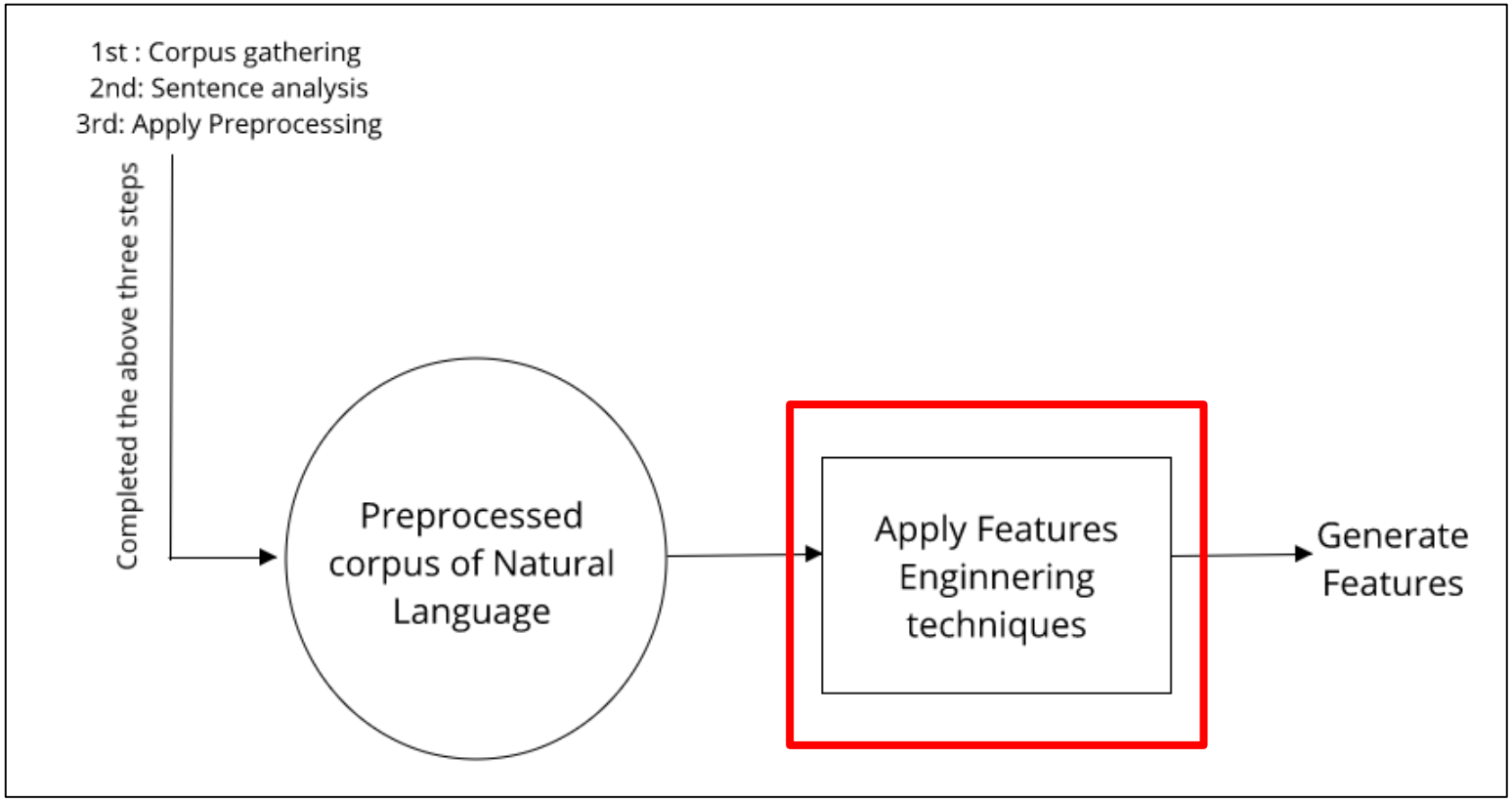
파이션 자연어 처리의 이론과 실제

5장. 피처 엔지니어링과 NLP 알고리즘

□ 피처 생성 과정에 대한 개요



□ 피처 엔지니어링 이해

- 피처 엔지니어링(feature engineering) : 원시 데이터 또는 코퍼스로부터 NLP 애플리케이션을 개발하거나 NLP 관련 문제를 해결하는데 도움이 되는 피처(어떤 현상의 개별 측정 가능한 특성 또는 속성)를 생성 또는 유도하는 과정

