the online experiment, we rewrote the queries for users in the treatment group and retrieved a new search in the search engine, targeting only the queries that had a bad search result (queries with zero search results returned by the search engine). We tested this system in the an e-commerce product search engine and measured the zero search result rate. We observed a big decrease in the zero search result rate ($p < 0.05$). By using the proposed solution, the zero search result rate defect rate decreased significantly.

6 CONCLUSION

In this work, we demonstrate empirically that implicit query parsing is critical in real product search engines. We then present our solution for implicit query parsing in an e-commerce search. Our solution is a unified framework that leverages recent advancements in knowledge graph reasoning, as well as customer behavior analysis. The results of our offline data experiment highlight the effectiveness of our proposed methods. Additionally, we provide insight into the deployment and utilization of the framework in an e-commerce search, with the aim of enhancing the shopping experience.

REFERENCES

[1] Xin Luna Dong, Xiang He, Andrey Kan, Xian Li, Yan Liang, Jun Ma, Yifan Ethan Xu, Chenwei Zhang, Tong Zhao, Gabriel Blanco Saldana, et al. 2020. AutoKnow: Self-driving knowledge collection for products of thousands of types. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2724–2734.

[2] Hanqing Lu, Yunwen Xu, Qingyu Yin, Tianyu Cao, Boris Aleksandrovsky, Yiwei Song, Xianlong Fan, and Bing Yin. 2021. Unsupervised Synonym Extraction for Document Enhancement in E-commerce Search. (2021).

[3] Chen Luo, Vihan Lakshman, Anshumali Shrivastava, Tianyu Cao, Sreyashi Nag, Rahul Goutam, Hanqing Lu, Yiwei Song, and Bing Yin. 2022. ROSE: Robust Caches for Amazon Product Search. (2022).

[4] Parikshit Sondhi, Mohit Sharma, Pranam Kolari, and ChengXiang Zhai. 2018. A Taxonomy of Queries for E-commerce Search. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 1245–1248.

[5] Tao Yang, Chen Luo, Hanqing Lu, Parth Gupta, Bing Yin, and Qingyao Ai. 2022. Can Clicks Be Both Labels and Features? Unbiased Behavior Feature Collection and Uncertainty-Aware Learning to Rank. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Madrid, Spain) *(SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 6–17. https://doi.org/10.1145/3477495.3531948

[6] Danqing Zhang, Zheng Li, Tianyu Cao, Chen Luo, Tony Wu, Hanqing Lu, Yiwei Song, Bing Yin, Tuo Zhao, and Qiang Yang. 2021. QUEACO: Borrowing Treasures from Weakly-labeled Behavior Data for Query Attribute Value Extraction. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 4362–4372.

[7] Wen Zhang, Yanbin Lu, Bella Dubrov, Zhi Xu, Shang Shang, and Emilio Maldonado. 2021. Deep Hierarchical Product Classification Based on Pre-Trained Multilingual Knowledge. (2021).