AVEN-GR: Attribute Value Extraction and Normalization using product GRaphs

Donato Crisostomi (*)
Amazon
Sapienza University Of Rome
crisostomi@di.uniroma1.it

Thomas Ricatte (*)
Amazon
tricatte@amazon.com

Abstract

Getting a good understanding of the user intent is vital for e-commerce applications to surface the right product to a given customer query. Query Understanding (QU) systems are essential for this purpose, and many e-commerce providers are working on complex solutions that need to be data efficient and able to capture early emerging market trends. Query Attribute Understanding (QAU) is a sub-component of QU that involves extracting named attributes from user queries and linking them to existing e-commerce entities such as brand, material, color, etc. While extracting named entities from text has been extensively explored in the literature, QAU requires specific attention due to the nature of the queries, which are often short, noisy, ambiguous, and constantly evolving. This paper makes three contributions to QAU. First, we propose a novel end-to-end approach that jointly solves Named Entity Recognition (NER) and Entity Linking (NEL) and enables open-world reasoning for QAU. Second, we introduce a novel method for utilizing product graphs to enhance the representation of query entities. Finally, we present a new dataset constructed from public sources that can be used to evaluate the performance of future QAU systems.

1 Introduction

Search queries are the main point of interaction between the customer and the search system. As such, extracting information from the queries is pivotal in surfacing the relevant products, making the task directly responsible for the quality of the overall customer experience. Query Understanding (QU) not only inherits all the challenges of standard natural language understanding but poses additional difficulties: queries are short and lack context, which makes them challenging to understand. They often contain implicit knowledge that is difficult to capture without external reference. For example, the query "M2 laptop" refers to Apple laptops since M2 processors are only sold by Apple. Furthermore, customers do not have technical writing skills, which can result in queries that are noisy or use inappropriate search terms.

In this work, we focus on the task of Query Attribute Understanding (QAU), which aims to extract the attribute values from the queries and make them usable for other downstream applications in the Search Engine (see fig. 1). QAU is related to another important task, Document Attribute Understanding (DAU), which aims to extract attributes from product descriptions. DAU has received significant attention from the community in the past years ((Zheng et al., 2018; Xu et al., 2019; Dong et al., 2020; Karamanolakis et al., 2020)) and does not suffer from the difficulties mentioned above and that are specific to queries. Both QAU and DAU are specific instances of Named Entity Recognition and Linking (NER/NEL), which aims to extract typed mentions from text. However, in contrast to classic NER, which usually handles fewer attribute types (such as Person, Location, and Organization), QAU and DAU deal with a larger number of attribute types (which can reach thousands in e-commerce as noted in (Xu et al., 2019)).

We claim that three critical elements need to be addressed to get a practical solution to QAU. Firstly, named entity recognition should be performed jointly with entity linking, in order to map the detected entities to our knowledge base. Solving these tasks separately is not practical in an industrial context, as it leads to error propagation (linking module cannot make up for a wrong attribute prediction by the NER module) and more generally hidden technical debt (see (Sculley et al., 2015)). Furthermore, separating the tasks precludes the possibility of inductive transfer, which has been shown to be crucial in related tasks (Zhang and Yang, 2021; Caruana, 1997; Ruder, 2017).

[*]These authors contributed equally to this work

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 5: Industry Track, pages 126–133

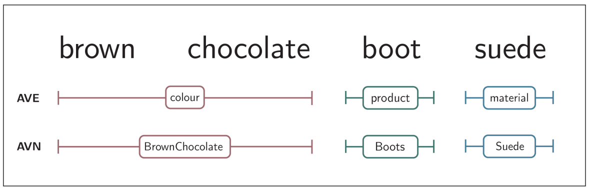

Figure 1: Overview of the task. We ultimately want to detect that this query contains three mentions: (brown chocolate), (boot) and (suede). The first annotation row shows the ground truth for the attribute value extraction task, while the second one shows that of the normalization step, which may be understood as entity linking over the detected mentions.

