Structure-Aware Language Model Pretraining Improves Dense Retrieval on Structured Data

Xinze Li¹, Zhenghao Liu¹*, Chenyan Xiong², Shi Yu², Yu Gu¹, Zhiyuan Liu³ and Ge Yu¹

¹Department of Computer Science and Technology, Northeastern University, China
²Microsoft Research, United States
³Department of Computer Science and Technology, Institute for AI, Tsinghua University, China

Abstract

This paper presents Structure Aware DeNse ReTrieval (SANTA) model, which encodes user queries and structured data in one universal embedding space for retrieving structured data. SANTA proposes two pretraining methods to make language models structure-aware and learn effective representations for structured data: 1) Structured Data Alignment, which utilizes the natural alignment relations between structured data and unstructured data for structure-aware pretraining. It contrastively trains language models to represent multi-modal text data and teaches models to distinguish matched structured data for unstructured texts. 2) Masked Entity Prediction, which designs an entity-oriented mask strategy and asks language models to fill in the masked entities. Our experiments show that SANTA achieves state-of-the-art on code search and product search and conducts convincing results in the zero-shot setting. SANTA learns tailored representations for multi-modal text data by aligning structured and unstructured data pairs and capturing structural semantics by masking and predicting entities in the structured data. All codes are available at https://github.com/OpenMatch/OpenMatch.

1 Introduction

Dense retrieval has shown strong effectiveness in lots of NLP applications, such as open domain question answering (Chen et al., 2017), conversational search (Qu et al., 2020; Yu et al., 2021), and fact verification (Thorne et al., 2018). It employs pretrained language models (PLMs) to encode unstructured data as high-dimensional embeddings, conduct text matching in an embedding space and return candidates to satisfy user needs (Xiong et al., 2021b; Karpukhin et al., 2020).

Besides unstructured data, structured data, such as codes, HTML documents and product descriptions, is ubiquitous in articles, books, and Web

*indicates corresponding author.
Lots of structure-aware pretraining methods are proposed to continuously train PLMs to be structure-aware and better represent structured data (Wang et al., 2021; Feng et al., 2020). They design task-specific masking strategies and pretrain PLMs with mask language modeling. Nevertheless, only using mask language modeling may not sufficiently train PLMs to conduct effective representations for structured data (Li et al., 2020; Fang et al., 2020). Some natural alignment signals between structured and unstructured data, such as code-description documentation and product description-bullet points, provide an opportunity to pretrain the structured data representations. Using these alignment signals, PLMs can be contrastively trained (Wu et al., 2020; Karpukhin et al., 2020) to match the representations of aligned structured and unstructured data and understand the semantics of structured data with the help of natural language.

In this paper, we propose Structure Aware DeNse ReTrieVAl (SANTA), a dense retrieval method on structured data. As shown in Figure 1, SANTA encodes queries and structured data in an embedding space for retrieval. SANTA designs two pretraining tasks to continuously train PLMs and make PLMs sensitive to structured data. The Structured Data Alignment task contrastively trains PLMs to align matched structured-unstructured data pairs in the embedding space, which helps to represent structured data by bridging the modality gap between structured and unstructured data. The Masked Entity Prediction task masks entities and trains PLMs to fill in the masked parts, which helps to capture semantics from structured data.

Our experiments show that SANTA achieves state-of-the-art in retrieving structured data, such as codes and products. By aligning structured and unstructured data, SANTA maps both structured and unstructured data in one universal embedding space and learns more tailored embeddings for multi-modal text data matching. The masked entity prediction task further guides SANTA to capture more crucial information for retrieval and better distinguish structured and unstructured data. Depending on these pretraining methods, SANTA can even achieve comparable retrieval results with existing code retrieval models without finetuning, showing that our structure-aware pretraining can benefit structured data understanding, multi-modal text data representation modeling and text data matching between user queries and structured data.

2 Related Work

Dense retrieval (Yu et al., 2021; Karpukhin et al., 2020; Xiong et al., 2021b; Li et al., 2021) encodes queries and documents using pretrained language model (PLM) (Devlin et al., 2019; Liu et al., 2019; Raffel et al., 2020) and maps them in an embedding space for retrieval. However, during retrieving candidates, the documents can be passages in natural language (Nguyen et al., 2016; Kwiatkowski et al., 2019), images (Chen et al., 2015), structured data documents (Lu et al., 2021) or multi-modal documents (Chang et al., 2021), which challenges existing dense retrieval models to handle different kinds of modalities of knowledge sources to build a self-contained retrieval system.

