Simple and Effective Knowledge-Driven Query Expansion for QA-Based Product Attribute Extraction

Keiji Shinzato
Rakuten Institute of Technology,
Rakuten Group Inc.
keiji.shinzato@rakuten.com

Naoki Yoshinaga
Institute of Industrial Science,
the University of Tokyo
ynaga@iis.u-tokyo.ac.jp

Yandi Xia
Rakuten Institute of Technology,
Rakuten Group Inc.
yandi.xia@rakuten.com

Wei-Te Chen
Rakuten Institute of Technology,
Rakuten Group Inc.
weite.chen@rakuten.com

Abstract

A key challenge in attribute value extraction (AVE) from e-commerce sites is how to handle a large number of attributes for diverse products. Although this challenge is partially addressed by a question answering (QA) approach which finds a value in product data for a given query (attribute), it does not work effectively for rare and ambiguous queries. We thus propose simple knowledge-driven query expansion based on possible answers (values) of a query (attribute) for QA-based AVE. We retrieve values of a query (attribute) from the training data to expand the query. We train a model with two tricks, knowledge dropout and knowledge token mixing, which mimic the imperfection of the value knowledge in testing. Experimental results on our cleaned version of AliExpress dataset show that our method improves the performance of AVE (+6.08 macro F_1), especially for rare and ambiguous attributes (+7.82 and +6.86 macro F_1, respectively).

1 Introduction

One of the most challenging problems in attribute value extraction (AVE) from e-commerce sites is a data sparseness problem caused by the diversity of attributes.[1] To alleviate the data sparseness problem, recent researches (Xu et al., 2019; Wang et al., 2020) formalize the task as question answering (QA) to exploit the similarity of attributes via representation learning. Specifically, the QA-based AVE takes an attribute name as *query* and product data as *context*, and attempts to extract the value from the context. Although this approach mitigates the data sparseness problem, performance depends on the quality of query representations (Li et al., 2020).
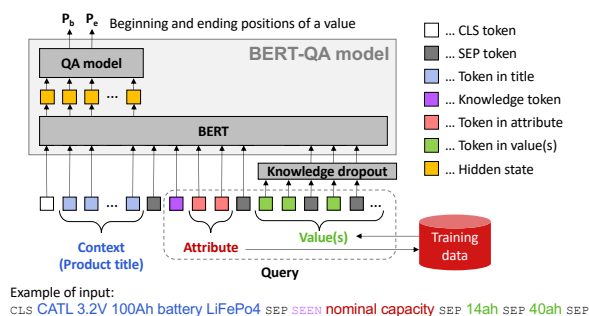

Figure 1: Our knowledge-based BERT-QA model for attribute value extraction.

Because attribute names are short and ambiguous as queries, the extraction performance drops significantly for rare attributes with ambiguous names (*e.g.*, *sort*) which do not represent their values well.

Aiming to perform more accurate QA-based AVE for rare and ambiguous attributes, we propose simple query expansion that exploits values for the attribute as knowledge to learn better query representations (Figure 1, § 3). We first retrieve possible values of each attribute from the training data, and then use the obtained values to augment the query (attribute). Since unseen values and attributes will appear in evaluation, we apply dropout to the seen values to mimic the incompleteness of the knowledge (§ 3.2), and perform multi-domain learning to capture the absence of the knowledge (§ 3.3).

We demonstrate the effectiveness of the query expansion for BERT-based AVE model (Wang et al., 2020) using the AliExpress dataset[2] released by Xu et al. (2019) (§ 4). In the evaluation process, we found near-duplicated data in this dataset. We thus construct, from this dataset, a more reliable dataset called cleaned AE-pub to evaluate our method.

---

[1] AliExpress.com classifies products in the Sports & Entertainment category using 77,699 attributes (Xu et al., 2019).

[2] https://github.com/lanmanok/ACL19_Scaling_Up_Open_Tagging

Our contribution is threefold:

- We proposed knowledge-driven query expansion for QA-based AVE (§ 3); the knowledge taken from the training data is valuable (§ 4.3).

- We revealed that rare, ambiguous attributes deteriorate the performance of QA-based AVE in the e-commerce domain (§ 4.3).

- We will release our cleaned version of AliExpress dataset for research purposes.

2 Related Work

Attribute value extraction has been modeled as a sequence labeling problem (Putthividhya and Hu, 2011; Shinzato and Sekine, 2013; More, 2016; Zheng et al., 2018; Rezk et al., 2019; Karamanolakis et al., 2020; Dong et al., 2020; Zhu et al., 2020; Mehta et al., 2021; Jain et al., 2021; Yan et al., 2021). However, since the number of attributes can exceed ten thousand in e-commerce sites, the models perform poorly for the majority of attributes that rarely appear in the labeled data (Xu et al., 2019).

