

텍스트 세미나

ToBig's 8기 김민정

# Text Preprocessing with Python

텍스트 전처리

## Unit 00 | 커리큘럼

### 8주간의 텍스트 세미나

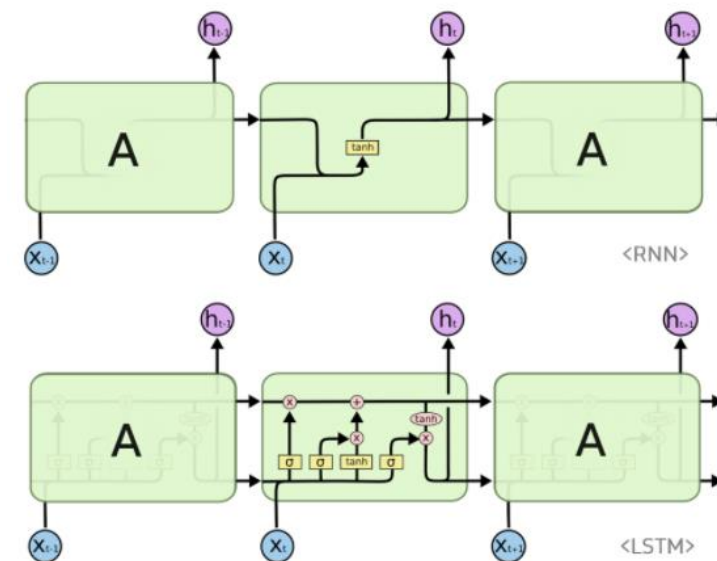
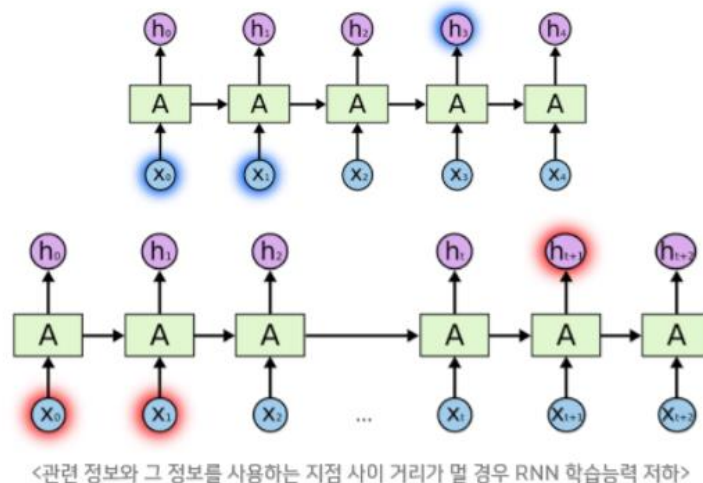
- 1주차 : 텍스트 전처리, 임베딩
- 2주차 : RNN, LSTM
- 3주차 : Seq2seq, GRU
- 4주차 : Text Classification
- 5~8주차 : 텍스트 또는 오디오 관련 논문
- Ex. Question Answering,
- Speech Recognition,
- Text to Speech
- Attention
- 발표자료 ?

1	분류	발표자	논문 제목
2	논문 스터디 1-1	박이삭	1. CNN을 이용한 음악 분류 2. 디컨볼루션을 이용한 CNN 이해 3. librosa 파이썬 음악 분석
3	논문 스터디 1-2	박현진	Generative adversarial nets
4	논문 스터디 2-1	이수빈	Deep Reinforcement Learning with Double Q-Learning
5	논문 스터디 2-2	곽대훈	Image Style Transfer Using Convolutional Neural Networks
6	논문 스터디 3-1	최수정	Jupyter notebook 사용하기
7	논문 스터디 3-2	류호성	You only look once: Unified, real-time object detection
8	논문 스터디 4-1	이유리	Deep Photo Style Transfer
9	논문 스터디 4-2	이다경	Git 사용법
10	논문 스터디 5-1	김은서	아마존AWS 사용법
11	논문 스터디 5-2	이현경	Fast R-CNN
12	논문 스터디 6-1	김민정	DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution,
13	논문 스터디 6-2	조혜진	Dynamic Routing Between Capsules
14			
15	분류	발표자	내용
16	분류기 만들기 1	이광록	이미지 수집/이미지 처리(PIL)/이미지 패킹(Pickle)/softmax 분류기 만들기(Tensorflow)
17	분류기 만들기 2	조양규	NN 분류기 제작 / 파라미터 튜닝 / Tensorboard 활용
18	분류기 만들기 3	최서현	CNN 분류기 제작 / 파라미터 튜닝 / Tensorboard 활용
19	분류기 만들기 4	김수현	양상블 / filter visualization
20	분류기 만들기 5	권문정	파인 튜닝 / fooling NN

## Unit 00 | 커리큘럼

### 8주간의 텍스트 세미나

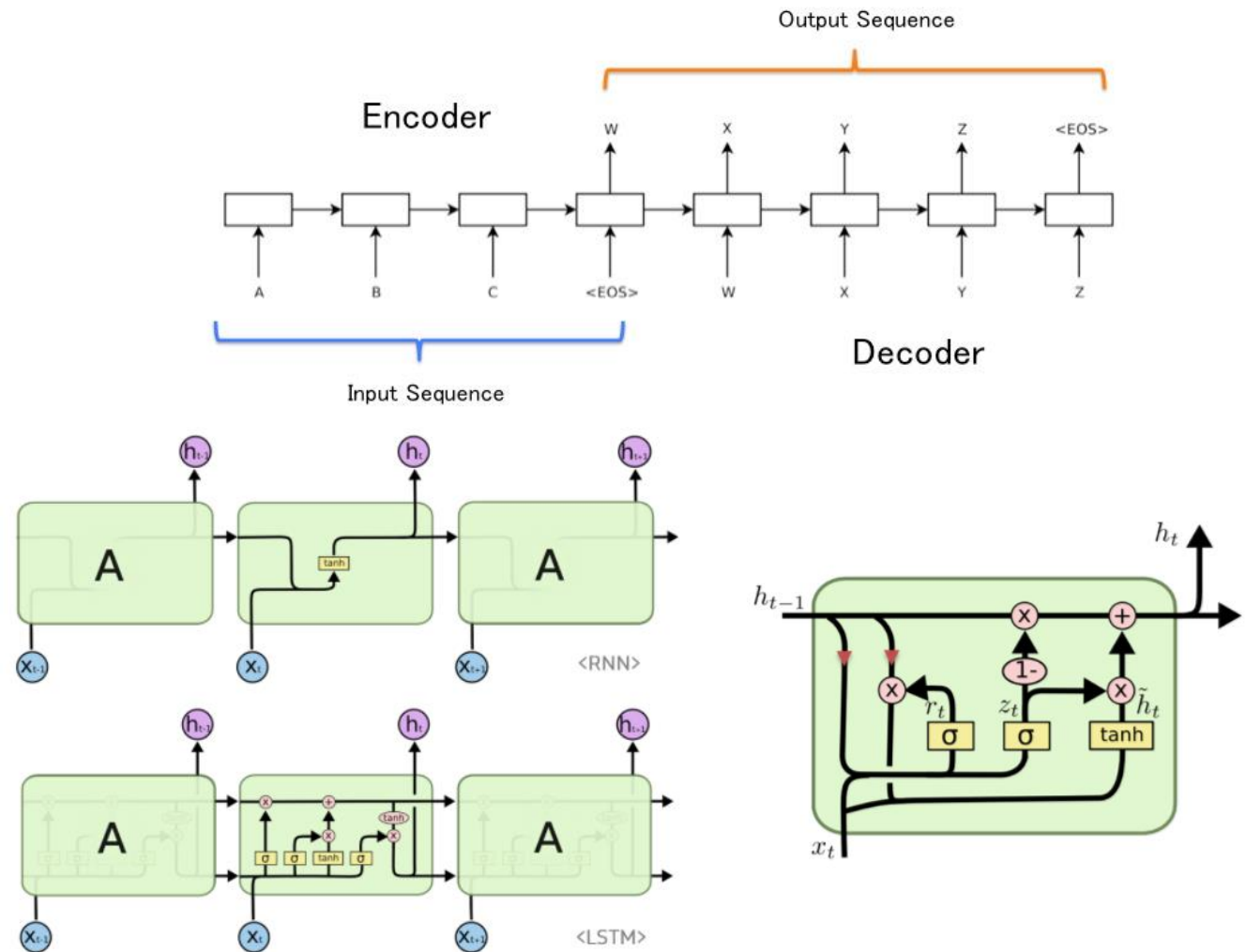
- 1주차 : 텍스트 전처리, 임베딩
- 2주차 : RNN, LSTM
- 3주차 : Seq2seq, GRU
- 4주차 : Text Classification
- 5~8주차 : 텍스트 또는 오디오 관련 논문
- Ex. Question Answering,
- Speech Recognition,
- Text to Speech
- Attention
- 발표자료 ?