Existing work (Guo et al., 2021) also builds dense retrievers for retrieving structured data and mainly focuses on learning representations for code data. Learning more effective representations with PLMs is crucial for dense retrieval (Gao and Callan, 2021; Luan et al., 2021), thus several continuous training models are proposed. They usually employ mask language modeling to train PLMs on structured data and help to memorize the semantic knowledge using model parameters (Wang et al., 2021; Peng et al., 2020; Roziere et al., 2021).

CodeBERT uses replaced token detection (Clark et al., 2020) and masked language modeling (Devlin et al., 2019) to learn the lexical semantics of structured data (Lu et al., 2021). DOBF (Roziere et al., 2021) further considers the characteristics of code-related tasks and replaces class, function and variable names with special tokens. CodeT5 (Wang et al., 2021) not only employs the span mask strategy (Raffel et al., 2020) but also masks the identifiers in codes to teach T5 (Raffel et al., 2020) to generate these identifiers, which helps better distinguish and comprehend the identifier information in code-related tasks. Nevertheless, the mask language modeling (Devlin et al., 2019) may not sufficiently train PLMs to represent texts and show less effectiveness in text matching tasks (Chen and He, 2021; Gao et al., 2019; Li et al., 2020; Reimers and Gurevych, 2019; Li et al., 2020).

The recent development of sentence representation learning methods has achieved convincing results (Fang et al., 2020; Yan et al., 2021). The work first constructs sentence pairs using back-translation (Fang et al., 2020), some easy deforma-
Figure 2: The Structure-Aware Pretraining Methods of SANTA. We use both Structured Data Alignment (SDA) and Masked Entity Prediction (MEP) methods for pretraining.

3 Methodology

In this section, we introduce our Structure Aware DeNse ReTrievAl (SANTA) model. First, we introduce the preliminary of dense retrieval (Sec. 3.1). And then we describe our structure-aware pretraining method (Sec. 3.2).

3.1 Preliminary of Dense Retrieval

Given a query \( q \) and a structured data document \( d \), dense retriever (Karpukhin et al., 2020; Xiong et al., 2021a) encodes queries and structured data documents with pretrained language models (Devlin et al., 2019; Liu et al., 2019) and maps them in an embedding space for retrieval.

Following previous work (Ni et al., 2022), we can use T5 (Raffel et al., 2020) to encode the query \( q \) and structured data document \( d \) as low dimensional representations \( h_q \) and \( h_d \), using the representation of the first token from the decoder:

\[
h_q = \text{T5}(q); \quad h_d = \text{T5}(d).
\]

Then we can calculate the similarity score \( f(q, d) \) between the representations of query \( h_q \) and structured data document \( h_d \):

\[
f(q, d) = \text{sim}(h_q, h_d),
\]

where \( \text{sim} \) is the dot product function to calculate the relevance between query \( q \) and structured data document \( d \).

Finally, we can finetune the representations of query and document by minimizing the loss \( \mathcal{L}_{DR} \):

\[
\mathcal{L}_{DR} = -\log \frac{e^{f(q, d^+)}}{\sum_{d^- \in D^-} e^{f(q, d^-)}},
\]

where \( d^+ \) is relevant to the given query \( q \). \( D^- \) is the collection of irrelevant structured data documents, which are sampled from inbatch negatives (Karpukhin et al., 2020) or hard negatives (Xiong et al., 2021a).

3.2 Structure Aware Pretraining

Existing language models are usually pretrained on unstructured natural languages with masked language modeling (Devlin et al., 2019; Liu et al., 2019). Nevertheless, these models struggle to better understand the semantics represented by data structures, which limits the effectiveness of language models in representing structured data for retrieval (Feng et al., 2020; Wang et al., 2021).

To get more effective representations for structured data, we come up with structure-aware pretraining methods, aiming to help language models better capture the semantics behind the text structures. As shown in Figure 2, we continuously fine-
tune T5 using two pretraining tasks by minimizing the following loss function $\mathcal{L}$:

$$
\mathcal{L} = \mathcal{L}_{\text{SDA}} + \mathcal{L}_{\text{MEP}},
$$

where $\mathcal{L}_{\text{SDA}}$ and $\mathcal{L}_{\text{MEP}}$ are two loss functions from structured data alignment (SDA) (Sec. 3.2.1) and masked entity prediction (MEP) (Sec. 3.2.2), which are two subtasks of our structure-aware language model pretraining method.

