1. 진행한 내용
2. 코드 위치
3. 전이학습 사용

진행사항

- DeepSoft-C

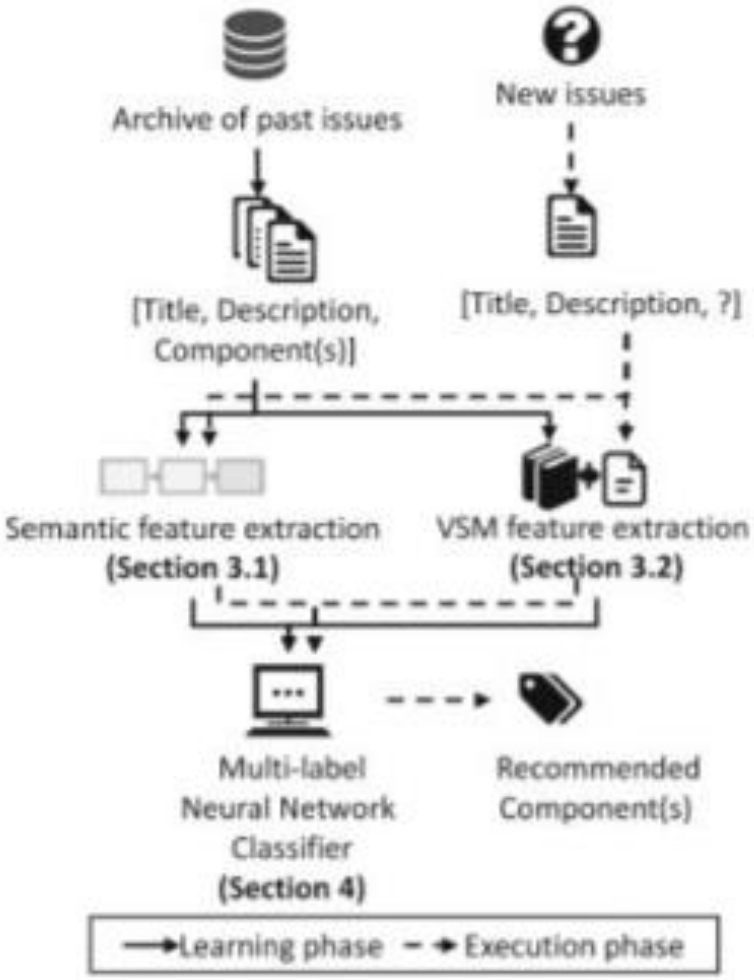
  Automatically recommending components for issue reports using deep learning

  워드 임베딩:

  LSTM (semanric feature) + VSM feature

  분류 모델:

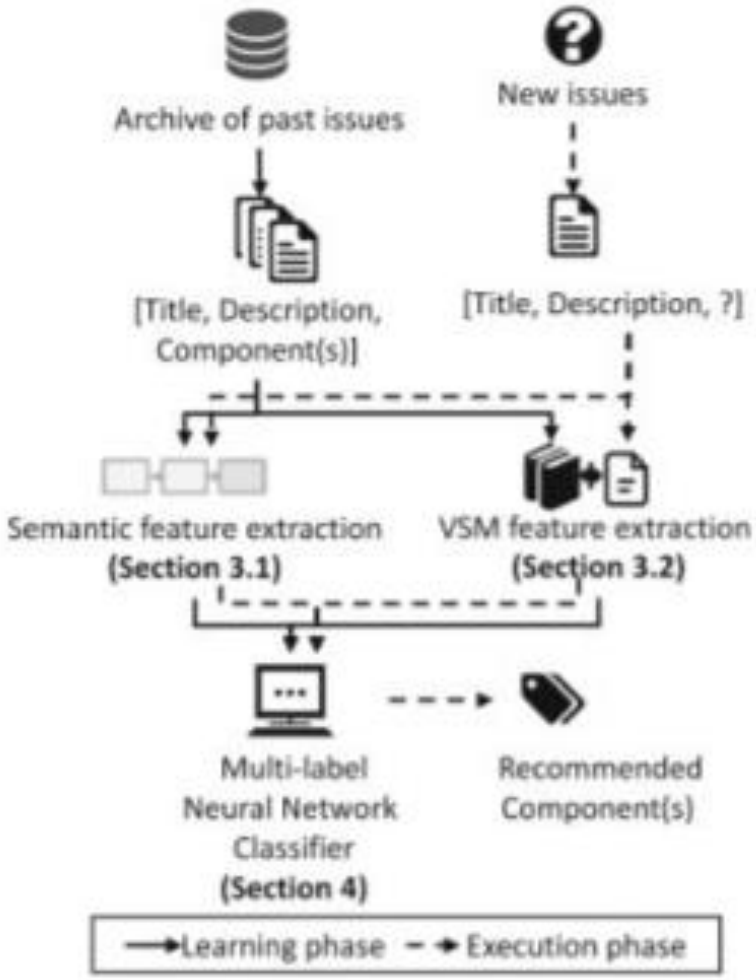
  CNN



진행사항

- DeepSoft-C와 차이점
- 임베딩
  -> BERT 임베딩

임베딩 과정 이해하기 위해 FastText로 연습
(데이터가 적을 때 좋다길래)



