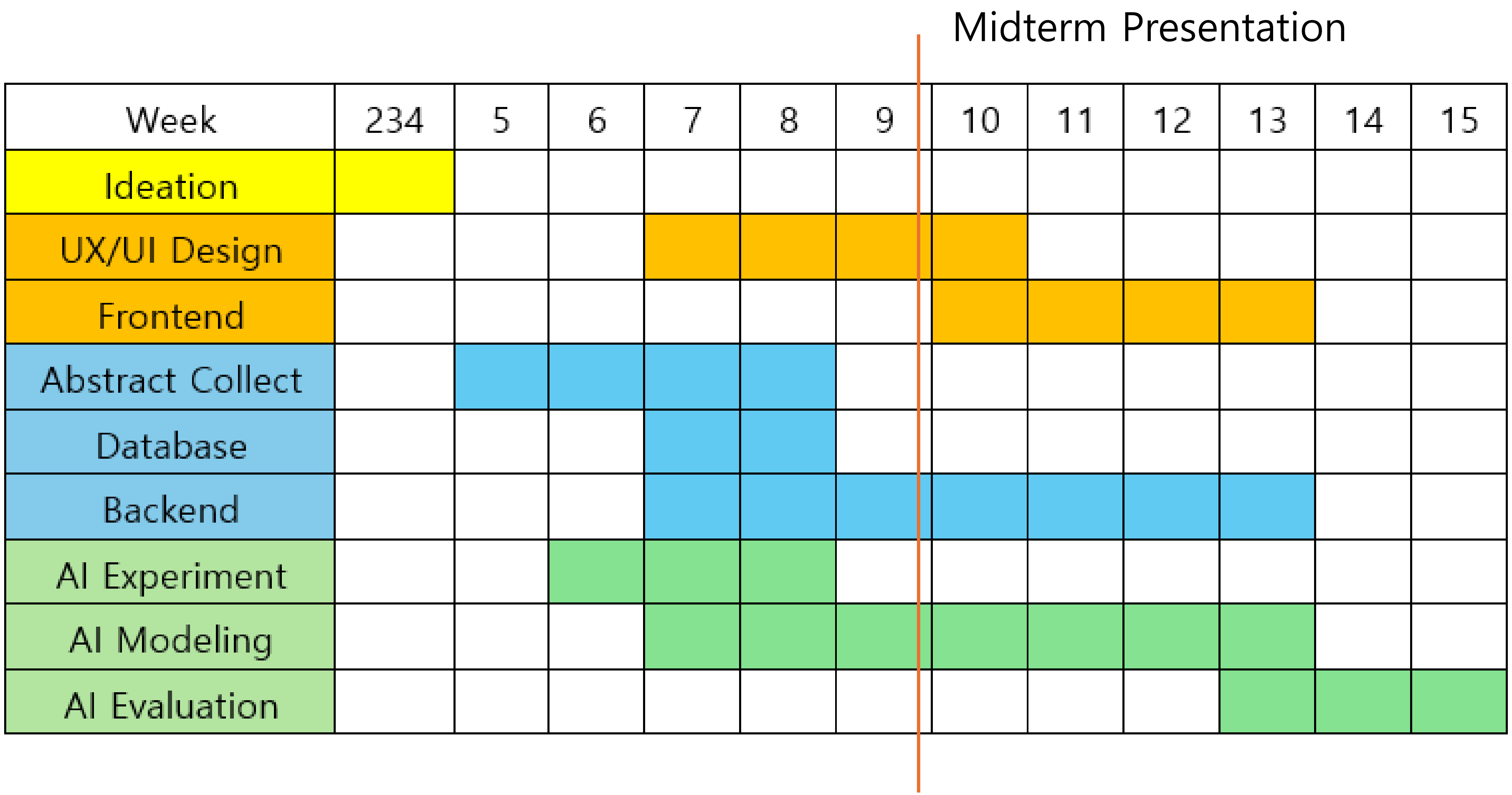
SKKU Lab Recommendation Service : FindMyLab
Midterm Presentation

