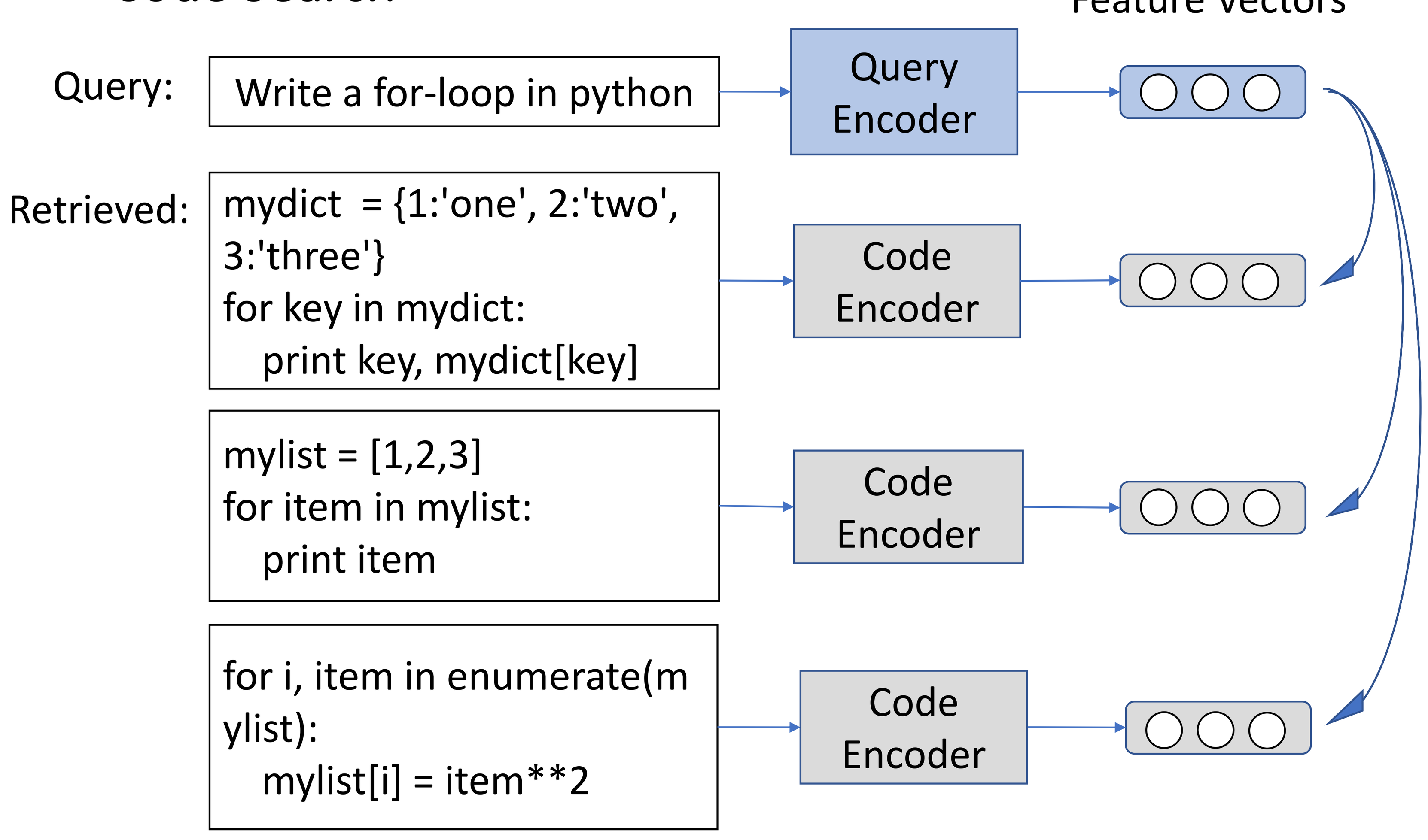
Exploring Representation-Level Augmentation for Code Search

Haochen Li[1], Chunyan Miao[1,2], Cyril Leung[1,2], Yanxian Huang[3], Yuan Huang[3], Hongyu Zhang[4], and Yanlin Wang[3]

[1]Nanyang Technological University, Singapore
[2]China-Singapore International Joint Research Institute
[3]Sun Yat-sen University [4]The University of Newcastle

Paper: https://arxiv.org/abs/2210.12285

Code Search



Code search, which aims at retrieving the most relevant code fragment for a given natural language query, is a common activity in software development practice.

