

# 2022 한국어 AI 경진대회

**NSML 소개**

**- Baseline code on NSML -**

대회운영사 (어반에이핏)

Baseline code on NSML

## 1. 데이터 구성

```
\_train
  \_ train_data (folder)
    \_ idx_000000 (audio data | pcm format)
    \_ idx_000001 (audio data | pcm format)
  \_ train_label(filename, text | csv format)
```

### 원천 데이터

- Sample Rate: 16000
- Channel: 1
- Sample Width: 2

### 라벨 파일

Filename.text

- idx\_000000, 자율 주행 기능 상태로 움직인 거리 말해 줘.
- idx\_000001, 내 옷이 문에 끼었어 조수석 문 다시 한번 열어 줘.
- idx\_000002, 이동 경로 갯길로 설정.

## Baseline code on NSML

## 2. 코드 구성

## 1. 제공 방식(접근 방법)

## 2. 구성 방법

## 1) setup.py

- pip 패키지
- 베이스 docker image

## 2) nsml\_package.txt

- apt-get / yum

## 3) main.py

**nsml\_package.txt** 19 bytes

```
1  libsndfile1
2  ffmpeg
3
```

**setup.py** 453 bytes

```
1  #nsml: pytorch/pytorch:1.6.0-cuda10.1-cudnn7-runtime
2  from distutils.core import setup
3
4  setup(
5      name='kospeech_nsml',
6      version='latest',
7      install_requires=[
8          # 'torch==1.7.0',
9          'levenshtein',
10         'librosa >= 0.7.0',
11         'numpy',
12         'pandas',
13         'tqdm',
14         'matplotlib',
15         'astropy',
16         'sentencepiece',
17         'torchaudio==0.6.0',
18         'pydub',
19         'glob2'
20     ],
21 )
22
```

## Baseline code on NSML

## 2. 코드 구성

## 1. main.py

## 1) nsml 함수 설명

- bind\_model -> 정의 및 호출위치
- args : mode, iteration, pause  
※ 세션 관련 Reserved parameter로  
수정 시, 제출이 불가할 수 있음
- pause / train 관련 분기
- report: 로그 항목 및 주기

```
if __name__ == '__main__':
    args = argparse.ArgumentParser()

    # DONOTCHANGE: They are reserved for nsml
    args.add_argument('--mode', type=str, default='train', help='submit일때 해당값이 test로 설정됩니다.')
    args.add_argument('--iteration', type=str, default='0',
                      help='fork 명령어를 입력할때의 체크포인트로 설정됩니다. 체크포인트 옵션을 안주면 마지막 wall time 의
                      model 을 가져옵니다.')
    args.add_argument('--pause', type=int, default=0, help='model 을 load 할때 1로 설정됩니다.')

def bind_model(model, optimizer=None):
    def save(path, *args, **kwargs):
        state = {
            'model': model.state_dict(),
            'optimizer': optimizer.state_dict()
        }
        torch.save(state, os.path.join(path, 'model.pt'))
        print('Model saved')

    def load(path, *args, **kwargs):
        state = torch.load(os.path.join(path, 'model.pt'))
        model.load_state_dict(state['model'])
        if 'optimizer' in state and optimizer:
            optimizer.load_state_dict(state['optimizer'])
        print('Model loaded')

    # 추론
    def infer(path, **kwargs):
        return inference(path, model)

    nsml.bind(save=save, load=load, infer=infer) # 'nsml.bind' function must be called at the end.

def inference(path, model, **kwargs):
    model.eval()

    results = []
    for i in glob(os.path.join(path, '*')):
        results.append(
            {
                'filename': i.split('/')[-1],
                'text': single_infer(model, i)[0]
            }
        )
    return sorted(results, key=lambda x: x['filename'])

collate_fn=collate_fn,
num_workers=config.num_workers
)

model, train_loss, train_cer = trainer(
    'train',
    config,
```

Baseline code on NSML

## 2. 코드 구성

### 1. NSML 상세 기능은 NSML docs 참고

- 대회 홈페이지 → FAQ  
→ NSML사용법은 어디서 확인할 수 있나요?
- [https://n-clair.github.io/ai-docs/\\_build/html/ko\\_KR/index.html](https://n-clair.github.io/ai-docs/_build/html/ko_KR/index.html)

#### nsml.bind

```
nsml.bind(save=None, load=None, **kwargs)
```

NSML에서 사용할 save와 load, infer함수를 바인드해줍니다.