Team J (Last Lap) 김현진 송민석 장민석 조재희

2024-Fall Capstone Design Project

Objectives & Milestones



Role of each member

MinSeok Jang | HyunJin Kim

UX/UI
Front-End

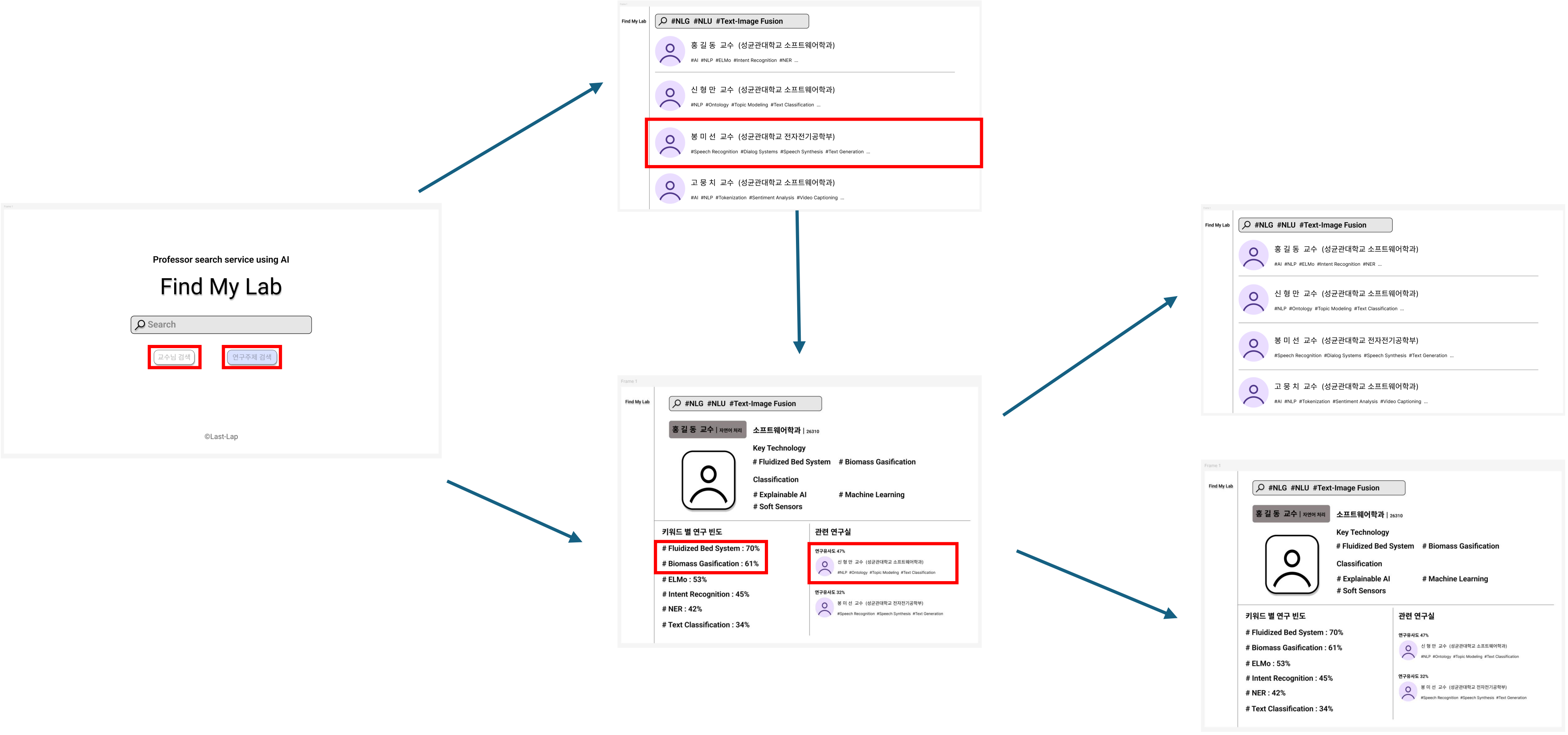
MinSeok Song

Abstract Extract
Back-End

JaeHee Jo

AI Modeling

UX/UI

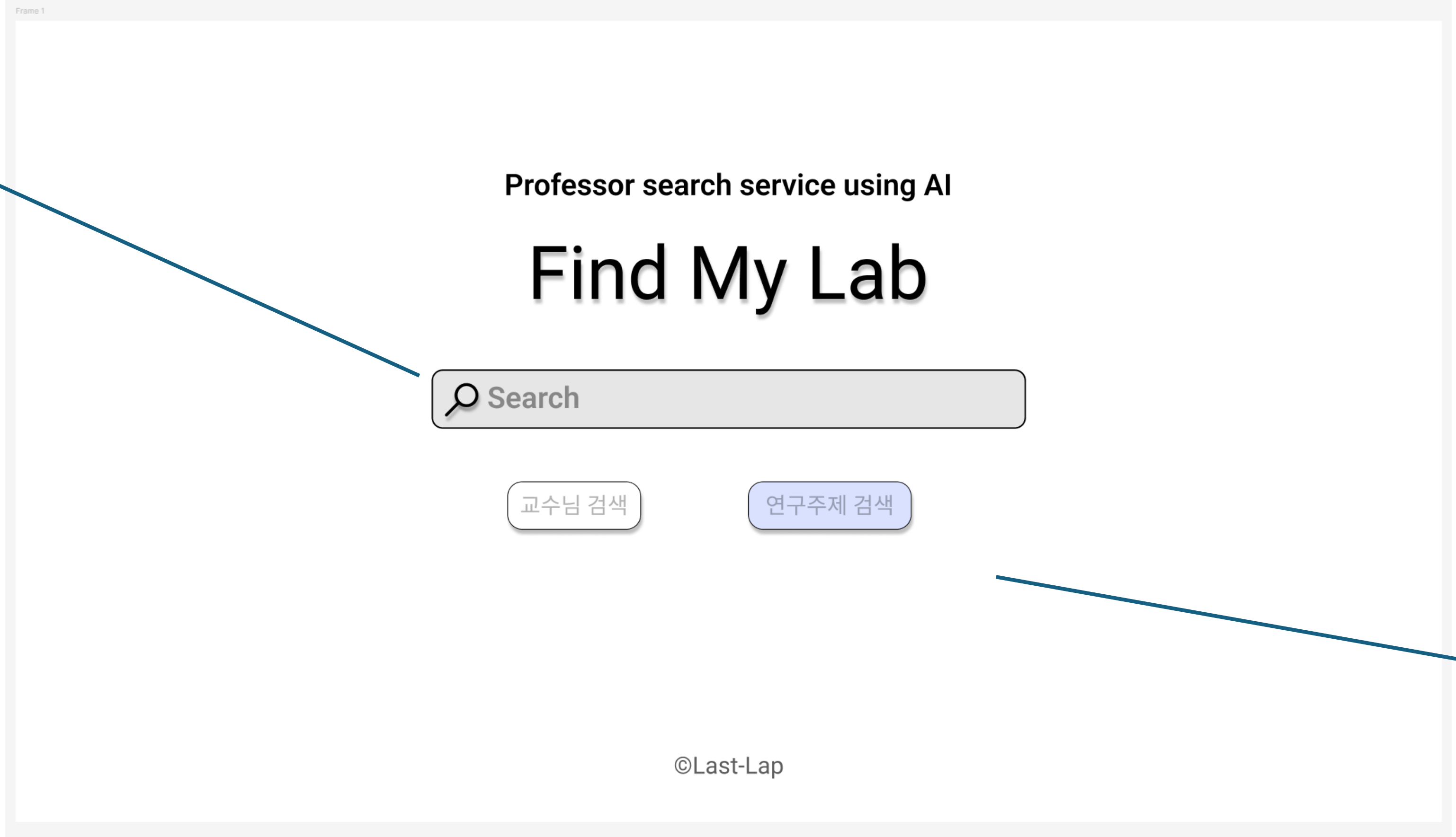


2024-Fall Capstone Design Project

Team J (Last Lap) 김현진 송민석 장민석 조재희

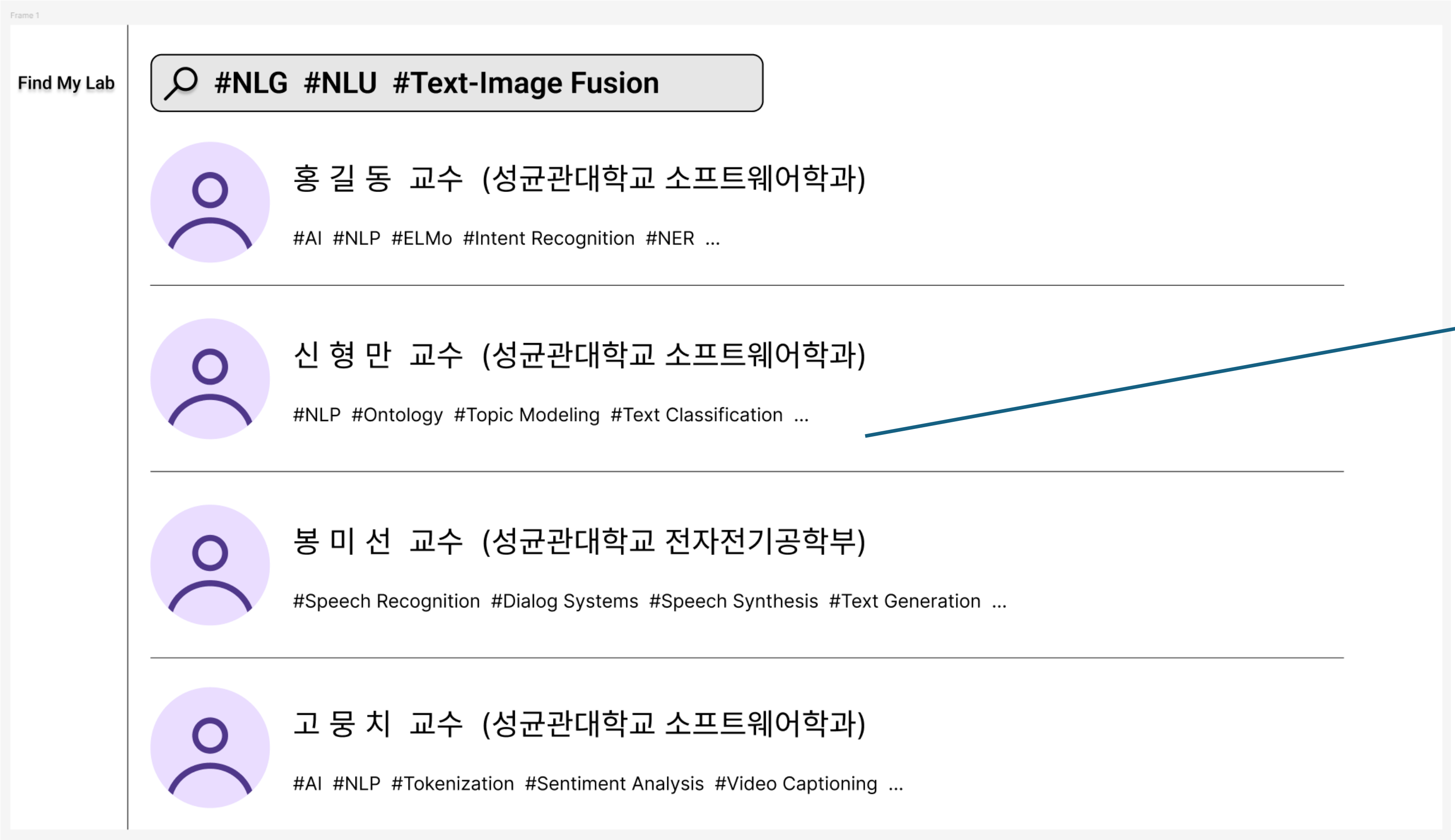
UX/UI - Search Page

Search Bar



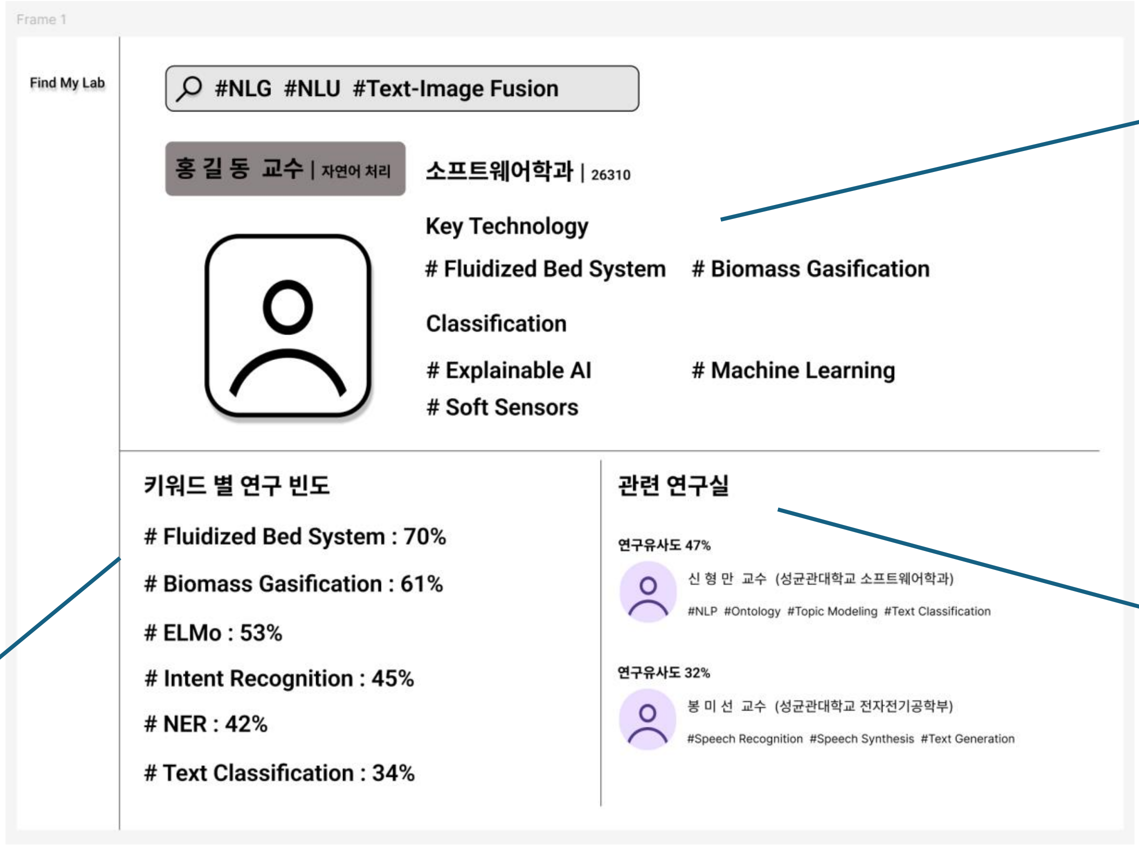
Change
Search Mode

UX/UI - Search Result Page



List of Recommended Lab
- Professor
- Department
- Main Research Keywords

UX/UI - Lab Introduction Page



Laboratory Key Information

Study frequency by keyword

Similar laboratory recommendations

Backend - Abstract Extractor

ORCID
Connecting Research
and Researchers