진행사항

```
In [26]: title_vocab

Out[26]: {'fcrepo': 1,
          'fedora': 2,
          'api': 3,
          'camel': 4,
          'rest': 5,
          'use': 6,
          'update': 7,
          'support': 8,
          'add': 9,
          'create': 10,
          'error': 11,
          'xml': 12,
          'remove': 13,
          'client': 14,
          'datastream': 15,
          'fesl': 16,
          'content': 17,
          'objects': 18,
          'fails': 19,
```

```
title_embedding={}
for word in title_vocab:
    title_embedding[word] = model_title.wv[word].tolist()
print(title_embedding)
```

```
{'fcrepo': [-0.4575629340244293, 0.03214785084128938, -0.0254
9815, 0.19127802550792964, 0.07796320556402206, 1.3487921953
7243704796807, -0.11547456915802542, -0.5441123843193054, -1
0.03844872489671514, -0.0950339275598526, 1.18775236606897
081058946, -0.48824238777150645, -0.2634963568444977, 0.3845
6, -1.737955503845214], 'api': [-0.3987962305545807, 0.0584
6638, -0.0582193112871 3752, 0.19704480469226837, 0.0366734 82
2862061022062798, 0.4379383062152323, -0.1305004209280014, -C
```

```
np.array(train_title_images)

array([[[-4.08269987e-02, -1.72736347e-02, -5.28760906e-03, ...
         -2.62541454e-02, -2.47510187e-02, -8.57250890e-02],
        [-2.03262850e-01,  4.73857018e-03, -1.43443113e-02, ...
         -5.67936487e-02, -2.37839535e-01, -6.28923858e-01],
        [-5.49637906e-01,  2.55172681e-02, -3.78330126e-02, ...
         -1.38435807e-01, -6.30841740e-01, -1.54610270e+00],
        ....
```

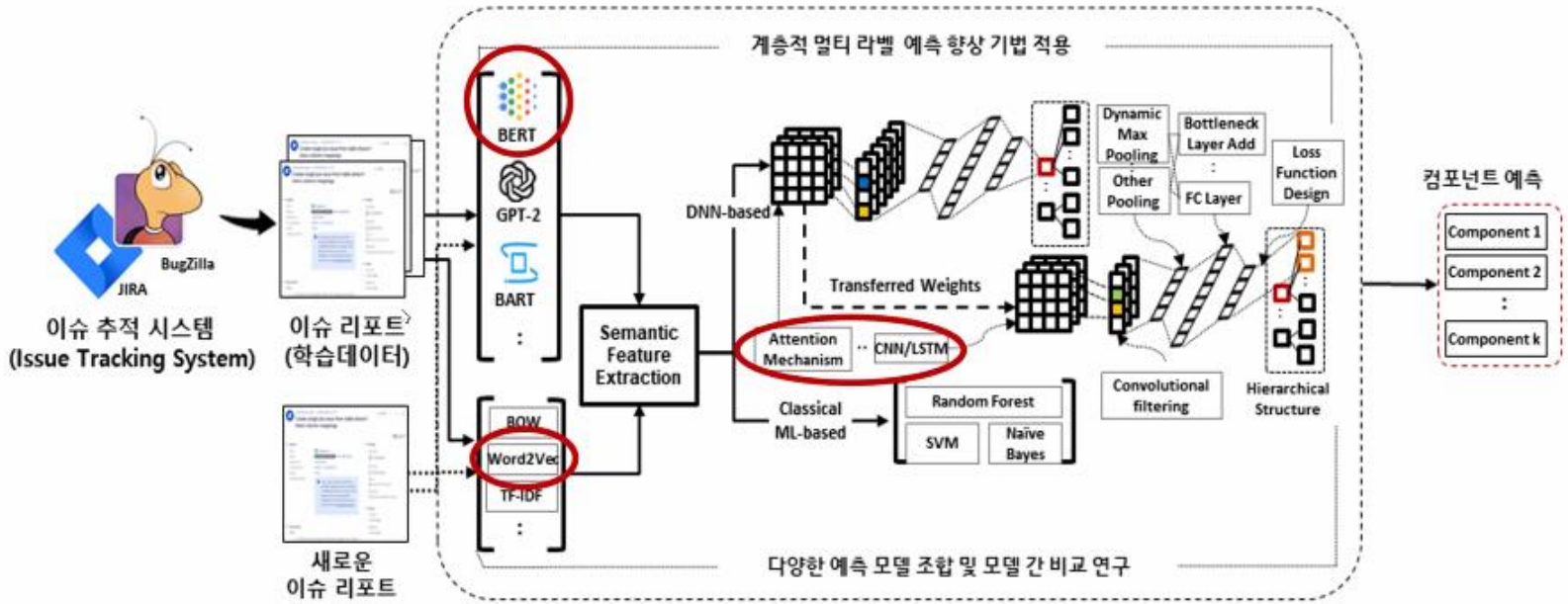
```
train_title_inputs

array([[   9.,   20.,  563., ...,    0.,    0.,    0.],
       [   9.,   39.,  330., ...,    0.,    0.,    0.],
       [   9.,   33.,   26., ...,    0.,    0.,    0.],
       ...,
       [   5.,    3.,    8., ...,    0.,    0.,    0.],
       [2247.,  178., 2248., ...,    0.,    0.,    0.],
       [ 914., 2249.,   64., ...,    0.,    0.,    0.]], dtype=int32)
```

```
train_title_images[0]

[[-0.04082699865102768,
  -0.01727363467216491 7,
  -0.005287609063088894,
  -0.01359899435192346 6,
  0.05898496136069298,
  -0.02006738260388374 3,
  0.02364125475287437 4,
  0.00024750936427153647,
  0.1140063181515 9084,
  -0.04139322042465 21,
  -0.00786075089126825 3,
  0.03035734035074711,
  0.04376852139830589,
  -0.02625414542853832 2,
  -0.02475101873278618,
  -0.08572506904602051],
```

