Prototype-Representations for Training Data Filtering in Weakly-Supervised Information Extraction

Nasser Zalmout and Xian Li
Amazon.com
{nzalmout,xianlee}@amazon.com

Abstract

The availability of high quality training data is still a bottleneck for the practical utilization of information extraction models, despite the breakthroughs in zero and few-shot learning techniques. This is further exacerbated for industry applications, where new tasks, domains, and specific use cases keep arising, which makes it impractical to depend on manually annotated data. Therefore, weak and distant supervision emerged as popular approaches to bootstrap training, utilizing labeling functions to guide the annotation process. Weakly-supervised annotation of training data is fast and efficient, however, it results in many irrelevant and out-of-context matches. This is a challenging problem that can degrade the performance in downstream models, or require a manual data cleaning step that can incur significant overhead. In this paper we present a prototype-based filtering approach, that can be utilized to denoise weakly supervised training data. The system is very simple, unsupervised, scalable, and requires little manual intervention, yet results in significant precision gains. We apply the technique in the task of attribute value extraction in e-commerce websites, and achieve up to 9% gain in precision for the downstream models, with a minimal drop in recall.

1 Introduction

Weak supervision and data programming have recently emerged as powerful techniques to support information extraction models. Weak supervision is useful for dynamic environments, where new tasks or deployment domains keep emerging, and using manually annotated data is impractical. Weakly supervised data programming aims to reformulate the training data annotation process into a programming paradigm. Instead of manually annotating each training sample, the annotation process is handled through labeling functions, which are then used to automatically annotate the training corpora. These labeling functions are usually based on manually predefined patterns or regular expressions, that are matched against the target unannotated data. The weak supervision setup allows for an efficient and scalable training data collection process, but at the expense of accuracy. Labeling functions can over-match, resulting in many irrelevant and out-of-context matches. This hinders the performance, and could require manual cleaning steps to lift quality, adding to the overall overhead.

In this paper we present a data filtering system based on prototype-learning. *Prototype* here refers to the correct contexts where a given target value is usually mentioned, based on a sample dataset. This can be drawn from a manually annotated corpus (making it more supervised), or it can be based on the centroid of the raw dataset (more noisy, but totally unsupervised). At runtime, we get the embedding of the prototype, along with context embedding of the annotated label for each training sample, and use outlier detection constructs to remove outliers. We calculate the distance between the prototype and the in-context label representation, and if the distance is above a certain threshold we filter it out. These out-of-context annotations are out-of-distribution, so approaches that utilize prototype-learning as part of the end-to-end network, or other noise-robust strategies, would not handle them properly. Removing them before modeling is the best approach to reduce noise. The intuition behind this filtering logic is that value context embeddings that are distant from the prototype tend to be more noisy, likely reflecting out-of-context matches. We use several prototype and data embedding techniques.

We utilize the filtering technique for attribute value extraction (AVE), an information extraction task that has recently been gaining much momentum in e-commerce applications. The goal of the AVE task is to obtain structured product features from the unstructured natural language description of the product's page in e-commerce websites. This

Proceedings of EMNLP 2022 Industry Track, pages 477–484
December 9–11, 2020.

Figure 1: Sample product profile and relevant attributes.

has several downstream applications in areas including product search, comparison, question answering, among others. Due to the dynamic and large scale nature of this domain, contributions for the AVE task often use weak and distant learning techniques to train the extraction models. Our results show up to 9% absolute improvement in precision, with minimal drop in recall (about 1%), for the extraction model trained on the filtered data.

