KU KONKUK UNIVERSITY KAIST

형태소 분석
(Morphological Analysis)

Practical Exercise

자연어처리연구실

과제: Tabular Parsing

- 오른쪽 우선 tabular parsing 알고리즘을 이용하여 다음 어절을 형태소 분석하는 프로그램을 작성하시오. (10점)

  * Java, Python, C/C++ 등 어떤 언어를 사용해도 됨

  - 어절: 건국대가
  - 사전: 건국/NNG, 국/NNG, 대/NNB, 대가/NNG, 건국대/NNP, 가/NNG, 가/JKS
  - 접속 정보: NNG+NNG, NNB+JKS, NNG+JKS, NNP+NNG, NNP+JKS

	ㄱ	ㅓ	ㄴ	ㄱ	ㅜ	ㄱ	ㄷ	ㅐ	ㄱ	ㅏ
ㄱ						건국/NNG		건국대/NNP		**건국/NNG+대가/NNG 건국대/NNP+가/NNG 건국대/NNP+가/JKS**
ㅓ										
ㄴ										
ㄱ			국/NNB							
ㅜ										
ㄱ										
ㄷ		대/NNB		**대가/NNG 대/NNB+가/JKS**						
ㅐ										
ㄱ		**가/NNG 가/JKS**								
ㅏ										

사전: 건국/NNG, 국/NNG, 대/NNB, 대가/NNG, 건국대/NNP, 가/NNG, 가/JKS

접속 정보: NNG+NNG, NNB+JKS, NNG+JKS, NNP+NNG, NNP+JKS

자소 분리 및 결합

예제 코드 다운로드: https://github.com/KUNLP/KTAI-Practice

자소 분리

```
# 자소 분리 및 결합 코드
cho_list = ['ㄱ', 'ㄲ', 'ㄴ', 'ㄷ', 'ㄸ', 'ㄹ', 'ㅁ', 'ㅂ', 'ㅃ', 'ㅅ', 'ㅆ', 'ㅇ', 'ㅈ', 'ㅉ', 'ㅊ', 'ㅋ', 'ㅌ', 'ㅍ', 'ㅎ']
jung_list = ['ㅏ', 'ㅐ', 'ㅑ', 'ㅒ', 'ㅓ', 'ㅔ', 'ㅕ', 'ㅖ', 'ㅗ', 'ㅘ', 'ㅙ', 'ㅚ', 'ㅛ', 'ㅜ', 'ㅝ', 'ㅞ', 'ㅟ', 'ㅠ', 'ㅡ', 'ㅢ',
             'ㅣ']
jong_list = ['', 'ㄱ', 'ㄲ', 'ㄳ', 'ㄴ', 'ㄵ', 'ㄶ', 'ㄷ', 'ㄹ', 'ㄺ', 'ㄻ', 'ㄼ', 'ㄽ', 'ㄾ', 'ㄿ', 'ㅀ', 'ㅁ', 'ㅂ', 'ㅄ', 'ㅅ',
             'ㅆ', 'ㅇ', 'ㅈ', 'ㅊ', 'ㅋ', 'ㅌ', 'ㅍ', 'ㅎ']

def jaso_split(korean_word):
    result = []
    for c in list(korean_word.strip()):
        # 현재 문자가 한글일 경우
        if '가' <= c <= '힣':
            chosung = (ord(c) - ord('가')) // 588
            jungsung = ((ord(c) - ord('가')) - (588 * chosung)) // 28
            jongsung = (ord(c) - ord('가')) - (588 * chosung) - (28 * jungsung)
            result.append(cho_list[chosung])
            result.append(jung_list[jungsung])
            if jongsung:
                result.append(jong_list[jongsung])
        # 현재 문자가 한글이 아닐 경우
        else:
            result.append(c)

    return result
```

초성: 19, 중성: 21, 종성: 28

"중성*종성=588"로 나눈 몫

가, 각, ..., 갛, 개, 객, ... , 갱, ..., 기, 긱, ..., 깋, 까, 깨, ...