Secondly, product graphs (PG) are becoming a new standard to represent e-commerce concepts and the relations between searchable products. Therefore, QAU systems should be able to leverage this new source of knowledge to improve their performance. Finally, QAU systems should always be designed with an "open-world" setup in mind to deal dynamically with new concepts. For instance, if we consider the query '*Sony A95K TV*', we should be able to detect that '*A95K*' is a mention representing a product line even if this product does not exist in our knowledge base.

Note that extreme classification (Jain et al., 2016) is a possible alternative to classic NER/NER stacking, but it does not consider the coarse-grained nature of attributes (entities belong to different attribute types) and does not easily take into account the open-world nature of the task. Users can search for attribute values that are not yet in the knowledge base or not associated with any known product, making it difficult to predict normalized attribute values directly.

Overview of our approach

To overcome the aforementioned limitations of existing approaches, we propose an *end-to-end multitask approach* that jointly predicts mentions, attribute types and entities. We build a shared representation of the text spans via a pre-trained transformer architecture (Liu et al., 2019). The shared span representation is used to determine the probability of the span being a mention, containing a particular attribute type, and representing a specific entity instance of that attribute. Our method can handle an open-world scenario where an attribute value does not have a matching entity in the knowledge base. In such cases, the model can still predict the attribute type of the value. Note that this approach is also data-efficient and can effectively utilize weakly labeled data points without entity annotations. Importantly, the end-to-end approach avoids error propagation since the entity-level prediction is conditioned but not solely reliant on the attribute-level information. Additionally, our approach can handle overlapping spans without requiring additional adjustments. Finally, if the entities are structured in a knowledge graph, our approach can leverage its topology to enrich the entity embeddings.

In order for our approach to be tested in scenarios with varying difficulties we need a dataset of queries of controllable complexity, along with a knowledge graph involving the entities there mentioned. To this end, we propose leveraging the products in the *Amazon Berkeley Objects* dataset (Collins et al., 2022) to construct a knowledge graph consisting of products related to their attribute values by relations encoding the attribute type. The product graph is used both as knowledge base for the approach and as starting artifact to generate a dataset of public synthetic queries. As public, non-confidential resources, we aim to release both artifacts for reproducibility and to encourage research in the field. Summarizing, our contributions are three-fold:

1. we propose AVEN, a novel end-to-end method that can effectively solve QAU in an open world setting;

2. we propose a way to use Product Graph to enrich the representation of the entities

3. we present a novel evaluation that combines a public product graph with a set of synthetic queries involving associated entities, aimed at promoting research on knowledge-based methods for QAU.

2 Related work

Document Attribute Understanding As previously noted, Query Attribute Understanding (QAU) shares similarities with Document Attribute Understanding (DAU), which has been previously addressed in the literature. (Zheng et al., 2018) proposed an early solution based on a classic NER pipeline that assigns each attribute type with a set of BIO (Beginning, Inside, Outside) tags. However, this approach suffers from scalability issues when dealing with a large number of attributes, and also hinders data sharing between head attributes (such as color) and tail attributes (such as glass color).

To solve this issue, several approaches (Xu et al., 2019; Dong et al., 2020) based on Question Answering were pushed in the subsequent years. These approaches consider each attribute as a separate question to be answered leveraging the product description. The main advantage of Question Answering approaches is that they do not require a specific set of BIO tags for each attribute and are therefore more scalable. However, they are also harder to train and highly depend on the semantic representations of the attribute types. In practical cases in which the detected entity mentions must also be linked to normalized entities, Entity Linking is performed independently over the output of the NER step. While all these works consider an attribute value to be just a span of unstructured text, we aim to directly obtain normalized entities as attribute values, hence requiring performing Entity Linking over the detected spans.

Entity Linking Entity Linking has been mostly studied in scenarios involving long documents with lot of context, while only few works exist for short sentences like queries. Most relevant to our work is ELQ (Li et al., 2020), in which a bi-encoder is employed to jointly perform mention detection and EL in a multi-task setup. Analogously, in Oliya et al. (2021) mention detection and entity linking are coupled with question answering in an end-to-end pipeline. We take inspiration from both works to tackle AVEN by injecting a new stage in the end-to-end mention detection and entity linking pipeline, responsible for classifying the span attribute.