### 3.2.1 Structured Data Alignment

The structured data alignment task teaches language models to optimize the embedding space by aligning structured data with unstructured data.

For the structured data document $d$, there are usually some natural language passages that share the same semantics with $d$, e.g., the descriptions of codes and bullet points of products. With the help of these text passages $p$ in natural language, we can enhance the model’s ability in representing structured data by continuously training language models to align the semantics between structured and unstructured data. Through text data alignment, the representations of structured data are benefited from the intrinsic natural language knowledge of pretrained language models.

Specifically, we can use T5 to encode the text passage and structured data document as $h_p$ and $h_d$, respectively, calculate the similarity score $f(p, d)$ between text passage $p$ and structured data document $d$, and then continuously train language models using the contrastive loss $\mathcal{L}_{\text{SDA}}$:

$$
\mathcal{L}_{\text{SDA}} = -\log \frac{e^{f(p, d^+)}}{e^{f(p, d^+) + \sum_{d^- \in D^-} e^{f(p, d^-)}}} = -f(p, d^+) + \log(e^{f(p, d^+) + \sum_{d^- \in D^-} e^{f(p, d^-)}}),
$$

where $D^-$ consists of the irrelevant structured data sampled from in-batch negatives.

As shown in Eq. 5, the structured data alignment training task helps to optimize the pretrained language models to assign similar embedding features to $<p, d^+>$ pairs and pull $d^-$ away from $p$ in the embedding space (Wang and Isola, 2020). Such a contrastive training method can bridge the semantic gap between structured and unstructured data and map them in one universal embedding space, benefiting learning representations of multi-modal text data (Liu et al., 2023).

### 3.2.2 Masked Entity Prediction

The masked entity prediction guides the language models to better understand the semantics of structured data by recovering masked entities. SANTA masks entities for continuous training instead of using the random masking in mask language modeling (Devlin et al., 2019; Raffel et al., 2020).

As shown in previous work (Sciavolino et al., 2021; Zhang et al., 2019), entity semantics show strong effectiveness in learning text data representations during retrieval. Thus, we first recognize mentioned entities that appeared in the structured data document $X_d = \{x_1, \text{ent}_1, x_2, \text{ent}_2, ..., \text{ent}_n\}$ and mask them as the input for T5 encoder module:

$$
X_d^{\text{mask}} = \{x_1, \langle\text{mask}\rangle_1, x_2, \langle\text{mask}\rangle_2, ..., x_n\},
$$

where $\langle\text{mask}\rangle_i$ is a special token to denote the $i$-th masked span. We replace the same entity with the same special token. Then we continuously train T5 to recover these masked entities using the following loss function:

$$
\mathcal{L}_{\text{MEP}} = \sum_{j=1}^{k} -\log P(Y_d(t_j) | X_d^{\text{mask}}, Y_d(t_{1,...,j-1})),
$$

where $Y_d(t_j)$ denotes the $j$-th token in the sequence $Y_d$. And $Y_d = \{\langle\text{mask}\rangle_1, \text{ent}_1, ..., \langle\text{mask}\rangle_n, \text{ent}_n\}$ denotes the ground truth sequence that contains masked entities. During training, we optimize the language model to fill up masked spans and better capture entity semantics by picking up the necessary information from contexts to recover the masked entities, understanding the structure semantics of text data, and aligning coherent entities in the structured data (Ye et al., 2020).

### 4 Experimental Methodology

In this section, we describe the datasets, evaluation metrics, baselines, and implementation details in our experiments.

**Dataset.** The datasets in our experiments consist of two parts, which are used for continuous training and finetuning, respectively.

**Continuous Training.** During continuous training, two datasets, CodeSearchNet (Husain et al., 2019) and ESCI (large) (Reddy et al., 2022), are employed to continuously train PLMs to conduct structure-aware text representations for codes and shopping products. In our experiments, we regard code documentation descriptions and product bullet
points as unstructured data for aligning structured data, codes and product descriptions, during training. More details of pretraining data processing are shown in Appendix A.2.