2 Background

2.1 Attribute Value Extraction

The AVE task aims to extract the corresponding values for a given attribute, out of a number of attributes of interest, from the textual sequence of a product profile (Zheng et al., 2018). Given a text sequence $X = [x_1, \ldots, x_n]$ in a product profile, where n is the number of words, and an attribute $r \in R$, where R is a predefined set of attributes, the model is expected to extract all text spans from X that could be valid values for attribute r characterizing this product's features. When there are no corresponding values mentioned in X, the model should return an empty set. For example, for the product in Figure 1, given its title as X, the model is expected to return ("8 Fl Ounce") if $r =$"Size", and an empty set if $r =$"Hair Type". The various products are categorized into diverse product types (PTs), like *Shampoo*, *Tea*, *TVs*, etc. The products within a given PT are homogeneous, sharing similar overall product features, including the set of relevant attributes. And different PTs can have a different set of relevant attributes.

The content in e-commerce websites is very dynamic, where new products are frequently added. The PT categorizations themselves also change overtime with new products, along with the set of relevant attributes. This makes manual training data annotation with a predefined set of PTs and attributes infeasible for attribute value extraction. Zero and few-shot learning for new attributes have also shown limited success (Yang et al., 2022). Therefore, most of the AVE contributions rely on distant or weak supervision, which is very susceptible to noisy and out-of-context annotations.

2.2 Training Data Denoising

To better understand how weak supervision with simple regular expressions or labeling functions can result in very noisy annotations, we use the product in Figure 1 as an example. The figure shows the product profile, which includes the title, description, along with the product image, for a *Skin Moisturizer* PT. Some of the relevant attribute values are highlighted with different colors. These are the values that an AVE model is expected to return, out of the target space of each separate attribute and PT. The target space for the *ItemForm* attribute and Skin Moisturizer PT, for example, includes "lotion", "cream", "oil", among others. The ItemForm label for this product should be "lotion" as shown. However, a weakly supervised training data, that does not consider contextual understanding when assigning labels, could have chosen the more frequent "oil" as the label. The same behavior could happen with the *Scent* attribute, where "Coconut" could have been selected instead, or in addition to, "Island Mango", which is the right value for this product. This paper suggests a filtering step on top of the weakly supervised training data, that eliminates such out-of-context annotations.

Our filtering setup is on the <attribute, PT> pair level. So we learn a different prototype for each <attribute, PT> pair, and apply the filtering approach for product attribute values in each pair separately.

2.3 Desiderata

There are a few constraints and specific desiderata that should guide the filtering technique to be deployed actively in a production system, without being disruptive to the advantages of distant-supervision. The filtering approach should be unsupervised, with minimal manual overhead. Ideally, the filtering approach should also provide an easy mechanism to control the balance between precision and recall, if needed. And finally, the filtering

approach should be easy to deploy without causing much disruption to existing pipelines.

3 Related Work

Prototype-based approaches for NLP have traditionally been used in the word representation literature (Huang et al., 2012; Reisinger and Mooney, 2010). Interest for prototype-based approaches increased significantly with the onset of Prototypical Networks (ProtoNet), with successful utilization in few-shot learning classification in computer vision tasks (Snell et al., 2017), and contrastive learning models (Gao et al., 2021). ProtoNet-based approaches compute one prototype per class as the class mean. These prototypes are then used in a nearest neighbour classifier to update the objective function. This is consistent with our overall setup. There have been many contributions since then utilizing Prototypical Networks for NLP tasks, mostly in few-shot learning in information extraction (Cui et al., 2021; Lai et al., 2021; Gao et al., 2019). But as far as we are aware, this paper is the first to use prototype representations for training data filtering in weak supervision models. There are other contributions utilizing data centroids for outlier detection, mostly through k-nearest neighbor formulation as well. The idea is to remove samples that are far from the cluster's centroid, in a clustering setup (Wang et al., 2021; Pamula et al., 2011). And we in fact use a similar formulation in our filtering approach, with the context prototype as the target.

The attribute value extraction task has traditionally been modeled using distant-supervision (Ding et al., 2022; Yang et al., 2022; Lin et al., 2021; Yan et al., 2021; Wang et al., 2020; Zheng et al., 2018) which is prone to noise. Practical utilization of the AVE task in production makes further use of weak-supervision and data programming techniques for training data collection (Zalmout et al., 2021). This further amplifies the noise issue, and makes training data filtering techniques more important.

