한국어 용언의 활용 함수 (Korean conjugation)

11

한국어의 단어는 9 품사로 이뤄져 있습니다. 그 중 용언에 해당하는 형용사와 동사는 활용 (conjugation) 이 됩니다. 용언은 어간 (stem) 과 어미 (ending) 로 구성되어 있으며, 용언의 원형은 어간의 원형에 종결어미 '-다'가 결합된 형태입니다. 예를 들어 '하다'라는 동사는 '하/어간 + 다/어미'로 구성되어 있습니다. '-다'가 '-니까'와 같은 다른 어미로 치환되어 '하 + 니까'로 동사의 형태가 변할 수 있습니다. 이처럼 어간과 어미의 모양이 변하지 않으면서 **어미만 치환되는 활용을 용언의 규칙 활용**이라 합니다. 하지만 '-았어' 라는 어미를 '하/어간'과 결합하려면 어간의 뒷부분과 어미의 앞부분의 모양이 변합니다. '하 + 았어 -> 했어' 처럼 어간과 **어미의 형태가 변하는 경우를 용언의 불규칙 활용**이라 합니다. 이번 포스트에서는 **어간과 어미의 원형이 주어졌을 때 불규칙 활용이 된 용언을 만드는 conjugate 함수를 구현**합니다.

Lemmatization vs Conjugation

주어진 용언에서 어간과 어미의 원형을 찾는 작업을 lemmatization 이라 합니다. 반대로 어간과 어미의 원형이 주어졌을 때 적절한 모양으로 용언을 변형시키는 작업을 conjugation 이라 합니다.

활용은 규칙 활용과 불규칙 활용으로 나뉩니다. **규칙 활용**은 규칙에 따라 용언이 변하는 경우로, 영어에서는 과거형을 만들기 위해 '-ed' 라는 suffix 를 붙입니다. 한국어의 용언은 어간 (stem) 과 어미 (ending) 라는 형태소로 구성되는되, **어간의 형태는 변하지 않고 어미만 다른 어미로 교체**되는 경우입니다.

```
가다/동사 = 가/어간 + 다/어미
가니까/동사 = 가/어간 + 니까/어미
가라고/동사 = 가/어간 + 라고/어미
```

이 경우에는 **주어진 어간과 어미를 결합 (concatenation) 함으로써 활용된 용언**을 만들 수 있습니다.

Lemmatization vs Conjugation

불규칙 활용은 어간이나 어미의 형태가 변하는 활용입니다. 이 경우에는 문법 규칙을 고려하여 용언의 모습을 변화하여야 합니다.

```
갔어/동사 = 가/어간 + ㅆ어/어미
간거야/동사 = 가/어간 + ㄴ거야/어미
꺼줘/동사 = 끄/어간 + 어줘/어미
```

이전의 lemmatizer post 에서는 어간 원형 사전이 주어졌을 때, 용언의 어간과 어미의 원형 후보를 복원하는 함수를 구현하였습니다.

이번 포스트에서는 이를 응용하여 **어간과 어미의 원형이 주어졌을 때 이를 활용하는 conjugate 함수를 구현**합니다.

Ready for conjugate function

Conjugate 함수는 따로 사전을 지닐 필요가 없기 때문에 class 가 아닌 함수 형태로 구현합니다. 이번에도 **어간의 마지막 글자와 초/중/종성과 어미의 첫글자의 초/중/종성을 분해하여 살펴봐야 합니다.** 이를 위한 준비를 합니다.

한 가지 더, 어미는 반드시 empty 가 아닌 str 이 입력되어야 합니다. 이를 확인하기 위하여 assert ending 을 추가합니다.

Conjugation 은 lemmatization 보다 구현이 더 쉽습니다. 문법을 그대로 구현하면 됩니다.