To alleviate the data sparseness problem, Xu et al. (2019) introduced a QA-based approach for the AVE task. It separately encodes product titles and attributes using BERT (Devlin et al., 2019) and bi-directional long-short term memory (Hochreiter and Schmidhuber, 1997), and then combines the resulting vectors via an attention layer to learn spans of values for the attributes from the titles. Wang et al. (2020) proposed a purely BERT-based model, which feeds a string concatenating the given title and attribute to BERT. These QA-based AVE models, however, do not fully enjoy the advantage of the QA model, since attribute queries are much shorter than sentential questions in the original QA task.

To build better queries in solving named entity recognition via QA, Li et al. (2020) exploited annotation guideline notes for named entity classes as queries. Although this approach will be also effective for QA-based AVE, it requires substantial labors to prepare manual annotations for more than ten thousand attributes in e-commerce site.

3 Proposed Method

This section proposes a simple but effective query expansion method for QA-based AVE (Wang et al., 2020) by utilizing attribute values. Given a product data (title) $x = \{x_1, ..., x_n\}$ and an attribute $a = \{a_1, ..., a_m\}$, where n and m denote the number of

tokens, the model returns the beginning position, P_b, and ending position, P_e, of a value.

Figure 1 depicts the model architecture with our approach. Although our query expansion is essentially applicable to any QA-based AVE models, we here employ the state-of-the-art model using BERT proposed by Wang et al. (2020). In addition to the QA component for AVE, their model has other two components; the no-answer classifier and the distilled masked language model. Since those components slightly decrease the overall micro F_1, we employ the QA component from their model (hearafter, referred to as BERT-QA).

3.1 Knowledge-Driven Query Expansion for QA-Based AVE

It is inherently difficult for QA-based AVE models to induce effective query representations for rare attributes with ambiguous names. It is also hard to develop expensive resources such as annotation guideline notes (Li et al., 2020) for more than ten thousand of attributes in e-commerce domain.

Then, is there any low-cost resource (knowledge) we can leverage to understand attributes? Our answer to this question is values (answers) for the attributes; we can guess what attributes means from their values. In this study, we exploit attribute values retrieved from the training data[3] of the target AVE model as *run-time knowledge* to induce better query representations.

Our query expansion allows the QA-based AVE model, M_{QA}, to utilize the seen values for attribute a in the whole training data to find beginning and ending positions of a value, $\langle P_b, P_e \rangle$ in title x:

$$\langle P_b, P_e \rangle = M_{QA}([\text{CLS}; x; \text{SEP}; a; \text{SEP}; v_a]) \quad (1)$$

Here, CLS and SEP are special tokens to represent a classifier token and a separator, respectively, and v_a is a string concatenating the seen values of the attribute a with SEP in descending order of frequency in the training data.

3.2 Knowledge Dropout

By taking all the seen values in the training data to augment input queries, the model may just learn to match the seen values with one in the given title. To avoid this, inspired from word dropout employed in language modeling (Gal and Ghahramani, 2016),

---

[3]We can utilize, if any, external resources for our method. For example, e-commerce sites may develop attribute-value databases to organize products in the marketplace.

we perform *knowledge dropout* over v_a in training before concatenating it with title x and attribute a.

$$v_a = [\text{drop}(v_{a,1}); \text{SEP}; \text{drop}(v_{a,2}); \text{SEP}; \ldots] \quad (2)$$

Here, drop is a function that replaces a value $v_{a,i}$ in v_a with padding tokens according to a dropout rate; we replace each token in $v_{a,i}$ with PAD. To decide if the dropout applies to a value, we take account of the number of examples labeled with the value. Given the dropout rate r and the number of training examples n_v, the dropout performs over the value v according to the probability of r^{n_v}. This implementation captures the fact that infrequent values are more likely to be unseen.

3.3 Knowledge Token Mixing

Since values are literally valuable to interpret attributes, the QA-based AVE model may rely more on values than an attribute name. This will hurt the performance on unseen attributes whose values are not available. To avoid this, we assume the availability of value knowledge to be *domain*, and perform multi-domain learning for QA-based model with and without our value-based query expansion. This will allow the model to handle not only seen attributes but also unseen attributes.