Theoretical Analysis

Theorem 1 Optimizing InfoNCE loss L_N improves lower bounds of mutual information $I(q,c)$ for a positive pair:

$$I(q,c) \geq \log(B) - L_N$$

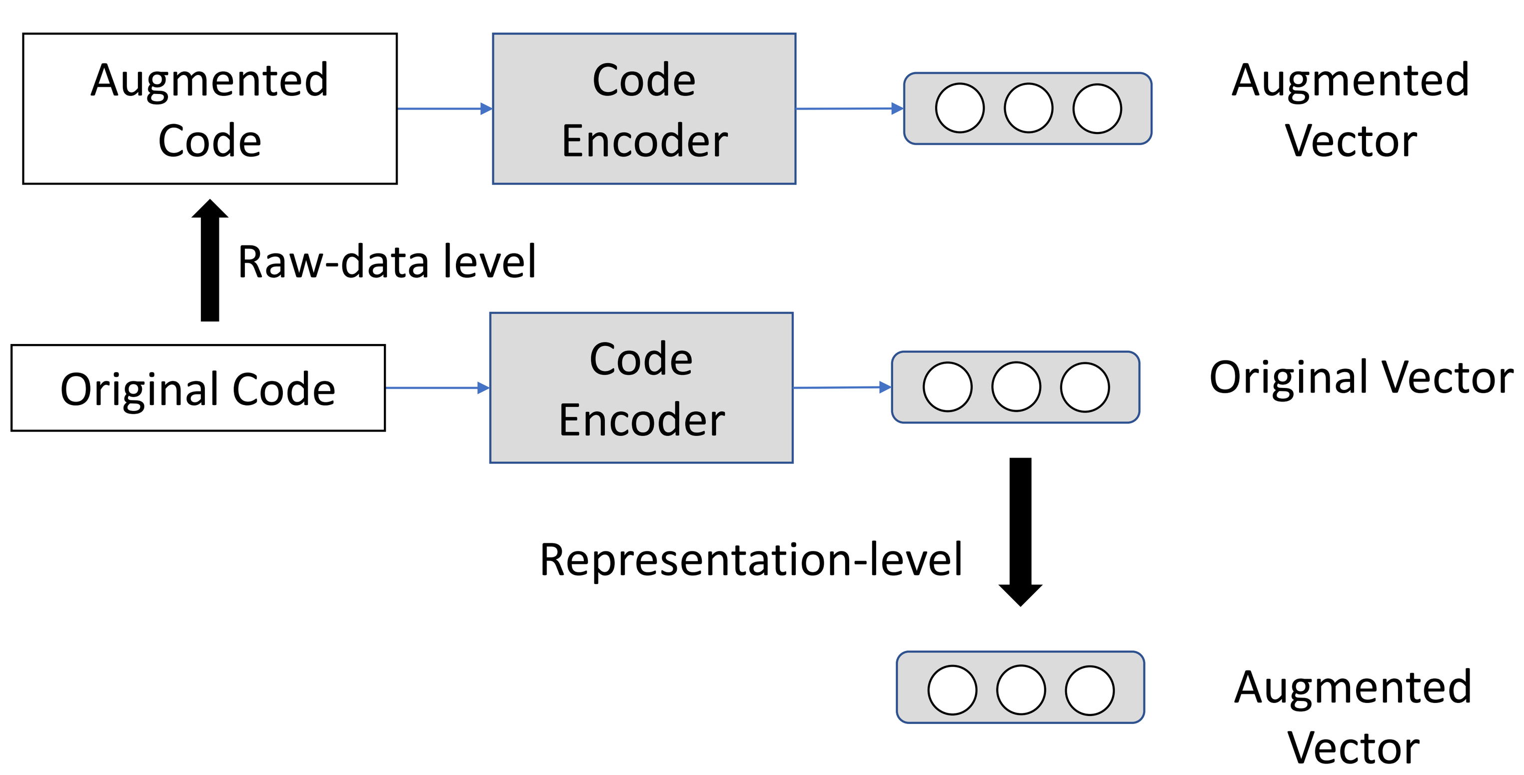
where $q \in Q, c \in C$, and B is the size of sets.