# Unit 00 | 커리큘럼

## 8주간의 텍스트 세미나

- 1주차 : 텍스트 전처리, 임베딩
- 2주차 : RNN, LSTM
- 3주차 : Seq2seq, GRU
- 4주차 : Text Classification
- 5~8주차 : 텍스트 또는 오디오 관련 논문
- Ex. Question Answering,
- Speech Recognition,
- Text to Speech
- Attention
- 발표자료 ?



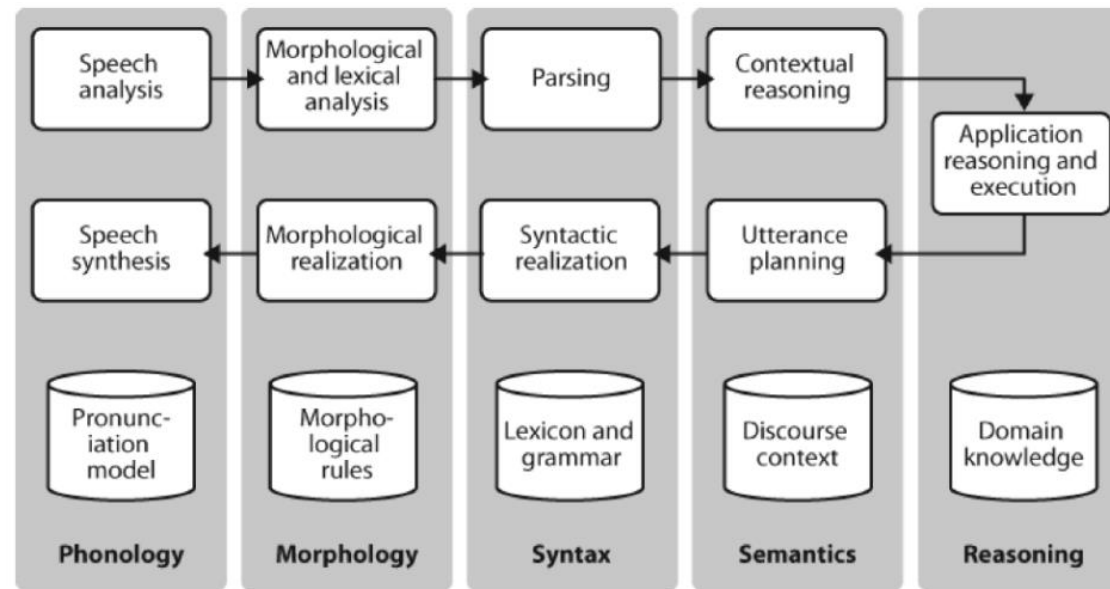
## Unit 00 | 커리큘럼

## 8주간의 텍스트 세미나

- 1주차 : 텍스트 전처리, 임베딩
- 2주차 : RNN, LSTM
- 3주차 : Seq2seq, GRU
- 4주차 : Text Classification
- 5~8주차 : 텍스트 또는 오디오 관련 논문
- Ex. Question Answering,
- Speech Recognition,
- Text to Speech
- Attention
- 발표자료 ?

## Unit 00 | 커리큘럼

### 자연어처리(NLP) 기본절차



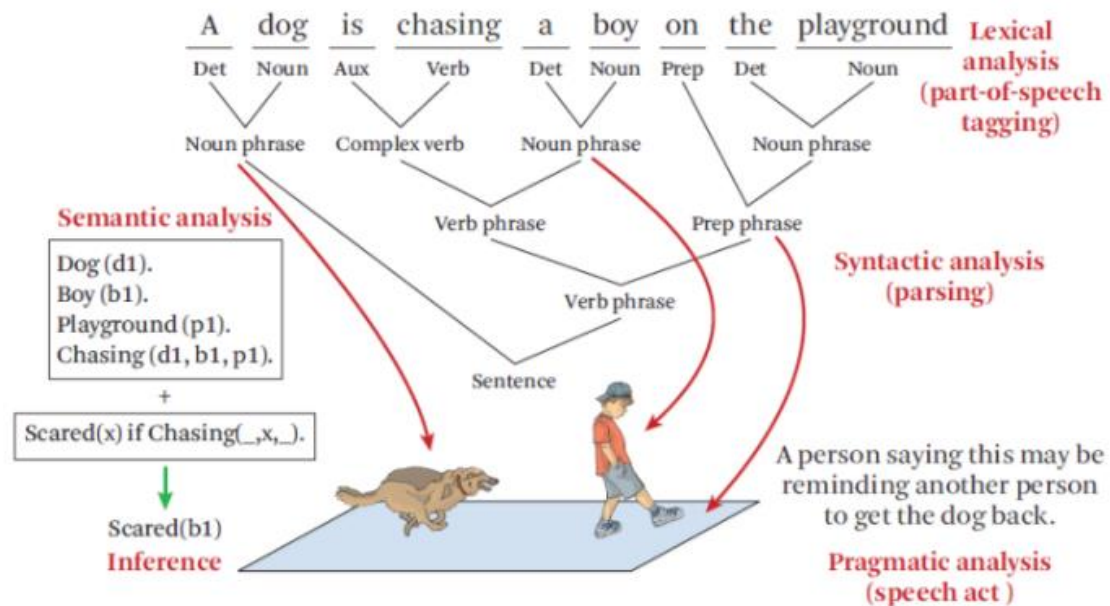
1. Phonology(음운론) : 말소리 연구 2. Morphology(형태론) : 단어와 형태소

3. Syntax(통사론) : 문법 4. Semantics : 맥락/담화

=> 음성 인식, 형태소 분석, 파싱(문장 문법적 구조 분석)등 !

