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

Xinze Li[1], **Zhenghao Liu**[1]*, **Chenyan Xiong**[2], **Shi Yu**[3], **Yu Gu**[1], **Zhiyuan Liu**[3] **and Ge Yu**[1]
[1]Department of Computer Science and Technology, Northeastern University, China
[2]Microsoft Research, United States
[3]Department of Computer Science and Technology, Institute for AI, Tsinghua University, China
Beijing National Research Center for Information Science and Technology, China

Abstract

This paper presents **S**tructure **A**ware De**N**se Re**T**riev**A**l (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 pages, and plays the same important roles in understanding text data. Learning the semantics behind text structures to represent structured data is crucial to building a more self-contained retrieval system. The structured data modeling stimulates researchers to build several benchmarks to evaluate model performance, such as code search and product search (Husain et al., 2019; Reddy et al., 2022). The structured data retrieval tasks require models to retrieve structured data according to user queries. Dense retrieval (Karpukhin et al., 2020; Li et al., 2022) shows a promising way to build a retrieval system on structured data by encoding user queries and structured data in an embedding space and conducting text matching using the embedding similarity. Nevertheless, without structure-aware pretraining, most PLMs lack the necessary knowledge to understand structured data and conduct effective representations for retrieval (Feng et al.,

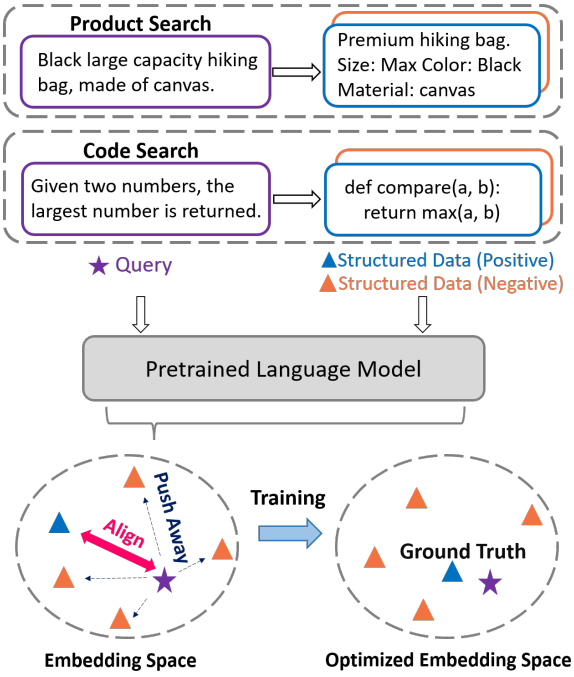


Figure 1: Dense Retrieval Pipeline on Structured Data.

*indicates corresponding author.

Findings of the Association for Computational Linguistics: ACL 2023, pages 11560–11574

2020; Hu et al., 2022; Gururangan et al., 2020).

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 **S**tructure **A**ware De**N**se Re**T**riev**A**l (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. Leaning 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; Feng 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-