Inspired from domain token mixing (Britz et al., 2017), we introduce two special domain tokens (*knowledge tokens*), and prepend either of the tokens to the attribute to express the knowledge status: SEEN and UNSEEN (with and without values).[4] In training, from an example with title x and attribute a, we build [CLS;x;SEP;SEEN;a;SEP;v_a] and [CLS;x;SEP;UNSEEN;a;SEP], and then put these examples to the same mini-batch. In testing, we use SEEN and UNSEEN tokens for seen attributes (with values) and unseen attributes, respectively.

4 Experiments

We evaluate our query expansion method for QA-based AVE on a public dataset,[2] which is built from product data under the Sports & Entertainment category in AliExpress, following (Wang et al., 2020).

4.1 Settings

Dataset The public AliExpress dataset consists of 110,484 tuples of ⟨product title, attribute, value⟩. When a value of the attribute is absent from the title,

---

[4]The original domain token mixing learns to induce domain tokens prior to generating outputs, whereas we prepend domain tokens to inputs since the knowledge status is known.

	Train	Dev.	Test
# of tuples	76,823	10,975	21,950
# of tuples with "NULL"	15,097	2,201	4,259
# of unique attribute-value pairs	11,819	2,680	4,431
# of unique attributes	1,801	635	872
# of unique values	9,317	2,258	3,671
# of tuples (Wang et al., 2020)	88,479	N/A	22,005

Table 1: Statistics of the cleaned AE-pub dataset.

the value in the tuple is set as "NULL." We manually inspected the tuples in the dataset, and found quality issues; some tuples contained HTML entities, and extra white spaces in titles, attributes, and values, and the same attributes sometimes have different letter cases. We thus decoded HTML entities, converted trailing spaces into a single space, and removed white spaces at the beginning and ending. We also normalized the attributes by putting a space between alphabets and numbers and by removing ':' at the endings (from 'feature1:' to 'feature 1'). As a result, we found 736 duplicated tuples. By removing these duplicated tuples, we finally obtained the *cleaned* AE-*pub* dataset of 109,748 tuples with 2,162 unique attributes and 11,955 unique values. We split this dataset into training, development, and test sets with the ratio of 7:1:2 (Table 1).

Evaluation Metrics We use precision (P), recall (R) and F_1 score as metrics. We adopt exact match criteria (Xu et al., 2019) in which the full sequence of extracted value needs to be correct.

4.2 Models

We apply our knowledge-driven query expansion method (§ 3) to BERT-QA (Wang et al., 2020), a QA-based AVE model on BERT. To perform the query expansion, we simply collect values other than "NULL" from tuples in the training data for each attribute (Table 1).

For comparison, we use SUOpenTag (Xu et al., 2019), AVEQA and vanilla BERT-QA (Wang et al., 2020), which achieved the state-of-the-art micro F_1 score on the AliExpress dataset. We also perform a simple dictionary matching; it returns the most frequent seen value for a given attribute among those included in the given title.

To convert tuples in the training set to beginning and ending positions, we tokenize both title and value, and then use matching positions if the token sequence of the value exactly matches a subsequence of the title. If the value matches multiple

Models	Macro			Micro		
	P (%)	R (%)	F_1	P (%)	R (%)	F_1
Dictionary	33.20 (±0.00)	30.37 (±0.00)	31.72 (±0.00)	73.39 (±0.00)	73.77 (±0.00)	73.58 (±0.00)
SUOpenTag (Xu et al., 2019)	30.92 (±1.44)	28.04 (±1.48)	29.41 (±1.44)	86.53 (±0.78)	79.11 (±0.35)	82.65 (±0.20)
AVEQA (Wang et al., 2020)	41.93 (±1.05)	39.65 (±0.96)	40.76 (±0.98)	86.95 (±0.27)	81.99 (±0.13)	84.40 (±0.09)
BERT-QA (Wang et al., 2020)	42.77 (±0.36)	40.85 (±0.22)	41.79 (±0.28)	87.14 (±0.54)	82.16 (±0.21)	84.58 (±0.24)
BERT-QA +vals	39.48 (±0.37)	35.60 (±0.44)	37.44 (±0.38)	**88.82** (±0.22)	81.77 (±0.14)	85.15 (±0.14)
BERT-QA +vals +drop	41.61 (±0.83)	38.22 (±0.80)	39.84 (±0.81)	88.46 (±0.26)	82.02 (±0.37)	85.12 (±0.14)
BERT-QA +vals +mixing	46.67 (±0.33)	43.32 (±0.50)	44.93 (±0.39)	88.30 (±0.69)	82.46 (±0.30)	**85.28** (±0.26)
BERT-QA +vals +drop +mixing	**47.74** (±0.54)	**44.82** (±0.75)	**46.23** (±0.64)	87.84 (±0.39)	**82.61** (±0.07)	85.14 (±0.19)

Table 2: Performance on the cleaned AE-pub dataset in Table 1; reported numbers are mean (std. dev.) of five trials.