- 매개변수:
- save (fn) - 모델을 저장하는 save 함수입니다.
  - load (fn) - 저장된 모델을 불러오는 load 함수입니다.

#### nsml.report

```
nsml.report(summary=False, scope=None, **kwargs)
```

변수의 변화량을 기록하여서 web 에서 scalar, tensorboard, visdom 에 그래프를 그릴 때 사용됩니다.

- 매개변수:
- summary (bool or None) - 값이 True이면, nsml ps 에 값이 보입니다.
  - scope - locals() 로 값을 주면, 해당 범위의 값에 nsml exec 으로 접근할 수 있습니다. 또는 python 의 eval 로 세션에 접근할 수 있습니다.
  - step - scalar 그래프에서 보여줄 x 축 값을 설정합니다.
  - \*\*kwargs (str or None) - 트래킹할 변수를 key=value 형식으로 입력합니다.  
(ex. loss=loss) (walltime key 값은 내부에서 자동으로 time.time() 으로 설정되므로 만약 walltime=variable 로 값을 넘겨질 경우 time.time() 의 값으로 overwrite됩니다.)

예외: **TypeError** - 트래킹할 변수에 json serialize 할 수 없는 값을 넘길경우 (ex. TensorType) 에러가 발생합니다.

#### NSML Constants

```
nsml.IS_ON_NSML
```

NSML server 에서 code 가 실행되는 상태면 True를 출력합니다.

```
nsml.HAS_DATASET
```

session 이 dataset 을 가지고 있으면 True를 출력합니다.

```
nsml.DATASET_PATH
```

nsml run -d DATASET 으로 실행시켰을 때 dataset 의 path를 출력합니다.

```
nsml.DATASET_NAME
```

nsml run -d DATASET 으로 실행시켰을 때 dataset 의 이름입니다.

#### reserved arguments

NSML에서는 세션을 재생성 하는 command 들(nsml fork, nsml submit)을 사용할 때 entry 파일에서 다음의 3가지 옵션을 받을 수 있어야 합니다.

```
--mode TEXT
```

들어온 command 가 nsml submit 일 경우 사용됩니다.

```
nsml submit
```

--mode test 로 설정됩니다.

```
--pause INTEGER
```

model 을 load 할때 1로 설정됩니다.

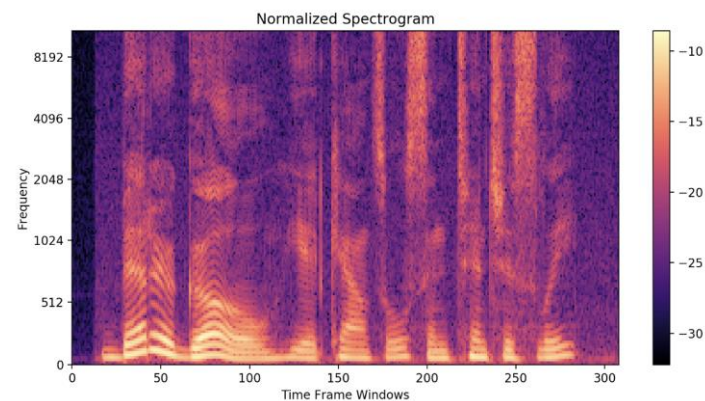
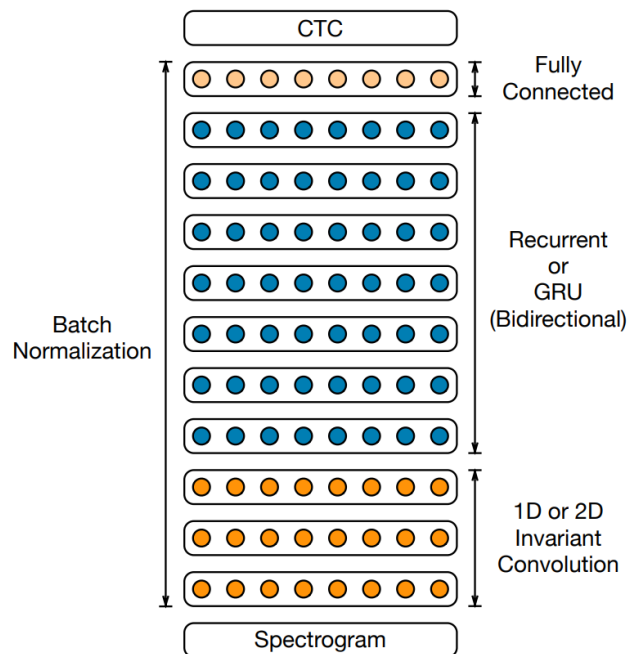
```
--iteration TEXT
```

nsml fork 할 때의 iteration로 설정됩니다. 이는 checkpoint와 동일한 개념입니다. iteration 옵션을 주지 않으면 마지막 wall time 의 model 을 가져옵니다.