Ready for conjugate function

```
from soynlp.hangle import compose, decompose

def conjugate(stem, ending):

    assert ending # ending must be inserted

    l_len = len(stem)
    l_last = decompose(stem[-1])
    l_last_ = stem[-1]
    r_first = decompose(ending[0])
    r_first_ = compose(r_first[0], r_first[1], ' ') if r_first[1] != ' ' else ending[0]

      candidates = set()
```

Types of conjugation

- ㄷ 불규칙 활용
- 르 불규칙 활용
- ㅂ 블규칙
- 어미의 첫글자가 종성일 경우 (-ㄴ, -ㄹ, -ㅂ, -ㅆ)
- ㅅ 불규칙 활용
- 우 불규칙
- 오 불규칙 활용 (가제, 본래는 규칙활용)
- ㅡ 탈락 불규칙 활용
- 거라, 너라 불규칙 활용
- 러 불규칙 활용
- 여 불규칙 활용
- ㅎ 불규칙 활용
- 규칙 활용

ㄷ 불규칙 활용

어간의 마지막 글자 종성이 'ㄷ' 이고 어미의 첫글자가 'ㅇ' 이면 (모음으로 시작하면) 'ㄷ' 이 'ㄹ' 로 바뀝니다.

```
깨닫 + 아 -> 깨달아
묻 + 었다 -> 물었다
```

```
if l_last[2] == 'ㄷ' and r_first[0] == 'ㅇ':
    l = stem[:-1] + compose(l_last[0], l_last[1], 'ㄹ')

    candidates.add(l + ending)
```

르 불규칙 활용

어간의 마지막 글자가 ‘르’ 이고 어미의 첫글자가 ‘-아/-어’이면 어간의 마지막 글자는 ‘ㄹ’ 로 변화하여 앞글자와 합쳐지고, 어미의 첫글자는 ‘-라/-러’로 바뀝니다.

```
구르 + 어 -> 굴러
들르 + 었다 -> 들렀다
```

어간의 마지막 글자가 ‘르’ 이고 어미의 첫글자가 ‘아/어’인지 확인합니다. 길이가 2 이상인 어간에 대해서만 어간의 축약이 가능하기 때문에 어간의 길이를 확인하는 l_len >= 2 를 추가합니다.

```
if (l_last_ == '르') and (r_first_ == '아' or r_first_ == '어') and
l_len >= 2:
    c0, c1, c2 = decompose(stem[-2])
    l = stem[:-2] + compose(c0, c1, 'ㄹ')
    r = compose('ㄹ', r_first[1], r_first[2]) + ending[1:]
      candidates.add(l + r)
```

ㅂ 불규칙

어간의 마지막 글자 종성이 'ㅂ' 이고 어미의 첫글자가 모음으로 시작하면 'ㅂ' 이 'ㅜ/ㅗ' 로 바뀝니다.

ㅂ 불규칙은 모음조화가 이뤄진 경우와 그렇지 않은 경우로 분류됩니다.
모음조화가 이뤄진 경우에는 어간의 마지막 글자의 중성과 어절의 첫글자의 중성이 모두 양성 혹은 음성 모음입니다.

```
더럽 + 어 -> 더러워
곱 + 아 -> 고와
```

모음조화가 이뤄지지 않은 예시입니다. '아름답다, 아니꼽다, 아깝다, 감미롭다'는 모음조화가 이뤄지지 않습니다.

```
아름답 + 아 -> 아름다워
아니꼽 + 아 -> 아니꼬워
```

ㅂ 불규칙

모음조화가 이뤄지지 않는 경우를 일반화 하기 위하여 어간의 길이가 2 이상이고 어간의 마지막 글자가 '-답, -곱, -깝, -롭'인 경우에는 어미의 첫글자의 중성을 '워'로 강제하였습니다.

```
if (l_last[2] == 'ㅂ') and (r_first_ == '어' or r_first_ == '아'):
    l = stem[:-1] + compose(l_last[0], l_last[1], ' ')
    if l_len >= 2 and (l_last_ == '답' or l_last_ == '곱' or l_last_ == '깝' or l_last_ == '롭'):
        c1 = '워'
    elif r_first[1] == 'ㅗ':
        c1 = '와'
    elif r_first[1] == 'ㅜ':
        c1 = '워'
    elif r_first_ == '어':
        c1 = '워'
    else: # r_first_ == '아'
        c1 = '와'
    r = compose('ㅇ', c1, r_first[2]) + ending[1:]

    candidates.add(l + r)
```