Models	cos	Macro			Micro		
		Number of training examples (median: 8)			Number of training examples (median: 8)		
		$[1, 8)$	$[8, \infty)$	all	$[1, 8)$	$[8, \infty)$	all
BERT-QA +vals	lo	42.41 (+0.80)	57.58 (+5.84)	49.96 (+3.31)	55.65 (+8.34)	78.51 (+1.33)	77.59 (+1.77)
	hi	41.05 (−1.75)	69.02 (+1.00)	56.58 (−0.23)	57.72 (+6.04)	88.54 (−0.06)	88.21 (+0.05)
	all	41.77 (−0.40)	63.63 (+3.29)	53.28 (+1.55)	56.65 (+7.25)	86.39 (+0.25)	85.89 (+0.46)
BERT-QA +vals +drop	lo	45.34 (+3.73)	57.89 (+6.15)	51.58 (+4.93)	58.61 (+11.30)	78.70 (+1.52)	77.87 (+2.05)
	hi	45.21 (+2.41)	70.11 (+2.09)	59.04 (+2.23)	60.71 (+9.03)	88.45 (−0.15)	88.16 (±0.00)
	all	45.28 (+3.11)	64.35 (+4.01)	55.32 (+3.59)	59.62 (+10.22)	86.37 (+0.23)	85.91 (+0.48)
BERT-QA +vals +mixing	lo	47.64 (+6.03)	57.91 (+**6.17**)	52.75 (+6.10)	58.13 (+10.82)	78.78 (+**1.60**)	77.90 (+**2.08**)
	hi	48.38 (+5.58)	70.48 (+2.46)	60.67 (+3.86)	62.10 (+10.42)	88.74 (+**0.14**)	88.45 (+**0.29**)
	all	47.99 (+5.82)	64.55 (+4.21)	56.71 (+4.98)	60.03 (+10.63)	86.61 (+**0.47**)	86.13 (+**0.70**)
BERT-QA +vals +drop +mixing	lo	49.15 (+**7.54**)	57.89 (+6.15)	53.51 (+**6.86**)	60.18 (+**12.87**)	78.55 (+1.37)	77.74 (+1.92)
	hi	50.94 (+**8.14**)	71.04 (+3.02)	62.10 (+**5.29**)	63.06 (+**11.38**)	88.56 (−0.04)	88.27 (+0.11)
	all	49.99 (+**7.82**)	64.84 (+**4.50**)	57.81 (+**6.08**)	61.56 (+**12.16**)	86.42 (+0.28)	85.96 (+0.53)

Table 3: Macro and micro F_1 gains over BERT-QA for 544 attributes (21,374 test examples) that took our value-based query expansion. 'lo' and 'hi' are similarity intervals, $[0.411, 0.929)$ and $[0.929, 1.0]$, respectively.

portions of the title, we use the match close to the beginning of the title. As beginning and ending positions of tuples whose value is "NULL," we use 0 which is a position of a CLS token in the title. The conversion procedure is detailed in Appendix A.1.

We implemented the above models using PyTorch (Paszke et al., 2019) (ver. 1.7.1), and used "bert-base-uncased" in Transformers (Wolf et al., 2020) as the pre-trained BERT (BERT$_{BASE}$). The implementation details and the training time are given in Appendix A.2 and Appendix A.3, respectively.

4.3 Results

Table 2 shows macro[5] and micro performance of each model that are averaged over five trials. The low recall of the model BERT-QA +vals suggests that this model learns to find strings that are similar to ones retrieved from the training data (overfitting). On the other hand, knowledge dropout and knowledge token mixing mitigates the overfitting, and improves both macro and micro F_1 performance.

---

[5]We ignored 70 attributes with only NULL since we cannot compute recall and F_1 for these attributes.

Impact on rare and ambiguous attributes To see if the query expansion improves the performance for rare attributes with ambiguous names, we categorized the attributes that took the query expansion according to the number of training examples and the appropriateness of the attribute names for their values. To measure the name appropriateness, we exploit embeddings of the CLS token using the BERT$_{BASE}$ for each attribute and its seen values; when the cosine similarity between the attribute embedding and averaged value embeddings is low, we regard the attribute name as ambiguous. We divide the attributes into four according to median frequency and similarity to values.