4 Approach

4.1 Context-Aware Embeddings

Each product in a given <attribute, PT> pair is represented through the context embedding of the mentioned attribute value, using pre-trained language models like BERT (Devlin et al., 2018). We use a masking vector on top of the text sequence, for each value. We then use BERT-like models to get the context embedding with the sequence and mask as input. Within this scope, we can use two embedding paradigms:

- Value-Based Embeddings: We get the context embedding for the value mention in each product profile directly, based on the target value. For multi-worded values, we take the average of the word embeddings.
- Name-Based Embeddings: We replace the value mention with the attribute and PT names, separated with [BOA] and [EOA] special tokens (BOA: beginning of attribute, EOA: end of attribute). Like "*[BOA] skin moisturizer item form [EOA]*", instead of the "*lotion*" value in the example in Figure 1. We then get the embedding as before.

We also fine-tune the pre-trained language model using the MLM objective. Fine-tuning is more critical for name-based embeddings, since the pattern of using the attribute and PT names instead of values, along with the additional special tokens, are not covered in the existing pretrained models. We follow the same value format mentioned above, and replace the attribute value with the PT and attribute names, along with the [EOA] and [BOA] special tokens. In the fine-tuning dataset, we randomly replace value mentions with the above name notation for $n\%$ of the overall corpus products.

It is worth noting that the masking setup in the name-based embeddings is used both during fine-tuning and context embedding retrieval at runtime.

4.2 Prototype Representation

The prototype embedding is the mean of the context embeddings of a representative sample of the values in a given <attribute, PT> pair. The prototype representation in prototype-learning is usually learnt from a small set of manually annotated data. However, in our case having annotated data for each PT is challenging, since production systems would be working with a large number of different PTs. And collecting annotated data for each PT is expensive. We therefore identify two different approaches to obtain the prototype representation; using a small annotated sample as typically done in classical prototype-learning, or using the centroid of the raw training data as a proxy for the prototype.

4.2.1 Gold Prototype

Manually annotated data for each PT would allow the model to capture more representative embed-

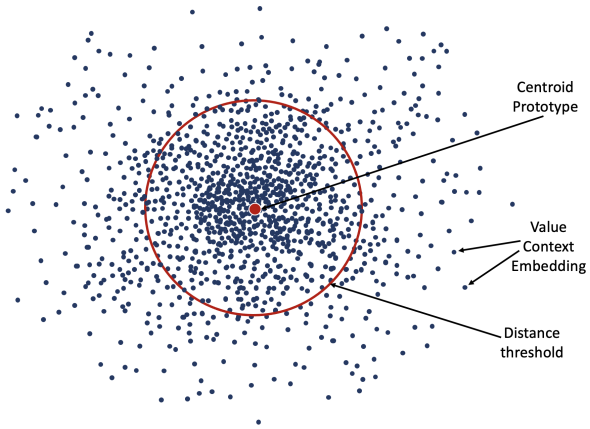

Figure 2: Filtering setup using centroid prototype.

dings. In this case, the PT embedding might not be at the centroid of the training data cluster, depending on how noisy and how representative it is of the real distribution. The main advantage of this setup is that we can have more accurate and meaningful representations, for the PTs with enough annotated data coverage. However, the size of the annotated sample for each PT would be small, given the large number of different PTs. This might lead to bias, misrepresentation, and limited coverage of all possible target values. Moreover, relying on annotated data incurs significant overhead, and creates a dependency between data annotation and the training data generation process, which hinders scalability.

4.2.2 Training Data Centroid Prototype

We also use the training data itself to calculate the prototype. The training data is noisy, so it cannot be used directly to get the prototype. However, in this case the goal would be to identify the centroid of the context embeddings, and then assign a numeric distance score for each context against the centroid as a proxy. The threshold for the distance is then used to eliminate outlier contexts relative to the centroid representation. This can work if the training data is not too excessively noisy, where the centroid is somewhat close to the PT representation if a large amount of gold data was used. The advantages of such setup is that the weakly supervised training data is cheap and does not require manual curation. We can also get sizable training data for each PT, that covers most of the relevant values. However, if the training data is too noisy, the centroid would not be capturing any meaningful representations. Figure 2 shows a sample distribution of the different contexts, centroid, and distance threshold.