진행사항

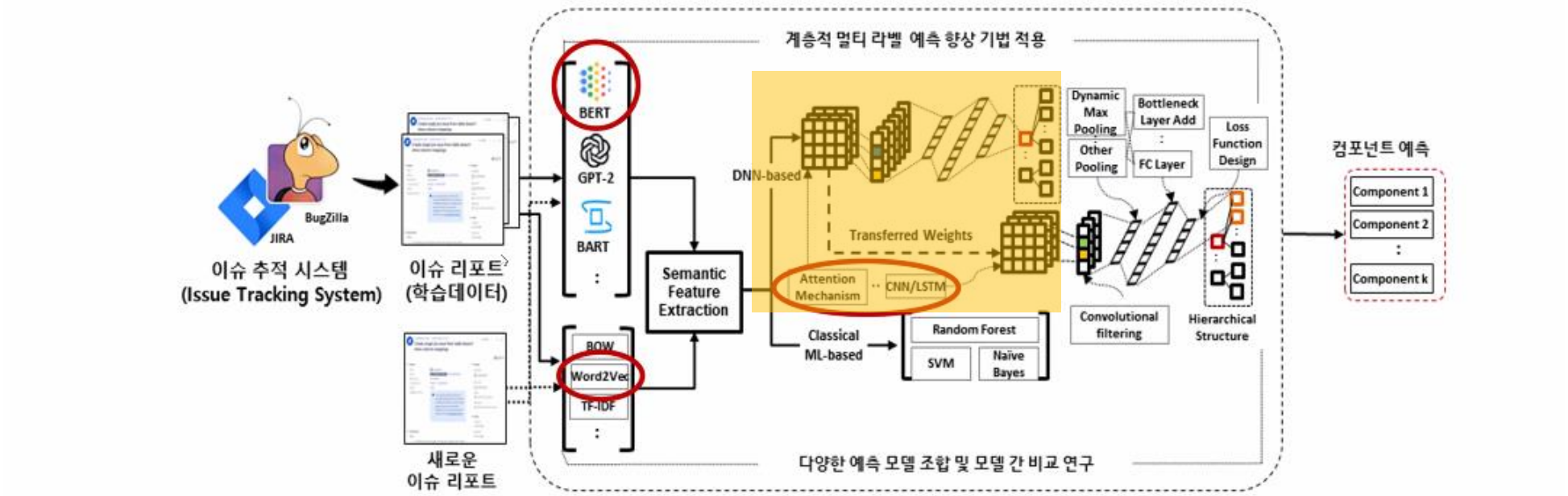
Process

- 현상황



역할 1 CNN model

1) Parent label을 분류하는 CNN 모델 구현



역할 1 CNN model

<Input> Semantic feature

|

CNN 모델

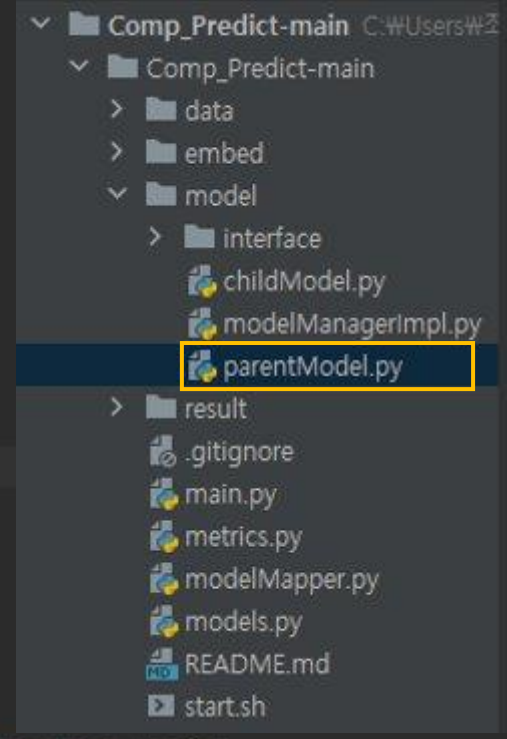
|

<Output> Parent Model weight 저장

|

Child Model 에 전달

```python
from tensorflow import keras
from model.interface.modelCollection import ModelCollection


class ParentModel(ModelCollection): # parent type
    def __init__(self, n_token, vec_dim, output_dim):
        super().__init__(n_token, vec_dim, output_dim)

    def mapping(self, model_name):
        if model_name == 'cnn':
            return self.cnn_model()
        elif model_name == 'lstm':
            return self.lstm_model()
        elif model_name == 'dense':
            return self.dense_model()

    def cnn_model(self):
        model = keras.models.Sequential([
            keras.layers.Conv2D(filters=12, kernel_size=(5, 5), activation='relu',
                                input_shape=(self.n_token, self.vec_dim, 1), name='conv_1'),
            keras.layers.MaxPool2D(pool_size=(1, 2)),
            keras.layers.Conv2D(filters=24, kernel_size=(5, 5), activation='relu', name='conv_2'),
            keras.layers.MaxPool2D(pool_size=(1, 2)),
            keras.layers.Flatten(),
            keras.layers.Dense(50, activation='softmax'),
            keras.layers.Dropout(0.5),
            keras.layers.Dense(self.output_dim, activation=None)])

        return model

    def lstm_model(self):
        pass

    def dense_model(self):
        pass
```