Query Attribute Understanding While it may be tempting to view Query Attribute Understanding (QAU) as a simplified version of Document Attribute Understanding (DAU), this assumption overlooks the unique challenges posed by queries, such as their inherent noisiness, lack of context, and ambiguity. To the best of our knowledge, the only existing work that deals with both attribute value extraction and subsequent entity linking is QUEACO (Zhang et al., 2021). Differently from our approach, QUEACO is a fragmented model that stacks a user-behavior based normalization module over a NER pipeline. While we use user behavior in the data collection, we don't require it for the training and inference pipelines.

3 Data

In order to have a controlled ground for experimentation, we need (i) a dataset of user queries, and (ii) a Knowledge Graph containing most entities involved in the user queries. Knowledge Graphs involving products and relative information are usually called product graphs.

3.1 Product Graphs

A Product Graph is a Knowledge Graph involving a set of products and their corresponding attributes. Formally, it is a bipartite graph consisting of a vertex set $V = (P \cup A)$ containing products P and attribute values A connected by edges $E = R_1 \cup R_2 \cup \cdots \cup R_m$, where $R_1, R_2, \ldots, R_m$ are set of edges for the different m attribute types. In practice, a triple (p, r, a) relates a product p with an attribute value a through an attribute relation r.

3.2 Synthetic data

Given the lack of a public Product Graph, we constructed one by leveraging the *Amazon Berkeley Objects* (ABO)[1] dataset (Collins et al., 2022). The constructed graph not only lends itself to the overall inference pipeline, but can also be used to generate a set of synthetic queries that involve the entities of interest by construction. The generation procedure simply constructs queries as bag of attributes by starting from product nodes and walking the relations related to the attributes of interest, then discarding the product node in the final query and only keeping its attribute values along with the attribute type annotations. The generation pipeline is formalized in appendix C. To increase the complexity of the dataset, we also replace product types with synonyms found in the same *WordNet* synsets (Fellbaum, 1998).

3.3 Real user queries

Given the huge number of possible attribute values, manual annotation of user queries with attribute and entity-level labels is unfeasible. For this reason, we leverage a pre-trained NER model to obtain the attribute-level labels and employ a deterministic heuristic to label the corresponding attribute values with entity-level annotations. Let P be a set of purchased items, and Q be the queries that led to the purchase. First, we create a Product Graph PG

[1] https://amazon-berkeley-objects.s3.amazonaws.com/index.html

from P by creating a triple (p, r, a) for each product p connected to an attribute value a through attribute type r. Then, for each query $q \in Q$, we iterate over each NER-annotated span (r, v, s), where span s holds value v for attribute type r. We now want to annotate the span s with two annotations, one at the attribute level and one at the entity level. For the former, we can keep the one detected from NER r. For the entity-level annotation instead, we choose to annotate s with the entity a such that $(p, r, a) \in PG$. In other words, given that NER has predicted the span to refer to an attribute type r, we annotate the span with the entity corresponding to the attribute value for r of the product that the user bought after searching for the query q. Assume for instance that an user looked for '*red Nike shoes*' and eventually bought some product p referring to a specific pair of shoes that are, in fact, red. In this case, the span $s_{0,1}$ with value '*red*' can be annotated to be a color as predicted by NER, while the entity label will be that of the value for p for the attribute color, which is the node corresponding to the value '*red*' in the knowledge base. Of course, the user may also have eventually bought a black pair of shoes instead: in this case, the heuristic makes a mistake, and therefore the annotation is expected to be noisy. Nevertheless, assuming the query keywords to encode strong preferences when present, these cases are expected to be rare enough for the model to eventually learn to discard them as noise.

4 Approach

The overall architecture of AVEN contains three different sub-modules, each responsible for a different task: (i) a mention detection module; (ii) an attribute classification module; (iii) an entity disambiguation module.

