Code Similarity Estimation

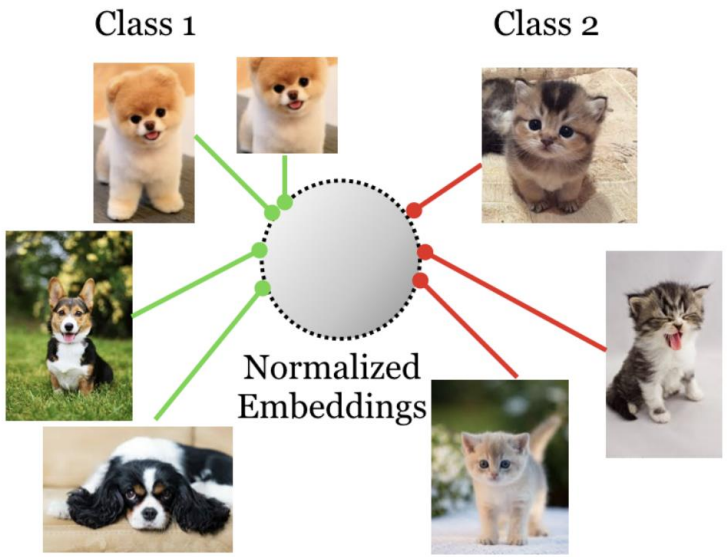
Pattern Recognition & Machine Learning Laboratory
Ha-Na Jo
Oct. 04, 2022

Project Description & Dataset

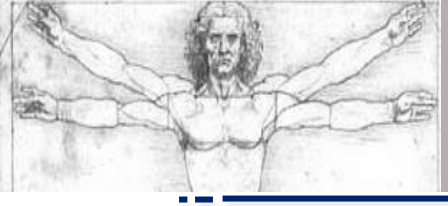
- **Goal**
  - Estimating code similarity
- **Necessity**
  - Lack of software engineers compared to an increase in demand
  - It is essential to have an automated way of analyzing, developing, and maintaining
- **Dataset**
  - code (folder)
    - 300 problems for training
      - Each 150 solution codes for 1 problem
  - sample_train.csv (file)
    - 17,970 pairs of extracted in solution codes of provided 300 problems
      - code 1: python code 1
      - code 2: python code 2
      - similar: 0 for different problems and 1 for the same problem
  - sample_submission.csv (file)
    - Sample form for submission
  - test.csv (file)
    - 179,700 pairs of extracted in solution codes of other 300 problems

Preprocessing

- **Delete the annotation line**
  - Skip the line that starts '#'
    - ex) # this is an annotation line
      → None
- **Skip saving the after annotation**
  - Stop saving when '#' found
    - ex) print('hello') # print 'hello'
      → print('hello')
- **Delete the newline letter and blank**
  - Delete '\n' and ' '
    - ex) print('hello')
      print('hi')
      → print('hello') print('hi')
- **Create dataset**
  - For training
    - 100 positive pairs and 100 negative pairs for 1 problem (total: 60,000)
  - For validation
    - 100 positive pairs and 100 negative pairs for 1 problem (total: 60,000)

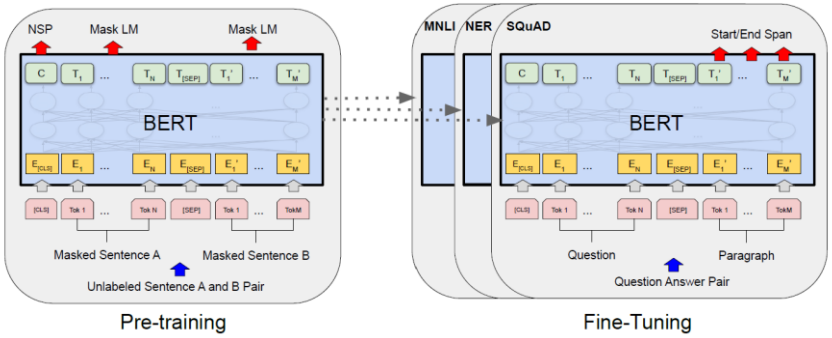


Examples of positive(green) and negative set(red)