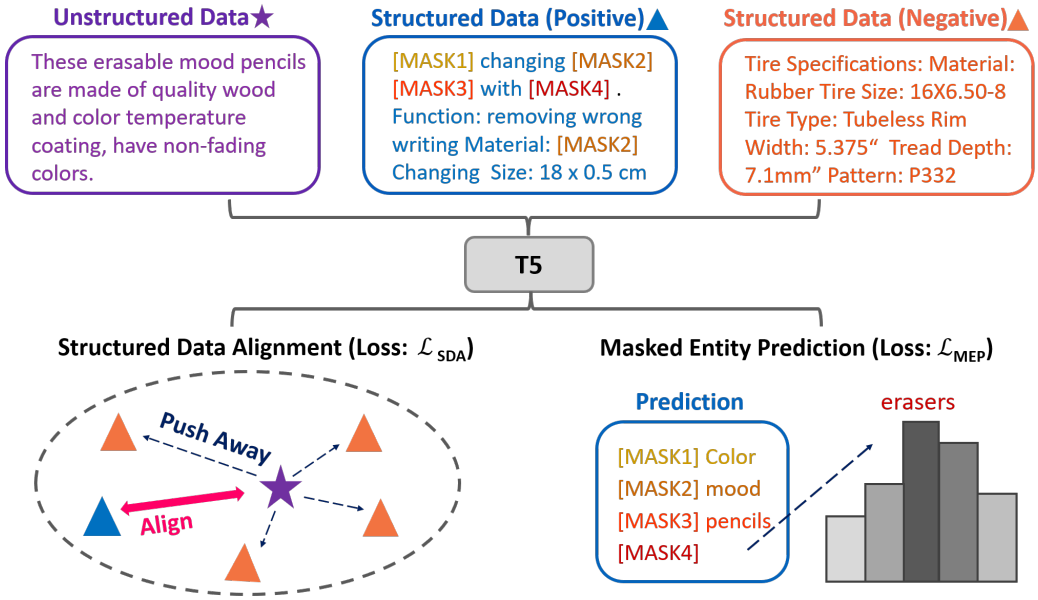


Figure 2: The Structure-Aware Pretraining Methods of SANTA. We use both Structured Data Alignment (SDA) and Masked Entity Prediction (MEP) methods for pretraining.

tion operations (Wu et al., 2020), original sequence cropping (Meng et al., 2021) or adding dropout noise (Gao et al., 2021). Then they contrastively train PLMs to learn sentence representations that can be used to distinguish the matched sentence pairs with similar semantics.

3 Methodology

In this section, we introduce our **S**tructure **A**ware De**N**se Re**T**riev**A**l (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:

$$h_q = \text{T5}(q); h_d = \text{T5}(d). \quad (1)$$

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

$$f(q, d) = sim(h_q, h_d), \quad (2)$$

where 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 $\mathcal{L}_{\text{DR}}$:

$$\mathcal{L}_{\text{DR}} = -\log \frac{e^{f(q,d^+)}}{e^{f(q,d^+)} + \sum_{d^- \in \mathcal{D}^-} e^{f(q,d^-)}}, \quad (3)$$

where d^+ is relevant to the given query q. $\mathcal{D}^-$ 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 $\mathcal{L}$:

$$\mathcal{L} = \mathcal{L}_{\text{SDA}} + \mathcal{L}_{\text{MEP}}, \tag{4}$$

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 $\mathcal{L}_{\text{SDA}}$:

$$\begin{aligned}\mathcal{L}_{\text{SDA}} &= -\log \frac{e^{f(p,d^+)}}{e^{f(p,d^+)} + \sum_{d^- \in D^-} e^{f(p,d^-)}} \\ &= -f(p,d^+) + \log(e^{f(p,d^+)} + \sum_{d^- \in D^-} e^{f(p,d^-)}),\end{aligned} \tag{5}$$

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:

$$X_d^{\text{mask}} = \{x_1, \text{<mask>}_1, x_2, \text{<mask>}_2, ..., x_n\}, \tag{6}$$

where <mask>$_i$ 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:

$$\mathcal{L}_{\text{MEP}} = \sum_{j=1}^{k} -\log P(Y_d(t_j) | X_d^{\text{mask}}, Y_d(t_{1,...,j-1})), \tag{7}$$

where $Y_d(t_j)$ denotes the j-th token in the sequence Y_d. And $Y_d = \{\text{<mask>}_1, \text{ent}_1, ..., \text{<mask>}_n, \text{ent}_n\}$ 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

Split	Code Search	Product Search	
	Query-Code Pair	Query	Product
Train	251,820	18,277	367,946
Dev	9,604	2,611	51,706
Test	19,210	8,956	181,701

Table 1: Data Statistics of Model Finetuning.

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

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

Model	Code	Product	
	MRR	NDCG	
		Two-C	Four-C
Zero-Shot			
T5 (Baseline)	0.03	70.21	71.25
T5 (w/ MEP)	0.03	70.56	71.58
T5 (w/ SDA)	45.01	76.64	77.40
SANTA (Span Mask)	35.88	**77.37**	**78.11**
SANTA (Entity Mask)	**46.08**	76.38	77.14
Fine-Tuning			
T5 (Baseline)	39.30	79.77	80.46
T5 (w/ MEP)	38.46	79.50	80.29
T5 (w/ SDA)	46.98	80.42	81.11
SANTA (Span Mask)	42.11	80.31	80.99
SANTA (Entity Mask)	**47.28**	**80.76**	**81.41**

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.

product search, respectively. It shows the critical role of structure-aware pretraining, which makes language models sensitive to text data structures and better represents structured data. On code retrieval, SANTA outperforms the state-of-the-art code retrieval model CodeRetriever with 4.3% improvements under the same inbatch training setting. SANTA also beats CodeRetriever (AR2), which is finetuned with more sophisticated training strategies (Zhang et al., 2022) and the larger batch size.

Besides, we show the retrieval performance of SANTA on CodeSearch dataset in Appendix A.4.