Baseline code on NSML

### 3. 주요 코드 (프레임워크 및 사용 모델)

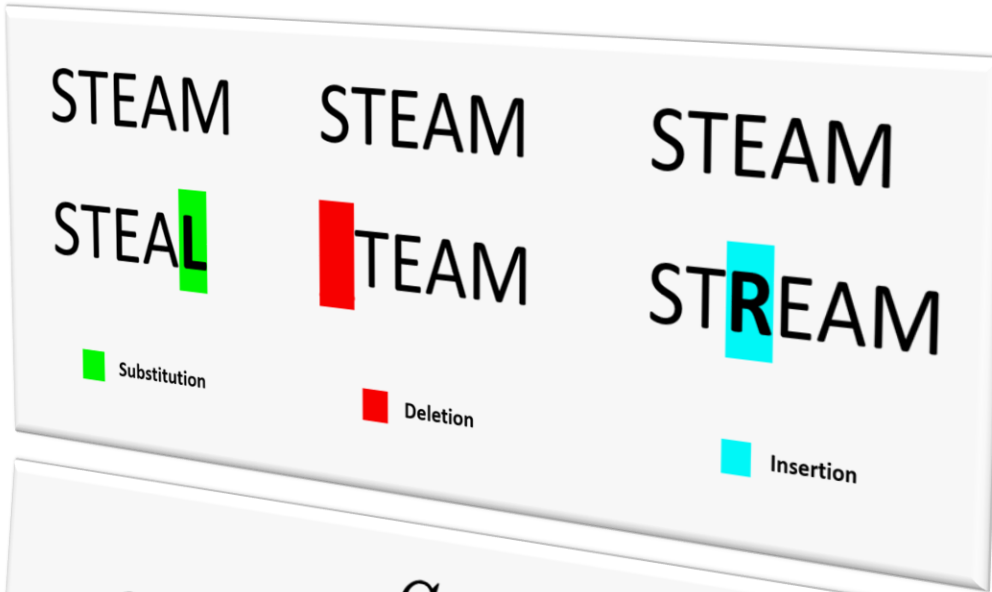
- 출처: Kospeech
- 모델: DeepSpeech2



Baseline code on NSML

### 3. 주요 코드

- 제 1 평가지표
  - CER (글자오류율)  
: 예측한 글자와 실제 글자가  
얼마나 다른지 나타내는 척도
- 평가 Metric & Optimizer & Loss function
  - 모듈 폴더의 Util에 위치



The diagram shows three examples of word corrections from a predicted word to a target word, illustrating the components of CER: Substitution, Deletion, and Insertion.

Target Word	Predicted Word	Errors
STEAM	STEAL	Substitution (L for A)
STEAM	TEAM	Deletion (S)
STEAM	STREAM	Insertion (R)

Legend:   
Green square: Substitution   
Red square: Deletion   
Blue square: Insertion

$$CER = \frac{S + D + I}{N}$$





## Baseline code on NSML

- preprocessing
- vocabulary.py
  - char2id, id2char

## 3. 주요 코드

```
class KoreanSpeechVocabulary(Vocabulary):
    def __init__(self, vocab_path, output_unit: str = 'character', sp_model_path=None):
        super(KoreanSpeechVocabulary, self).__init__()

        self.vocab_dict, self.id_dict = self.load_vocab(vocab_path, encoding='utf-8')
        self.sos_id = int(self.vocab_dict['< sos >'])
        self.eos_id = int(self.vocab_dict['< eos >'])
        self.pad_id = int(self.vocab_dict['< pad >'])
        self.blank_id = int(self.vocab_dict['< blank >'])
        self.labels = self.vocab_dict.keys()

        self.vocab_path = vocab_path
        self.output_unit = output_unit

    def __len__(self):

        return len(self.vocab_dict)

    def label_to_string(self, labels):
        """
        Converts label to string (number => Hangeul)

        Args:
            labels (numpy.ndarray): number label

        Returns: sentence
            - **sentence** (str or list): symbol of labels
        """

        if len(labels.shape) == 1:
            sentence = str()
            for label in labels:
                if label.item() == self.eos_id:
                    break
                elif label.item() == self.blank_id:
                    continue
                sentence += self.id_dict[label.item()]
            return sentence

        sentences = list()
        for batch in labels:
            sentence = str()
            for label in batch:
                if label.item() == self.eos_id:
                    break
                elif label.item() == self.blank_id:
                    continue
                sentence += self.id_dict[label.item()]
            sentences.append(sentence)
        return sentences
```