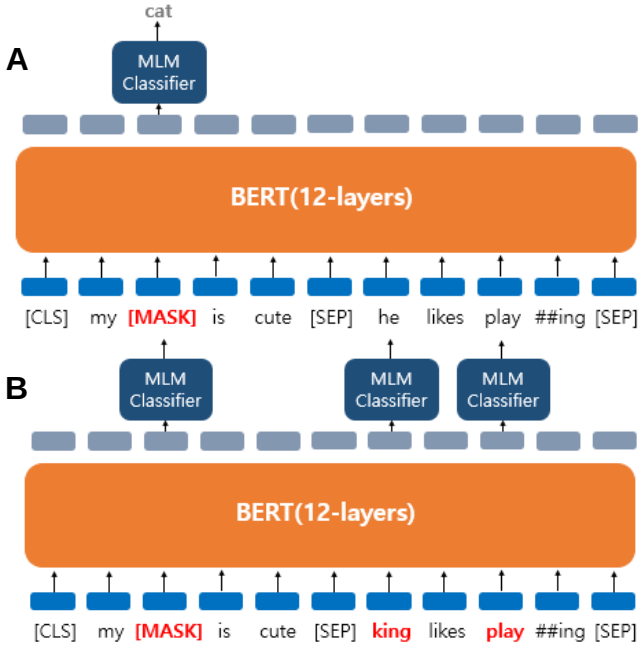


Baseline Model

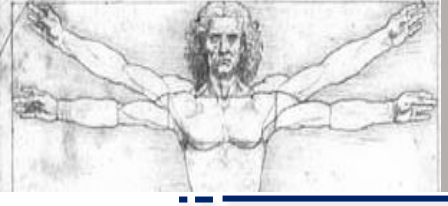
- **BERT**
  - **Goal**
    - Natural language processing
  - **Architecture**
    - Transformer encoder 12 or 24 layers
      - Word embedding with context
    - Pre-training
      - Using massive data without labels
      - Masked language model (MLM)
        - » Text random masking
      - Next sentence prediction (NSP)
        - » Train with randomly concatenated sentences
    - Fine-tuning
      - Using additional tasks with labels
      - Single text classification
        - » ex) emotion classification
      - Tagging
      - Question answering



Overall pre-training and fine-tuning architecture

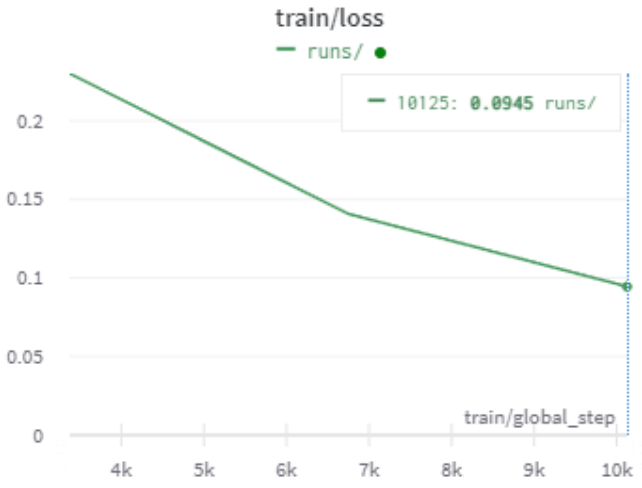


Example of pre-training (A) MLM (B) NSP

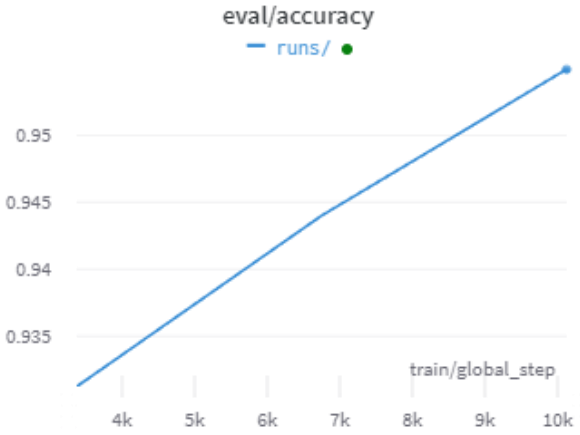


Baseline Model (Cont.)

- **Implementation environment**
  - **Hardware**
    - 64GB of RAM
    - GPU with 16GB of RAM
  - **Software**
    - Anaconda environment (local)
    - Python: 3.9.7
- **Result**
  - **Learning time: 50m 57s**
  - **Validation accuracy: 95.49%**
  - **Test (no label) accuracy: 89.23%**



Loss result of training



Accuracy result of validation

Additional Information

- **Git hub**
  - https://github.com/HanaJo-ku/NNAP
    - code.zip
      - Solution codes folders
    - run.ipynb
      - Preprocessing, training, and test run code
    - sample_submission.csv
    - sample_train.csv
    - test.csv.vol1~2.egg
      - Test set (no labels) file
- **E-mail**
  - hn_jo@korea.ac.kr