**Finetuning.** For downstream retrieval tasks on structured data, we use Adv (Lu et al., 2021), and ESCI (small) (Reddy et al., 2022) to finetune models for code search and product search, respectively. All data statistics are shown in Table 1. Each query in ESCI (small) has 20 products on average, which are annotated with four-class relevance labels: Exact, Substitute, Complement, and Irrelevant. We also establish a two-class testing scenario by only regarding the products that are annotated with the Exact label as relevant ones.

**Evaluation Metrics.** We use MRR@100 and NDCG@100 to evaluate model performance, which is the same as the previous work (Lu et al., 2021; Reddy et al., 2022; Feng et al., 2020).

**Baselines.** We compare SANTA with several dense retrieval models on code search and product search tasks.

We first employ three pretrained language models to build dense retrievers for structured data retrieval, including BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019) and T5 (Raffel et al., 2020), which are widely used in existing dense retrieval models (Karpukhin et al., 2020; Xiong et al., 2021a; Ni et al., 2022). All these models are trained with in-batch negatives (Karpukhin et al., 2020).

For the code search task, we also compare SANTA with three typical and task-specific models, CodeBERT (Feng et al., 2020), CodeT5 (Wang et al., 2021) and CodeRetriever (Li et al., 2022). CodeBERT inherits the BERT architecture and is trained on code corpus using both mask language modeling and replaced token detection. CodeT5 employs the encoder-decoder architecture for modeling different code-related tasks and teaches the model to focus more on code identifiers. CodeRetriever is the state-of-the-art, which continuously trains GraphCodeBERT (Guo et al., 2021) with unimodal and bimodal contrastive training losses.

**Implementation Details.** This part describes the experiment details of SANTA.

We initialize SANTA with T5-base and CodeT5-base for product search and code search. For masked entity prediction, we regard code identifiers and some noun phrases as entities in codes and product descriptions, respectively. More details about identifying entities are shown in Appendix A.3.

During continuous training, we set the learning rate as 1e-4 and 5e-5 for product search and code search, and the training epoch as 6. During finetuning, we conduct experiments by training SANTA using inbatch negatives and hard negatives. We set the training epoch to 60 and learning rate to 5e-5 for product search, while the training epoch and learning rate are 6 and 1e-5 for code search. And we follow ANCE (Xiong et al., 2021a), start from inbatch finetuned SANTA (Inbatch) model and continuously finetune it with hard negatives to conduct the SANTA (Hard Negative) model. The learning rates are set to 1e-5 and 1e-6 for product search and code search. These hard negatives are randomly sampled from the top 100 retrieved negative codes/product descriptions from the SANTA (Inbatch) model.

All models are implemented with PyTorch, Huggingface transformers (Wolf et al., 2019) and OpenMatch (Yu et al., 2023). We use Adam optimizer to optimize SANTA, set the batch size to 16 and set the warmup proportion to 0.1 in our experiments.

### 5 Evaluation Results

In this section, we focus on exploring the performance of SANTA on code search and product search tasks, the advantages of SANTA in representing structured data, and the effectiveness of proposed pretraining methods.

#### 5.1 Overall Performance

The performance of SANTA on structured data retrieval is shown in Table 2.

SANTA shows strong zero-shot ability by comparing its performance with finetuned models and achieving 6.8% improvements over finetuned CodeT5 on code search. Such impressive improvements demonstrate that our pretrained strategies have the ability to enable the advantages of PLMs in representing structured data without finetuning.

After finetuning, SANTA maintains its advantages by achieving about 8% and 2% improvements over CodeT5 and T5 on code search and product search, respectively.


<table>
<thead>
<tr>
<th>Model</th>
<th>Code Product</th>
<th>MRR</th>
<th>NDCG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Two-C</td>
<td>Four-C</td>
<td></td>
</tr>
<tr>
<td><strong>Zero-Shot</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BERT (Devlin et al., 2019)</td>
<td>0.20</td>
<td>71.46</td>
<td>72.45</td>
</tr>
<tr>
<td>RoBERTa (Liu et al., 2019)</td>
<td>0.03</td>
<td>71.25</td>
<td>72.24</td>
</tr>
<tr>
<td>CodeBERT (Feng et al., 2020)</td>
<td>0.03</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CodeRetriever (Li et al., 2022)</td>
<td>34.7</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>T5 (Raffel et al., 2020)</td>
<td>0.03</td>
<td>70.21</td>
<td>71.25</td>
</tr>
<tr>
<td>CodeT5 (Wang et al., 2021)</td>
<td>0.03</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SANTA</td>
<td>46.1</td>
<td>76.38</td>
<td>77.14</td>
</tr>
</tbody>
</table>