어미의 첫글자가 종성일 경우 (-ㄴ, -ㄹ, -ㅂ, -ㅆ)

어미 중에는 첫글자가 자음인 어미들이 있습니다. 혹은 자음 자체가 어미이기도 합니다. 어간의 종성이 없을 경우 이들은 어간의 받침으로 이용됩니다.

```
이 + ㅂ니다 -> 입니다
하 + ㅂ니다 -> 합니다
```

```
if l_last[2] == ' ' and r_first[1] == ' ' and (r_first[0] == 'ㄴ' or r_first[0] == 'ㄹ' or r_first[0] == 'ㅂ' or
r_first[0] == 'ㅆ'):
    l = stem[:-1] + compose(l_last[0], l_last[1], r_first[0])
    r = ending[1:]

    candidates.add(l + r)
```

ㅅ 불규칙 활용

어간의 종성이 'ㅅ' 이고 어미가 모음으로 시작하면 'ㅅ' 이 탈락합니다. 단, '벗다' 는 예외입니다.

```
낫 + 아 -> 나아
긋 + 어 -> 그어
선긋 + 어 -> 선그어
벗 + 어 -> 벗어
옷벗 + 어 -> 옷벗어
```

'옷벗다, 선긋다' 처럼 명사와 ㅅ 불규칙을 따르는 동사가 결합된 단어가 단일 형태소로 이용될 것을 고려하여 어간의 마지막 글자가 '벗'인지 확인하도록 구현합니다.

```
if (l_last[2] == 'ㅅ') and (r_first[0] == 'ㅇ'):
    if stem[-1] == '벗':
        l = stem
    else:
        l = stem[:-1] + compose(l_last[0], l_last[1], ' ')
     candidates.add(l + ending)
```

우 불규칙

어간의 중/종성이 ‘우’이고 어미의 첫글자가 ‘어’일 때 ‘ㅜ’ 가 탈락하는 활용입니다. ‘푸다’가 유일하며, 그 외에는 ‘ㅜ + ㅓ = ㅝ’로 규칙 활용이 됩니다.

```
푸 + 어갔어 -> 퍼갔어
주 + 어 -> 줘
주 + 었어 -> 줬어
```

이 역시 ‘물푸다’ 처럼 ‘푸다’와 결합된 동사가 사용될 수 있음을 고려하여 어간의 마지막 글자가 ‘푸’인지 확인하도록 구현합니다. 그 외에는 ‘ㅜ + ㅓ’ 형태인지 확인합니다.

```
if l_last[1] == 'ㅜ' and l_last[2] == ' ' and r_first[0] == 'ㅇ' and r_first[1] == 'ㅓ':
    if l_last_ == '푸':
        l = '퍼'
    else:
        l = stem[:-1] + compose(l_last[0], 'ㅝ', r_first[2])
    r = ending[1:]
     candidates.add(l + r)
```

오 불규칙 활용 (가제, 본래는 규칙활용)

어간의 중성/종성이 ‘ㅗ’, ‘ ‘ 이고 어미의 첫글자가 ‘아’이면 ‘ㅗ + ㅏ = ㅘ’에 의하여 어간의 마지막 글자의 중성이 ‘ㅘ’로 변합니다.

```
오 + 았어 -> 왔어
```

```
   if l_last[1] == 'ㅗ' and l_last[2] == ' ' and r_first[0] == 'ㅇ' and
r_first[1] == 'ㅏ':
        l = stem[:-1] + compose(l_last[0], 'ㅘ', r_first[2])
        r = ending[1:]

         candidates.add(l + r)
```