종성: 28

중성*종성: 588

자소 분리 및 결합

자소 결합

jongsung=''

```
def jaso_combine(chosung, jungsung, jongsung):
    if chosung and jungsung:
        result = chr((28 * int(jung_list.index(jungsung))) + int(jong_list.index(jongsung)) + (588 * cho_list.index(chosung)) + ord('가'))
    else:
        result = ''.join([chosung, jungsung, jongsung])
    return result
```

Main

```
if __name__ == '__main__':
    morpheme = {"건국": ["NNG"], "국": ["NNG"], "대": ["NNB"], "대가": ["NNG"], "건국대": ["NNP"], "가": ["NNG", "JKS"]}
    grammar = ["NNG+NNG", "NNB+JKS", "NNG+JKS", "NNP+NNG", "NNP+JKS"]

    # input_text = input("Input Sentence : ")
    input_text = "건국대가"
    jaso_list = jaso_split(input_text)
    print(jaso_list)
    print(tabular_parsing(jaso_list, morpheme, grammar))
```

```
['ㄱ', 'ㅓ', 'ㄴ', 'ㄱ', 'ㅜ', 'ㄱ', 'ㄷ', 'ㅐ', 'ㄱ', 'ㅏ']
[['건국대/NNP', '가/NNG'], ['건국대/NNP', '가/JKS'], ['건국/NNG', '대가/NNG']]
```

KU KONKUK UNIVERSITY

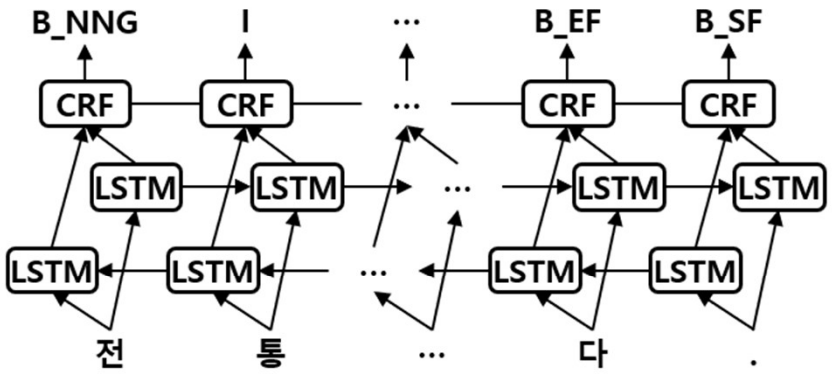
품사 부착
(Part-Of-Speech Tagging)

Practical Exercise

자연어처리연구실

실습: Seq. Labeling for POS Tagging

- BiLSTM-CRF를 이용하여 형태소 분석을 수행하는 프로그램을 작성하시오.



예제 코드 다운로드: https://github.com/KUNLP/KTAI-Practice

음절	전	통	<SP>	문	화	와	<SP>	…
라벨	B_NNG	I	<SP>	B_NNG	I	B_JC	<SP>	…

  - 데이터 형식
    - 한글 음절 열 ₩t 레이블 열
    - 예제: 전 통 〈SP〉 문 화 와 〈SP〉 … ₩t B_NNG I 〈SP〉 B_NNG I B_JC 〈SP〉 …