4.3 Outlier Detection

After obtaining the prototype representation, whether using the gold prototype or data centroid, along with the individual context embeddings, we formulate the cleaning task as an outlier detection task. We use distance metrics to calculate distance between each training sample context embedding and the prototype. And eliminate training samples with a distance above a tunable threshold. We experiment with several distance metrics, in addition to Euclidean distance, including:

- Mahalonobis Distance: Mahalonabis distance is a multivariate distance metric, that considers the potential covariance between the different variables. It is commonly used in anomaly detection literature. However, excessive noise can bias the covariance matrix, hence resulting in biased distance calculations.

$$d(x, \mu) = \sqrt{(x - \mu)^T C^{-1} (x - \mu)}$$

  Where x is the vector representation of the given sample. μ is the vector representation of the centroid or prototype, C^{-1} is the inverse covariance matrix estimate for the samples.

- Cosine Distance: The inverse of the cosine of the angle between sample and prototype vectors, through the dot product of the vectors divided by the product of their lengths.

$$d(x, \mu) = 1 - \frac{\sum_{i=1}^{n} x_i \mu_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \sqrt{\sum_{i=1}^{n} \mu_i^2}}$$

4.4 Evaluation Criteria

Throughout the various experiments we evaluate the filtering setup based on two different criteria, precision/recall for the training data itself, and precision/recall for the downstream extraction model.

Training data recall vs model recall. We optimize mainly for precision in the training data evaluation. Recall in the training data evaluation is calculated based on the intersection of the benchmarking and training datasets, and does not necessarily correlate with the recall of the model itself. Therefore, lower recall in the training data evaluation is not problematic, as long as it does not cause a significant bias in specific values, as in eliminating certain values completely or near completely. Whereas for the actual trained model, we optimize

Attribute	# PTs	Training Set	Gold Testing Set
ContainerType*	34	265,822	2,119
ItemForm*	14	1,195,256	5,432
Pattern	33	126,153	2,665
ItemShape	80	511,977	2,825
ChocolateType	1	6,797	75
Material	21	41,245	2,198
ControlType	9	72,489	892
Sum	**192**	**2,219,739**	**16,206**

Table 1: Statistics of the training and evaluation datasets for the various attributes. *We use two attributes (ItemForm and ContainerType) for the ablations. And we expand to the remaining attributes afterwards.

for both precision and recall. The goal is to maximize precision, with minimal sacrifice in recall. We confirm this behavior in Table 4, where the recall of the model does not drop significantly, even though the training data evaluation reflects a bigger drop. Throughout the training data evaluation experiments we mainly focus on the precision results, but also report recall as a sanity check. But we opt not to report F1 scores, since it does not reflect a meaningful metric in this case.

5 Experiments and Results

5.1 Dataset

We collect our raw dataset from the product profiles (title, bullets, and description) from the public web pages at Amazon.com. The goal is to collect training data that is entirely weakly supervised, without any manual cleaning or intervention. This is why we opted not to use available public data like MAVE (Yang et al., 2022), which has been extensively processed. We selected seven different attributes, and identified the set of relevant values. The value identification is the only manual step in this setup, gathered from Amazon pages. The training data is then collected through labeling functions based on regular expressions for each of the target values. This setup is commonly used in the AVE task, usually followed by a manual curation step to fix erroneous matching patterns.

To better understand the limits of our setup, we also worked on enhancing the quality of the training data manually, to compare against the automatic filtering system. We selected a sample of products per PT, and worked with annotators to identify patterns of erroneous value annotations in the training data and fix them. The goal is to update the labeling function with negative patterns that it should avoid matching, through look ahead and behind phrases in the regular expression. For example, "whole" is a valid value for the ItemForm attribute, used in cases like "whole beans". A negative matching pattern would be phrases like "whole foods". A manually curated pattern in this case is to avoid matching the "whole" value if it is followed by the "foods" word. We call the resulting dataset *manually curated* throughout the experiments.