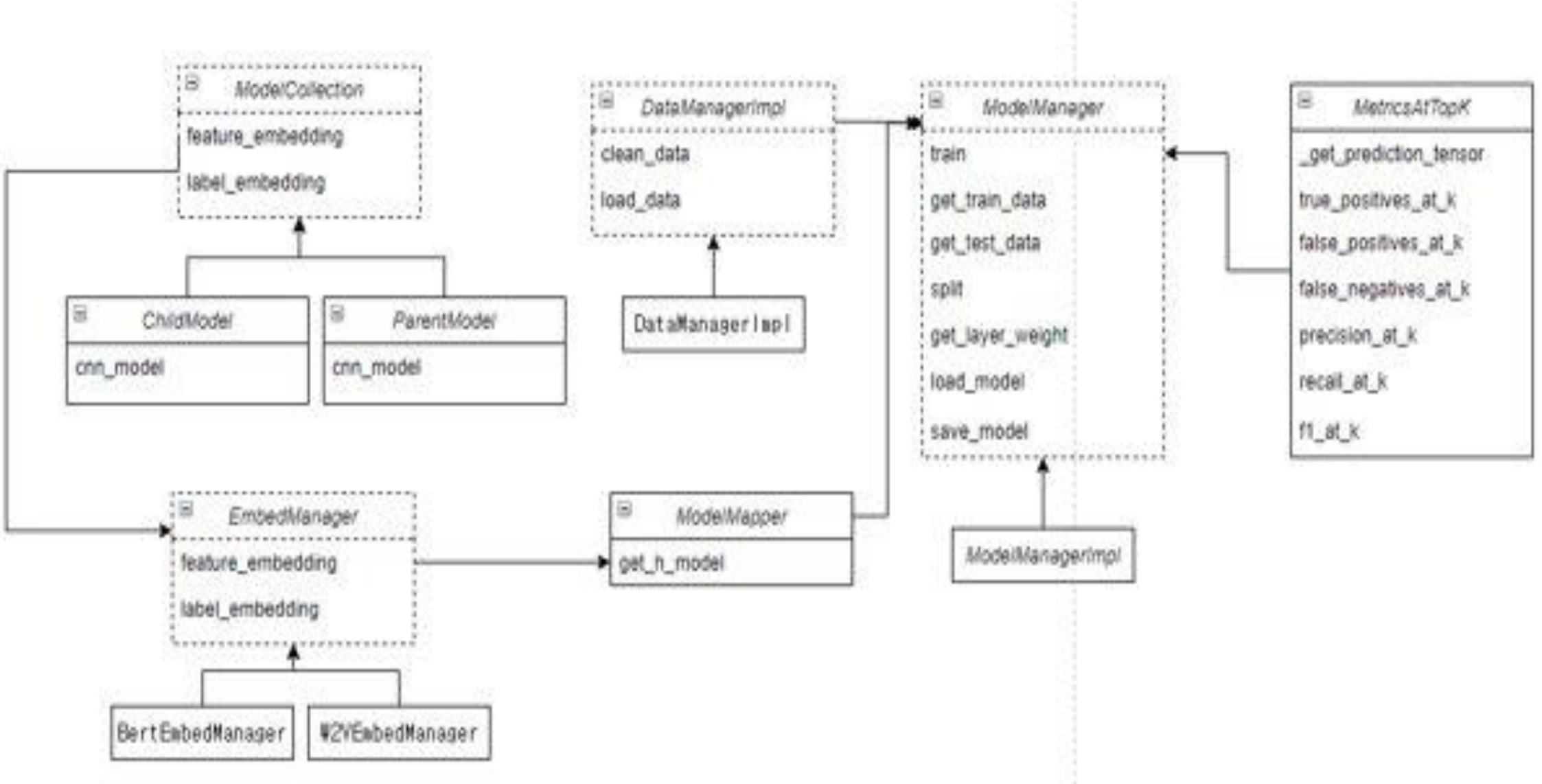
역할 1 CNN model

임베딩 결과를
받는 Parent Model

```
Comp_Predict-main C:₩Users₩3
  Comp_Predict-main
    data
    embed
    model
      interface
      childModel.py
      modelManagerImpl.py
      parentModel.py
    result
    .gitignore
    main.py
    metrics.py
    modelMapper.py
    models.py
    README.md
    start.sh
```

```python
from model.childModel import ChildModel
from model.parentModel import ParentModel
from embed.interface.embedManager import EmbedManager
from data.interface.dataManager import DataManager


class ModelMapper:
    def __init__(self, embed_manager: EmbedManager):
        self.e_manager = embed_manager

    def get_h_model(self):
        """
        :param model_name: such as 'cnn', 'lstm'
        :return: model: Model
        """
        embed_dim = self.e_manager.max_n_title_token + self.e_manager.max_n_token

        if self.e_manager.h_level == 'parent':
            p_model = ParentModel(embed_dim, self.e_manager.embed_dim, self.e_manager.output_dim)
            return p_model
        elif self.e_manager.h_level == 'child':
            c_model = ChildModel(embed_dim, self.e_manager.embed_dim, self.e_manager.output_dim)