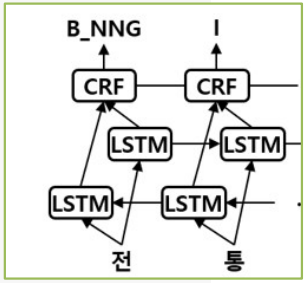
실습: Seq. Labeling for POS Tagging

모델 설계

```
class BiLSTM_CRF(nn.Module):

    def __init__(self, config):
        super(BiLSTM_CRF, self).__init__()

        # 전체 음절 개수
        self.eumjeol_vocab_size = config["eumjeol_vocab_size"]

        # 음절 임베딩 사이즈
        self.embedding_size = config["embedding_size"]

        # LSTM 히든 사이즈
        self.hidden_size = config["hidden_size"]

        # 분류할 태그의 개수
        self.number_of_tags = config["number_of_tags"]

        # 입력 데이터에 있는 각 음절 index를 대응하는 임베딩 벡터로 치환해주기 위한 임베딩 객체
        # 기존에 사전학습 된 음절 임베딩을 사용할 수도 있고 랜덤으로 초기화 한 후,
        # 모델 학습 과정 중에 같이 학습 시키는 것도 가능
        # 예제 코드는 랜덤으로 초기화 한 후 같이 학습하도록 설정
        self.embedding = nn.Embedding(num_embeddings=self.eumjeol_vocab_size, embedding_dim=self.embedding_size,
                                      padding_idx=0)

        # Bi-LSTM layer
        self.bi_lstm = nn.LSTM(input_size=self.embedding_size, hidden_size=self.hidden_size,
                               num_layers=3, batch_first=True, dropout=config["dropout"],
                               bidirectional=True)

        # CRF layer
        self.crf = CRF(num_tags=self.number_of_tags, batch_first=True)

        # fully_connected layer를 통하여 출력 크기를 number_of_tags에 맞춰
        # (batch_size, max_length, hidden_size*2) -> (batch_size, max_length, number_of_tags)
        self.hidden2num_tag = nn.Linear(in_features=self.hidden_size*2, out_features=self.number_of_tags)
```

3층 (그림과 다름)

전방향+역방향



CRF 설치

```
!pip install pytorch-crf

root_dir = "/gdrive/My Drive/colab/6-2. POS Tagging"
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pytorch-crf in /usr/local/lib/python3.8/dist-packages (0.7.2)
```

실습: Seq. Labeling for POS Tagging

모델 설계

```
def forward(self, inputs, labels=None):
    # (batch_size, max_length) -> (batch_size, max_length, embedding_size)
    eumjeol_inputs = self.embedding(inputs)

    # hidden_outputs : (batch_size, max_length, hidden_size)
    hidden_outputs, _ = self.bi_lstm(eumjeol_inputs)

    # (batch_size, max_length, hidden_size*2) -> (batch_size, max_length, number_of_tags)
    hidden_outputs = self.hidden2num_tag(hidden_outputs)

    if(labels is not None):
        log_likelihood = self.crf(emissions=hidden_outputs, tags=labels, reduction="mean")
        loss = log_likelihood * -1.0

        return loss
    else:
        output = self.crf.decode(emissions=hidden_outputs)

        return output
```

Cell States (사용 안함)

정답 레이블에 해당하는 확률 값들의 평균을 리턴 (Log Likelihood: -∞~0.0)

Loss 함수로 만들기 위해 Negative Log Likelihood로 변경 (∞~0.0)

데이터 읽기

```
# 학습 or 평가 데이터를 읽어 리스트에 저장
def read_data(file_path):
    with open(file_path, "r", encoding="utf8") as inFile:
        lines = inFile.readlines()

    datas = []
    for line in lines:
        # 입력 문장을 ₩t으로 분리
        pieces = line.strip().split("₩t")

        # 입력 문자열을 음절 단위로 분리
        eumjeol_sequence, label_sequence = pieces[0].split(), pieces[1].split()

        datas.append((eumjeol_sequence, label_sequence))

    return datas
```

