한국어 용언의 원형 복원 (Korean lemmatization)

10

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. 어미 부분이 다른 어미로 교체되기만 한 활용을 규칙활용이라 합니다. 반면 활용 도중 어간의 뒷부분 혹은 **어미의 앞부분의 형태가 변하는 활용을 불규칙 활용**이라 합니다. **불규칙 활용은 용언의 형태소 분석을 어렵게 하는 원인 중 하나**입니다. 이번 포스트에서는 **불규칙 활용의 경우를 유형화하여 어간과 어미의 원형 후보를 만드는 lemmatizer 의 candidate 함수를 구현**합니다.

Conjugation, Lemmatization, Stemming

용언의 활용과 관련된 단어들의 정의입니다. **활용 (conjutagion) 은 한 동사나 형용사인 용언이 더 정밀한 의미를 표현하기 위하여 다양한 형태로 변하는 현상**입니다. '가다/동사'는 '가고, 가니까, 갔었다' 처럼 변할 수 있다. 영어에서도 'act' 는 'acting, acts, acted' 처럼 변합니다.

활용은 규칙 활용과 불규칙 활용으로 나뉩니다. 규칙 활용은 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 교체되는 경우입니다.

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

Conjugation, Lemmatization, Stemming

이 경우에는 용언으로 구성된 어절을 L + R 구조로 분해하면 손쉽게 용언의 원형인 '가다'를 복원할 수 있습니다. '가다, 가니까, 가라고'는 모두 '가- + {-다, -니까, -라고}'의 형태로 string split 을 합니다.

불규칙 활용은 어간이나 어미의 형태가 변하는 활용입니다. 이때에는 **단순한 string split 만으로는 용언의 원형을 복원하기가 어렵습니**다. **불규칙 활용도 몇 가지의 문법 규칙이 있습니다. 이 규칙들을 이용하여 lemmatizer 를 만들 수 있습니다.**

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

Conjugation, Lemmatization, Stemming

한 단어는 원형 (canonical form) 과 활용되는 형 (surfacial form) 이 있습니다. 사전에 등제된 단어의 형태가 원형입니다. 한국어의 용언은 '어간 + -다/어미' 형태로 원형을 기술합니다. 영어에서는 명사도 단/복수에 따라 surfacial form 이 달라집니다. 'car'의 복수형은 'cars' 입니다.

Stemming 과 lemmatization 은 모두 단어의 canonical form 을 인식하기 위한 방법입니다. **둘의 차이는 stemming 은 규칙들로 이뤄진 string processing** 입니다. '단어의 끝부분 -ed 를 제거한다'는 규칙을 적용하면 -ed 로 끝나는 단어의 원형을 찾을 수 있습니다.

```
studies = studi + es
studying = study + ing
```

Conjugation, Lemmatization, Stemming

Lemmatization 은 단어의 원형 (lemma)으로 복원을 합니다.

```
studies = study + (-y / +i) + es
studying = study + ing
```

Lemmatization 을 위해서는 단어의 품사 추정이 **함께 이뤄져야 합**니다. 또한 데이터 분석에서 '꺼줘'와 '끈'을 모두 '끄다/동사'로 표현하기 위해서는 lemmatization 이 이뤄져야 합니다.

한국어 lemmatization

- 규칙 활용을 따르는 용언의 원형 복원
- 불규칙 활용을 따르는 용언의 원형 복원을 위한 준비
- ㄷ 불규칙 활용
- 르 불규칙 활용
- ㅂ 블규칙 (1)
- 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

한국어 lemmatization

한국어 용언의 불규칙 활용은 나무위키에 예제와 함께 잘 정리가 되어있습니다. 이를 Python 으로 구현합니다.