ㅡ 탈락 불규칙 활용

어간의 중성이 ‘ㅡ’ 이고 받침이 없고 어미가 ‘-아/-어’로 시작하면 ‘ㅡ’가 탈락합니다.

```
끄 + 었다 -> 껐다
트 + 었어 -> 텄어
```

‘끄다, 크다, 트다’가 결합된 다른 용언을 고려하여 어간의 마지막 글자가 ‘끄, 크, 트’인지 확인합니다.

```
if (l_last_ == '끄' or l_last_ == '크' or l_last_ == '트') and (r_first[0] == 'ㅇ') and (r_first[1] == 'ㅓ'):
    l = stem[:-1] + compose(l_last[0], r_first[1], r_first[2])
    r = ending[1:]
      candidates.add(l + r)
```

거라, 너라 불규칙 활용

명령형 어미 '-아라/-어라'가 '-거라/-너라'로 바뀌는 활용입니다.

```
가 + 아라 -> 가거라
오 + 어라 -> 오너라
```

Lemmatizer 에서는 '-거라/-너라'를 어미로 취급하면 규칙 활용으로 생각할 수 있습니다. 하지만 conjugation 에서는 이를 구현해야 합니다. '-어라니까' 와 같이 '-어라'가 포함된 어미를 고려하여 어미의 앞부분 두 글자가 '어라/아라'인지 확인합니다.

```
if ending[:2] == '어라' or ending[:2] == '아라':
    if l_last[1] == 'ㅏ':
        r = '거' + ending[1:]
    elif l_last[1] == 'ㅗ':
        r = '너' + ending[1:]
    else:
        r = ending
     candidates.add(stem + r)
```

러 불규칙 활용

어간의 마지막 글자가 ‘르’이고 어미의 첫글자가 ‘-어’ 일 때 ‘-어’가 ‘-러’로 바뀝니다.
이때도 ‘러’를 포함하는 형태소를 어미로 생각하면 규칙 활용에 해당합니다.

```
이르 + 어 -> 이르러
푸르 + 어 -> 푸르러
```

```
if l_last_ == '르' and r_first[0] == 'ㅇ' and r_first[1] == 'ㅓ':
    r = compose('ㄹ', r_first[1], r_first[2]) + ending[1:]

    candidates.add(stem + r)
```

여 불규칙 활용

‘-하다’로 끝나는 용언에서 어미의 첫글자 ‘-아’가 ‘-여’로 바뀌는 활용입니다.

```
아니하 + 았다 -> 아니하였다
영원하 + 아 -> 영원하여
```

어간의 마지막 글자가 ‘하’이고 어미의 초성이 ‘ㅇ’, 중성이 ‘ㅏ/ㅕ’인지 확인합니다. 또한 어간의 마지막 글자 ‘하’와 어미의 초/중성 ‘아’가 결합되어 ‘해’로 바뀌기도 합니다.

```
if l_last_ == '하' and r_first[0] == 'ㅇ' and (r_first[1] == 'ㅏ' or r_first[1] == 'ㅕ'):
    # case 1
    r = compose(r_first[0], 'ㅕ', r_first[2]) + ending[1:]
    candidates.add(stem + r)
    # case 2
    l = stem[:-1] + compose('ㅎ', 'ㅐ', r_first[2])
    r = ending[1:]
      candidates.add(l + r)
```

ㅎ 불규칙 활용

어간의 마지막 글자의 종성이 'ㅎ'일 경우 'ㅎ'이 탈락하거나 축약되는 활용입니다.
어간의 종성이 'ㅎ'인 형용사 중에서 '좋다'를 제외한 모든 형용사에서 발생합니다.