# Unit 00 | 커리큘럼

## 자연어처리(NLP) 기본절차



1. Phonology(음운론) : 말소리 연구 2. Morphology(형태론) : 단어와 형태소

3. Syntax(통사론) : 문법 4. Semantics : 맥락/담화

=> 음성 인식, 형태소 분석, 파싱(문장 문법적 구조 분석)등 !

The diagram illustrates the NLP pipeline for Named Entity Recognition (NER) and Relation Extraction (RE). The pipeline starts with 'Input text (unstructured)', followed by 'preprocessing' (Sentence splitting, Tokenizing, Part-of-Speech (POS) tagging), 'Expanded Tagging with semantics' (Named entity recognition), and 'Chunking' (Parsing: Shallow, Full). The final step is 'Relation extraction' (Co-occurrence, Pattern-based, Rule-based, ML-based). The diagram also shows the NLP process for the sentence 'IL-8 recognizes and activates CXCR1...', including tokenization, POS tagging, and a parse tree.

**Input text (unstructured)**

**preprocessing**

- Sentence splitting
- Tokenizing
- Part-of-Speech (POS) tagging

**Expanded Tagging with semantics**

**Named entity recognition**

**Chunking**

**Parsing**

- Shallow
- Full

**Relation extraction**

- Co-occurrence
- Pattern-based
- Rule-based
- ML-based

**NLP process for the sentence: IL-8 recognizes and activates CXCR1...**

...IL-8 recognizes and activates CXCR1...

IL-8 recognizes and activates CXCR1

IL-8\_recognizes\_and\_activates\_CXCR1

IL-8/NNP recognizes/VBZ and/CC activates/VBZ CXCR1/NNP

**IL-8, CXCR1**

**Parse tree:**

```

    graph TD
      S[s] --- NP1[NP]
      S --- VP[VP]
      S --- NP2[NP]
      NP1 --- NNP1[NNP]
      VP --- VBZ1[VBZ]
      VP --- CC[CC]
      VP --- VBZ2[VBZ]
      NP2 --- NNP2[NNP]
      NNP1 --- IL8[IL-8]
      VBZ1 --- recognizes[recognizes]
      CC --- and[and]
      VBZ2 --- activates[activates]
      NNP2 --- CXCR1[CXCR1]
  
```

IL-8 recognizes and activates CXCR1

**Recognize (IL-8, CXCR1)**

**Activate (IL-8, CXCR1)**



# contents

---

Unit 01 | 정규표현식 복습

---

Unit 02 | Tokenizing

---

Unit 03 | POS-tagging

---

Unit 04 | Parsing

---

Unit 05 | 그 외

---

## Unit 01 | 정규표현식 복습

>>> Import re

- 정규표현식 : 특정한 규칙을 가진 문자열 집합을 표현하는데 사용하는 형식 언어
- re : 파이썬에서 정규표현식을 지원하는 모듈
- 대표적 메소드에는 match, search, findall, split, sub 등이 있다

match()	문자열의 처음부터 정규식과 매치되는지 조사한다.
search()	문자열 전체를 검색하여 정규식과 매치되는지 조사한다.
findall()	정규식과 매치되는 모든 라인의 문자열(substring)을 리스트로 리턴한다
finditer()	정규식과 매치되는 모든 라인의 문자열(substring)을 iterator 객체로 리턴한다
sub()	정규식과 매치되면 변경시킴
split()	매칭되면 패턴별로 쪼개서 리턴

# Unit 01 | 정규표현식 복습

정규표현식	표현	설명
$\wedge x$		문자열이 x로 시작합니다.
$x\$$		문자열이 x로 끝납니다.
$.x$		임의의 한 문자를 표현합니다. (x가 마지막으로 끝납니다.)
$x+$		x가 1번이상 반복합니다.
$x?$		x가 존재하거나 존재하지 않습니다.
$x^*$		x가 0번이상 반복합니다.
$x y$		x 또는 y를 찾습니다. (or연산자를 의미합니다.)
$(x)$		( )안의 내용을 캡처하며, 그룹화 합니다.
$(x)(y)$		그룹화 할 때, 자동으로 앞에서부터 1번부터 그룹 번호를 부여해서 캡처합니다. 결과값에 그룹화한 Data가 배열 형식으로 그룹번호 순서대로 들어갑니다.
$(x)(?:y)$		캡처하지 않는 그룹을 생성할 경우 ?를 사용합니다. 결과값 배열에 캡처하지 않는 그룹은 들어가지 않습니다.
$x\{n\}$		x를 n번 반복한 문자를 찾습니다.
$x\{n, \}$		x를 n번이상 반복한 문자를 찾습니다.
$x\{n, m\}$		x를 n번이상 m번이하 반복한 문자를 찾습니다.

정규표현식	표현	설명
$[xy]$		x,y중 하나를 찾습니다.
$[^xy]$		x,y를 제외하고 문자 하나를 찾습니다. (문자 클래스 내의 ^는 not을 의미합니다.)
$[x-z]$		x~z 사이의 문자중 하나를 찾습니다.
$\wedge\wedge$		^(특수문자)를 식에 문자 자체로 포함합니다. (escape)
$\Wb$		문자와 공백사이의 문자를 찾습니다.
$\WB$		문자와 공백사이가 아닌 값을 찾습니다.
$\Wd$		숫자를 찾습니다.
$\WD$		숫자가 아닌 값을 찾습니다.
$\Ws$		공백문자를 찾습니다.
$\WS$		공백이 아닌 문자를 찾습니다.
$\Wt$		Tab 문자를 찾습니다.
$\Wv$		Vertical Tab 문자를 찾습니다.
$\Ww$		알파벳 + 숫자 + _ 를 찾습니다.
$\WW$		알파벳 + 숫자 + _ 을 제외한 모든 문자를 찾습니다.

## Unit 01 | 정규표현식 복습

## 메타문자

: 원래 그 문자가 가진 뜻이 아닌 특별한 용도로 사용되는 문자

. ^ \$ \* + ? { } [ ] \ | ( )

## Unit 01 | 정규표현식 복습

### 메타문자

. ^ \$ \* + ? { } [ ] \ | ( )

: [] 사이의 문자들과 매치

: 정규 표현식이 [abc] -> a, b, c 중 한 개의 문자와 매치

검사할 문자열	정규 표현식	결과
a	[abc]	일치
bef <b>a</b> re	[abc]	일치
dude	[abc]	불일치

## Unit 01 | 정규표현식 복습

### 메타문자

. ^ \$ \* + ? { } [ ] \ | ( )

: [] 사이의 문자들과 매치

: [] 안에 - (하이픈) 사용하면, 문자 사이의 범위 의미

Ex. [a-c]는 [abc]

[0-5]는 [012345]

[a-zA-Z] : 알파벳 모두

[0-9] : 숫자

## Unit 01 | 정규표현식 복습

### 메타문자

. ^ \$ \* + ? { } [ ] \ | ( )