The three modules are learnt jointly as shown in fig. 2 and each of them contributes to the final loss. The latter is obtained as a weighted sum of the three losses. While the coefficients are currently set to 1 for all the three tasks, we aim to eventually use GradNorm (Chen et al., 2018) to tune the loss weights.

More formally, let us define $q = q_1, \dots, q_n$ as an input query with n tokens/words. We denote by $s_{[i,j]}$ the sub-span $q_i q_{i+1} \dots q_j$. We are interested in three different quantities: M_{ij} refers to span $s_{[i,j]}$ being a mention, A^a_{ij} refers to the same span being an attribute value for attribute a, and finally E^e_{ij} refers to $s_{[i,j]}$ being an instance of entity e. In the next sections, we will review the three different components.

Mention Detection

For a span $s_{[i,j]}$, we denote the span embedding by $\mathbf{s}_{ij} = f_\theta(s_{[i,j]})$. A simple version of $f_\theta(s_{[i,j]})$ is the mean of the RoBERTa (Liu et al., 2019) embeddings of the tokens in $s_{[i,j]}$. We can define the probability of span $s_{[i,j]}$ being an actual mention to be

$$\mathrm{P}\left(M_{ij}\right) = \sigma\left(g_\mu(\mathbf{s}_{ij})\right) \quad ,$$

where σ is the sigmoid function and $g_\mu(\cdot)$ is a parametric function taking in input the span representation and returning an unnormalized score. In our current implementation, this is realized as a Multi-Layer Perceptron (MLP). Note that we employ the sigmoid as we assume that the probability of a span $s_{[i,j]}$ to be a mention does not depend on the probability of another span $s_{[k,l]}$ to be a mention. Note that, this assumption is questionable, especially as soon as $s_{[i,j]}$ and $s_{[k,l]}$ have a non-null intersection. Nevertheless, this choice allows the model to detect overlapping spans when faced with cases such as those exemplified in section 1. Note that it's always possible to add a *Non-Maximum Suppression* (NMS) step if we want to avoid producing overlapping annotations. The mention detector is trained by minimizing a *Binary Cross Entropy* loss ℓ_{MD}.

Attribute classification

We are now interested in the probability that a span $s_{[i,j]}$ has attribute type a knowing that it is a mention

$$\mathrm{P}\left(A^{(a)}_{i,j} \mid M_{ij}\right) = \frac{\exp\left(h^{(a)}_\nu(\mathbf{s}_{i,j})\right)}{\sum_{a' \in A} \exp\left(h^{(a')}_\nu(\mathbf{s}_{i,j})\right)},$$

where $h^{(\cdot)}_\nu()$ is a parametric function taking into input the span representation. As for the mention detector, we employ a MLP. Note that we adopt a multi-task approach where we use the exact same span representation for the three different tasks, fostering information transfer among the latter. The attribute classifier is trained with a simple cross entropy loss ℓ_{AC} and only considers actual ground-truth mentions at train time.

Entity disambiguation

In the entity disambiguation module, our goal is to estimate the probability $\mathrm{P}\left(E^{(e)}_{ij} \mid M_{ij}\right)$. Given the

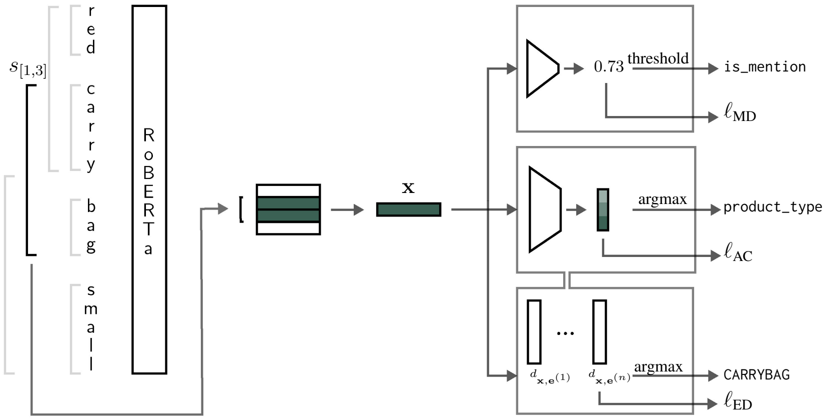

Figure 2: Our multi-task architecture with the three corresponding losses ℓ_{MD}, ℓ_{AC} and ℓ_{ED}