전 통 〈SP〉 문 화 와 〈SP〉 … ₩t B_NNG I 〈SP〉 B_NNG I B_JC 〈SP〉 …

실습: Seq. Labeling for POS Tagging

사전 읽기

```
# 데이터를 읽고 대응하는 딕셔너리 생성
def read_vocab_data(vocab_data_path):
    term2idx, idx2term = {}, {}

    with open(vocab_data_path, "r", encoding="utf8") as inFile:
        lines = inFile.readlines()

    for line in lines:
        term = line.strip()
        term2idx[term] = len(term2idx)
        idx2term[term2idx[term]] = term

    return term2idx, idx2term
```

eumjeol_vocab.txt

1	[PAD]
2	!
3	"
4	#
5	%
6	&
7	'
8	(

label_vocab.txt

1	[PAD]
2	<SP>
3	B_#EC
4	B_#EC+E
5	B_#EF
6	B_#EP
7	B_#ETM
8	B_#JKB

데이터 전처리

```
# 입력 데이터 전처리
def convert_data2feature(datas, max_length, eumjeol2idx, label2idx):
    # 음절 데이터, 각 데이터의 실제 길이, 라벨 데이터를 담을 리스트
    eumjeol_features, label_features = [], []

for eumjeol_sequence, label_sequence in datas:

    # 사전 설정한 max_length의 길이를 갖는 numpy array 생성
    eumjeol_feature = np.zeros(shape=(max_length), dtype=int)
    label_feature = np.zeros(shape=(max_length), dtype=int)

    # 음절 sequence와 라벨 sequence의 각 값들을 index로 치환하고 위에서 생성한 numpy array에 저장
    for index in range(len(eumjeol_sequence[:max_length])):
        eumjeol_feature[index] = eumjeol2idx[eumjeol_sequence[index]]
        label_feature[index] = label2idx[label_sequence[index]]

    # 변환한 데이터를 각 리스트에 저장
    eumjeol_features.append(eumjeol_feature)
    label_features.append(label_feature)

return eumjeol_features, label_features
```

```
datas = [
    (['전', '통', ' <SP> ', '문', '화', '와', ' <SP> ', '외', '래', ' <SP> ', ...], ['B_NNG', 'I', ' <SP>

    (...),

    ]

eumjeol_features : Tensor([1307, 1664, 15, 723, 1831, 1197, 15, 1208, ... ], [...], ...)
label_features : Tensor([ 99, 199, 1, 99, 199, 54, 1, 99, 199, 1, 99, 199, ... ], [...], ...)
```

실습: Seq. Labeling for POS Tagging

Train

```
def train(config):
    # 학습 데이터 읽기
    train_datas = read_data(file_path=config["train_data_path"])

    # 음절 및 라벨 딕셔너리 생성
    eumjeol2idx, idx2eumjeol = read_vocab_data(vocab_data_path=config["eumjeol_vocab_data_path"])
    label2idx, idx2label = read_vocab_data(vocab_data_path=config["label_vocab_data_path"])

    # 입력 데이터 전처리
    train_eumjeol_features, train_label_features = convert_data2feature(datas=train_datas, max_length=config["max_length"],
                                                                        eumjeol2idx=eumjeol2idx, label2idx=label2idx)
```

```
    for epoch in range(config["epoch"]):
        model.train()

        losses = []
        for batch in train_dataloader:
            batch = tuple(t.cuda() for t in batch)

            # 음절 데이터, 라벨 데이터
            inputs, labels = batch[0], batch[1]

            # 모델 학습
            loss = model(inputs, labels)

            # 역전파 단계를 실행하기 전에 변화도를 0으로 변경
            optimizer.zero_grad()

            # loss 값으로부터 모델 내부 각 매개변수에 대하여 gradient 계산
            loss.backward()

            # 모델 내부 각 매개변수 가중치 갱신
            optimizer.step()
```

학습모드

학습된 모델 저장

```
torch.save(model.state_dict(), os.path.join(output_dir, "trained_model.pt"))
```