‘ㅎ’ 탈락

‘ㅎ’이 탈락하는 경우입니다.

```
파랗 + 면 -> 파라면
동그랗 + ㄴ -> 동그란
```

어미의 첫글자가 자음인 경우만 예외적으로 확인합니다.

```
if l_last[2] == 'ㅎ' and l_last_ != '좋' and not (r_first[1] == 'ㅏ' or r_first[1] == 'ㅓ'):
    if r_first[1] == ' ':
        l = l = stem[:-1] + compose(l_last[0], l_last[1], r_first[0])
    else:
        l = stem[:-1] + compose(l_last[0], l_last[1], ' ')
    if r_first_ == '으':
        r = ending[1:]
    elif r_first[1] == ' ':
        r = ''
    else:
        r = ending
     candidates.add(l + r)
```

‘ㅎ’ + ‘ㅏ/ㅓ’ -> ‘ㅐ/ㅔ’

어간의 마지막 글자의 종성 ‘ㅎ’와 어미의 첫글자 ‘ㅏ/ㅓ’가 합쳐져 ‘ㅐ/ㅔ’로 축약되는 경우입니다.

```
파랗 + 았다 -> 파랬다
그렇 + 아 -> 그래
시퍼렇 + 었다 -> 시퍼렜다
```

```
if l_last[2] == 'ㅎ' and l_last_ != '좋' and (r_first[1] == 'ㅏ' or r_first[1] == 'ㅓ'):
    l = stem[:-1] + compose(l_last[0], 'ㅐ' if r_first[1] == 'ㅏ' else 'ㅔ', r_first[2])
    r = ending[1:]

    candidates.add(l + r)
```

‘ㅎ + 네’ 불규칙

어간의 마지막 글자의 종성 ‘ㅎ’과 어미의 첫글자의 초/중성 ‘ㄴ’, ‘네’가 만날 경우 ‘ㅎ’이 탈락하기도 유지되기도 합니다. 맞춤법 개정에 의하여 둘 모두 문법을 따르는 표현입니다.

```
그렇 + 네 -> 그렇네 / 그러네
노랗 + 네요 -> 노랗네요 / 노라네요
```

```
if l_last[2] == 'ㅎ' and r_first[0] == 'ㄴ' and r_first[1] != ' ':

    candidates.add(stem + ending)
```

규칙 활용

위 경우를 모두 확인하였는데 **candidates 가 empty set 이라면 이는 불규칙 활용 문법이 적용되지 않는, 즉 규칙 활용을 따르는 어간과 어미라는 의미**입니다.
Concatenation 한 단어를 candidates 에 추가합니다.

단, 어미의 첫글자가 자음이 아닌 경우에만 concatenation 을 합니다.

```
if not candidates and r_first[1] != ' ':

    candidates.add(stem + ending)
```

구현된 conjugate 함수

어간과 어미가 주어졌을 때 용언 활용을 하는 함수를 구현하였습니다.

구현체는 soynlp.lemmatizer._conjugation.py 에 구현되어 있습니다.