fact that each entity e is associated with an unique attribute type $a = type\,(e)$, we can argue that this is actually equivalent to estimating the joint probability $\mathrm{P}\left(E_{ij}^{(e)}, A_{ij}^{(a)} \mid M_{ij}\right)$ Since the probability of a span to be type a is already given by the attribute classifier, we can just estimate for each possible attribute a

$$\mathrm{P}\left(E_{ij}^{(e)} \mid A_{ij}^{(a)}, M_{ij}\right) = \frac{\exp\left(v_{\xi}^{(a)}(\mathbf{s}_{i,j}, \mathbf{e})\right)}{\sum_{e' \in E} \exp\left(v_{\xi}^{(a)}(\mathbf{s}_{i,j}, \mathbf{e}')\right)},$$

where $v_{\xi}^{(a)}(\cdot)$ is a parametric function taking into input the span representation and the entity e to be scored. The main advantage of this last expression is that it allows us to adopt a *divide-and-conquer* approach since for each attribute a, we only have to consider its compatible entities. Similarly to the attribute classifier, the entity disambiguator is learnt with a simple cross entropy loss ℓ_{ED} on actual groundtruth mentions. Our first implementation of $v_{\xi}^{(a)}(\cdot)$ is a simple similarity scorer between the span representation and the embedding of the considered entity. Entity embeddings are computed by embedding a textual representation of their neighborhood in the knowledge graph, as illustrated in Figure 3.

Inference

To compute the probability of each span $s_{[i,j]}$ being a mention of entity e at inference time, we simply multiply the mention probability by the entity classification score. To improve efficiency, we exclude all spans $s_{[i,j]}$ with a mention probability $\mathrm{P}\left(M_{ij}\right)$ lower than a pre-defined threshold p_{min}, such as 0.5 in our experiments.

Advantages

Our approach shares the span representation across all three tasks: mention detection, attribute classification, and entity disambiguation, benefiting from the effectiveness of multi-task learning (Caruana, 1997; Ruder, 2017) in transferring knowledge between similar tasks. This is particularly relevant for our method as the tasks require different levels of label details: mention detection only requires weak labeling, while the attribute/entity tasks rely on associations between mentions and knowledge graph entities. Sharing the representation allows the entity disambiguation module to leverage weakly-labeled mention data, leading to a more data-efficient approach.

5 Experimental Results

In this section, we present experimental results on two datasets described in section 3.2 and section 3.3. We provide a brief overview of the protocol used in both cases.

5.1 Considered metrics

Mention Detection

We report both (micro) `Precision` and `Recall` for the mention detection task to validate the performance of the mention detector. The percentage of recalled mention will be a natural upper bound for the following metrics on attribute classification. Indeed, if we are not able to retrieve a mention, we will consider that we cannot be right at the subsequent tasks.

Attribute Classification

We report the multiclass `Accuracy` for the attribute classification task; This metric is computed on the set of ground-truths mentions and thus ignoring

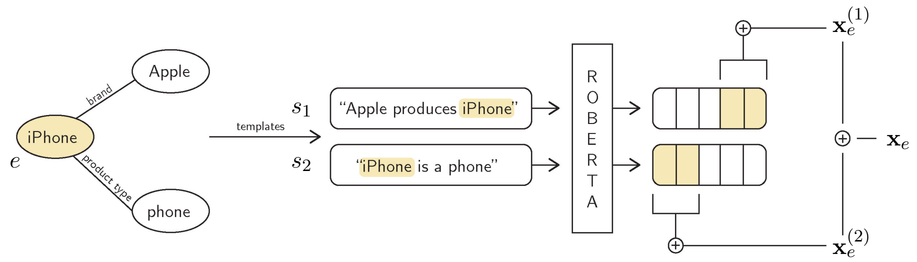

Figure 3: We use predefined templates to format encoded relations in the graph into natural language sentences for each entity. These sentences are then embedded using RoBERTa (Liu et al., 2019) to obtain an in-sentence representation, which is further averaged to obtain an overall entity representation.