| **Fine-Tuning**        |        |       |
| BERT (Devlin et al., 2019) | 16.7  | 78.29 | 79.06 |
| RoBERTa (Liu et al., 2019) | 18.3  | 79.59 | 80.29 |
| CodeBERT (Feng et al., 2020) | 27.2  | -     | -     |
| CodeRetriever          | 43.0   | -     | -     |
| CodeRetriever (AR2) (Li et al., 2022) | 46.9  | -     | -     |
| T5 (Raffel et al., 2020) | 23.8  | 79.77 | 80.46 |
| CodeT5 (Wang et al., 2021) | 39.3  | -     | -     |
| SANTA (Inbatch)        | 47.3   | 80.76 | 81.41 |
| SANTA (Hard Negative)  | 47.5   | 82.59 | 83.15 |

Table 2: Retrieval Effectiveness of Different Models on Structured Data. For product search, there are two ways to evaluate model performance. Two-C regards the query-product relevance as two classes, Relevant (1) and Irrelevant (0). Four-C is consistent with the ESCI dataset (Reddy et al., 2022) and sets the relevance labels with the following four classes: Exact (1), Substitute (0.1), Complement (0.01), and Irrelevant (0).

Table 3: The Retrieval Performance of Ablation Models of SANTA on Structured Data Retrieval. Masked Entity Prediction (MEP) and Structured Data Alignment (SDA) are two pretrained tasks that are proposed by SANTA.

Compared with our baseline model, MEP and SDA show distinct performance in structured data retrieval. As expected, MEP shows almost the same performance as the baseline model. It shows that only mask language modeling usually shows less effectiveness in learning representations for structured data, even using different masking strategies. Different from MEP, SDA shows significant improvements in both structured data retrieval tasks, especially the code retrieval task. Our SDA training method contrastively trains T5 models using the alignment relations between structured data and unstructured data, which helps to bridge the modality gap between structured and unstructured data, maps structured and unstructured data in one universal embedding space, and learns more effective representations for retrieval. When adding additional task MEP to T5 (w/ SDA), the retrieval performance of SANTA is consistently improved. This phenomenon shows that mask language modeling is still effective to teach T5 to better capture the structure semantics and conduct more effective text representations for structured data by filling up the masked entities of structured data.

5.2 Ablation Study

In this subsection, we conduct ablation studies to further explore the roles of different components in SANTA on structured data retrieval.

We start from CodeT5/T5 models and continuously train CodeT5/T5 using two proposed training tasks, Masked Entity Prediction (MEP) and Structured Data Alignment (SDA) to show their effectiveness in teaching models to better learn semantics from structured data. Meanwhile, we compare MEP with the random span masking strategy (Raffel et al., 2020; Wang et al., 2021) to evaluate the effectiveness of different masking strategies. The retrieval performance in both zero-shot and finetuning settings is shown in Table 3.
Figure 3: Retrieval Effectiveness on Code Search. We sample several query-code pairs from the test split of code search data and show the ranking probability distribution of query-related codes in Figure 3(a). Then Figure 3(b) presents the learned embedding space of structured data of codes.

5.3 Embedding Visualization of Structured and Unstructured Data

This section further explores the characteristics of embedding distributions of structured and unstructured data learned by SANTA.

As shown in Figure 3, we first conduct experiments to show the retrieval effectiveness of CodeT5 and SANTA under the zero-shot setting. The ranking probability distribution of relevant query-code pairs is shown in Figure 3(a). Even though CodeT5 is pretrained with code text data, it seems that CodeT5 learns ineffective representations for structured data, assigns a uniform ranking probability distribution for all testing examples and fails to pick up the related structured data for the given queries. On the contrary, SANTA assigns much higher ranking probabilities to matched structured documents, demonstrating that our structured data alignment task has the ability to guide the model to conduct more effective text data representations to align queries with its relevant structured documents. Then we plot the embedding distribution of structured data in Figure 3(b). Distinct from the embedding distribution of CodeT5, the embeddings learned by SANTA, are more distinguishable and uniform, which are two criteria of learning more effective embedding space under contrastive training (Li et al., 2021; Wang and Isola, 2020).