우리가 구현할 lemmatizer 는 한국어 어절의 L-R 구조를 이용합니다. L-R 구조에 대해서는 이전 포스트를 참고하세요. 예를 들어 우리가 다음과 같은 동사 원형 사전을 가지고 있을 때, 다음의 어절을 (L, R) 로 분해한 뒤, L 이 우리가 알고 있는 동사의 어간인지 확인합니다.

```
verb_dict = {'깨닫다'}
eojeol = '깨달아'

check(l='깨달아', r='')
check(l='깨달', r='아') # -> '깨닫' + '아'
check(l='깨', r='달아')
```

한국어 lemmatization

일단 lemmatizer 가 용언의 원형 사전을 이용하기 때문에 이를 class 형태로 만들면 좋습니다. 단어 word 가 입력되었을 때 이를 (L, R) 로 나눠 가능한 원형의 후보들을 출력하는 함수도 만들어 둡니다.

```
class Lemmatizer:
    def __init__(self, stems):
        self._stems = stems

    def is_stem(self, w):
        return w in self._stems

    def lemmatize(self, word):
        return None

    def candidates(self, word):
        candidates = set()
        for i in range(1, len(word)+1):
            l, r = word[:i], word[i:]
            candidates.update(self._candidates(l, r))
        return candidates

    def _candidates(self, l, r):
        candidates = set()
        # TODO
          return candidates
```

규칙 활용을 따르는 용언의 원형 복원

규칙 활용의 경우에는 L 자체가 어간의 원형이기 때문에 확인이 쉽습니다.

```
def _candidates(self, l, r):
    candidates = set()

    if self.is_stem(l):
          candidates.add((l, r))
```

불규칙 활용을 따르는 용언의 원형 복원을 위한 준비

‘불규칙 활용’이라는 단어는 의미가 잘 전달되지 않는 말 같습니다. **사실 불규칙 활용은 어간과 어미의 형태가 변하는 ‘문법 규칙’** 입니다. 에를 들어 ‘ㄷ불규칙 활용’은 어간의 ㄷ 받침과 어미의 모음이 만나 ㄷ 이 ㄹ 로 변하는 규칙이기 때문입니다. **규칙을 따르지 않는 불규칙은 ‘예외 경우’**라 합니다.

여하튼 **‘불규칙 활용’도 문법 규칙이 있습니다.** 이를 이용하여 용언의 어간과 어미를 인식하는 lemmatizer 를 만듭니다.

용언의 불규칙 활용은 어간과 어절이 만나는 부분에서 발생합니다. 주어진 어절을 (L, R) 로 나눈 뒤, L 의 끝부분과 R 의 첫부분이 불규칙 활용이 발생하는지 확인합니다. 이를 위하여 주어진 str 인 l 의 마지막 글자와 r 의 첫글자의 초/중/종성을 분해한 l_last, r_first 를 만듭니다. 그리고 l_last 의 종성 (받침)을 없엔 l_last_ 와 r 의 종성을 없엔 r_first_ 도 만듭니다.

불규칙 활용을 따르는 용언의 원형 복원을 위한 준비

```
l_last = decompose(l[-1])
l_last_ = compose(l_last[0], l_last[1], ' ')
r_first = decompose(r[0]) if r else ('', '', '')

r_first_ = compose(r_first[0], r_first[1], ' ') if r else ' '
```

구현된 lemmatizer candidate 함수

한 어절이 주어졌을 때 이로부터 가능한 용언의 어간과 어미의 원형 후보를 생성하는 lemmatizer 를 정리하면 아래와 같습니다.

위 코드와 테스트는 soynlp.lemmatizer 에 구현되어 있습니다.