wrongly detected mentions (for which no attribute exists). We also present a complementary version of this metric, which focuses exclusively on ground-truth mentions that contain previously unseen "unknown" entities. This metric is only applicable to the second, more realistic dataset that includes novel entities in the test set.

Entity Disambiguation

We report the multiclass `Accuracy` for the entity disambiguation task; This one is computed only on the subset of ground-truth mentions containing entities seen at train time.

5.2 Baselines

We consider the following models (i) **NER+Dict**: A RoBERTa-based NER baseline with dictionary lookup over the detected attributes. (ii) **NER+NN**: A RoBERTa-based NER baseline with nearest neighbor between detected attribute embeddings and entity embeddings. (iii) **AVEN/AVEN-NC/AVEN-GR**: Our end–to-end approach in three different flavours: with plain entity embedding, with plain entity embedding and no contextual span embedding and with product-graph based embeddings.

5.3 Results

We report in fig. 4 (resp. fig. 5) the results from the synthetic dataset described in section 3.2 (resp. the actual user queries described in section 3.3). Overall, our AVEN- methods outperform the "stacked" methods (NER + separate entity linker), particularly on the task of known entity classification. Among our methods, AVEN-NC has a lower mention recall due to the lack of contextual span embedding. However, our methods are effective in predicting the attribute type of unseen entities, as

Model	**Mention**		**Attribute**	**Entity**
	Precision	Recall	Accuracy	Accuracy
`NER+Dict`	98.5	97.9	97.6	68.3
`NER+NN`	98.1	97.7	97.5	64.3
`AVEN`	95.2	93.5	93.3	83.2
`AVEN-NC`	69.8	96.2	95.3	89.5
`AVEN-GR`	97.8	97.4	97.2	76.3

Figure 4: Results on synthetic data (see section 3.2)

Model	**Mention**		**Attribute**		**Entity**
	Precision	Recall	Acc.	Acc. (unseen)	Acc.
`NER+Dict`	89.9	93.6	93.2	88.1	81.5
`NER+NN`	91.6	92.5	92.3	86.6	81.9
`AVEN`	96.3	94.0	90.2	89.4	93.0
`AVEN-NC`	88.2	93.8	91.5	82.4	95.3
`AVEN-GR`	96.0	95.4	93.0	89.7	93.9

Figure 5: Results on real user queries (see section 3.3)

evidenced by their performance on this task. It is worth noting that the attribute classification performance is lower for unseen attributes, which is expected.

6 Conclusions and future directions

In this paper, we introduced a novel approach to tackle QAU in a multi-task fashion. We demonstrated its effectiveness on two datasets, compared to some simple baselines. However, further ablation studies on more datasets / baselines (e.g. Ayoola et al. 2022) are necessary to assess its generalization power. Additionally, future work will focus on improving the multitasking efficiency of AVEN, for instance by implementing (Chen et al., 2018).

References

Tom Ayoola, Shubhi Tyagi, Joseph Fisher, Christos Christodoulopoulos, and Andrea Pierleoni. 2022. ReFinED: An efficient zero-shot-capable approach to end-to-end entity linking. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Track*, pages 209–220, Hybrid: Seattle, Washington + Online. Association for Computational Linguistics.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 794–803. PMLR.

Jasmine Collins, Shubham Goel, Kenan Deng, Achleshwar Luthra, Leon Xu, Erhan Gundogdu, Xi Zhang, Tomas F Yago Vicente, Thomas Dideriksen, Himanshu Arora, Matthieu Guillaumin, and Jitendra Malik. 2022. Abo: Dataset and benchmarks for real-world 3d object understanding. *CVPR*.