테스트 코드 및 결과

용언의 어간과 어미가 주어졌을 때 이를 활용하는 테스트 함수 및 결과입니다.

```
testset = [
    ('깨닫', '아'), # ㄷ 불규칙
    ('구르', '어'), ('구르', '었다'), # 르 불규칙
    ('덥', '어'), ('줍', '어'), ('곱', '아'), ('곱', '어'), ('곱', '아서'),  # ㅂ 불규칙 모음조화
    ('아름답', '았다'), ('아니꼽', '어서'), ('아깝', '아서'), ('아깝', '어서'), ('감미롭', '아서'), # ㅂ 불규칙 모음조화가 깨진 경우
    ('이', 'ㅂ니다'), ('이', 'ㄹ지라도'), ('이', 'ㄴ'), ('이', '쓰다'), # 어미의 첫글자가 초성일 경우
    ('벗', '어서'), ('긋', '어서'), ('긋', '었어'), ('낫', '아야지'), # ㅅ 불규칙
    ('푸', '어'), ('주', '어'), ('주', '었다'), # ㅜ 불규칙
    ('오', '았어'), ('사오', '았다'), ('돌아오', '았지용'), # ㅗ 규칙 활용
    ('끄', '었다'), ('끄', '어'), ('트', '었던건데'), ('들', '었다'),  # ㅡ 탈락 불규칙
    ('가', '아라'), ('삼가', '어라'), ('삼가', '아라니까'), ('돌아오', '아라'), # 거라/너라 불규칙
    ('이르', '어'), ('푸르', '어'), ('이르', '었다던'), # 러 불규칙
    ('아니하', '았다'), ('영원하', '었던'), # 여 불규칙
    ('파랗', '으면'), ('파랗', '면'), ('동그랗', 'ㄴ'), # ㅎ (탈락) 불규칙
    ('파랗', '았다'), ('시퍼렇', '었다'), # ㅎ (축약) 불규칙
    ('그렇', '네'), ('파랗', '네요'), # ㅎ + 네 불규칙
    ('좋', '아'), ('좋', '았어'), # ㅎ 불규칙 예외
    ('하', '았다'), ('하', '었다') # 여 불규칙 (2)
]

for stem, eomi in testset:

    print('{} + {} -> {}'.format(stem, eomi, conjugate(stem, eomi)))
```

깨닫 + 아 -> {'깨달아'}
구르 + 어 -> {'구르러', '굴러'}
구르 + 었다 -> {'구르렀다', '굴렀다'}
덥 + 어 -> {'더워'}
춥 + 어 -> {'추워'}
곱 + 아 -> {'고와'}
곱 + 어 -> {'고워'}
곱 + 아서 -> {'고와서'}
아름답 + 았다 -> {'아름다웠다'}
아니꼽 + 어서 -> {'아니꼬워서'}
아깝 + 아서 -> {'아까워서'}
아깝 + 어서 -> {'아까워서'}
감미롭 + 아서 -> {'감미로워서'}
이 + ㅂ니다 -> {'입니다'}
이 + ㄹ지라도 -> {'일지라도'}
이 + ㄴ -> {'인'}
이 + ㅆ다 -> {'있다'}
벗 + 어서 -> {'벗어서'}
긋 + 어서 -> {'그어서'}
긋 + 었어 -> {'그었어'}
낫 + 아야지 -> {'나아야지'}
푸 + 어 -> {'퍼'}
주 + 어 -> {'줘'}
주 + 었다 -> {'줬다'}
오 + 았어 -> {'왔어'}
사오 + 았다 -> {'사왔다'}
돌아오 + 았지용 -> {'돌아왔지용'}
끄 + 었다 -> {'껐다'}
끄 + 어 -> {'꺼'}
트 + 었던건데 -> {'텄던건데'}
들 + 었다 -> {'들었다'}
가 + 아라 -> {'가거라'}
삼가 + 어라 -> {'삼가거라'}
삼가 + 아라니까 -> {'삼가거라니까'}
돌아오 + 아라 -> {'돌아오너라', '돌아와라'}
이르 + 어 -> {'일러', '이르러'}
푸르 + 어 -> {'푸르러', '풀러'}
이르 + 었다던 -> {'일렀다던', '이르렀다던'}
아니하 + 았다 -> {'아니하였다', '아니했다'}
영원하 + 었던 -> {'영원하였던', '영원했던'}
파랗 + 으면 -> {'파라면'}
파랗 + 면 -> {'파라면'}
동그랗 + ㄴ -> {'동그란'}
파랗 + 았다 -> {'파랬다'}
시퍼렇 + 었다 -> {'시퍼렜다'}
그렇 + 네 -> {'그러네', '그렇네'}
파랗 + 네요 -> {'파랗네요', '파라네요'}
좋 + 아 -> {'좋아'}
좋 + 았어 -> {'좋았어'}
하 + 았다 -> {'했다', '하였다'}
하 + 었다 -> {'했다', '하였다'}