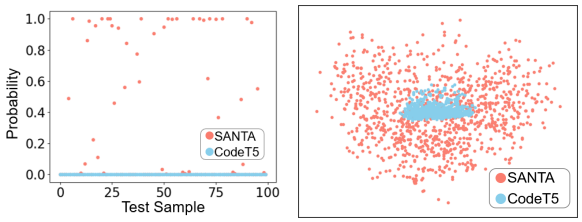
5.2 Ablation Study

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

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.

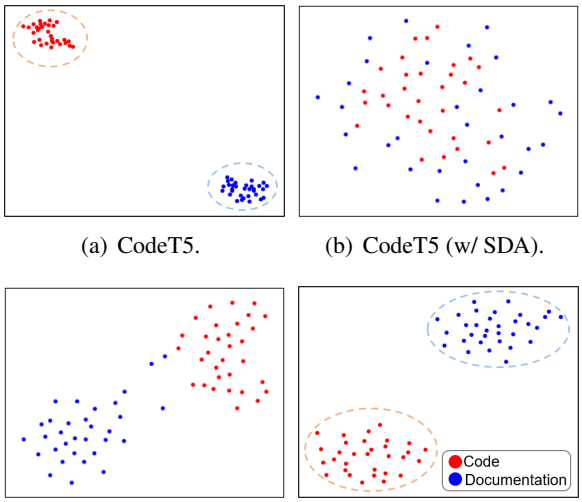
Compared with our baseline model, MEP and SDA show distinct performance in structured data retrieval. As expected, MEP shows almost the same performance as the baseline model. It shows that only mask language modeling usually shows less effectiveness in learning representations for structured data, even 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.

We also compare different masking strategies that are used during mask language modeling. Our entity masking strategy usually outperforms the random span masking strategy, showing the crucial role of entities in structured data understanding. With the masked entity prediction task, SANTA achieves comparable ranking performance with finetuned models, which illustrates that structure-



(a) Ranking Probability of Matched Text Data Pairs. (b) Embedding Distribution of Structured Data.

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.



(a) CodeT5. (b) CodeT5 (w/ SDA).

(c) CodeT5 (w/ MEP). (d) SANTA.

Figure 4: Embedding Visualization of Different Models using T-SNE. We randomly sample 32 codes and 32 code documentation texts from the testing set of code retrieval and plot their embedding distribution.

aware pretraining is starting to benefit downstream tasks, such as structured data retrieval. The next experiment further explores how these pretraining strategies guide models to learn representations of structured/unstructured data.

5.3 Embedding Visualization of Structured and Unstructured Data

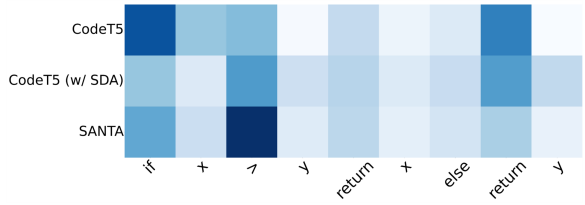
This section further explores the characteristics of embedding distributions of structured and unstructured data learned by SANTA.

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).

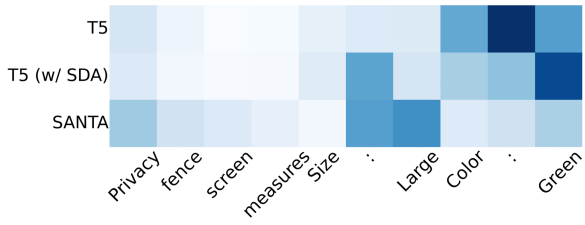
Then we present the embedding distribution of documentation texts and their corresponding codes in Figure 4. Overall, depending on our structure-aware pretraining methods, SANTA conducts a more uniform embedding space than CodeT5 and makes the representations of structured and unstructured data more 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.