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*, pages 2724–2734.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Himanshu Jain, Yashoteja Prabhu, and Manik Varma. 2016. Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 935–944, New York, NY, USA. Association for Computing Machinery.

Giannis Karamanolakis, Jun Ma, and Xin Luna Dong. 2020. Txtract: Taxonomy-aware knowledge extraction for thousands of product categories. *arXiv preprint arXiv:2004.13852*.

Belinda Z Li, Sewon Min, Srinivasan Iyer, Yashar Mehdad, and Wen-tau Yih. 2020. Efficient one-pass end-to-end entity linking for questions. *arXiv preprint arXiv:2010.02413*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Armin Oliya, Amir Saffari, Priyanka Sen, and Tom Ayoola. 2021. End-to-end entity resolution and question answering using differentiable knowledge graphs. *arXiv preprint arXiv:2109.05817*.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. 2015. Hidden technical debt in machine learning systems. *Advances in neural information processing systems*, 28.

Huimin Xu, Wenting Wang, Xinnian Mao, Xinyu Jiang, and Man Lan. 2019. Scaling up open tagging from tens to thousands: Comprehension empowered attribute value extraction from product title. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5214–5223.

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*, pages 4362–4372.

Yu Zhang and Qiang Yang. 2021. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1.

Guineng Zheng, Subhabrata Mukherjee, Xin Luna Dong, and Feifei Li. 2018. Opentag: Open attribute value extraction from product profiles. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1049–1058.

A Limitations

Despite the promising results achieved by our approach, some limitations must be acknowledged. First, the use of product graphs as a knowledge source is a double-edged sword. Indeed, while it provides a valuable resource to exploit, the constant evolution of product graphs may create a strong coupling between the algorithm and the knowledge source, thus reducing the method's robustness over time. Second, our method's span-based approach makes it computationally expensive, requiring setting a maximum span size to circumvent this issue

B Ethics Statement

Our approach aims to boost the effectiveness of e-commerce search engines. However, by jointly optimizing multiple tasks, we run the risk of creating a less transparent system that could be susceptible to biases. These biases may lead to certain less frequent entities being overlooked or misclassified as more common ones, thereby reducing the overall fairness and accuracy of the system.

C Synthetic queries generation

Algorithm 1 outlines the synthetic query generation procedure.

Algorithm 1 Synthetic queries generation.

procedure GENERATE QUERIES(pg: ProductGraph)
  $P \leftarrow$ pg.products
  $A_{cons} \leftarrow$ considered attributes
  $Q \leftarrow []$ ▷ queries
  for all product p in P **do**
    $A_p \leftarrow []$ ▷ attributes for the product
    $T \leftarrow$ all triples $(p, *, *)$ in pg
    for all triple (p, a, r) in T **do**
      if a in A_{cons} **then**
        $A_p \leftarrow A_p \cup a$ ▷ attribute values
        $R_p \leftarrow R_p \cup r$ ▷ attribute types
      end if
    end for
    shuffle A_p and R_p accordingly
    $q_{text} = str(A_p)$ ▷ query is a bag of attribute values
    $q_{ann} = R_p$ ▷ annotations
  end for
  return Q
end procedure

D Prediction inspection

We present in fig. 6, an example of our qualitative evaluation within the QAU framework we have presented.

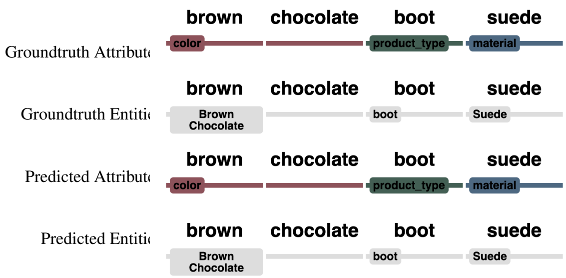

Figure 6: Predictions over one sample, with each row consisting of query text and corresponding annotations. The first two rows represent ground truth attributes and entities, while the last two represent predicted attributes and entities.