- 피처는 machine learning model (ML model)에 입력 변수로 사용

- 피처 엔지니어링의 목적
  - 머신러닝 기술을 사용해 NLP 애플리케이션을 개발 할때 중요한 역할
  - 코퍼스를 대표할 수 있는 속성과 머신러닝 알고리즘이 이해할 수 있는 속성이 필요
  - ML 모델의 정확성, 효율성은 피처에 따라 달라짐
  - 피처를 생성후에 사용할 피처를 선택하는 과정(feature selection)도 중요

□ NLP의 기본 피처

- **파서와 파싱**
- **POS 태깅과 POS 태거**
- 개체명 인식
- N그램
- BOW (Bag of words)
- **NLP에 대한 기본 통계 피처**

□ 파서의 기본 이해

- 파서는 문장을 Grammer Rules과 Lexical Entries 을 사용해서 파스 트리를 생성함.

문장 : John is a student.

- S는 문장
- NP는 명사구
- VP는 동사구
- V는 동사
- N은 명사
- ART는 관사인 a, an, the

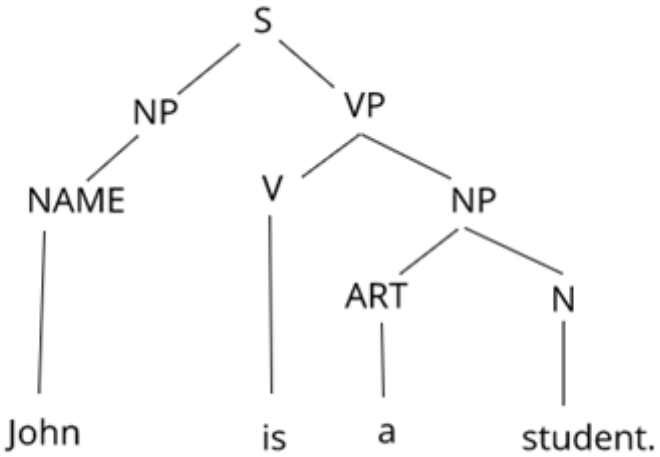
Grammar Rules

S → NP VP
NP → NAME
VP → V NP
NP → ART N

Lexical Entries

NAME → John
V → is
ART → a
N → student

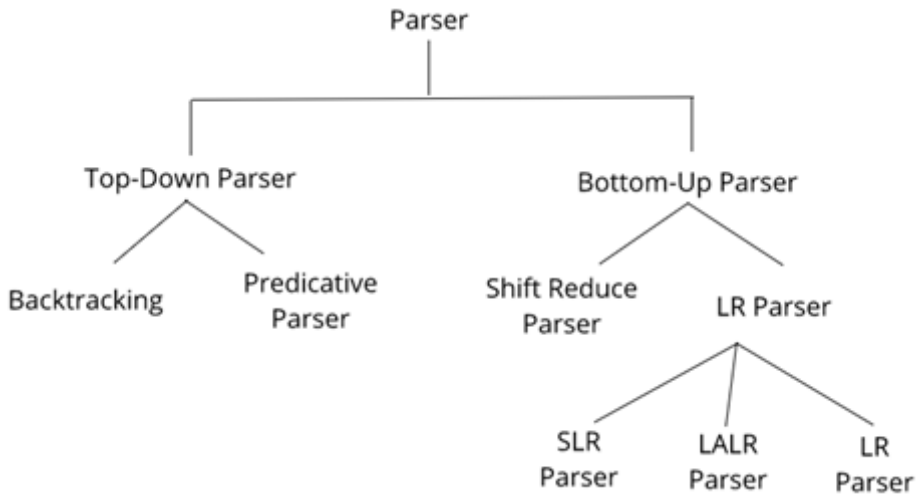
사용



□ 파서와 파싱

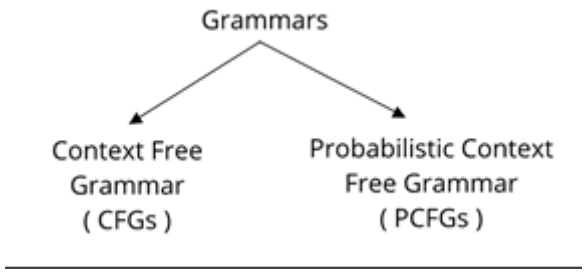
□ 파싱의 개념 이해

- 파싱이란 문장이나 토큰 스트림을 사용하는 과정, 또는 형식 분석이며, 정의된 형식 문법 규칙의 도움으로 문장 구조와 의미를 이해.
- 파싱은 문장안의 단어의 성분 구조를 이해하고 결정
- 파서와 파싱이 하는 일
  - 문법 규칙에 따라 파싱 과정을 수행하고 파스 트리를 생성
  - 생성된 파스 트리 구조는 문장의 구문 구조를 확인하는데 사용
  - 파싱이 끝나면 문장의 모호성을 감지하는데 도움이 되는 파스 트리가 출력
  - 파스 트리는 여러 개가 될 수 있음.