Table 3 lists macro and micro F_1 of each model and the improvements over the BERT-QA for each category. We can see that our query expansion tends to be more effective for attributes with low similarity. This means that the query expansion can generate more informative queries than ambiguous attributes alone. Moreover, by using knowledge dropout and knowledge token mixing, we can improve macro and micro F_1 for rare attributes. These

Models	Seen Attr. (Seen Values)		Seen Attr. (Unseen Values)		Unseen Attr.	
	Micro F_1	Macro F_1	Micro F_1	Macro F_1	Micro F_1	Macro F_1
Dictionary	87.19 (±0.00)	83.89 (±0.00)	n/a	n/a	n/a	n/a
BERT-QA (Wang et al., 2020)	92.26 (±0.15)	73.30 (±2.30)	<u>46.11</u> (±0.86)	**25.43** (±1.24)	<u>25.86</u> (±2.53)	<u>20.92</u> (±1.92)
BERT-QA +vals	**92.85** (±0.13)	**86.93** (±0.61)	42.11 (±0.70)	10.98 (±1.01)	6.90 (±1.19)	4.03 (±0.76)
BERT-QA +vals +drop	92.74 (±0.20)	86.21 (±0.58)	44.14 (±0.19)	16.40 (±0.80)	11.17 (±2.89)	7.21 (±2.16)
BERT-QA +vals +mixing	<u>92.82</u> (±0.15)	<u>86.40</u> (±0.79)	45.59 (±0.48)	19.87 (±1.81)	25.39 (±2.63)	20.14 (±2.13)
BERT-QA +vals +drop +mixing	92.67 (±0.11)	86.34 (±0.72)	**46.14** (±0.34)	<u>22.52</u> (±0.93)	**27.54** (±1.35)	**21.95** (±1.25)

Table 4: Performance on the cleaned AE-pub dataset in terms of the types of the attribute values; reported numbers are mean (std. dev.) of five trials. The best score is in bold face and the second best score is underlined.

results are remarkable since the knowledge used to enhance the model comes from its training data; the model could use more parameters to solve the task itself by taking the internal knowledge induced from the training data as runtime input.

Impact on seen and unseen attribute values To see for what types of attribute values the query expansion is effective, we categorize the test examples according to the types of the training data used to solve the examples. We first categorize the test examples into seen or unseen attributes. Next, we further classify the examples for the seen attributes into either seen or unseen attribute values.

Table 4 shows the performance in terms of the attribute value types. The query expansion improved macro F_1 by 13 points on the seen values for the seen attributes; these improvements were yielded by the large performance gains for rare attributes in Table 3. Although BERT-QA +vals performed the best on the seen values, it performed the worst on the unseen values for the seen attributes and unseen attributes; the model is trained to match seen values in a query with a given title. Meanwhile, the two tricks enable the model to maintain the micro F_1 performance of BERT on the unseen values for the seen attributes. The lower macro F_1 against BERT suggests that there is still room for improvements in query representation for rare seen attributes. Lastly, the knowledge token mixing successfully recovered the performance of BERT for the unseen attributes, and even improved the performance when it is used together with the knowledge dropout. This is possibly because the knowledge token mixing allows the model to switch its behavior for seen and unseen attributes, and the knowledge dropout strengthens the ability to induce better query representations.

Example outputs Table 5 shows examples of the actual model outputs for a given context and query (attribute (seen values)). In the first two examples, *function 1* and *nominal capacity* are ambiguous

C: aeronova [bicycle [**carbon mtb handlebar**]ours]BERT-QA mountain bikes flat handlebar mtb integrated handlebars with stem bike accessories
Q: function 1 (skiing goggles, carbon road bicycle handlebar, cycling glasses, bicycle mask, gas mask, . . .)

C: lfp [3.2v [**100ah**]ours]BERT-QA lifepo4 prismatic cell deep cycle diy lithium ion battery 72v 60v 48v 24v 100ah 200ah ev solar storage battery
Q: nominal capacity (14ah, 40ah, 17.4ah)

C: camel outdoor softshell [**men**]BERT-QA's hiking jacket windproof thermal jacket for [camping]ours ski thick warm coats
Q: suitable (men, camping, kids, saltwater/freshwater, women, 4-15y, mtb cycling shoes, . . .)

Table 5: Example outputs of BERT-QA with and without query expansion for given C(ontext) and Q(uery).

and rare attributes, respectively, and are thereby hard for the BERT-QA to extract **correct** values without the help of our query expansion. As shown in the last example, when there are more than one candidates as values of a given attribute, our query expansion is still unstable.