Model	SANTA	CodeT5/T5
Query	Construct the command to **poll the driver status**	
Rank	1	1
Snippet	... arg_0 . _connection ['master']] if arg_0 . _driver_id : arg_1 += ["–**status**" , arg_0 . _driver_id] else : raise AirflowException ("–**Invalid status: attempted to poll driver** ...	def Func (arg_0) : return os . path . join (get_user_config_dir (arg_0 . app_name , arg_0 . app_author) , arg_0 . filename)
Query	Attempt to **copy path with storage**.	
Rank	1	1
Snippet	... if arg_2 in arg_0 . copied_files : return arg_0 . log ("Skipping '%s' (already –**copied** earlier)" % arg_1) if not arg_0 . delete_file (arg_1 , arg_2 , arg_3) : return arg_4 = arg_3 . –**path** (arg_1) ...	'... arg_0) : if arg_0 . _api_arg : arg_1 = str (arg_0 ._api_arg) else : arg_1 = arg_0 . _name if arg_0 . _parent : return '/' . join (filter (None , [arg_0 . _parent . Func , arg_1])) ...'
Query	#1 black natural **hair dye without ammonia or peroxide**	
Rank	1	1
Snippet	... naturcolor Haircolor **Hair Dye** - Light Burdock, 4 Ounce (5N) naturcolor 5n light burdock permanent herbal Ingredients: haircolor gel utilizes herbs to cover grey as **opposed to chemicals** ...	... Naturtint Permanent Hair Color 5N Light Chestnut Brown (Pack of 1), Ammonia Free, Vegan, Cruelty Free, up to 100% Gray Coverage, Long Lasting Results...
Query	!**qscreen fence** without holes	
Rank	2	2
Snippet	... Material: HDPE+Brass Color: Green Size(L x W): About 6'x50" Package included: Garden **fence privacy screen***1 Straps*80 ...	... Windscreen Cover Fabric Shade Tarp Netting Mesh Cloth - Commercial Grade 170 GSM - Cable Zip Ties Included - We Make Custom Size..

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**.



(a) Code Search.



(b) Product Search.

Figure 5: Visualization of Attention Distribution of SANTA. The cross attention weight distributions from the decoder module to encoded token embeddings are plotted. Darker blue indicates a higher attention weight.

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, 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

Yingshan Chang, Mridu Narang, Hisami Suzuki, Guihong Cao, Jianfeng Gao, and Yonatan Bisk. 2021. Webqa: Multihop and multimodal qa. In *Proceedings of CVPR*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of ACL*, pages 1870–1879.

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*.

Xinlei Chen and Kaiming He. 2021. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758.

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*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Hongchao Fang, Sicheng Wang, Meng Zhou, Jiayuan Ding, and Pengtao Xie. 2020. Cert: Contrastive self-supervised learning for language understanding.

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. 2020. CodeBERT: A pre-trained model for programming and natural languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1536–1547.

Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2019. Representation degeneration problem in training natural language generation models. In *Proceedings of ICLR*.

Luyu Gao and Jamie Callan. 2021. Condenser: a pre-training architecture for dense retrieval. In *Proceedings of EMNLP*, pages 981–993.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of EMNLP*, pages 6894–6910.

Daya Guo, Shuo Ren, Shuai Lu, Zhangyin Feng, Duyu Tang, Shujie Liu, Long Zhou, Nan Duan, Alexey Svyatkovskiy, Shengyu Fu, Michele Tufano, Shao Kun Deng, Colin B. Clement, Dawn Drain, Neel Sundaresan, Jian Yin, Daxin Jiang, and Ming Zhou. 2021. Graphcodebert: Pre-training code representations with data flow. In *Proceedings of ICLR*.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of ACL*, pages 8342–8360.

Xiaomeng Hu, Shi Yu, Chenyan Xiong, Zhenghao Liu, Zhiyuan Liu, and Ge Yu. 2022. P3 ranker: Mitigating the gaps between pre-training and ranking fine-tuning with prompt-based learning and pre-finetuning. In *Proceedings of SIGIR*, pages 1956–1962.

Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. 2019. Codesearchnet challenge: Evaluating the state of semantic code search. *CoRR*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of EMNLP*, pages 6769–6781.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, pages 452–466.

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. 2020. On the sentence embeddings from pre-trained language models. In *Proceedings of EMNLP*, pages 9119–9130.

Xiaonan Li, Yeyun Gong, Yelong Shen, Xipeng Qiu, Hang Zhang, Bolun Yao, Weizhen Qi, Daxin Jiang, Weizhu Chen, and Nan Duan. 2022. Coderetriever: Unimodal and bimodal contrastive learning.

Yizhi Li, Zhenghao Liu, Chenyan Xiong, and Zhiyuan Liu. 2021. More robust dense retrieval with contrastive dual learning. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*, pages 287–296.

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.

Zhenghao Liu, Chenyan Xiong, Yuanhuiyi Lv, Zhiyuan Liu, and Ge Yu. 2023. Universal vision-language dense retrieval: Learning a unified representation space for multi-modal retrieval. In *Proceedings of ICLR*.