□ 처음부터 파서 개발하기

- 유명한 스탠포드 파서의 절차와 통계파서를 개발하는데 사용된 알고리즘을 알아봄.
- 문법 타입
  - 문맥 자유 문법
  - 확률론적 문맥 자유 문법



□ 문맥 자유 문법

- 구문 구조 문법과 같은 개념
- 문법에 따라서 문장 구조를 결정하는 과정

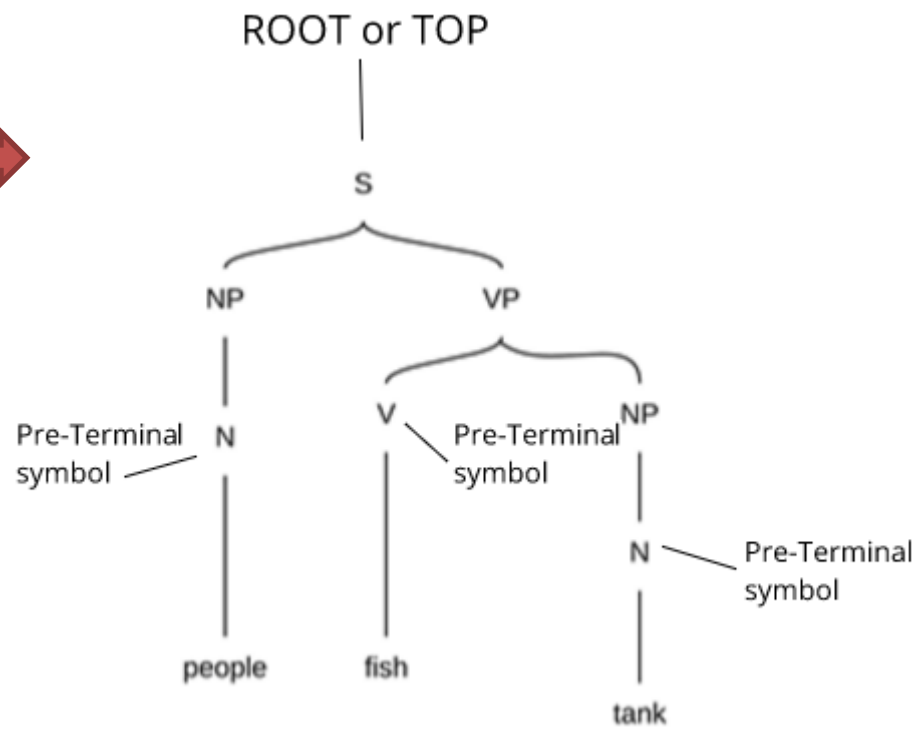
CFG 규칙, 어휘, 문장

S → NP VP
VP → V NP
VP → V NP PP
NP → NP NP
NP → NP PP
NP → N
NP → e
PP → P NP

N → People
N → fish
N → tank
N → rods
V → people
V → fish
V → tanks
P → with

Sentences: people fish tank
People fish tank with rods

주어진 문법에 의해 생성된 문장 중 하나의 파스 트리 표현



□ 문맥 자유 문법

G = (T, C, N, S, L, R)

T 는 어휘 기호.

C 는 프리 터미널(pre-terminal) 기호.

N 는 비터미널 기호.

S 는 비터미널 N에 속하는 시작 기호다. (S ε N)

L 은 X -> x 규칙을 따르는 항목 집합인 어휘 터미널. , Here X → P and x → T

R 은 X -> y 규칙을 따르는 항목 세트인 문법 , here X ε N and γ ε (N U C)*.

CFG에 대해 더 자연스런 형태의 형식 표현

- * 기호는 빈 시퀀스의 존재 의미
- S 기호부터 시작하고 TOP 또는 ROOT인 단계를 더 추가할 수 있음.
- 빈 문자열(e) 규칙
  - people fish tank => fish tank 또는 people fish 라는 구로 추출 가능
  - 두 예제 모두 누락된 명사가 있음
  - 이 누락된 명사를 e fish tank 또는 people fish e로 표현
- 빈 규칙 (empty rule) : NP -> e와 같이 오른쪽에 빈 문자열만 있는 규칙
- 단항 규칙(unary rule) : NP -> N과 같이 왼쪽과 오른쪽에 하나의 기호만 있는 규칙
- 이항 규칙(binary rule) : VP -> V NP와 같이 오른쪽에 2개의 기호가 있는 규칙