시작 끝

## Unit 01 | 정규표현식 복습

### 메타문자

. ^ \$ \* + ? { } [ ] \ | ( )

검사할 문자열

정규 표현식

결과

ahelloasd

hello

패턴과 일치

ahelloasd

^hello

패턴과 불일치  
(hello로 시작하지 않음)

applepi

^a

패턴과 일치

edcan

^a

패턴과 불일치  
(a로 시작하지 않음)



## Unit 01 | 정규표현식 복습

### 메타문자

. ^ \$ \* + ? { } [ ] \w | ( )

검사할 문자열

정규 표현식

결과

sun**ringo**

in

패턴과 일치

sun**ringo**

in\$

패턴과 불일치  
(in으로 끝나지 않음)

sun**rin**

in\$

패턴과 일치

hansub**in**

in\$

패턴과 일치

## Unit 01 | 정규표현식 복습

### 메타문자

. ^ \$ \* + ? { } [ ] \ | ( )

검사할 문자열

정규 표현식

결과

sunringo

`^in$`

패턴과 불일치

subin

`^in$`

패턴과 불일치

in

`^in$`

패턴과 일치

injeong

`^in$`

패턴과 불일치

## Unit 01 | 정규표현식 복습

### 메타문자

. ^ \$ \* + ? { } [ ] \ | ( )

검사할 문자열

정규 표현식

결과

"

'^\$'

패턴과 일치

'blabla'

'^\$'

패턴과 불일치

"

"

패턴과 일치

'blabla'

"

패턴과 일치

## Unit 01 | 정규표현식 복습

### 주의

정규표현식	표현	설명
<code>^x</code>		문자열이 x로 시작합니다.
<code>[^xy]</code>		x,y를 제외하고 문자 하나를 찾습니다. (문자 클래스 내의 ^는 not을 의미합니다.)

```
In [25]: # 정규표현식을 사용해서 특수문자를 제거
import re
# 소문자와 대문자가 아닌 것은 공백으로 대체한다.
letters_only = re.sub('[^a-zA-Z]', ' ', example1.get_text())
letters_only[:700]
```

시작을 의미하는 ^과 문자클래스[] 내부의 ^(not) 혼동하지 말자 !

## Unit 01 | 정규표현식 복습

### 메타문자

. ^ \$ \* + ? { } [ ] ~~W~~ | ( )

`\d` - 숫자(0~9).

`\D` - 숫자가 **아닌 것**.

`\s` - whitespace 문자(공백, \t, \n, \r, \f, \v).

`\S` - whitespace 문자가 **아닌 것**.

`\w` - 알파벳 문자 또는 숫자(0~9, a~z, A~Z).

`\W` - 알파벳 문자 또는 숫자가 **아닌 것**.

: 다른 문자와 조합되어 정해지지 않은 특정 종류의 문자자리를 나타냄

## Unit 01 | 정규표현식 복습

### 메타문자

. ^ \$ \* + ? { } [ ] ~~W~~ | ( )

검사할 문자열

정규 표현식

결과

35 (혹은 35)

`\d`

패턴과 일치

35

`^\d$`

패턴과 불일치

5km이다

`^\dkm`

패턴과 일치

53km이다

`^\dkm`

패턴과 불일치

## Unit 01 | 정규표현식 복습

### 메타문자

. ^ \$ \* + ? { } [ ] \ | ( )

: '0번 이상' 반복

정규식	문자열	Match 여부	설명
<code>ca*t</code>	ct	Yes	"a"가 0번 반복되어 매치
<code>ca*t</code>	cat	Yes	"a"가 0번 이상 반복되어 매치 (1번 반복)
<code>ca*t</code>	caaat	Yes	"a"가 0번 이상 반복되어 매치 (3번 반복)

## Unit 01 | 정규표현식 복습

### 메타문자

. ^ \$ \* + ? { } [ ] \ | ( )

: '1번 이상' 반복

정규식	문자열	Match 여부	설명
<code>ca+t</code>	ct	No	"a"가 0번 반복되어 매치되지 않음
<code>ca+t</code>	cat	Yes	"a"가 1번 이상 반복되어 매치 (1번 반복)
<code>ca+t</code>	caaat	Yes	"a"가 1번 이상 반복되어 매치 (3번 반복)



## Unit 01 | 정규표현식 복습

### 메타문자

. ^ \$ \* + ? { } [ ] \w | ( )

: '1번 이상' 반복

검사할 문자열

정규 표현식

결과

accccdddqr

`^a+c+d+`

패턴과 일치

'시작 -> a문자 -> c문자 -> d문자'으로 생각하면 된다

35

`^\d+$`

패턴과 일치

5km이다

`^\d+km`

패턴과 일치

53km이다

`^\d+km`

패턴과 불일치

## Unit 01 | 정규표현식 복습

### 메타문자

. ^ \$ \* + ? { } [ ] \ | ( )

{m} : 반드시 m번 반복

정규식	문자열	Match 여부	설명
<code>ca{2}t</code>	cat	No	"a"가 1번만 반복되어 매치되지 않음
<code>ca{2}t</code>	caat	Yes	"a"가 2번 반복되어 매치

{m,n} : m~n회 반복

정규식	문자열	Match 여부	설명
<code>ca{2,5}t</code>	cat	No	"a"가 1번만 반복되어 매치되지 않음
<code>ca{2,5}t</code>	caat	Yes	"a"가 2번 반복되어 매치
<code>ca{2,5}t</code>	caaaaat	Yes	"a"가 5번 반복되어 매치

## Unit 01 | 정규표현식 복습

### 메타문자

. ^ \$ \* + ? { } [ ] \ | ( )

{m} : 반드시 m번 반복

정규식	문자열	Match 여부	설명
<code>ca{2}t</code>	cat	No	"a"가 1번만 반복되어 매치되지 않음
<code>ca{2}t</code>	caat	Yes	"a"가 2번 반복되어 매치

{m,n} : m~n회 반복

정규식	문자열	Match 여부	설명
<code>ca{2,5}t</code>	cat	No	"a"가 1번만 반복되어 매치되지 않음
<code>ca{2,5}t</code>	caat	Yes	"a"가 2번 반복되어 매치
<code>ca{2,5}t</code>	caaaaat	Yes	"a"가 5번 반복되어 매치

? 는 {0,1}과 같음

-> 있어도 되고 없어도 됨~

정규식	문자열	Match 여부	설명
<code>ab?c</code>	abc	Yes	"b"가 1번 사용되어 매치
<code>ab?c</code>	ac	Yes	"b"가 0번 사용되어 매치

## Unit 01 | 정규표현식 복습

### 메타문자

. ^ \$ \* + ? { } [ ] \ | ( )

: 'OR'

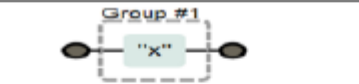
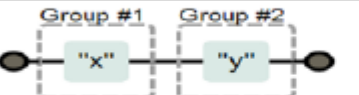