Shuai Lu, Daya Guo, Shuo Ren, Junjie Huang, Alexey Svyatkovskiy, Ambrosio Blanco, Colin B. Clement, Dawn Drain, Daxin Jiang, Duyu Tang, Ge Li, Lidong Zhou, Linjun Shou, Long Zhou, Michele Tufano, Ming Gong, Ming Zhou, Nan Duan, Neel Sundaresan, Shao Kun Deng, Shengyu Fu, and Shujie Liu. 2021. Codexglue: A machine learning benchmark dataset for code understanding and generation. In *Proceedings of NeurIPS*.

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics*, pages 329–345.

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

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human-generated machine reading comprehension dataset. In *CoCo@ NIPs*.

Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. 2022. Sentence-t5: Scalable sentence encoders from pretrained text-to-text models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1864–1874.

Chen Qu, Liu Yang, Cen Chen, Minghui Qiu, W. Bruce Croft, and Mohit Iyyer. 2020. Open-retrieval conversational question answering. In *Proceedings of SIGIR*, pages 539–548.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, (140):1–67.

Chandan K. Reddy, Lluís Màrquez, Fran Valero, Nikhil Rao, Hugo Zaragoza, Sambaran Bandyopadhyay, Arnab Biswas, Anlu Xing, and Karthik Subbian.

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

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of EMNLP*, pages 3982–3992.

Baptiste Roziere, Marie-Anne Lachaux, Marc Szafraniec, and Guillaume Lample. 2021. Dobf: A deobfuscation pre-training objective for programming languages. In *Proceedings of NeurIPS*.

Christopher Sciavolino, Zexuan Zhong, Jinhyuk Lee, and Danqi Chen. 2021. Simple entity-centric questions challenge dense retrievers. In *Proceedings of EMNLP*, pages 6138–6148.

James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal. 2018. The fact extraction and VERification (FEVER) shared task. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 1–9.

Tongzhou Wang and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *Proceedings of ICML*, pages 9929–9939.

Yue Wang, Weishi Wang, Shafiq Joty, and Steven C.H. Hoi. 2021. CodeT5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In *Proceedings of EMNLP*, pages 8696–8708.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing.

Zhuofeng Wu, Sinong Wang, Jiatao Gu, Madian Khabsa, Fei Sun, and Hao Ma. 2020. Clear: Contrastive learning for sentence representation.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021a. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *Proceedings of ICLR*.

Wenhan Xiong, Xiang Lorraine Li, Srini Iyer, Jingfei Du, Patrick S. H. Lewis, William Yang Wang, Yashar Mehdad, Scott Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oguz. 2021b. Answering complex open-domain questions with multi-hop dense retrieval. In *Proceedings of ICLR*.

Yuanmeng Yan, Rumei Li, Sirui Wang, Fuzheng Zhang, Wei Wu, and Weiran Xu. 2021. ConSERT: A contrastive framework for self-supervised sentence representation transfer. In *Proceedings of ACL*, pages 5065–5075.

Deming Ye, Yankai Lin, Jiaju Du, Zhenghao Liu, Peng Li, Maosong Sun, and Zhiyuan Liu. 2020. Coreferential Reasoning Learning for Language Representation. In *Proceedings of EMNLP*, pages 7170–7186.

Shi Yu, Zhenghao Liu, Chenyan Xiong, Tao Feng, and Zhiyuan Liu. 2021. Few-shot conversational dense retrieval. In *Proceedings of SIGIR*.

Shi Yu, Zhenghao Liu, Chenyan Xiong, and Zhiyuan Liu. 2023. Openmatch-v2: An all-in-one multi-modality plm-based information retrieval toolkit. In *Proceedings of SIGIR*.

Hang Zhang, Yeyun Gong, Yelong Shen, Jiancheng Lv, Nan Duan, and Weizhu Chen. 2022. Adversarial retriever-ranker for dense text retrieval. In *Proceedings of ICLR*.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced language representation with informative entities. In *Proceedings of ACL*, pages 1441–1451.