Theorem 2 Optimizing InfoNCE loss L_N with representation-level augmentation improves a tighter lower bounds of mutual information $I(q,c)$ for a positive pair:

$$I(q,c) \geq \frac{1}{\alpha^2}(\log(NB) - L_N - \alpha\beta \cdot I(q,c^-) - \alpha\beta \cdot I(q^-,c) - \beta^2 \cdot I(q^-,c^-))$$

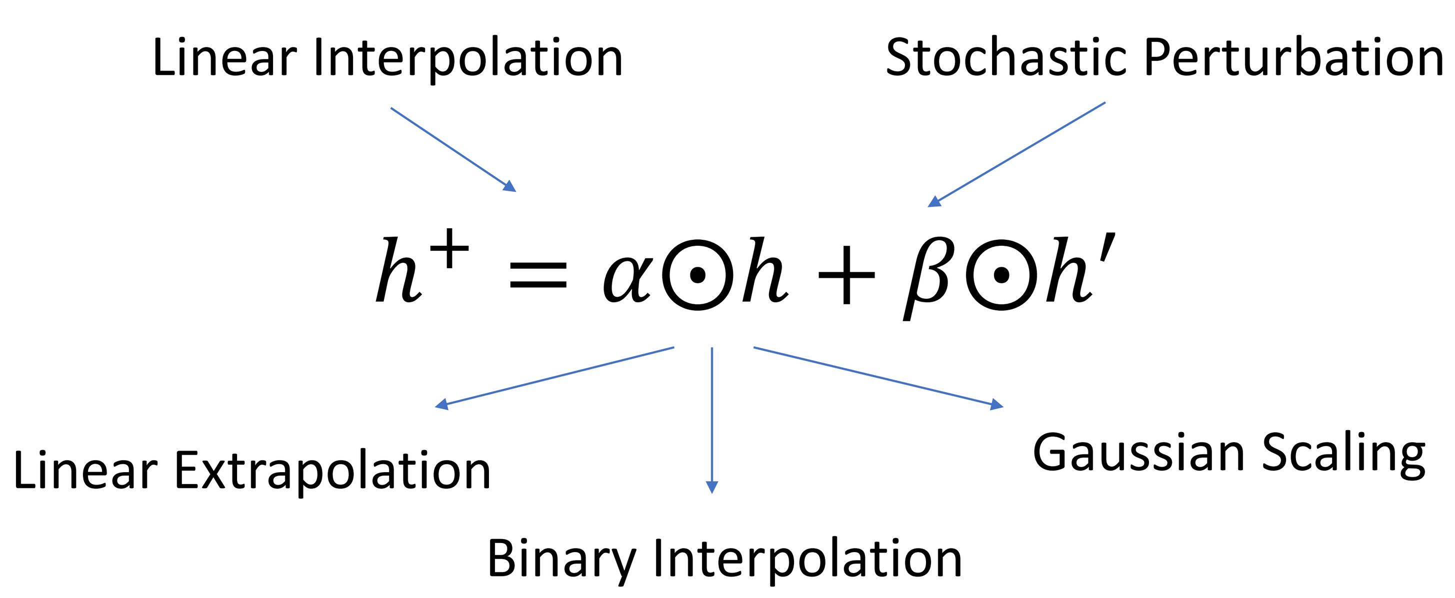
where $q,q^- \in Q, c,c^- \in C, (q,c^-)$, (q^-,c) and (q^-,c^-) are all negative pairs, α and β are coefficients, B is the size of sets, and N is the augmentation time.

Motivation



Existing augmentation methods are applied to raw data, which is resource-consuming and limited. For representation-level augmentation, we do not need to encode raw data again.