실습: Seq. Labeling for POS Tagging

Test

```
def test(config):
    # 평가 데이터 읽기
    test_datas = read_data(file_path=config["test_data_path"])
    test_datas = test_datas[:10]

    # 음절 및 라벨 딕셔너리 생성
    eumjeol2idx, idx2eumjeol = read_vocab_data(vocab_data_path=config["eumjeol_vocab_data_path"])
    label2idx, idx2label = read_vocab_data(vocab_data_path=config["l

    test_eumjeol_features, test_label_features = convert_data2featur

    # BiLSTM_CRF 모델 객체 생성
    model = BiLSTM_CRF(config).cuda()
    # 사전학습한 모델 파일로부터 가중치 불러옴
    model.load_state_dict(torch.load(os.path.join(config["sa

model.eval()
```

학습된 모델 로드

평가모드

```
    for batch in test_dataloader:
        batch = tuple(t.cuda() for t in batch)

        # 음절 데이터, 라벨 데이터
        inputs, labels = batch[0], batch[1]

        # 모델 평가
        predicts = model(inputs)
        inputs, predicts, labels = inputs[0], predicts[0], labels[0]

        # Tensor를 numpy array로 변경하고 입력 데이터의 실제 길이만큼 추출
        inputs = inputs.cpu().numpy().tolist()
        labels = labels.cpu().numpy().tolist()

        input_length = config["max_length"] - inputs.count(0)

        inputs = [idx2eumjeol[piece] for piece in inputs[:input_length]]
        predicts = [idx2label[piece] for piece in predicts[:input_length]]
        labels = [idx2label[piece] for piece in labels[:input_length]]

        print("inputs : {}".format(inputs))
        print("predicts : {}".format(predicts))
        print("labels : {}".format(labels))
        print()
```

예시로 배치의 첫번째 것만 출력

실습: Seq. Labeling for POS Tagging

Main

```
if(__name__=="__main__"):
    root_dir = "/gdrive/My Drive/colab/Morphology_Analysis"
    save_dir = os.path.join(root_dir, "save")
    output_dir = os.path.join(root_dir, "output")
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    config = {"mode": "train",
              "trained_model_name": "trained_model.pt",
              "train_data_path": os.path.join(root_dir, "train_datas.txt"),
              "test_data_path": os.path.join(root_dir, "test_datas.txt"),
              "save_dir_path": save_dir,
              "output_dir_path": output_dir,
              "eumjeol_vocab_data_path": os.path.join(root_dir, "eumjeol_vocab.txt"),
              "label_vocab_data_path": os.path.join(root_dir, "label_vocab.txt"),
              "eumjeol_vocab_size": 1892,
              "embedding_size": 128,
              "hidden_size": 128,
              "max_length": 100,
              "number_of_tags": 302,
              "epoch": 10,
              "batch_size": 64,
              "dropout": 0.1
              }

    if(config["mode"] == "train"):
        train(config)
    else:
        test(config)
```

```
Average loss : 39.693843398579254

Average loss : 7.514955607272079

Average loss : 5.412512378363142

Average loss : 4.399688872632047
```

```
inputs : ['전', '통', '<SP>', '문', '화', '와', '<SP>', '외', '래',
predicts : ['B_NNG', 'I', '<SP>', 'B_NNG', 'I', 'B_JC', '<SP>', 'B_N
labels : ['B_NNG', 'I', '<SP>', 'B_NNG', 'I', 'B_JC', '<SP>', 'B_NNG

inputs : ['경', '제', '적', '<SP>', '측', '면', '에', '서', '<SP>',
predicts : ['B_NNG', 'I', 'B_XSN', '<SP>', 'B_NNG', 'I', 'B_JKB', 'I
labels : ['B_NNG', 'I', 'B_XSN', '<SP>', 'B_NNG', 'I', 'B_JKB', 'I',

inputs : ['그', '의', '<SP>', '사', '상', '은', '<SP>', '현', '실',
predicts : ['B_NP', 'B_JKG', '<SP>', 'B_NNG', 'I', 'B_JX', '<SP>', '
labels : ['B_NP', 'B_JKG', '<SP>', 'B_NNG', 'I', 'B_JX', '<SP>', 'B_
```