테스트 코드 및 결과

용언의 어간 사전이 주어졌을 때 이를 이용하여 가능한 어간과 어미의 원형을 찾는 테스트를 수행합니다.

```
stems = {
    '깨닫', '가', # ㄷ 불규칙
    '구르', '들르', # 르 불규칙
    '더럽',  '곱', '감미롭', # ㅂ 불규칙 (1)
    '이', '하', '푸르', # # 어미의 첫글자가 종성일 경우
    '낫', '긋', '벗', # ㅅ 불규칙
    '푸', '주', '누', # 우 불규칙
    '오', # 오 불규칙 (가제, 규칙 활용 ㅗ + ㅏ = ㅘ)
    '끄', '트', # ㅡ 탈락 불규칙
    '파랗', '하얗', '그렇', '시퍼렇', '노랗', # ㅎ (탈락) 불규칙
    '다하', # 여 불규칙 활용 (2)
}

testset = [
    '깨달아', '가고', # ㄷ 불규칙
    '굴러', '구르라니까', '들러', '들렀다', # 르 불규칙
    '더러워서', '더럽다', '고와', '감미로워서',  # ㅂ 불규칙 (1)
    '입니다', '합니다', '합니까', '한답니다', '할껄', '있어요', '푸른', # 어미의 첫글자가 종성일 경우
    '나았어', '그어버려', '벗어던져',  # ㅅ 불규칙
    '퍼갔어', '줬습니다', '눴어', # 우 불규칙
    '왔다', # 오 불규칙
    '껐다', '껐어', '텄어', # ㅡ 탈락 불규칙
    '파란', '파라면', '하얀', '노란', # ㅎ (탈락) 불규칙
    '파랬다', '그래', '그랬다', '그랬지', '시퍼렜다', #ㅎ (축약) 불규칙
    '했다', '했었다', '다했다', # 여 불규칙활용 (2)
]

lemmatizer = Lemmatizer(stems = stems)

for word in testset:
    candidates = lemmatizer.candidates(word)


    print('{} : {}'.format(word, candidates))
```

테스트 코드 및 결과

결과는 아래와 같습니다. **‘하얀’의 경우에는 ‘하다’와 ‘하얗다’가 어간의 원형의 후보로 생성**됩니다. 물론 ‘-얀’이라는 어미는 존재하지 않기 때문에 ‘하얗다’가 정답입니다. 이 부분은 lemmatize 함수에서 구현할 부분입니다.

깨달아 : {('깨닫', '아')}
가고 : {('가', '고')}
굴러 : {('구르', '어')}
구르라니까 : {('구르', '라니까')}
들러 : {('들르', '어')}
들렀다 : {('들르', '었다')}
더러워서 : {('더럽', '어서')}
더럽다 : {('더럽', '다')}
고와 : {('곱', '아')}
감미로워서 : {('감미롭', '어서')}
입니다 : {('이', 'ㅂ니다')}
합니다 : {('하', 'ㅂ니다')}
합니까 : {('하', 'ㅂ니까')}
한답니다 : {('하', 'ㄴ답니다')}
할껄 : {('하', 'ㄹ껄')}
있어요 : {('이', 'ㅆ어요')}
푸른 : {('푸르', 'ㄴ'), ('푸', '른')}
나았어 : {('낫', '았어')}
그어버려 : {('긋', '어버려')}
벗어던져 : {('벗', '어던져')}
퍼갔어 : {('푸', '어갔어')}
줬습니다 : {('주', '었습니다')}
눴어 : {('누', '었어')}
왔다 : {('오', '았다')}
껐다 : {('끄', '었다')}
껐어 : {('끄', '었어')}
텼어 : {('트', '였어')}
파란 : {('파랗', 'ㄴ')}
파라면 : {('파랗', '면')}
하얀 : {('하얗', 'ㄴ'), ('하', '얀')}
노란 : {('노랗', 'ㄴ')}
파랬다 : {('파랗', '았다')}
그래 : {('그렇', '아')}
그랬다 : {('그렇', '았다')}
그랬지 : {('그렇', '았지')}
시퍼렜다 : {('시퍼렇', '었다')}
했다 : {('하', '았다')}
했었다 : {('하', '았었다')}
다했다 : {('다하', '았다')}

References

https://lovit.github.io/nlp/2018/06/07/lemmatizer/

나무위키: 용언 활용 https://en.wikipedia.org/wiki/Lemmatisation