Then we present the embedding distribution of documentation texts and their corresponding codes in Figure 4. Overall, depending on our structure-aware pretraining methods, SANTA conducts a more uniform embedding space than CodeT5 and makes the representations of structured and unstructured data more distinguished in the embedding space. Then we analyze the effectiveness of our continuous training methods, Masked Entity Prediction (MEP) and Structured Data Alignment (SDA). By comparing Figure 4(b) with Figure 4(a), our structured data alignment task indeed helps PLMs to align the representations of code and documentation, which reduces the distance between matched unstructured-structured data pairs and mixes the multi-modal embeddings thoroughly in the embedding space. After adding the masked entity prediction training task to CodeT5 (w/ SDA) (from Figure 4(b) to Figure 4(d)), the embedding distributions of code and documentation become distinguished again, demonstrating that masked entity prediction can help models capture different semantics from different data modalities to represent unstructured/structured data. Besides, by comparing Figure 4(d) with Figure 4(c), the structured data alignment task also makes the boundary of the embedding clusters of code and documentation clearer. The main reason lies in that these embeddings are assigned to appropriate positions for aligning matched code-documentation pairs with the help of our structured data alignment task.
Table 4: Case Studies. We sample four cases from the test datasets of code search and product search to show the effectiveness of SANTA. The matched text phrases are highlighted.

5.4 Attention Mechanism of SANTA

This section presents the attention mechanism of SANTA during encoding structured data. In Figure 5, we randomly sample a small piece of code and a text sequence of product descriptions to plot the attention distribution.

The attention weight distributions on code search are shown in Figure 5(a). Compared with CodeT5 (w/ SDA) and SANTA calibrate the attention weights from the “if” token to the “>” token. The “>” token is a logical operation, which indicates the usage of the code. SANTA thrives on the structured data alignment task and captures these important semantic clues to represent codes. Compared with CodeT5 (w/ SDA), SANTA decreases its attention weights on code identifiers, such as “x” and “y”, and shares more attention weights to “If” and “>”. These identifiers can be replaced with attribute ones and are less important than these logical operations to understand code semantics. Thus, SANTA adjusts its attention weights to logical tokens to understand structured data, which is benefited from pretraining with the masked entity prediction task.

Figure 5(b) shows the attention distribution on product search. T5 (w/ SDA) assigns more attention weights to the product attribute “Green” than T5, as well as highlights the sequence boundary tokens of product attributes. Nevertheless, for the product “privacy fence screen”, “Large” is a more important attribute than “Green”. SANTA captures such semantic relevance, which confirms that our masked entity prediction task indeed helps to improve the semantic understanding ability of language models on structured data.
5.5 Case Studies

Finally, we show several cases in Table 4 to analyze the ranking effectiveness of SANTA.

In the first case, SANTA directly matches queries and codes through the text snippet “poll the driver status”. It demonstrates that SANTA has the ability to distinguish the differences between code and documentation and pick up the necessary text clues for matching queries and codes. Then the second case illustrates that SANTA is effective in understanding codes by capturing the structure semantics of codes and matching queries and codes by capturing some keywords in codes, such as “copied” and “path”. The last two cases are from product search and the product description is more like natural language. SANTA also shows its effectiveness on identifying some important entities, such as “Hair Dye” and “fence screen”, to match queries and products.

6 Conclusion

This paper proposes SANTA, which pretrains language models to understand structure semantics of text data and guides language models to map both queries and structured texts in one universal embedding space for retrieval. SANTA designs both structured text alignment and masked entity prediction tasks to continuously train pretrained language models to learn the semantics behind data structures. Our experiments show that SANTA achieves state-of-the-art on code and product search by learning more tailored representations for structured data, capturing semantics from structured data and bridging the modality gap between structured and unstructured data.