```
StatusCode        : 200
StatusDescription : OK
Content           : {"status":"ok","message-type":"work-list","message-version":"1.0.0","message":{"facets":{},"total-r
                    esults":19,"items":[{"indexed":{"date-parts":[[2024,7,3]],"date-time":"2024-07-03T23:32:09Z","times
                    ta...
RawContent        : HTTP/1.1 200 OK
                    transfer-encoding: chunked
                    access-control-expose-headers: Link
                    access-control-allow-headers: X-Requested-With, Accept, Accept-Encoding, Accept-Charset, Accept-Lan
                    guage, Accept-Range...
Forms             : {}
Headers           : {[transfer-encoding, chunked], [access-control-expose-headers, Link], [access-control-allow-headers
                    , X-Requested-With, Accept, Accept-Encoding, Accept-Charset, Accept-Language, Accept-Ranges, Cache-
                    Control], [access-control-allow-origin, *]...}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 197746
```


[Crossref API]

학술지 논문

- (2022) Pivotal B plus tree for Byte-Addressable Persistent Memory. IEEE ACCESS. 10, -
- (2021) Failure-Atomic Byte-Addressable R-tree for Persistent Memory. IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS. 32, 3
- (2021) Failure-Atomic Byte-Addressable R-tree for Persistent Memory. IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS. 32, 3
- (2020) B-3-Tree: Byte-Addressable Binary B-Tree for Persistent Memory. ACM TRANSACTIONS ON STORAGE. 16, 3

Google 학술검색 (2022) Pivotal B plus tree for Byte-Addressable Persistent Memory. IEEE AC

학술자료 검색결과 약 126개 (0.10초)

모든 날짜
2024 년부터
2023 년부터
2020 년부터
기간 설정...