            return c_model
        else:
            print("[ERROR] there is no existed model.")
```

Implemented Code



Code review- 9/16

Project	Method	Recall@k					
		k = 5	%imp	k = 10	%imp	k = 15	%imp
Average	DeepSoft-C	0.508		0.631		0.694	

1) Bert Padding error

해결

-bert parent

-bert child 1 & 2

```
Epoch 50/50
30/30 [==============================] - 3s 101ms/step - loss: 0.2438 - acc: 0.5
799 - recall_at_k: 0.5800 - val_loss: 0.2432 - val_acc: 0.5831 - val_recall_at_k
: 0.5832
10/10 - 0s - loss: 0.2446 - acc: 0.5737 - recall_at_k: 0.5738
```

```
Epoch 50/50
1/1 [==============================] - 0s 22ms/step - loss: 0.2258 - acc: 0.2917
 - recall_at_k: 0.1944 - val_loss: 0.2156 - val_acc: 1.0000 - val_recall_at_k: 0
.3333
1/1 - 0s - loss: 0.2273 - acc: 0.7500 - recall_at_k: 0.2500
```

```
Epoch 50/50
1/1 [==============================] - 0s 27ms/step - loss: 0.3404 - acc: 0.1905 - recall_at_
k: 0.2857 - val_loss: 0.3342 - val_acc: 0.4286 - val_recall_at_k: 0.2381
1/1 - 0s - loss: 0.3590 - acc: 0.1429 - recall_at_k: 0.2857
```

2) W2V Padding error

미해결

-Wv2 parent

-Wv2 child

어려운 부분 -> 숫자 결과 읽는 거

역할 2

*임베딩 (semantic feature + VSM feature) 방식이 다른데 괜찮은가?