id	char	freq
0	<pad>	0
1	<sos>	0
2	<eos>	0
3	,	5774244
4	.	640923
5	그	556372
6	이	509291
7	는	374559
8	아	370443
9	가	369698
10	고	356378
11	어	333842
12	거	306987
13	지	276453
14	데	249269
15	?	235024
16	나	229646
17	하	226136
18	다	221216
19	서	211193
20	에	204330
21	도	190561
22	게	177140
23	니	173284
24	기	149467
25	은	144674
26	면	142025
27	아	137552
28	있	133155
29	한	121564
30	을	121048
31	까	119483
32	해	115148
33	리	111854
34	라	111479
35	래	105784
36	사	100533
37	크	99781



## Baseline code on NSML

## 3. 주요 코드 (데이터 로딩)

- modules/data.py
  - data loader
    - train:valid = 8:2
    - SpectrogramDataset Class
    - feature ( input ) / transcript (label)
  - collate\_fn을 사용하여 label padding

```
train_dataset = SpectrogramDataset(  
    train_audio_paths,  
    train_transcripts,  
    vocab.sos_id, vocab.eos_id,  
    config=config,  
    spec_augment=config.spec_augment,  
    dataset_path=config.dataset_path,  
    audio_extension=config.audio_extension,  
)
```

```
class SpectrogramDataset(Dataset, SpectrogramParser):  
    """  
    Dataset for feature & transcript matching  
    """  
  
    Args:  
        audio_paths (list): list of audio path  
        transcripts (list): list of transcript  
        sos_id (int): identification of <start of sequence>  
        eos_id (int): identification of <end of sequence>  
        spec_augment (bool): flag indicating whether to apply spec augmentation  
        config (DictConfig): set of configurations  
        dataset_path (str): path of dataset  
    """  
  
    def __init__(  
        self,  
        audio_paths: list, # List  
        transcripts: list, # List  
        sos_id: int, # identification of <start of sequence>  
        eos_id: int, # identification of <end of sequence>  
        config, # set of arguments  
        spec_augment: bool = False,  
        dataset_path: str = None,  
        audio_extension: str = 'pcm'  
    ) -> None:  
        super(SpectrogramDataset, self).__init__() # Call the init method of the parent class  
        feature_extract_by=config.feature_extract_by  
        n_mels=config.n_mels, frame_length=config.frame_length  
        del_silence=config.del_silence  
        normalize=config.normalize  
        time_mask_num=config.time_mask_num  
        sos_id=sos_id, eos_id=eos_id  
        audio_extension=audio_extension  
  
        self.audio_paths = list(audio_paths)  
        self.transcripts = list(transcripts)  
        self.augment_methods = [self._spec_augment, self._time_mask]  
        self.dataset_size = len(self.audio_paths)  
        self.shuffle()  
  
    def __getitem__(self, idx):  
        """ get feature vector & transcript  
        """  
        feature = self.parse_audio(os.path.join(self.audio_paths[idx]))  
  
        if feature is None:  
            return None  
  
        transcript = self.parse_transcript(self.transcripts[idx])  
  
        return feature, transcript  
  
    def collate_fn(batch):  
        """ functions that pad to the maximum sequence length """  
  
        def seq_length(p):  
            return len(p[0])  
  
        def target_length(p):  
            return len(p[1])  
  
        # sort by sequence length for rnn.pack_padded_sequence()  
        try:  
            batch = [i for i in batch if i is not None]  
            batch = sorted(batch, key=lambda sample: sample[0].size(0), reverse=True)  
  
            seq_lengths = [len(s[0]) for s in batch]  
            target_lengths = [len(s[1]) - 1 for s in batch]  
  
            max_seq_sample = max(batch, key=seq_length)[0]  
            max_target_sample = max(batch, key=target_length)[1]  
  
            max_seq_size = max_seq_sample.size(0)  
            max_target_size = len(max_target_sample)  
  
            feat_size = max_seq_sample.size(1)  
            batch_size = len(batch)  
  
            seqs = torch.zeros(batch_size, max_seq_size, feat_size)  
            targets = torch.zeros(batch_size, max_target_size).to(torch.long)  
            targets.fill_(pad_id)  
  
            for x in range(batch_size):  
                sample = batch[x]  
                tensor = sample[0]  
                target = sample[1]  
                seq_length = tensor.size(0)  
  
                seqs[x].narrow(0, 0, seq_length).copy_(tensor)  
                targets[x].narrow(0, 0, len(target)).copy_(torch.LongTensor(target))  
  
            seq_lengths = torch.IntTensor(seq_lengths)  
            return seqs, targets, seq_lengths, target_lengths
```