5 Conclusions

We have proposed simple query expansion based on possible values of a given query (attribute) for QA-based attribute extraction. With the two tricks to mimic the imperfection of the value knowledge, we retrieve values of given attributes from the training data, and then use the obtained values as knowledge to induce better query representations. Experimental results on our cleaned version of the public AliExpress dataset demonstrate that our method improves the performance of product attribute extraction, especially for rare and ambiguous attributes.

We will leverage external resources to handle unseen attributes (preliminary experiments are shown in Appendix A.4). We will release the script to build our cleaned AE-pub dataset.[6]

---

[6]http://www.tkl.iis.u-tokyo.ac.jp/~ynaga/acl2022/

Acknowledgements

References

Denny Britz, Quoc Le, and Reid Pryzant. 2017. Effective domain mixing for neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 118–126, Copenhagen, Denmark. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Xin Luna Dong, Xiang He, Andrey Kan, Xian Li, Yan Liang, Jun Ma, Yifan Ethan Xu, Chenwei Zhang, Tong Zhao, Gabriel Blanco Saldana, Saurabh Deshpande, Alexandre Michetti Manduca, Jay Ren, Surender Pal Singh, Fan Xiao, Haw-Shiuan Chang, Giannis Karamanolakis, Yuning Mao, Yaqing Wang, Christos Faloutsos, Andrew McCallum, and Jiawei Han. 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*, page 2724–2734, New York, NY, USA. Association for Computing Machinery.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems 29 (NIPS)*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Mayank Jain, Sourangshu Bhattacharya, Harshit Jain, Karimulla Shaik, and Muthusamy Chelliah. 2021. Learning cross-task attribute - attribute similarity for multi-task attribute-value extraction. In *Proceedings of The 4th Workshop on e-Commerce and NLP*, pages 79–87, Online. Association for Computational Linguistics.

Giannis Karamanolakis, Jun Ma, and Xin Luna Dong. 2020. TXtract: Taxonomy-aware knowledge extraction for thousands of product categories. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8489–8502, Online. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the third International Conference on Learning Representations*, San Diego, California, USA.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020. A unified MRC framework for named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5849–5859, Online. Association for Computational Linguistics.

Kartik Mehta, Ioana Oprea, and Nikhil Rasiwasia. 2021. LATEX-numeric: Language agnostic text attribute extraction for numeric attributes. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Papers*, pages 272–279, Online. Association for Computational Linguistics.

Ajinkya More. 2016. Attribute extraction from product titles in ecommerce. In *KDD 2016 Workshop on Enterprise Intelligence*, San Francisco, California, USA.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., Red Hook, NY, USA.

Duangmanee Putthividhya and Junling Hu. 2011. Bootstrapped named entity recognition for product attribute extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1557–1567, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Martin Rezk, Laura Alonso Alemany, Lasguido Nio, and Ted Zhang. 2019. Accurate product attribute extraction on the field. In *Proceedings of the 35th IEEE International Conference on Data Engineering*, pages 1862–1873, Macau SAR, China. IEEE.

Keiji Shinzato and Satoshi Sekine. 2013. Unsupervised extraction of attributes and their values from product description. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1339–1347, Nagoya, Japan. Asian Federation of Natural Language Processing.

Qifan Wang, Li Yang, Bhargav Kanagal, Sumit Sanghai, D. Sivakumar, Bin Shu, Zac Yu, and Jon Elsas. 2020. Learning to extract attribute value from product via question answering: A multi-task approach. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 47–55, Online. ACM.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz,

Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 38–45, Online. Association for Computational Linguistics.

Huimin Xu, Wenting Wang, Xin 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, Florence, Italy. Association for Computational Linguistics.

Jun Yan, Nasser Zalmout, Yan Liang, Christan Grant, Xiang Ren, and Xin Luna Dong. 2021. AdaTag: Multi-attribute value extraction from product profiles with adaptive decoding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4694–4705, Online. Association for Computational Linguistics.

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 and Data Mining*, pages 1049–1058, London, United Kingdom. ACM.

Tiangang Zhu, Yue Wang, Haoran Li, Youzheng Wu, Xiaodong He, and Bowen Zhou. 2020. Multimodal joint attribute prediction and value extraction for E-commerce product. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2129–2139, Online. Association for Computational Linguistics.