Pivotal B+ tree for Byte-Addressable Persistent Memory
J Yoo, H Cha, W Kim, WH Kim, SS Park, B Nam - IEEE Access, 2022 - ieeexplore.ieee.org
... for **byte-addressable persistent memory**. In this work, we design and implement PB+**tree** (**Pivotal B**+**tree**) ... In this work, we present **Pivotal B**+**tree** (PB+**Tree**) that simplifies **memory access** ...
☆ 저장 ❞ 인용 1회 인용 관련 학술자료 전체 4개의 버전

Backend - Abstract Extractor

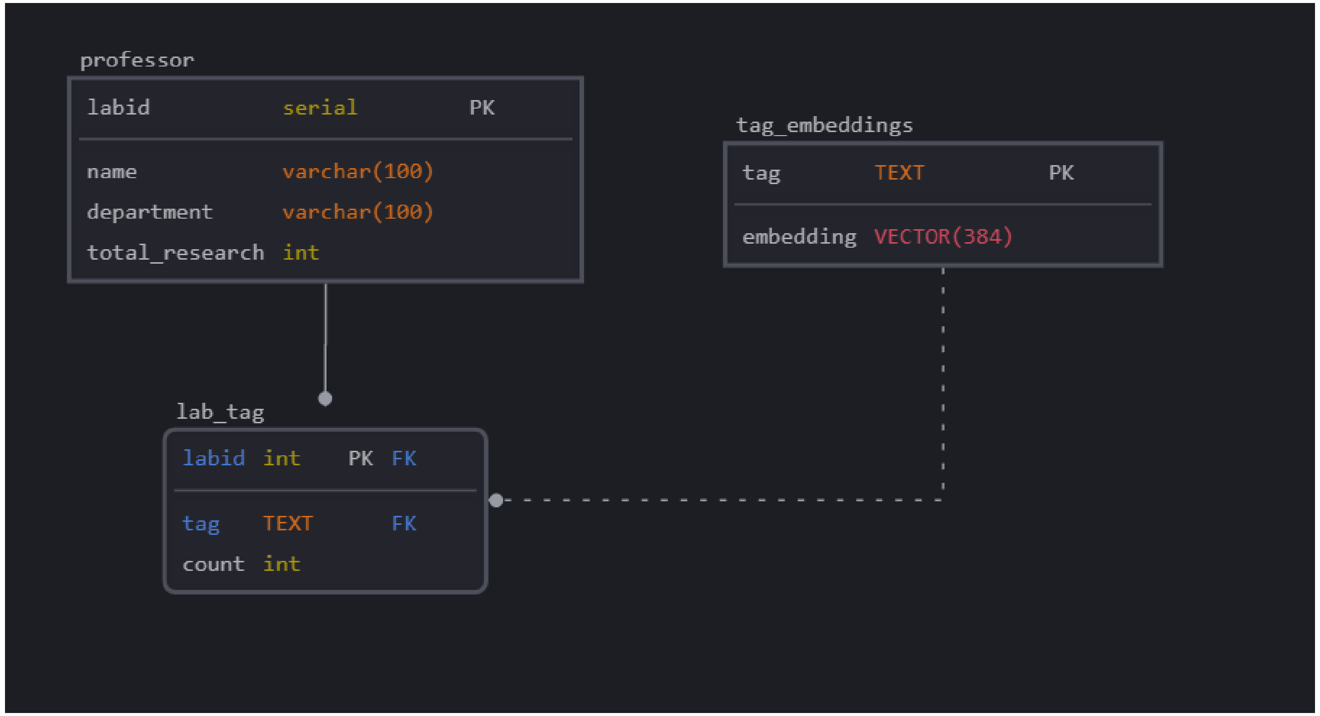
SPRINGER NATURE
Data Solutions

Developer Documentation   Our API's

BeautifulSoup

Se

Backend - Database

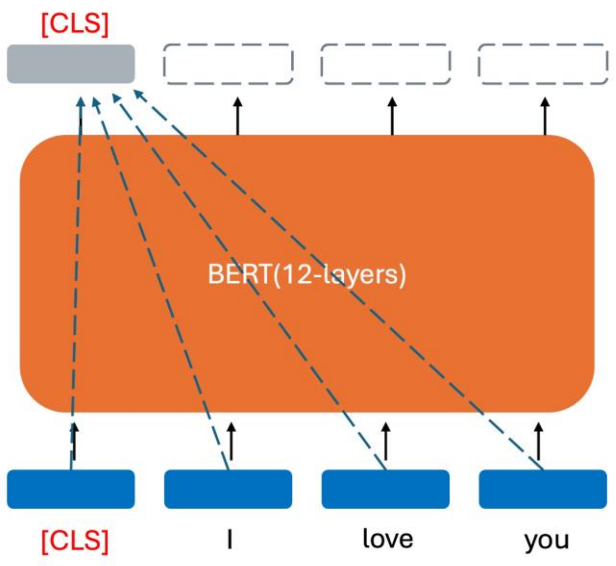


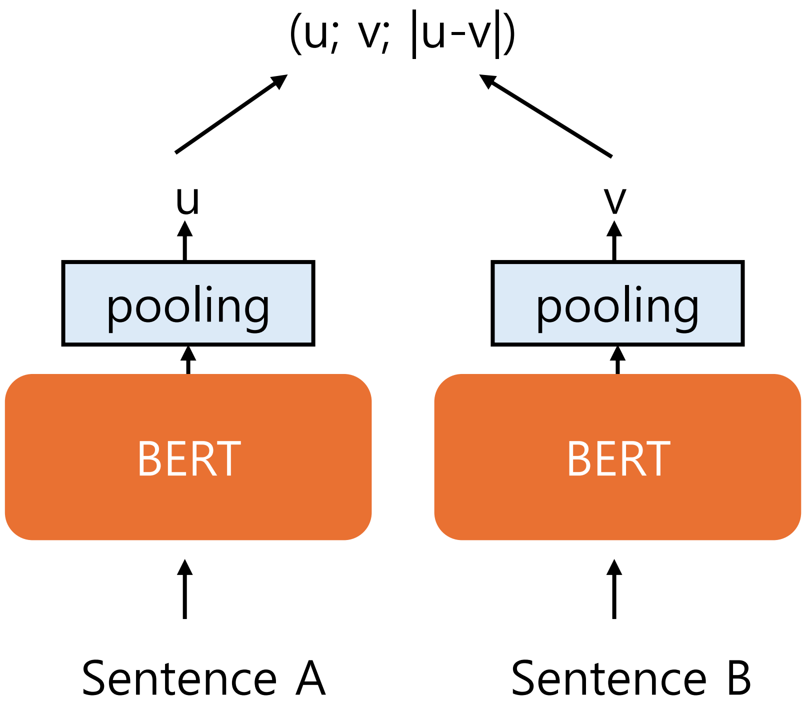
Backend - Restful server

AI - SBERT

Wiki/Library corpuses

Hugging Face : SBERT-bas-nli-v2

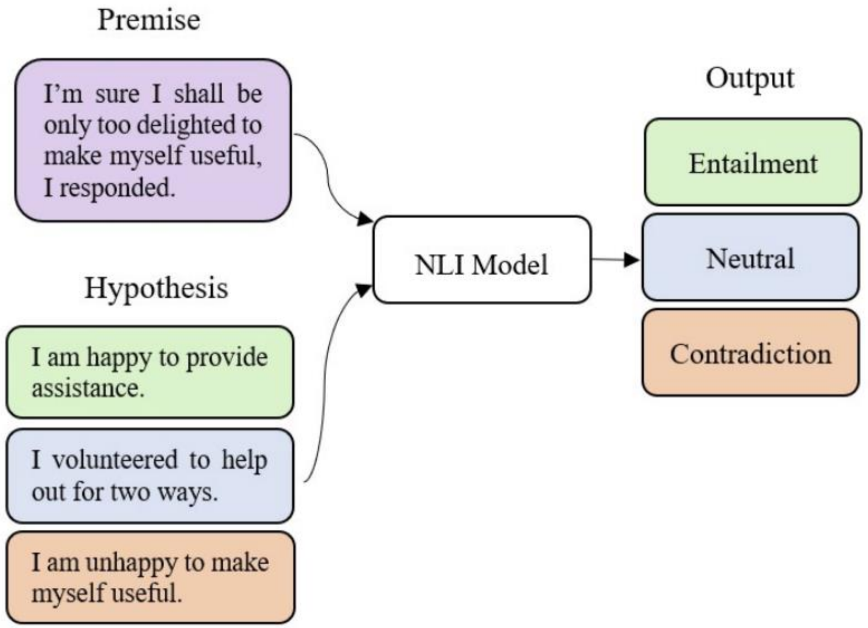




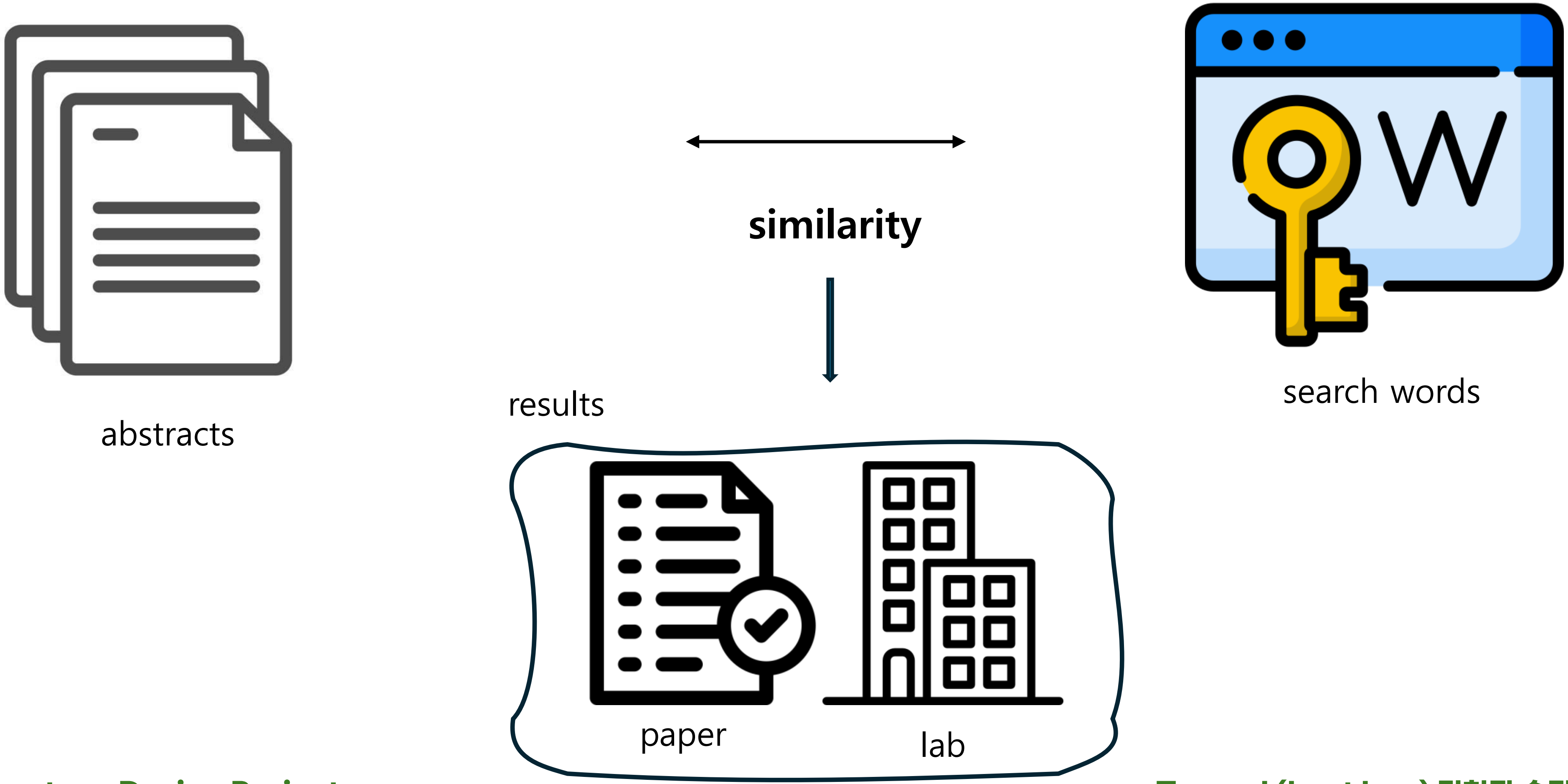
AI - NLI Task

NLI (**N**atural **L**anguage **I**nference)

Goal: understanding the relationships between sentences



AI - Goal



AI - Experiment