2) 성능 비교

a) 기존의 Flat한 label을 분류하는 모델과 성능 비교하기

- deepsoft – C (CNN)
- lstm?
- dense?

```
@abc.abstractmethod
def lstm_model(self):
    pass

@abc.abstractmethod
def dense_model(self):
    pass
```

b) different text encoders

-bert

-wv2

역할 2

0. Dataset

Dataset	Taxonomy	#Label	#Training	#Test
RCV1	Tree	103	23,149	781,265
Yelp	DAG	539	87,375	37,265

1. Compare with state-of-the-art

Text Encoders	Micro-F1 on RCV1	Micro-F1 on Yelp
HR-DGCNN (Peng et al., 2018)	0.7610	-
HMCN (Wehrmann et al., 2018)	0.8080	0.6640
Ours	**0.8313**	**0.6704**

2. Different text encoders

Text Encoders	RCV1		Yelp	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1
TextCNN	0.7717	0.5246	0.6281	0.3657
TextRNN	0.8152	0.5458	**0.6704**	0.4059
RCNN	**0.8313**	**0.6047**	0.6569	0.3951
FastText	0.6887	0.2701	0.6031	0.2323
DRNN	0.7846	0.5147	0.6579	0.4401
DPCNN	0.8220	0.5609	0.5671	0.2393
VDCNN	0.7263	0.3860	0.6395	0.4035
AttentiveConvNet	0.7533	0.4373	0.6367	0.4040
RegionEmbedding	0.7780	0.4888	0.6601	**0.4514**
Transformer	0.7603	0.4274	0.6533	0.4121
Star-Transformer	0.7668	0.4840	0.6482	0.3895

3. Hierarchical vs Flat

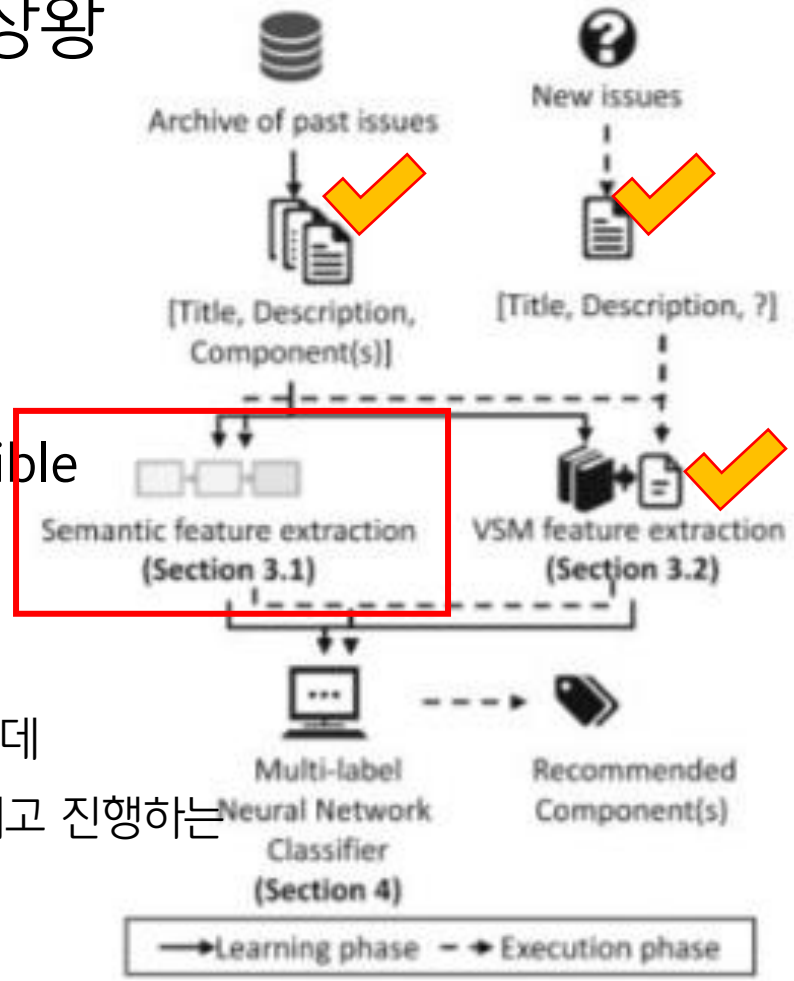
Text Encoders	Hierarchical		Flat	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1
TextCNN	0.7717	0.5246	0.7367	0.4224
TextRNN	0.8152	0.5458	0.7546	0.4505
RCNN	**0.8313**	**0.6047**	**0.7955**	**0.5123**
FastText	0.6887	0.2701	0.6865	0.2816
DRNN	0.7846	0.5147	0.7506	0.4450
DPCNN	0.8220	0.5609	0.7423	0.4261
VDCNN	0.7263	0.3860	0.7110	0.3593
AttentiveConvNet	0.7533	0.4373	0.7511	0.4286
RegionEmbedding	0.7780	0.4888	0.7640	0.4617
Transformer	0.7603	0.4274	0.7602	0.4339
Star-Transformer	0.7668	0.4840	0.7618	0.4745