We also collected a benchmarking set of manually annotated set of products in each PT, for general evaluation. Table 1 shows the statistics of the datasets we collected. We also collected a dataset of about 3 million products, from the public pages at Amazon.com. We use this dataset for the pre-trained language model fine-tuning, using the MLM objective.

5.2 Training Data Evaluation

In this part we evaluate the resulting training datasets directly, through a manually labeled sample from the raw training data. Since the various ablations aim to filter out erroneous annotations, the recall of the raw data would be the upper bound for all subsequent variations. Recall of training data is not as important as precision, since the downstream extraction model is expected to cover the recall gap, as we highlighted earlier. So we report recall in the various results, but we focus on precision gain. We use the raw data as the main baseline. We also compare against the manually curated datasets, that were handled through manual inspection and sets of manually curated rules to fix them. The K value, at the P@R=K metric, were chosen for each case to match the recall for the manually curated datasets, to facilitate easier comparison.

Results in Table 2 show significant improvement compared to the unfiltered data, along with large improvements compared to the manually curated data as well. Value-based embeddings outperform name-based embeddings across the various settings. And the centroid approach seems to outperform the Gold Prototype approach. This is significant, since it indicates that we do not need manually annotated dataset to utilize the filtering approach. This is probably due to the more representative nature of the centroid, despite the noise, compared to a small annotated sample. We also experiment with the different outlier detection methods. Results in Table 3 show that Cosine distance outperforms the other metrics. One theory for why Mahalanobis distance did not perform well is that covariance

Training Data		ItemForm		ContainerType	
		Precision	Recall	Precision	Recall
Raw Data		86.5%	44.0%	78.1%	24.1%
Manually Curated		93.2%	29.6%	80.3%	20.2%
Centroid Prototype	Value-based max precision	**96.4%**	21.2%	**82.7%**	18.8%
	Value-based P@R=K*	95.6%	30.0%	81.6%	20.0%
	Name-based max precision	89.2%	38.0%	80.0%	21.0%
	Name-based P@R=K*	88.5%	30.0%	77.6%	20.0%
Gold Prototype	Value-based max precision	95.1%	21.5%	81.4%	18.5%
	Value-based P@R=K*	94.7%	29.3%	80.5%	20.4%
	Name-based max precision	87.1%	**38.2%**	80.5%	**21.5%**
	Name-based P@R=K*	84.5%	30.5%	75.4%	19.4%

Table 2: Training data evaluation results after the various filtering approaches. These results reflect the training data evaluation, not the extraction model evaluation. Therefore, precision gain is more important than recall, and raw data recall is not a baseline. Check Section 4.4 for more details. *The K value, for P@R=K, is 0.3 for ItemForm, and 0.2 for ContainerType, as described in Section 5.2.

Distance Metric	ContainerType		ItemForm	
	Precision	Recall	Precision	Recall
Raw Data	78.1%	24.1%	86.5%	44.0%
Cosine Distance	**81.6%**	20.0%	**95.6%**	30.0%
Mahalanobis Distance	78.6%	**20.8%**	88.4%	**30.9%**
Euclidean Distance	78.9%	20.3%	92.7%	29.8%

Table 3: Training data evaluation results for the various distance metrics. The data is filtered using value-based embeddings, centroid prototype, and the P@R=K setup. We do not include F1 results, as explained in Section 4.4.