A Appendix

A.1 How to Convert Tuples to Labeled Data

Let's say, we have a tuple of ⟨product title, attribute, value⟩ = ⟨*golf clubs putter pu neutral golf grip, material, pu*⟩, and try to obtain beginning and ending positions of the value in the title. First, we tokenize both title and value using BertTokenizer, and then find a partial token sequence of the title that exactly matches with the token sequence of the value. By performing the match over the tokenization results, we can avoid matching a part of tokens in the title to the value. In case of this example, we can prevent the value *pu* from matching to the first two characters of *putter*. As a result, the value *pu* matches to the token *pu* in the title, and we properly obtain the beginning and ending positions of *pu* in the title.

A.2 Implementation Details

We implemented all the models used in our experiments using PyTorch (Paszke et al., 2019) (ver. 1.7.1),[7] and used "bert-base-uncased" in Transformers (Wolf et al., 2020)[8] as the pre-trained BERT (BERT$_{\text{BASE}}$). The dimension of the hidden states (D) is 768, and the maximum token length of the product title is 64. We set the maximum token length of the query to 32 for all models with the exception of models with the query expansion. To make as many attribute values as possible, we set 192 to the maximum token length of the query for the models using the query expansion, and truncate the concatenated string if the length exceeds 192. We set a rate of dropout over values to 0.2. The total number of parameters in BERT-QA with our query expansion is 109M. We train the models five times with varying random seeds, and average the results.

Regarding to AVEQA, the loss of the distilled masked language model got NaN if we followed the algorithm in the paper. We instead used BERTMLMHead class implemented in Transformers.[8]

We use Adam (Kingma and Ba, 2015) with a learning rate of 10^{-5} as the optimizer. We trained the models up to 20 epochs with a batch size of 32 and chose the models that perform the best micro F_1 on the development set for the test set evaluation.

A.3 Training Time

We used an NVIDIA Quadro M6000 GPU on a server with an Intel® Xeon® E5-2643 v4 3.40GHz CPU with 512GB main memory for training. It took around two hours per epoch for training BERT-QA with our query expansion, while it took around 25 minutes per epoch for training the BERT-QA.

A.4 Preliminary experiments using external resource to obtain the value knowledge

As we have discussed in § 3.1, we can utilize external resource other than the training data of the model to perform the query expansion. We here evaluate the BERT-QA models that have been already trained with our query expansion, using the development data as external (additional) resource to obtain the value knowledge in testing. If new values are retrieved from the development data, the models will build longer queries for attributes. We