Identifying compiler toolchain provenance serves as a basis for both benign and malicious binary analyses. A wealth of prior studies mostly focuses on the inference of a popular compiler toolchain for C and C++ languages from stripped binaries that are built with GCC or clang. Lately, the popularity of an emerging compiler is on the rise such as Rust, Go, and Nim programming languages that complement the downsides of C and C++ (e.g., security), which little has been explored on

⋮

target abstract

Compiler toolchain provenance

BERT-based system

Toolchain classification

Binary code similarity detection

Machine learning

positive

Dialogue-specific pre-training task

Encoder-decoder transformer

AI dialogue systems

Language understanding and generation

semi negative

(same domain but different)

AI - Experiment

positive

```
positive similarity (Compiler toolchain provenance): 0.7811970114707947
positive similarity (BERT-based system): 0.41414642333984375
positive similarity (Toolchain classification): 0.5638635754585266
positive similarity (Binary code similarity detection): 0.57719486951828
positive similarity (Machine learning): 0.5563946962356567
positive similarity (Fine-tuning process): 0.43460023403167725
positive similarity (Binary analysis): 0.5318818092346191
positive similarity (Signature-based tool): 0.48783546686172485
positive similarity (Emerging compilation toolchains (Rust Go Nim)): 0.6662441492080688
positive similarity (C and C++ compilers (GCC clang)): 0.7179384231567383
```

0.5 ↑ (almost)

negative

```
negative similarity (Dialogue-specific pre-training task): 0.3611958622932434
negative similarity (Encoder-decoder transformer): 0.43461769819259644
negative similarity (AI dialogue systems): 0.4394277334213257
negative similarity (Language understanding and generation): 0.4609677195549011
negative similarity (Task-oriented dialogue systems): 0.41865217685699463
negative similarity (SIMMC2.0 challenge): 0.33725640177726746
negative similarity (Auxiliary task for pre-training): 0.32983070611953735
negative similarity (Model performance improvement): 0.43528273701667786
```