Limitations

Even though SANTA shows strong effectiveness on learning the representation of structured data, it heavily depends on the alignment signals between structured and unstructured data. Such alignment relations can be witnessed everywhere, but the quality of constructed pairs of structured and unstructured data directly determines the effectiveness of SANTA. Besides, we use the product bullet points and code descriptions as the unstructured data in our experiments, which is designed for specific tasks and limits the model’s generalization ability. On the other hand, SANTA mainly focuses on evaluating the structured data understanding ability through text data representation and matching. It is still unclear whether SANTA outperforms baseline models in all downstream tasks, such as code summarization and code generation.

Acknowledgments

This work is supported by the Natural Science Foundation of China under Grant No. 62206042, No. 62137001 and No. 62272093, the Fundamental Research Funds for the Central Universities under Grant No. N2216013 and No. N2216017, China Postdoctoral Science Foundation under Grant No. 2022M710022, and National Science and Technology Major Project (J2019-IV-0002-0069).

References


Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. Microsoft coco captions: Data collection and evaluation server. CoRR.


Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In Proceedings of ICLR.


Yu Meng, Chenyan Xiong, Payal Bajaj, Saurabh Tiwary, Paul Bennett, Jiawei Han, and Xia Song. 2021. Coco-im: Correcting and contrasting text sequences for language model pretraining. In Proceedings of NeurIPS.


2022. Shopping queries dataset: A large-scale ESCI benchmark for improving product search. CoRR.


Table 5: Data Statistics of Pretraining Data. “Entities” denotes the proportion of identified entities in the structured data.

<table>
<thead>
<tr>
<th>Task</th>
<th>Positive Pairs</th>
<th>Entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>429,596</td>
<td>28.6%</td>
</tr>
<tr>
<td>PHP</td>
<td>514,127</td>
<td>17.8%</td>
</tr>
<tr>
<td>Go</td>
<td>317,824</td>
<td>17.1%</td>
</tr>
<tr>
<td>Java</td>
<td>454,433</td>
<td>24.4%</td>
</tr>
<tr>
<td>JavaScript</td>
<td>122,682</td>
<td>15.4%</td>
</tr>
<tr>
<td>Ruby</td>
<td>487,90</td>
<td>28.8%</td>
</tr>
<tr>
<td>Product</td>
<td>331,590</td>
<td>20.1%</td>
</tr>
</tbody>
</table>

A Appendix

A.1 License

For all datasets in our experiments, Adv and CodeSearchNet use MIT License, while ESCI uses Apache License 2.0. All of these licenses and agreements allow their data for academic use.

A.2 Construction of Pretraining Data

In this subsection, we show how to process the pretraining data and construct structured-unstructured data for code/product search. During pretraining, we use inbatch negatives to optimize SANTA and all data statistics are shown in Table 5.

As shown in Figure 6, we show some examples to show how to construct structured-unstructured data pairs for pretraining. For code retrieval tasks, code snippets have corresponding documentation descriptions, which describe the purpose and function of these code snippets. Thus, the code documentation and its corresponding code snippet are regarded as a positive training pair.

For product retrieval tasks, structured product descriptions usually have corresponding unstructured bullet points, which provide key points about the products. We randomly select one bullet point of items and use its corresponding product description to construct a positive training pair.

A.3 Additional Experimental Details of Entities Identification on Structured Data

We show some examples of entity identifications on structured data in Figure 7.

For codes, we follow Wang et al. (2021) and regard code identifiers as entities such as variables, function names, external libraries and methods.
Table 6: Code Retrieval Evaluations of SANTA. Because of the GPU memory limitation, we set the batch size as 128 during pretraining and finetuning, which is different with Li et al. (2022). All models are evaluated using MRR.

<table>
<thead>
<tr>
<th>Language</th>
<th>Query</th>
<th>CodeSearch</th>
<th>Adv</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language</td>
<td>Train</td>
<td>Dev</td>
<td>Test</td>
</tr>
<tr>
<td>Python</td>
<td>251,820</td>
<td>13,914</td>
<td>14,918</td>
</tr>
<tr>
<td>PHP</td>
<td>241,241</td>
<td>12,982</td>
<td>14,014</td>
</tr>
<tr>
<td>Go</td>
<td>167,288</td>
<td>7,325</td>
<td>8,122</td>
</tr>
<tr>
<td>Java</td>
<td>164,923</td>
<td>5,183</td>
<td>10,955</td>
</tr>
<tr>
<td>JavaScript</td>
<td>58,025</td>
<td>3,885</td>
<td>3,291</td>
</tr>
<tr>
<td>Ruby</td>
<td>24,927</td>
<td>1,400</td>
<td>1,261</td>
</tr>
</tbody>
</table>

Specifically, we use BytesIO and tree_sitter\(^1\) to identify entities in Python and other programming languages, respectively. For product descriptions, we use the NLTK tool\(^2\) to identify nouns and proper nouns that appear in both product descriptions and titles and regard them as entities.