A|B면 A 또는 B

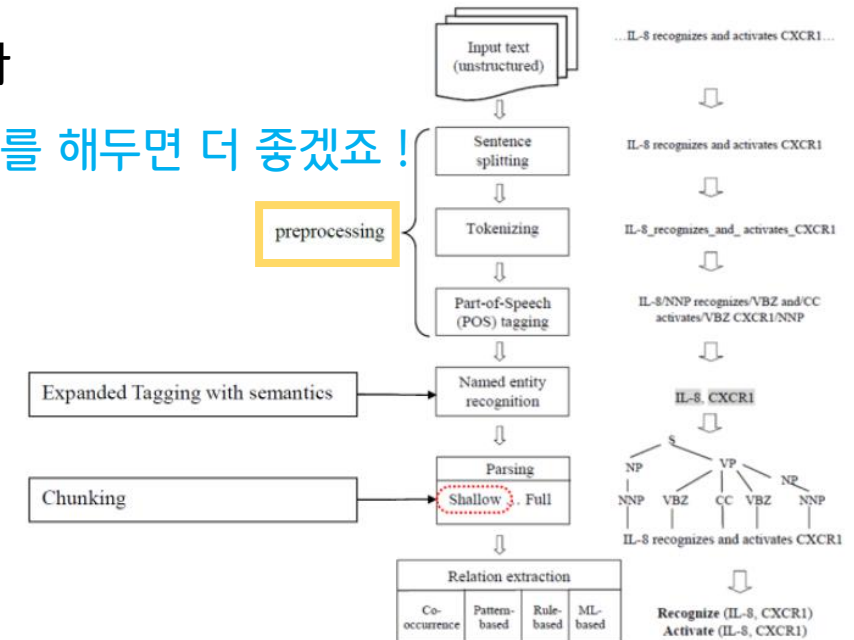
## Unit 01 | 정규표현식 복습

### 메타문자

. ^ \$ \* + ? { } [ ] \ | ( )

: Grouping

(x)		( )안의 내용을 캡처하며, 그룹화 합니다.
(x)(y)		그룹화 할 때, 자동으로 앞에서부터 1번부터 그룹 번호를 부여해서 캡처합니다. 결과값에 그룹화한 Data가 배열 형식으로 그룹번호 순서대로 들어갑니다.
(x)(?:y)		캡처하지 않는 그룹을 생성할 경우 ?:를 사용합니다. 결과값 배열에 캡처하지 않는 그룹은 들어가지 않습니다.



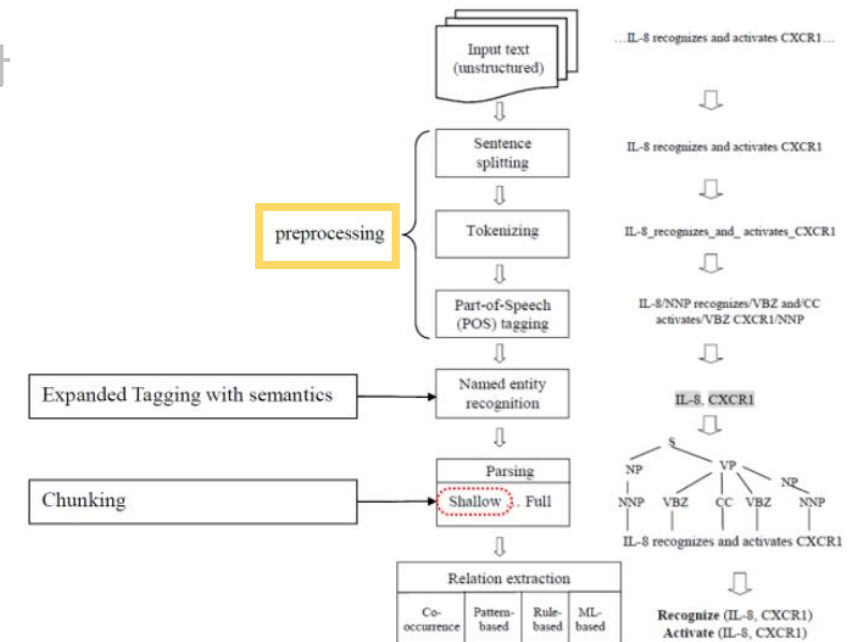
## Unit 02 | Tokenizing

### 1. Sentence Splitting

- corpus를 문장 단위로 끊어서 입력
- .,!? 등의 기준으로
- LDA 같은 특정 알고리즘이나 방법론의 경우 splitting 꼭 하지 않아도 된다

### 2. Tokenizing

- 말그대로 token으로 나누기
- Token : 의미를 가지는 문자열
- 형태소(뜻을 가진 최소 단위) + 단어(자립하여 쓸 수 있는 최소단위)
- 영어 : 공백 만으로도 충분, 중국어 : 띄어쓰기 거의 하지 않아 난제



## Unit 02 | Tokenizing

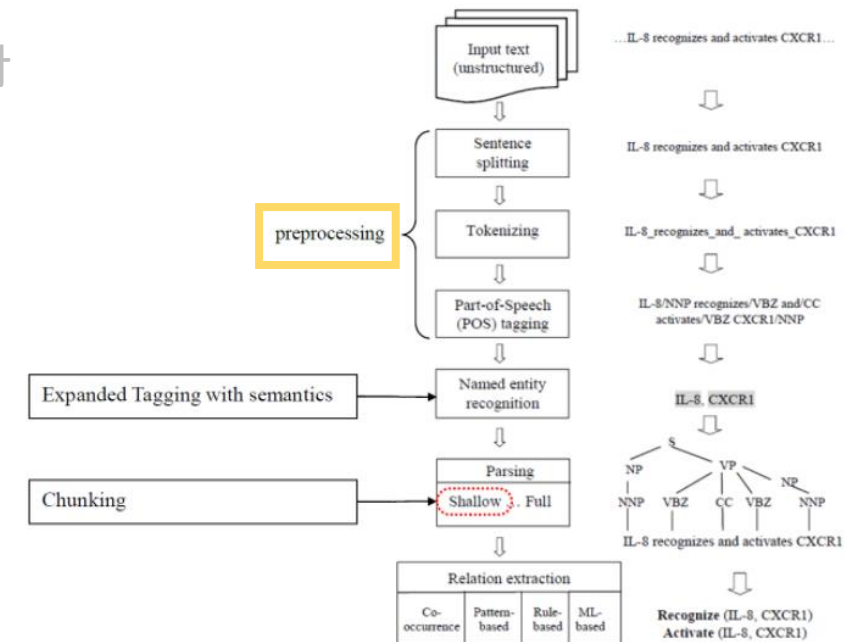
### 1. Sentence Splitting

- corpus를 문장 단위로 끊어서 입력
- .,! ? 등의 기준으로
- LDA 같은 특정 알고리즘이나 방법론의 경우 splitting 꼭 하지 않아도 된다

### 2. Tokenizing

- 말그대로 token으로 나누기
- Token : 의미를 가지는 문자열
- 형태소(뜻을 가진 최소 단위) + 단어(자립하여 쓸 수 있는 최소단위)
- 영어 : 공백 만으로도 충분, 중국어 : 띄어쓰기 거의 하지 않아 난제

### 3. POS-tagging





Mrs. Clinton previously worked for Mr. Obama, but she is now distancing herself from him.