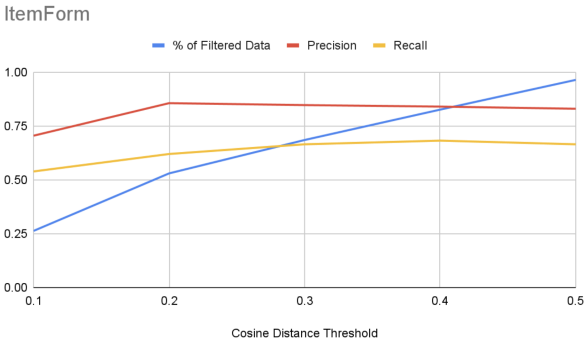

Figure 3: Results for the extraction models trained using the filtered training data for the ItemForm attribute, as a function of the cosine distance threshold.

matrices are susceptible to noise. To test if this is more prominent in centroid-based filtering, we also used the Gold Prototypes approach, and results are actually lower, in accordance to the results for Gold Prototypes in general.

We also experimented with a number of pre-trained language models, including BERT (Devlin et al., 2018) (base and large), RoBERTa (Liu et al., 2019) (base and large), and GPT2, all fine-tuned using the same dataset. Results are very close to each other, besides GPT2 which is significantly lower, so we opted to use BERT base.

5.3 Downstream Extraction Models Results

The training data results show significant precision gain, at the expense of some recall drop, which is not a problem as we highlighted earlier. To assess the impact of the filtering setup on the downstream extraction models themselves, and investigate the role of cosine distance threshold, we train several models using the filtered data. There are several architectures used for the AVE task in literature, with a varying degree of complexity (Zalmout et al., 2021). In this part we opt for the original OpenTag model (Zheng et al., 2018). Table 4 shows the results of the filtered compared to raw data, and Figure 3 shows the results as a function of the cosine distance threshold. The filtering setup achieves up to 10% absolute gain in precision, with a minimal recall drop of around 1%, after filtering more than 50% of the original training dataset. Interestingly, recall seems to be doing well overall across most of the filtering thresholds, even though we are doing significant filtering of the training data.

5.4 Experimenting with Additional Attributes

We also expanded the experiments to five additional attributes, to further evaluate the consistency of the improvement. We evaluated the resulting training data compared to the unfiltered datasets. Results in Table 5 show gains across all attributes, with an average precision gain of about 9% absolute.

6 Conclusion

We presented an automatic filtering approach using prototype-based representations. We applied the approach on the AVE task, and showed that using centroid-based prototypes outperforms gold-data

Training Data	ContainerType			ItemForm		
	Precision	Recall	F1 Score	Precision	Recall	F1 Score
Raw data	71.3%	**34.6%**	46.7%	82.4%	**67.1%**	73.9%
Filtered data*	**80.6%**	33.5%	**47.4%**	**85.1%**	66.8%	**74.9%**

Table 4: Results for the extraction model evaluation, trained using the filtered data compared to the raw data. *The data is filtered using cosine distance, value-based embeddings, centroid prototype, and the P@R=K setup.

Attribute	Raw Data		Filtered Data	
	Precision	Recall	Precision	Recall
Pattern	74.4%	14.5%	**90.8%**	14.1%
ItemShape	59.7%	17.8%	**70.3%**	16.7%
ChocolateType	88.9%	32.9%	**93.1%**	27.8%
Material	71.6%	16.1%	**74.9%**	14.5%
ControlType	69.7%	18.0%	**80.1%**	12.4%
Average	72.9%	19.9%	**81.8%**	17.1%

Table 5: Training data results for five additional attributes. As explained earlier, recall drop is not as important as precision gain for training data evaluation, and we do not report F1 scores. Check Section 4.4 for more details.

prototypes. We also showed that cosine distance outperforms other outlier detection techniques. We also showed that although recall in the filtered training data drops, the precision gain would still provide the downstream model with the capacity to cover any recall gaps. Model results show significant precision gain, with a minimal drop in recall.

Future work in this direction include tying the filtering process to the underlying task, which would help learn more meaningful representations. Along with developing an iterative filtering process, through which we get the centroids, filter data, then use filtered data to learn centroids again. Such iterative process could improve the quality of the filtering process.