0.5 ↓

AI - Experimer

Identifying compiler toolchain provenance serves as a basis for both benign and malicious binary analyses. A wealth of prior studies mostly focuses on the inference of a popular compiler toolchain for C and C++ languages from stripped binaries that are built with GCC or clang. Lately, the popularity of an emerging compiler is on the rise such as Rust, Go, and Nim programming languages that complement the downsides of C and C++ (e.g., security), which little has been explored on

positive

Compiler toolchain provenance

BERT-based system

Toolchain classification

Binary code similarity detection

Machine learning

target words

The next generation of artificial intelligence (AI) is required to be capable of proper communication to enable eloquent interaction with human beings. Thus, AI models require powerful language understanding and generation capabilities. Therefore, designing an adequate dialogue-specific task is an important topic in dialogue research. In this article, we improve our model that won in two out of four subtasks in SIMMC2.0 challenge using a dialogue-specific pre-training task, which learns

semi - negative

Over the past few years, various indexes have been redesigned for byte-addressable persistent memory. In this work, we design and implement PB+tree (Pivotal B+tree) that resolves the limitations of state-of-the-art fully persistent B+trees. First, PB+tree reduces the number of expensive shift operations by up to half by managing two sub-arrays separated by a pivot key. Second, PB+tree reads cachelines in ascending order, which makes PB+tree benefit from hardware

negative

AI - Experiment

clear
domain specific

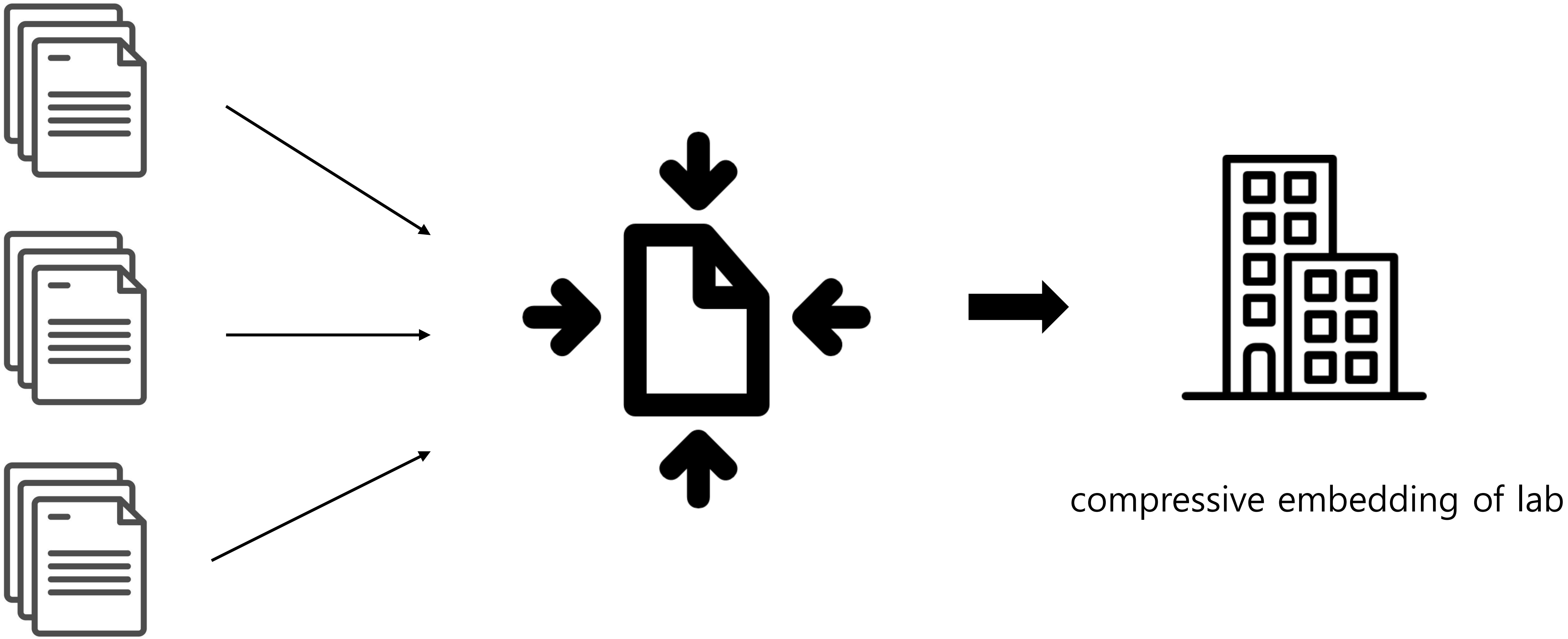
```
key word (Compiler toolchain provenance)
Positive similarity: 0.7811970114707947
semi-Negative similarity: 0.4616994857788086
Negative similarity: 0.4929749369621277
```

```
key word (Binary code similarity detection)
Positive similarity: 0.57719486951828
semi-Negative similarity: 0.4377126693725586
Negative similarity: 0.4894843101501465
```

ambiguous
generic words

```
key word (Machine learning)
Positive similarity: 0.5563946962356567
semi-Negative similarity: 0.5957521200180054
Negative similarity: 0.4678981602191925
```

AI - To do



Q & A