Mrs. Clinton previously worked for Mr. Obama, but she is now distancing herself from him.

Mrs. Clinton previously worked for Mr. Obama, but she is now distancing herself from him.

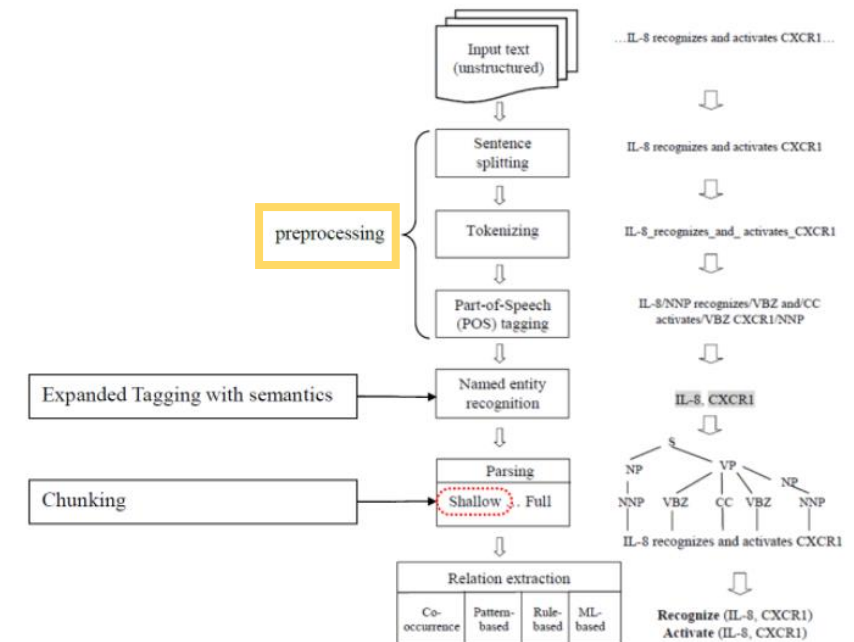
The diagram illustrates coreference chains for the sentence "Mrs. Clinton previously worked for Mr. Obama, but she is now distancing herself from him." The chains are as follows:

- Chain 1: "Mrs. Clinton" (Mention) is coreferent with "she" (Ment).
- Chain 2: "she" (Ment) is coreferent with "herself" (M).
- Chain 3: "herself" (M) is coreferent with "him" (M).

Basic dependencies:

The diagram illustrates the basic dependencies of the sentence "Mrs. Clinton previously worked for Mr. Obama, but she is now distancing herself from him". The sentence is broken down into tokens with their part-of-speech tags: Mrs. (NNP), Clinton (NNP), previously (RB), worked (VBD), for (IN), Mr. (NNP), Obama, (NNP), but (CC), she (PRP), is (VBZ), now (RB), distancing (VBG), herself (PRP), from (IN), him (PRP). The dependencies are shown as arrows connecting the tokens to their grammatical functions: compound (Mrs. Clinton), nsubj (Clinton), advmod (previously), nmod (Obama), case (for), compound (Mr. Obama), conj (but), nsubj (she), aux (is), advmod (now), nsubj (distancing), dobj (herself), nmod (from him), and case (him).

**의존관계분석 :** 구구조문법(성분에 따라 문장구조 정의)과 달리 단어 간 의존관계로 문장구조를 분석



## Unit 03 | POS-tagging

POS-tagging 전에 Morphological Analysis ?

- Text normalization
- Token들을 일반적인 형태로 만들어 단어 수를 줄임으로서 분석의 효율성을 높임
- Apples -> apple, Giving -> give, Love -> love (folding)
- Stemming : 단어를 축약형으로 바꿔 줌
- Lemmatization : 품사 정보가 보존된 형태의 기본형으로 바꿔 줌

## Unit 03 | POS-tagging

POS-tagging 전에 **Morphological Analysis** ?

- Stemming : 단어를 축약형으로 바꿔 줌
- Lemmatization : 품사 정보가 보존된 형태의 기본형으로 바꿔 줌

Word	stemming	lemmatization
Love	Lov	Love
Loves	Lov	Love
Loved	Lov	Love
Loving	Lov	Love
Innovation	Innovat	Innovation
Innovations	Innovat	Innovation
Innovate	Innovat	Innovate
Innovates	Innovat	Innovate
Innovative	Innovat	Innovative

## Unit 03 | POS-tagging

## POS-tagging ?

- token에 품사정보를 붙이는 것
- POS(Part of Speech)는 품사를 의미 (noun, adverbs, adjectives, pronouns, conjunction..)
- KoNLPy는 '꼬꼬마, 코모란, 트위터, 한나눔, 은전한닢' 이렇게 다섯개의 형태소 분석기를 묶어서 편리하게 사용할 수 있게 해 놓은 한국어 처리 패키지
  - > 꼬꼬마(kkma), 트위터(Twitter)
- nltk는 영어일 때 사용하는 패키지

## Unit 03 | POS-tagging

### POS-tagging ?

- token에 품사정보를 붙이는 것
- POS(Part of Speech)는 품사를 의미 (noun, a
- KoNLPy는 '꼬꼬마, 코모란, 트위터, 한나눔, 은  
 묶어서 편리하게 사용할 수 있게 해 놓은 한국  
 -> 꼬꼬마(kkma), 트위터(Twitter)
- nltk는 주로 영어일 때 사용하는 패키지

Tag	Description	설명	Example
CC	coordinating conjuntion		
CD	cardinal digit		
DT	determiner		
EX	existential there		'there' is.., 'there' exists..
FW	foreign word		
IN	preposition/subordiateing conjuction		
JJ	adjective	형용사	'big'
JJR	adjective, comparative	형용사, 비교급	'bigger'
JJS	adjective, superlative	형용사, 최상급	'biggest'
LS	list marker		'1)'
MD	modal		'could', 'will'
NN	noun, singular	명사, 단수형	'desk'
NNS	noun, plural	명사, 복수형	'desks'
NNP	proper noun, singular	고유명사, 단수형	'Harrison'
NNPS	proper noun, plural	고유명사, 복수형	'Americans'
PDT	predeterminer		'all the kids'
POS	possessive ending		'parent's '
PRP	personal pronoun	인칭 대명사	'I', 'he', 'she'
PRP\$	possessive pronoun	소유 대명사	'my', 'his', 'hers'
RB	adverb	형용사	'very', 'silently'
RBR	adverb, comparative	형용사, 비교급	'better'
RBS	adverb, superlative	형용사, 최상급	'best'
RP	particle		'give up'
TO	to		go 'to' the store
UH	interjection		'errrrrm'
VB	verb, base form	동사, 원형	'take'
VBD	verb, past tense	동사, 과거형	'took'
VBG	verb, gerund/present participle	동사, 현재분사	'taking'
VBN	verb, past participle	동사, 과거분사	'taken'
VBP	verb, sing. Present, non-3d		'take'
VBZ	verb, 3rd person sing. Present		'takes'
WDT	wh-determiner		'which'
WP	wh-pronoun		'who', 'what'
WP\$	possessive		'wh-pronoun', 'whose'
WRB	wh-adverb		'where', 'when'

## Unit 03 | POS-tagging

### POS-tagging ?

- Decision Tree, Hidden Markov Models, Support Vector Machines 등 많은 방법론
- KoNLPy는 Sentence Splitting, tokenize, lemmatization, POS-tagging 등 전 과정 지원 ! 굳 ㅎ
- 그런데,, 한국어는 어근에 파생접사나 어미가 붙어서 단어를 만들기 때문에,
- 이를 적절하게 나누는 것도 어렵고
- 따라서 정확한 분석이 어렵다 @\_@
- 예시) 깨뜨리시었겠더군요.