Limitations

Despite the impressive overall performance, along with the simplicity of the approach, the filtering system covers a subset of all possible errors. The goal is to address out-of-context annotations, so errors that are not far off contextually would be more difficult to filter out. Moreover, even for the out-of-context matches, the filtering system is relatively crude and aggressive. The filtering decisions are not fine-grained, so false positives and negatives can still happen. Finally, the centroid prototype, which provides the best results in our setup, is highly dependent on the level of noise in the raw datasets. So we would expect the filtering process to be more biased for attributes that are excessively noisy. Albeit, we still think the filtering system is powerful, useful, yet simple enough for successful utilization in production.

References

Li Cui, Deqing Yang, Jiaxin Yu, Chengwei Hu, Jiayang Cheng, Jingjie Yi, and Yanghua Xiao. 2021. Refining sample embeddings with relation prototypes to enhance continual relation extraction. In *Proceedings of ACL-IJCNLP'21 (Volume 1: Long Papers)*, pages 232–243.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina N. Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Yifan Ding, Yan Liang, Nasser Zalmout, Xian Li, Christan Grant, and Tim Weninger. 2022. Ask-and-verify: Span candidate generation and verification for attribute value extraction. In *Proceedings of EMNLP'22 Industry Track*, Abu Dhabi, UAE.

Tianyu Gao, Xu Han, Zhiyuan Liu, and Maosong Sun. 2019. Hybrid attention-based prototypical networks for noisy few-shot relation classification. In *Proceedings of AAAI'19*, volume 33, pages 6407–6414.

Yizhao Gao, Nanyi Fei, Guangzhen Liu, Zhiwu Lu, and Tao Xiang. 2021. Contrastive prototype learning with augmented embeddings for few-shot learning. In *Uncertainty in Artificial Intelligence*, pages 140–150. PMLR.

Eric Huang, Richard Socher, Christopher Manning, and Andrew Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of ACL'12 (Volume 1: Long Papers)*, pages 873–882, Jeju Island, Korea.

Viet Lai, Franck Dernoncourt, and Thien Huu Nguyen. 2021. Learning prototype representations across few-shot tasks for event detection. In *Proceedings of EMNLP'21*, pages 5270–5277, Online and Punta Cana, Dominican Republic.

Rongmei Lin, Xiang He, Jie Feng, Nasser Zalmout, Yan Liang, Li Xiong, and Xin Luna Dong. 2021. Pam: Understanding product images in cross product category attribute extraction. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '21, page 3262–3270, New York, NY, USA. Association for Computing Machinery.

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.

Rajendra Pamula, Jatindra Kumar Deka, and Sukumar Nandi. 2011. An outlier detection method based on clustering. In *2011 Second International Conference on Emerging Applications of Information Technology*, pages 253–256.

Joseph Reisinger and Raymond J. Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Proceedings of NAACL-HLT'10*, pages 109–117, Los Angeles, California.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, volume 30.

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. KDD '20, New York, NY, USA. Association for Computing Machinery.

Xiaochun Wang, Xiali Wang, and Mitch Wilkes. 2021. A k-nearest neighbor centroid-based outlier detection method. In *New Developments in Unsupervised Outlier Detection: Algorithms and Applications*, pages 71–112, Singapore. Springer Singapore.

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.

Li Yang, Qifan Wang, Zac Yu, Anand Kulkarni, Sumit Sanghai, Bin Shu, Jon Elsas, and Bhargav Kanagal. 2022. Mave: A product dataset for multi-source attribute value extraction. In *Proceedings of WSDM '22*, WSDM '22, page 1256–1265, New York, NY, USA.

Nasser Zalmout, Chenwei Zhang, Xian Li, Yan Liang, and Xin Luna Dong. 2021. All you need to know to build a product knowledge graph. In *Proceedings of KDD'21*, page 4090–4091, New York, NY, USA.

Guineng Zheng, Subhabrata Mukherjee, Xin Luna Dong, and Feifei Li. 2018. Opentag: Open attribute extraction from product profiles. In *Proceedings of KDD'18*.