In our experiments, we replace the same entities with the same special tokens and ask SANTA to generate these masked entities (Eq. 7). These special tokens come from the predefined vocabulary of T5, such as \{<extra_id_0>, <extra_id_1>, ..., <extra_id_99>\}. The proportions of identified entities in pretraining data are shown in Table 5.

A.4 Additional Evaluation Results of SANTA

In this experiment, we follow Li et al. (2022), keep the same evaluation settings and evaluate the retrieval effectiveness of SANTA on CodeSearch dataset. The dataset consists of code retrieval tasks on six programming languages, including Ruby, Javascript, Go, Python, Java, and PHP. We show the data statistics of CodeSearch in Table 7. Since CodeT5 and CodeRetriever don’t release their data processing code for pretraining, we can only refer to the tutorial\(^3\) to process data. When we evaluate SANTA on CodeSearch, the instances in testing and development sets are filtered out from CodeSearchNet dataset for pretraining. Some codes that can not be parsed are also filtered out, because the data processing details are not available\(^4\).

During continuous pretraining, we set the batch size, learning rate and epoch as 128, 5e-5 and 10, respectively. During finetuning, we set the learning rate as 2e-5 and 1e-5 for CodeSearch and Adv, and set batch size and epoch as 128 and 12. We use inbatch negatives with one hard negative for finetuning and the hard negative is randomly sampled from the top 100 retrieved negative codes by pretrained SANTA. The warm-up ratio is 0.1.

The performance of SANTA on CodeSearch and Adv is shown in Table 6. Under the zero-shot setting, SANTA still outperforms CodeRetriever (Li et al., 2022) with about 2% improvements, which shows that the advances of SANTA can be generalized to different structured data retrieval tasks. Moreover, SANTA also shows strong zero-shot ability by achieving comparable performance with the finetuned CodeBERT, GraphCodeBERT and CodeT5 models. After finetuning, SANTA achieves more than 3.7% improvements over CodeT5 on CodeSearch. All these encouraged experiment results further demonstrate that our structure-aware pretraining method indeed helps language models to capture the structure semantics behind the text data. The retrieval performance on Adv dataset illustrates that the retrieval effectiveness of SANTA can be further improved by increasing the batch size from 16 to 128.

\(^1\)https://github.com/tree-sitter/tree-sitter
\(^2\)https://www.nltk.org/
\(^3\)https://github.com/github/CodeSearchNet/blob/master/notebooks/ExploreData.ipynb
\(^4\)https://github.com/salesforce/CodeT5/issues/64
ACL 2023 Responsible NLP Checklist

A For every submission:

✔️ A1. Did you describe the limitations of your work?
   In the section of Limitations.

✘ A2. Did you discuss any potential risks of your work?
   Our structure-aware language model uses public datasets and pretrained language model, so there are no potential risks.

✔️ A3. Do the abstract and introduction summarize the paper’s main claims?
   In abstract and Section 1.

✘ A4. Have you used AI writing assistants when working on this paper?
   Left blank.

B ✔️ Did you use or create scientific artifacts?
   In Section 4.

✔️ B1. Did you cite the creators of artifacts you used?
   In Section 4.

✔️ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
   In Appendix A.1.

✘ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
   Not applicable. Left blank.

✘ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
   Not applicable. Left blank.

✘ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
   Not applicable. Left blank.

✔️ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
   In Section 4.

C ✔️ Did you run computational experiments?
   In Section 4.

✔️ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
   In Section 4.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.
C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
   *In Section 4.*

C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
   *In Section 4.*

C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
   *In Section 4.*

D  ✗ Did you use human annotators (e.g., crowdworkers) or research with human participants?
   *Left blank.*

D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
   *No response.*

D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants’ demographic (e.g., country of residence)?
   *No response.*

D3. Did you discuss whether and how consent was obtained from people whose data you’re using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
   *No response.*

D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
   *No response.*

D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
   *No response.*