Methodology



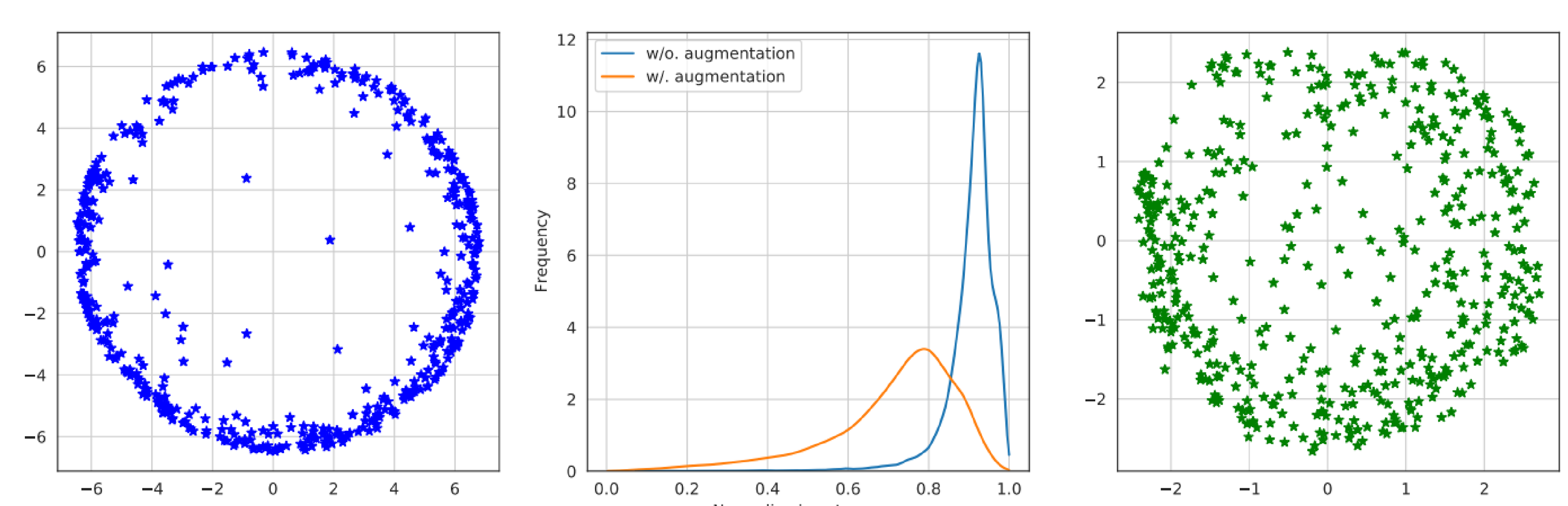
We unify the existing two approaches, Linear Interpolation and Stochastic Perturbation, to a general format. Based on it, we propose three novel augmentation approaches, Linear Extrapolation, Binary Interpolation, and Gaussian Scaling.

- Code: https://github.com/Alex-HaochenLi/RACS
- Contact: haochen003@ntu.edu.sg

Experiments

Model	Ruby		JavaScript		Go		Python		Java		PHP	
	Original	w/ RA	Original	w/ RA	Original	w/ RA	Original	w/ RA	Original	w/ RA	Original	w/ RA
RoBERTa (code)	0.641	**0.665**	0.583	**0.612**	0.867	**0.892**	0.610	**0.663**	0.634	**0.674**	0.584	**0.617**
CodeBERT	0.648	**0.664**	0.594	**0.608**	0.878	**0.890**	0.636	**0.654**	0.663	**0.674**	0.615	**0.619**
GraphCodeBERT	0.705	**0.721**	0.647	**0.671**	0.896	**0.903**	0.690	**0.708**	0.691	**0.708**	0.648	**0.656**

Results on CodeSearchNet Dataset. Performance of different models under MRR. ``w/ RA'' stands for ``with representation-level augmentation''.



Visualization of code vector distribution with and without representation-level augmentation. **Left**: without augmentation. **Right**: with augmentation. **Middle**: distribution of vector norms. With augmentation, representations better take advantage of norms.

Augmentations	RoBERTa	CodeBERT	GraphCodeBERT
no augmentations	0.629	0.636	0.690
linear interpolation	0.644	0.648	0.702
linear extrapolation	0.640	0.646	0.704
stochastic perturbation	0.658	0.648	0.698
binary interpolation	0.655	**0.655**	0.705
Gaussian scaling	0.657	0.649	0.696
all augmentations	**0.663**	0.654	**0.708**

Five augmentation methods have similar effects on MRR since they come from the same general format.

Model	FiQA-2018		NFCorpus	
	Original	w/ RA	Original	w/ RA
DistilBERT	0.352	**0.400**	0.481	**0.505**
RoBERTa	0.343	**0.356**	0.367	**0.389**

We also evaluate the proposed methods on two passage retrieval datasets.