---

[7] https://github.com/pytorch/pytorch/
[8] https://huggingface.co/models

Models	Macro			Micro		
	P (%)	R (%)	F$_1$	P (%)	R (%)	F$_1$
Seen attributes (seen values)						
BERT-QA (Wang et al., 2020)	73.10 (±3.99)	66.86 (±2.97)	69.83 (±3.42)	95.50 (±0.13)	92.14 (±0.21)	93.79 (±0.10)
w/ values in the training data						
BERT-QA +vals	**87.66** (±1.21)	**82.60** (±1.13)	**85.06** (±1.15)	95.76 (±0.21)	92.54 (±0.12)	94.13 (±0.15)
BERT-QA +vals +drop	87.28 (±0.65)	81.83 (±0.90)	84.47 (±0.70)	**95.84** (±0.19)	92.81 (±0.36)	94.30 (±0.17)
BERT-QA +vals +mixing	86.98 (±0.91)	81.36 (±1.13)	84.07 (±0.88)	**95.84** (±0.16)	92.80 (±0.25)	94.29 (±0.10)
BERT-QA +vals +drop +mixing	86.43 (±0.98)	81.48 (±0.70)	83.88 (±0.78)	95.79 (±0.20)	**93.12** (±0.11)	**94.44** (±0.15)
w/ values in the training and development data						
BERT-QA +vals	86.71 (±1.14)	81.50 (±0.83)	84.02 (±0.96)	95.44 (±0.23)	92.00 (±0.13)	93.69 (±0.18)
BERT-QA +vals +drop	85.29 (±1.04)	80.29 (±0.94)	82.71 (±0.93)	95.44 (±0.20)	92.44 (±0.31)	93.92 (±0.12)
BERT-QA +vals +mixing	85.89 (±1.50)	80.51 (±2.26)	83.11 (±1.85)	95.77 (±0.16)	92.77 (±0.22)	94.25 (±0.08)
BERT-QA +vals +drop +mixing	85.65 (±0.57)	80.72 (±0.69)	83.11 (±0.56)	95.75 (±0.16)	93.06 (±0.12)	94.39 (±0.12)
Seen attributes (unseen values)						
BERT-QA (Wang et al., 2020)	**29.72** (±2.20)	**24.72** (±1.58)	**26.99** (±1.83)	34.44 (±3.47)	**21.28** (±1.80)	**26.28** (±2.26)
w/ values in the training data						
BERT-QA +vals	16.89 (±1.49)	12.94 (±1.46)	14.65 (±1.48)	31.56 (±2.40)	12.42 (±1.15)	17.83 (±1.55)
BERT-QA +vals +drop	22.32 (±1.48)	18.77 (±1.06)	20.39 (±1.24)	37.06 (±1.09)	16.99 (±0.70)	23.30 (±0.81)
BERT-QA +vals +mixing	24.10 (±1.45)	18.98 (±0.90)	21.23 (±1.09)	35.07 (±1.51)	16.77 (±1.02)	22.68 (±1.22)
BERT-QA +vals +drop +mixing	27.19 (±1.20)	22.31 (±1.00)	24.51 (±1.06)	36.60 (±0.50)	18.33 (±0.73)	24.42 (±0.64)
w/ values in the training and development data						
BERT-QA +vals	24.03 (±1.81)	17.94 (±1.56)	20.54 (±1.68)	36.54 (±2.25)	15.71 (±1.31)	21.97 (±1.67)
BERT-QA +vals +drop	27.27 (±1.09)	22.30 (±1.24)	24.53 (±1.17)	**39.49** (±2.31)	19.16 (±0.59)	25.79 (±0.91)
BERT-QA +vals +mixing	27.52 (±1.02)	21.56 (±1.02)	24.17 (±0.97)	37.35 (±1.02)	18.77 (±0.97)	24.98 (±1.02)
BERT-QA +vals +drop +mixing	28.57 (±1.11)	23.44 (±1.13)	25.75 (±1.12)	37.64 (±0.96)	19.67 (±0.60)	25.83 (±0.67)
Unseen attributes						
BERT-QA (Wang et al., 2020)	42.22 (±6.67)	42.22 (±6.67)	42.22 (±6.67)	59.23 (±8.78)	45.26 (±6.32)	51.22 (±7.14)
w/ values in the training data						
BERT-QA +vals	15.56 (±2.22)	15.56 (±2.22)	15.56 (±2.22)	64.00 (±9.70)	14.74 (±2.11)	23.91 (±3.30)
BERT-QA +vals +drop	19.44 (±2.48)	18.33 (±1.36)	18.85 (±1.85)	56.67 (±9.33)	18.95 (±2.58)	28.37 (±3.98)
BERT-QA +vals +mixing	42.22 (±4.44)	42.22 (±4.44)	42.22 (±4.44)	61.69 (±3.15)	45.26 (±4.21)	52.03 (±2.92)
BERT-QA +vals +drop +mixing	42.22 (±2.72)	42.22 (±2.72)	42.22 (±2.72)	54.42 (±2.48)	45.26 (±2.58)	49.41 (±2.51)
w/ values in the training and development data						
BERT-QA +vals	37.78 (±2.22)	37.78 (±2.22)	37.78 (±2.22)	69.33 (±1.33)	35.79 (±2.11)	47.19 (±2.17)
BERT-QA +vals +drop	43.33 (±2.22)	43.33 (±2.22)	43.33 (±2.22)	72.18 (±1.09)	41.05 (±2.11)	52.32 (±2.02)
BERT-QA +vals +mixing	50.00 (±3.51)	50.00 (±3.51)	50.00 (±3.51)	**74.67** (±2.88)	52.63 (±3.33)	61.69 (±2.86)
BERT-QA +vals +drop +mixing	**52.22** (±2.72)	**52.22** (±2.72)	**52.22** (±2.72)	73.38 (±4.20)	**54.74** (±2.58)	**62.66** (±2.80)

Table 6: Performance on seen and unseen attributes in Table 4 whose new values are retrieved from the development data and are used for the query expansion; reported numbers are mean (std. dev.) of five trials.

here evaluate such attributes with longer queries among the seen and unseen attributes in Table 4.

Table 6 shows the performance of the BERT-QA models with our query expansion on 288 seen values for 107 seen attributes, 339 unseen values for 131 seen attributes, and 19 values for 18 unseen attributes, for which new values are retrieved from the development data. We can observe that the new values retrieved from the development data boosted the performance of the BERT-QA models with our query expansion on the unseen values for the seen attributes and the unseen attributes, whereas they did not increase the performance on the seen values for the seen attributes. In the future, we will explore a better way to leverage the value knowledge in the external resources other than the training data of the QA-based models.