여기에서 '깨:'는 어근이며, '뜨리:'는 접사로서 '힘줄'의 뜻을 나타냅니다. '시:'는 높임, '었;' 겠, '더:'는 모두 시간을 보이는 어미들이며, '군:'은 감탄의 뜻을 나타내는 어미, '요'는 문장을 끝맺는 어미입니다. 위 문장을 제대로 분석하려면 8개 형태소로 쪼개야 합니다. 매우 고된 작업이지요.

## Unit 03 | POS-tagging

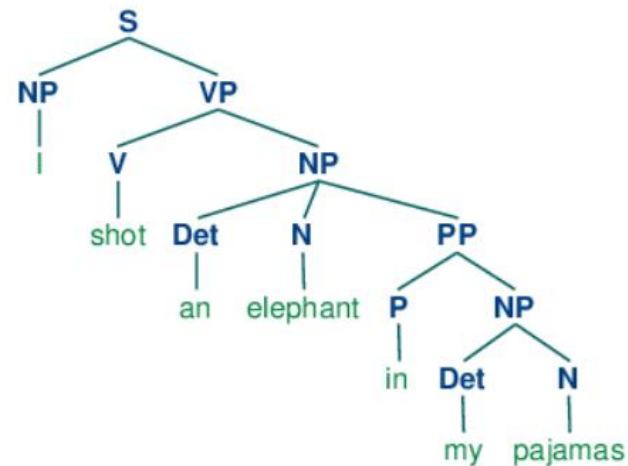
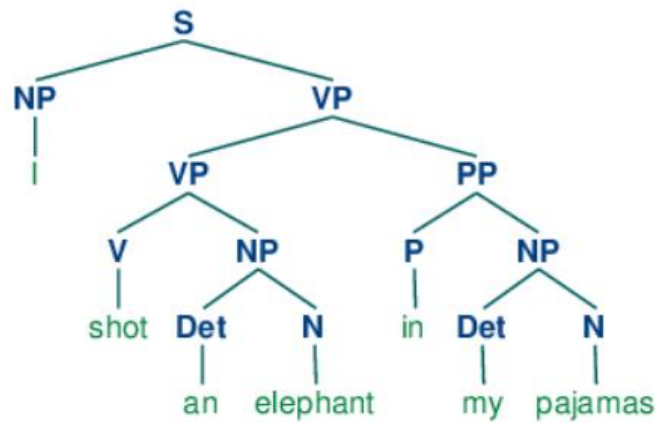
대체로,

- 정확한 품사 정보가 필요할 때 : 꼬꼬마
  - 정확성, 시간 모두 중요할 때 : 코모란
  - 빠른 분석이 중요할 때 : 트위터
- 시간빠르기 : 꼬꼬마 < 코모란 < 트위터
  - 정확한 품사 정보 : 트위터 < 코모란 < 꼬꼬마

꼬꼬마		코모란		트위터			
태그	설명	태그	설명	태그	설명		
EC	연결 어미	EC	연결 어미	Eomi	어미		
ECD	의존적 연결 어미						
ECE	대응 연결 어미						
ECS	보조적 연결 어미						
EF	종결 어미	EF	종결 어미				
EFA	청유형 종결 어미						
EFI	감탄형 종결 어미						
EFN	평서형 종결 어미						
EFO	명령형 종결 어미						
EFQ	의문형 종결 어미						
EFR	존칭형 종결 어미						
ET	전성 어미						
ETD	관형형 전성 어미	ETM	관형형 전성 어미				
ETN	명사형 전성 어미	ETN	명사형 전성 어미				
EP	선어말 어미	EP	선어말어미	PreEomi	선어말어미		
EPH	존칭 선어말 어미						
EPP	공손 선어말 어미						
EPT	시제 선어말 어미						
IC	감탄사	IC	감탄사	Exclamation	감탄사		
JK	조사			Josa	조사		
JC	접속 조사	JC	접속 조사				
JKC	보격 조사	JKC	보격 조사				
JKG	관형격 조사	JKG	관형격 조사				
JKI	호격 조사	JKV	호격 조사				
JKM	부사격 조사	JKB	부사격 조사				
JKO	목적격 조사	JKO	목적격 조사				
JKQ	인용격 조사	JKQ	인용격 조사				
JKS	주격 조사	JKS	주격 조사				
JX	보조사	JX	보조사				
MA	부사					Adverb	부사
MAG	일반 부사	MAG	일반 부사				
MAC	접속 부사	MAJ	접속 부사			Conjunction	접속사
MD	관형사	MM	관형사			Determiner	관형사
MDN	수 관형사						
MDT	일반 관형사			Noun	명사		
NN	명사	NNG	일반 명사				
NNG	보통명사						
NNB	일반 의존 명사	NNB	이존 명사				
NNM	단위 의존 명사						
NNP	고유명사	NNP	고유 명사				
NP	대명사	NP	대명사				
NR	수사	NR	수사				
OH	한자	SH	한자	Foreign	외국어 한자 및 기타기호		
OL	외국어	SL	외국어				
ON	숫자	SN	숫자	Alpha	알파벳		
SE	종임표	SE	종임표	Number	숫자		
SF	마침표, 물음표, 느낌표	SF	마침표, 물음표, 느낌표	Punctuation	구두점		
SO	불임표(물결, 숨결, 땀점)	SO	불임표(물결, 숨결, 땀점)				
SP	실표, 가운뎃점, 클론, 빗금	SP	실표, 가운뎃점, 클론, 빗금				
SS	따옴표, 괄호표, 줄표	SS	따옴표, 괄호표, 줄표				
SW	기타기호 (논리/수학기호, 화폐기호)	SW	기타기호 (논리/수학기호, 화폐기호)				
VA	형용사	VA	형용사			Adjective	형용사
VXA	보조 형용사						
VC	지정사						
VCN	부정지정사 형용사 '아니다'			VCN	부정 지정사		
VCP	긍정지정사, 서술격조사 '이다'	VCP	긍정 지정사	Verb	동사		
VV	동사	VV	동사				
VXV	보조 동사	VX	보조 용언				
VX	보조 용언						
XP	접두사		Suffix	접사			
XPN	제언 접두사	XPN			제언 접두사		
XPV	용언 접두사						
XSA	형용사 파생 접미사	XSA			형용사 파생 접미사		
XSN	명사파생 접미사	XSN			명사파생 접미사		
XSV	동사 파생 접미사	XSV			동사 파생 접미사		
XR	어근	XR	어근	Unknown	미등록어		
UN	명사추정범주	NA	분석불능범주				
		NF	명사추정범주				
		NV	용언추정범주				
				Hashtag	트위터 해쉬태그		
				KoreanParticle	이모티콘 (예:ㅋㅋ)		
				ScreenName	트위터 아이디		