## Baseline code on NSML

## 4. 주요 코드 (학습 부분)

## • trainer.py

- Dataloader를 통해 iteration 진행
- 매 iter마다 update
- config.print\_every 주기마다 print 진행
- 완료시 model, loss, cer 리턴

```
9 def trainer(mode, config, dataloader, optimizer, model, criterion, metric, train_begin_time, device):
10
11     log_format = "[INFO] step: {:4d}/{:4d}, loss: {:.6f}, " \
12                  "cer: {:.2f}, elapsed: {:.2f}s {:.2f}s {:.2f}s {:.2f}h, lr: {:.6f}"
13
14     total_num = 0
15     epoch_loss_total = 0.
16     print(f'[INFO] {mode} Start')
17     epoch_begin_time = time.time()
18     cnt = 0
19     for inputs, targets, input_lengths, target_lengths in dataloader:
20         begin_time = time.time()
21
22         optimizer.zero_grad()
23         inputs = inputs.to(device)
24         targets = targets.to(device)
25         input_lengths = input_lengths.to(device)
26         target_lengths = torch.as_tensor(target_lengths).to(device)
27         model = model.to(device)
28
29         outputs, output_lengths = model(inputs, input_lengths)
30
31         loss = criterion(
32             outputs.transpose(0, 1),
33             targets[:, 1:],
34             tuple(output_lengths),
35             tuple(target_lengths)
36         )
37
38         y_hats = outputs.max(-1)[1]
39
40         if mode == 'train':
41             optimizer.zero_grad()
42             loss.backward()
43             optimizer.step(model)
44
45         total_num += int(input_lengths.sum())
46         epoch_loss_total += loss.item()
47
48         torch.cuda.empty_cache()
49
50         if cnt % config.print_every == 0:
51             current_time = time.time()
52             elapsed = current_time - begin_time
53             epoch_elapsed = (current_time - epoch_begin_time) / 60.0
54             train_elapsed = (current_time - train_begin_time) / 3600.0
55             cer = metric(targets[:, 1:], y_hats)
56             print(log_format.format(
57                 cnt, len(dataloader), loss,
58                 cer, elapsed, epoch_elapsed, train_elapsed,
59                 optimizer.get_lr(),
60             ))
61             cnt += 1
62     return model, epoch_loss_total/len(dataloader), metric(targets[:, 1:], y_hats)
```

Baseline code on NSML

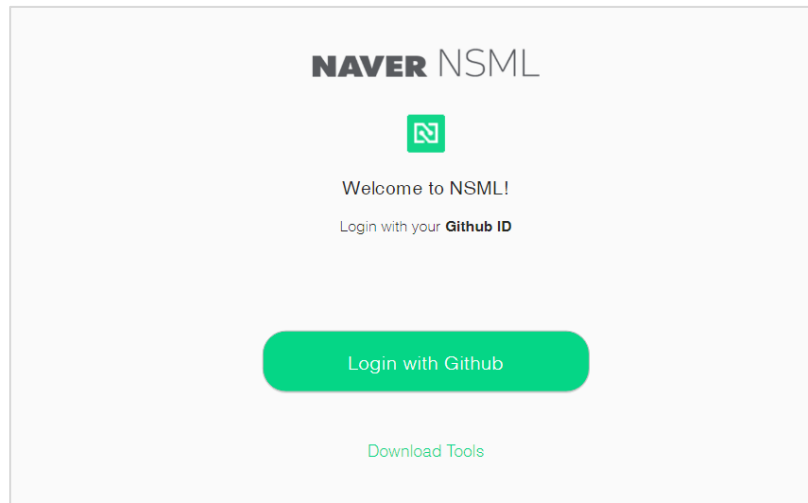
## 4. 코드 실행 (NSML 설치 및 주요 명령어)