역할 2 진행사항

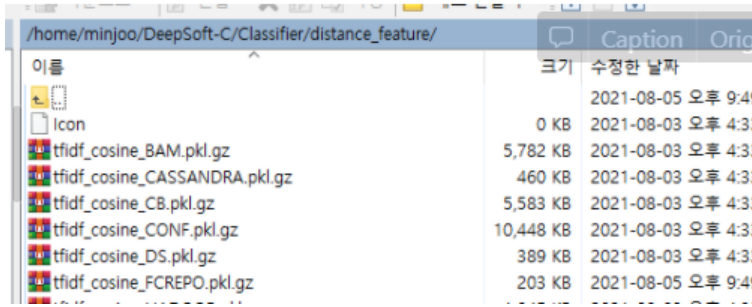
2) 7~8월 성능 비교 진행상황

데이터: FCREPO data

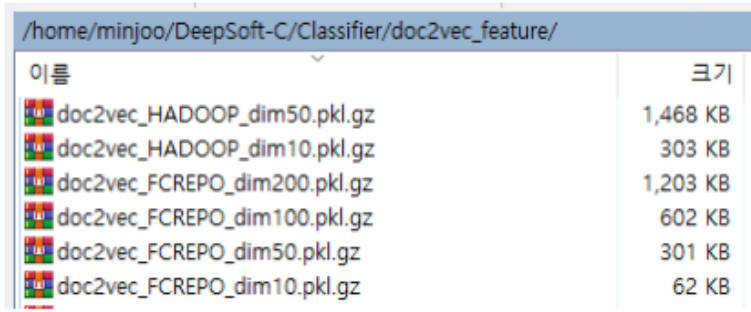
여기서 shape imcompatible
에러가 났음

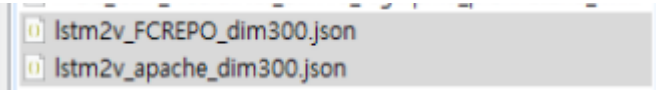


modeList = ['cosine_feature']



modeList = ['doc2vec']





계산량을 줄여주는 NCE 단계에서 문제인데
우리는 데이터에는 필요없는 단계라서 빼고 진행하는
방향으로 다시 돌려봐야할 듯

Dataset

Dataset

fcrepo	qpid
F4 (child_1) -auth -camel -core … **Audit(child_2)** -Fedora -FeSL -java-client	**(child_1)** -Broker -Build -Client **(child_2)** -Common -Tests -Documentation -JMS AMQP 0-x

아이디어

- 비슷한 상황(문제점) 언급한 논문

Hierarchical Transfer Learning for Multi-label Text Classification

-> multi-label 데이터의 불균형

-> transfer learning을 사용(HTrans)

: child binary-classifier가 parent classifier의 parameter을 가지고 초기화되고, child 분류를 fine-tune.

However, multi-label models suffer on categories with very few training examples (Krawczyk, 2016) due to data imbalance. Due to a large prediction space (all categories) of multilabel models, it is very difficult to optimize class weights to handle data imbalance. By contrast, binary classifiers provide more flexibility as class weights for each classifier can easily be optimized based on validation metrics. With a reasonable number of categories (few hundreds), collection of binary classifiers are a feasible option to solve MLHTC problems

아이디어

논문 깃허브 코드

https://github.com/francisco-perez-sorrosal/francisco-perez-sorrosal.github.io

How TO

부모 클래스가 P1 ~ P12로 나뉘어져 있고, 자식 클래스가 C1 ~ C12로 나뉘어져 있을 때, 가장 먼저 P1을 위한 이진 분류기를 훈련한다.
다음으로 C1 이진 분류기를 P1으로 학습한 분류기의 마지막 파라메터로 초기화해 훈련한다.