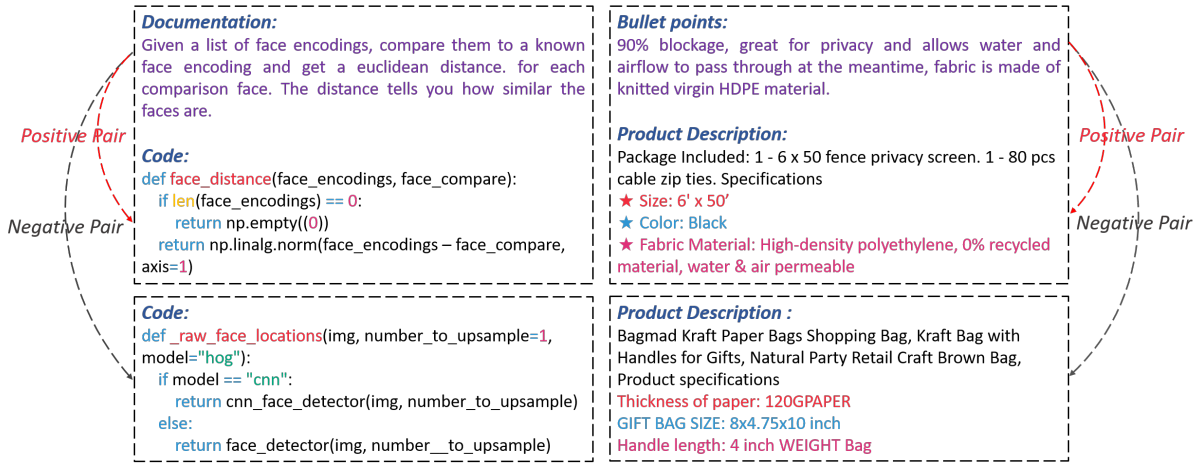
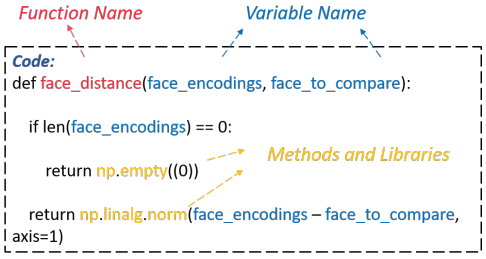


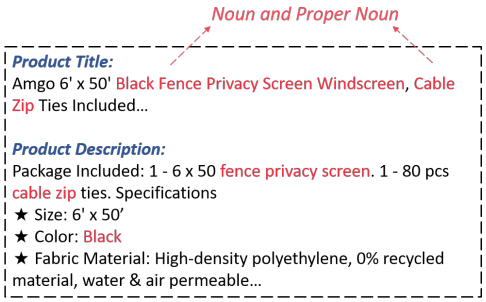
Figure 6: Examples of Positive and Negative Pairs of Pretraining Data.

Task	Positive Pairs	Entities
Python	429,596	28.6%
PHP	514,127	17.8%
Go	317,824	17.1%
Java	454,433	24.4%
JavaScript	122,682	15.4%
Ruby	487,90	28.8%
Product	331,590	20.1%

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



(a) Code Search.



(b) Product Search.

Figure 7: The Illustration of Identified Entities in Structured Data. All entities of different functions are annotated with different colors.

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.

Model	CodeSearch							Adv
	Ruby	Javascript	Go	Python	Java	PHP	Overall	
Zero-Shot								
GraphCodeBERT	1.5	0.4	0.2	0.4	0.7	2.1	0.88	0.5
CodeRetriever	68.7	**63.7**	87.6	67.7	**69.0**	62.8	69.1	34.7
SANTA	**72.6**	62.4	**88.9**	**70.0**	68.6	**62.8**	**70.9**	**48.1**
Fine-Tuning								
CodeBERT	67.9	62.0	88.2	67.2	67.6	62.8	69.3	27.2
GraphCodeBERT	70.3	64.4	89.7	69.2	69.1	64.9	71.3	35.2
CodeT5	71.9	65.5	88.8	69.8	68.6	64.5	71.5	39.3
CodeRetriever (Inbatch)	**75.3**	69.5	91.6	73.3	74.0	68.2	75.3	43.0
CodeRetriever (Hard Negative)	75.1	**69.8**	**92.3**	**74.0**	**74.9**	**69.1**	**75.9**	45.1
SANTA (Hard Negative)	74.7	68.6	91.8	73.7	73.7	68.6	75.2	**48.6**

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.

Language	Query			Document
	Train	Dev	Test	
Python	251,820	13,914	14,918	43,827
PHP	241,241	12,982	14,014	52,660
Go	167,288	7,325	8,122	28,120
Java	164,923	5,183	10,955	40,347
JavaScript	58,025	3,885	3,291	13,981
Ruby	24,927	1,400	1,261	4,360

Table 7: Data Statistics of CodeSearch Dataset. The document collections consist of candidate codes.

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.