실습: Seq. Labeling for POS Tagging

과제

배점: 10점 (출력형태를 두번째처럼 변경)

```
inputs : ['방', '바', '닥', '에', '<SP>', '주', '저', '앉', '았', '습', '니', '다', '.']
predicts : ['B_NNG', 'I', 'I', 'B_JKB', '<SP>', 'B_VV', 'I', 'I', 'B_EP', 'B_EF', 'I', 'I', 'B_SF']
labels : ['B_NNG', 'I', 'I', 'B_JKB', '<SP>', 'B_VV', 'I', 'I', 'B_EP', 'B_EF', 'I', 'I', 'B_SF']

inputs : ['그', '즈', '음', '<SP>', '그', '들', '이', '<SP>', '워', '싱', '턴', '을', '<SP>', '거', '처
predicts : ['B_NNG', 'I', 'I', '<SP>', 'B_NP', 'B_XSN', 'B_JKS', '<SP>', 'B_NNP', 'I', 'I', 'B_JKO', '<
labels : ['B_NNG', 'I', 'I', '<SP>', 'B_NP', 'B_XSN', 'B_JKS', '<SP>', 'B_NNP', 'I', 'I', 'B_JKO', '<SP

inputs : ['이', '것', '이', '<SP>', '소', '위', '<SP>', '역', '사', '적', '<SP>', '일', '회', '성', 'O
predicts : ['B_NP', 'I', 'B_JKS', '<SP>', 'B_MAG', 'I', '<SP>', 'B_NNG', 'I', 'B_XSN', '<SP>', 'B_NR',
labels : ['B_NP', 'I', 'B_JKS', '<SP>', 'B_MAG', 'I', '<SP>', 'B_NNG', 'I', 'B_XSN', '<SP>', 'B_NR', 'B

inputs : ['-', '청', '와', '대', '와', '의', '<SP>', '갈', '등', '이', '<SP>', '예', '상', '되', '는',
predicts : ['B_SS', 'B_NNP', 'I', 'I', 'B_JKB', 'B_JKG', '<SP>', 'B_NNG', 'I', 'B_JKS', '<SP>', 'B_NNG'
labels : ['B_SS', 'B_NNP', 'I', 'I', 'B_JKB', 'B_JKG', '<SP>', 'B_NNG', 'I', 'B_JKS', '<SP>', 'B_NNG',
```

```
predict_sentence : 방바닥/NNG 에/JKB <SP> 주저앉/VV 았/EP 습니다/EF ./SF
correct_sentence : 방바닥/NNG 에/JKB <SP> 주저앉/VV 았/EP 습니다/EF ./SF

predict_sentence : 그즈음/NNG <SP> 그/NP 들/XSN 이/JKS <SP> 워싱턴/NNP 을/JKO <SP> 거처/NNG 로/JKB <SP>
correct_sentence : 그즈음/NNG <SP> 그/NP 들/XSN 이/JKS <SP> 워싱턴/NNP 을/JKO <SP> 거처/NNG 로/JKB <SP>

predict_sentence : 이것/NP 이/JKS <SP> 소위/MAG <SP> 역사/NNG 적/XSN <SP> 일/NR 회/NNB 성/XSN 이/VCP 라
correct_sentence : 이것/NP 이/JKS <SP> 소위/MAG <SP> 역사/NNG 적/XSN <SP> 일/NR 회/NNB 성/XSN 이/VCP 라

predict_sentence : -/SS 청와대/NNP 와/JKB 의/JKG <SP> 갈등/NNG 이/JKS <SP> 예상/NNG 되/XSV 는데/EF ./SF
correct_sentence : -/SS 청와대/NNP 와/JKB 의/JKG <SP> 갈등/NNG 이/JKS <SP> 예상/NNG 되/XSV 는데/EF ./SF
```

질의응답

Q&A

Homepage: http://nlp.konkuk.ac.kr
E-mail: nlpdrkim@konkuk.ac.kr