### 1. 설치방법

- <https://ncp.nsml.navercorp.com>

▶ Cli 클라이언트도 다운로드 가능

※ Clova 버전과 다른 Client로 재설치 필요



### 2. 주요 명령어

#### 학습 시작 (nsml run)

```
nsml run -d <dataset name> ₩
        -e <main file> ₩
        -c <cpus> ₩
        -g <gpus> ₩
        --memory <memory size (GB)>
```

#### 학습 시작 후, 저장된 Checkpoint 확인 (nsml model ls)

```
nsml model ls <session id>
```

#### 학습 완료 후, 제출 시 (nsml submit)

```
nsml submit <session id> <checkpoint name>
```

※ 인퍼런스는 3,600초(1h) 이내 종료되어야 함

Baseline code on NSML

5. 리더보드

NSML 웹 인터페이스

custom · descending Public

Rank	Name	Score	Model	Count	Last Submit	Recorded
1	admin group	0.241	ncp-admin/face3/1/10	1	14 days ago	14 days ago
2	bluebrush	0.235	bluebrush/face3/2/10	2	17 days ago	17 days ago

오픈 리더보드

### NCP Hackathon 2022

NCP 2022 라운드 1 (face3)

Explore

RANKING BOARD  
END

Last updated 2022-08-08T15:49:39+09:00

Rank	Name	Score	Recorded	Count
1	admin group	0.241	14 days ago	1
2	bluebrush	0.235	17 days ago	2

Baseline code on NSML

## 6. FAQ

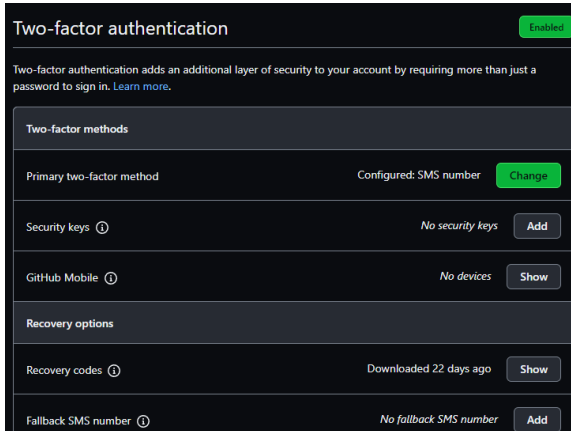
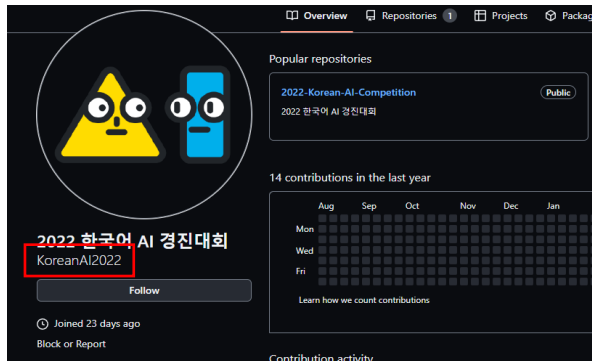
### Case 1. 로그인 불가

#### ① User ID

- Github 닉네임이 아닌 Username로 Cli 로그인
- 자신의 Github url의 / 뒷부분 확인

#### ② 2중 인증

- 2중 인증이 안된 경우, Github 규정 상 서드파티앱 동작 불가



### Case 2. 세션 관리

- 학습 진행 및 제출 시, 세션이 구성
- 실행 중이거나 저장된 세션이 너무 많을 경우, 전체적인 시스템 성능에 영향
- ▶ 사용하지 않는 세션은 주기적인 삭제 필요

※ 주의: 최고점수가 나온 세션은 삭제하지 않고 유지 필요

#### 기타 문의사항

- 대회 홈페이지 내 문의 게시판 (평일 09:00~18:00 실시간 답변 운영)



Baseline code on NSML

좋은 성과 있으시길 바랍니다