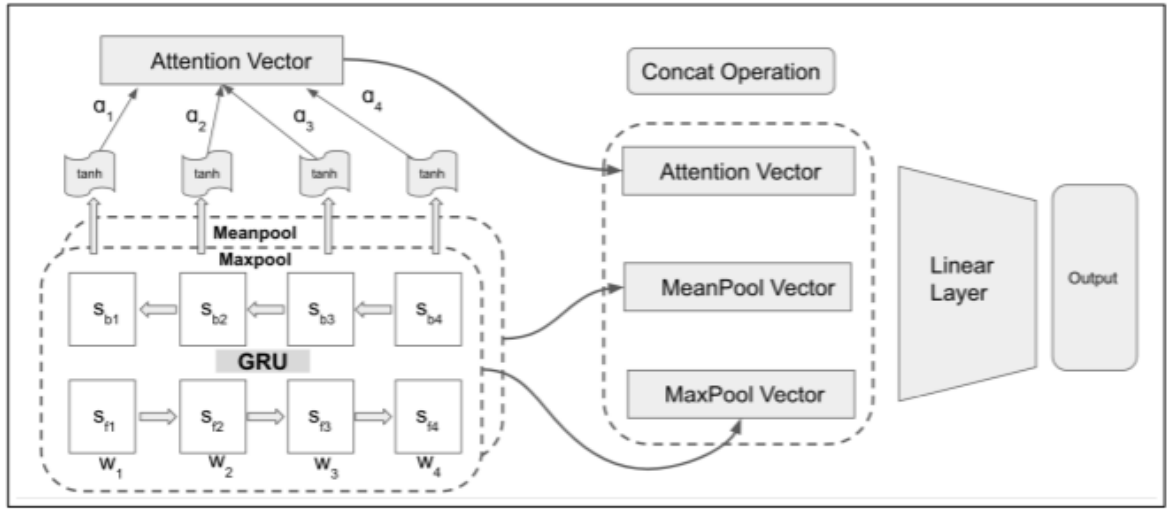


Figure 1: Architecture of our Proposed Model

입력 시퀀스를 세 갈래로 나누어 concatenate한다.

- bi-GRU -> 어텐션 2) MaxPool 3) MeanPool

concat한 벡터는 FC레이어로 넘겨 확률분포(시그모이드 활성함수)를 얻는다.

$$\mathbf{S} = [s_1, s_2, s_3, ...s_T]$$

$$R = [Att(\mathbf{S}), Maxpool(\mathbf{S}), Meanpool(\mathbf{S})]$$

아이디어

Bert Embedding -> transfer learning (parent)

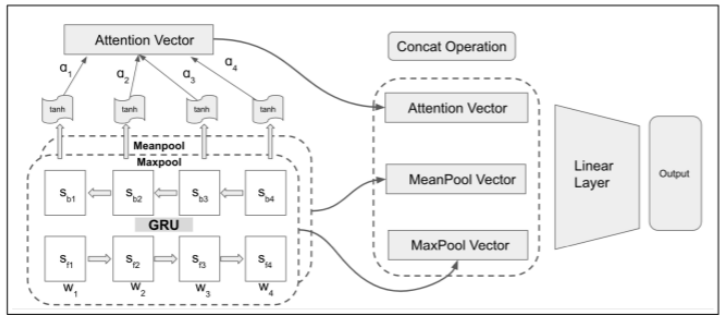


Figure 1: Architecture of our Proposed Model

-> CNN for child

아이디어

- Bert 임베딩의 특징
  - 다이나믹 **임베딩** / 동일 단어도 다른 임베딩 (중의성 해소)
  - Self-attention 기법

- Bert와 transfer learning이 잘 맞는가?

- Transfer learning 모델의 구현
  (예상되는 인풋, 아웃풋, 데이터 형태, 코드, layer 구성)
  -> https://jeinalog.tistory.com/13

정리

w2v padding error

성능 비교를 위한 모델 API 짜기

Transfer learning

https://github.com/francisco-perez-sorrosal/francisco-perez-sorrosal.github.io

https://jeinalog.tistory.com/13

아이디어

- 구현 및 오픈소스

- 궁금한 점: DeepSoft-C와 우리는
- - 임베딩 (semantic +) vs Bert
- - 분류 모델 CNN vs. transfer learning
- 이 다른데, 비교하는 게 의미가 있나? 다른

- 1) BERT 쓰는데 CNN인 거랑 (1차 실험)
- 2) 임베딩은 DeepSoft-C 꺼를 쓰고, transfer learning을 써야하는거 아닌가?
- 그그.. 차이점이 임베딩이랑 모델에서 둘다 있어도 되는?!