## Unit 04 | Parsing

- Parsing : 문장의 문법적 구조 분석
- 즉, 형태소 분석된 문장이 주어지면,
- 구문구조 및 문장성분에 대한 문법(syntactic grammar)을 적용해
- 분석가능한 모든 구문구조(parse tree)를 만들고
- 생성된 parse tree 중 가장 적합한 것 하나를 선택하는 과정을 구문 분석이라 함



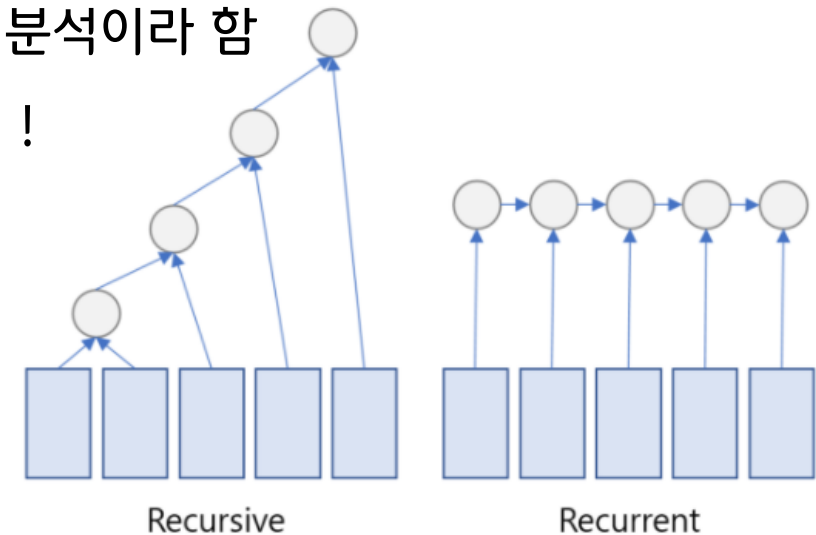


## Unit 04 | Parsing

- Parsing : 문장의 문법적 구조 분석
- 즉, 형태소 분석된 문장이 주어지면,
- 구문구조 및 문장성분에 대한 문법(syntactic grammar)을 적용해
- 분석가능한 모든 구문구조(parse tree)를 만들고
- 생성된 parse tree 중 가장 적합한 것 하나를 선택하는 과정을 구문 분석이라 함
  - 예문) 그가 산 사과를 다시 샀다.
    - (((그가/주어 산/술어) 사과를)/목적어 다시 샀다/술어) — 가장 적합한 구문구조
    - ((그가/주어 (산 사과를)/목적어) (다시 샀다)/술어)
    - ((그가/주어 산) (사과를/목적어 (다시 샀다)/술어))

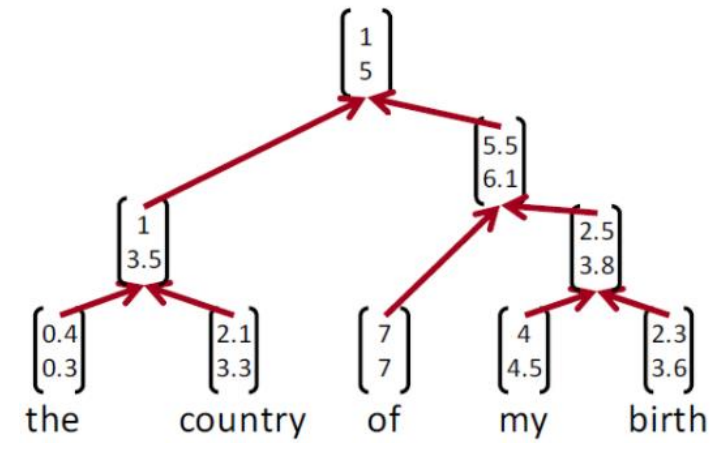
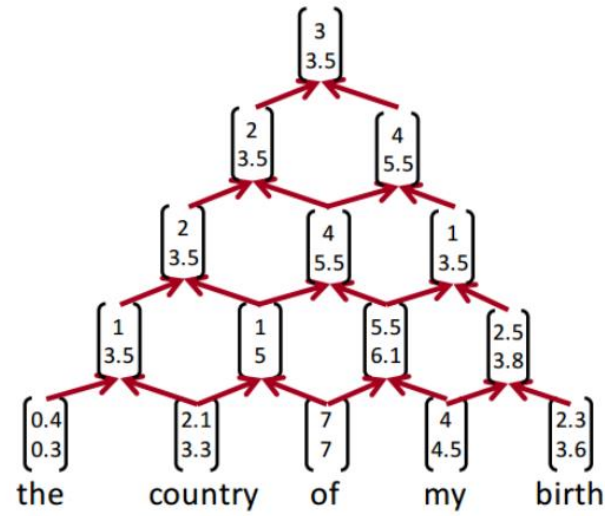
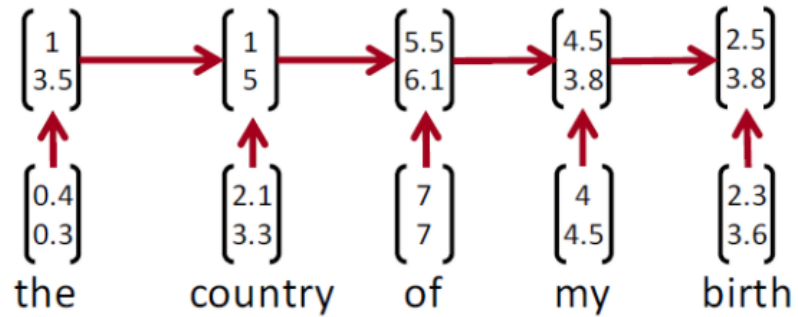
## Unit 04 | Parsing

- Parsing : 문장의 문법적 구조 분석
- 즉, 형태소 분석된 문장이 주어지면,
- 구문구조 및 문장성분에 대한 문법(syntactic grammar)을 적용해
- 분석가능한 모든 구문구조(parse tree)를 만들고
- 생성된 parse tree 중 가장 적합한 것 하나를 선택하는 과정을 구문 분석이라 함
- Recurrent Neural Network의 input이 이런 계층 구조의 데이터 임 !
- Ex. ( ( ( The ) ( actors ) ) ( ( ( are ) ( fantastic ) ) ( . ) ) )



## Unit 04 | Parsing

recurrent vs convolutional vs recursive



## Unit 05 | 그 외

이 외에도 전처리 하는 방법이나, 패키지는 다양해요 !

파이토치를 쓸 때는 토치텍스트라는 딥러닝 전용 텍스트 처리 패키지를 이용하기도하고

spaCy라는 외국어 패키지(spacy.io)도 있고

강의에서 다루지 않은 한나눔, 은전한닢이라는 형태소 분석기도 있고요~~

기본적으로 많이 쓰는 것 먼저 공부하시고, 다양한 전처리 방법들을 익혀보시면 좋을 것 같습니다 !

Q & A

들어주셔서 감사합니다.