□ 확률론적 문맥 자유 문법

G = (T, N, S, R, P)
T is a set of terminal symbols
N is a set of nonterminal symbols
S is the start symbol (S ∈ N)
R is a set of rules/productions of the form X → γ
P is the probability function
P is R → [0,1]

$$\forall X \in N, \sum_{X \to \gamma \in R} P(X \to \gamma) = 1$$

PCFG 형식 정의

S → NP VP	1.0
VP → V NP	0.6
VP → V NP PP	0.4
NP → NP NP	0.1
NP → NP PP	0.2
NP → N	0.7
PP → P NP	1.0

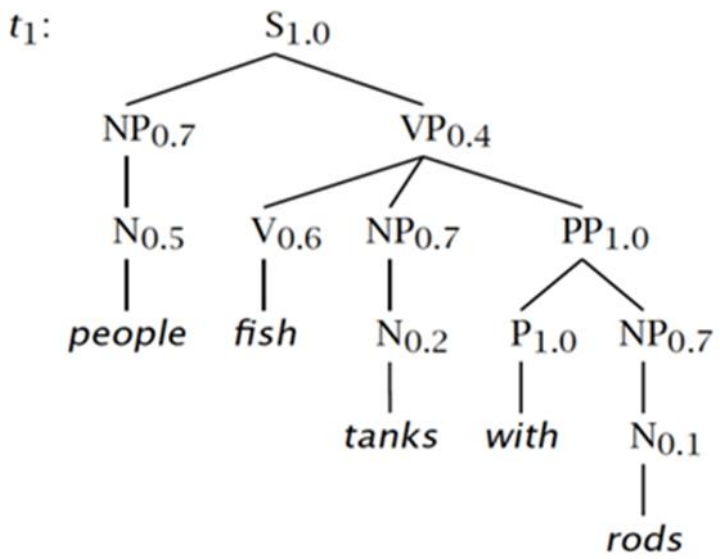
문법 규칙에 대한 확률

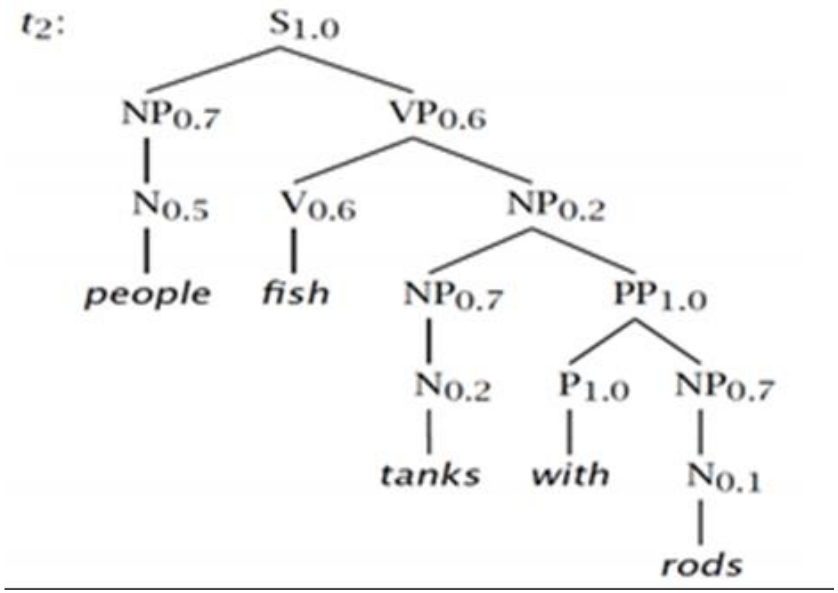
N → *people*	0.5
N → *fish*	0.2
N → *tanks*	0.2
N → *rods*	0.1
V → *people*	0.1
V → *fish*	0.6
V → *tanks*	0.3
P → *with*	1.0

어휘 규칙에 대한 확률

- T, N, S, R은 CFG와 유사
- 확률함수(Probability function) P 가 추가
- 비터미널에 대한 확률의 합이 1이 되어야 함.
- 3개의 NP 규칙
  - 확률의 합이 1 (0.1 + 0.2 + 0.7 = 1.0)
  - NP -> N 이 확률 0.7이므로 명사일 가능성이 높음.

□ 트리의 확률 계산과 문자열의 확률 계산





- 트리의 확률 계산은 어휘집과 문법 규칙의 확률 값을 곱.
- P(t1) = 1.0 * 0.7 * 0.4 * 0.5 * 0.6 * 0.7 * 1.0 * 0.2 * 1.0 * 0.7 * 0.1 = 0.008232
- P(t2) = 0.00024696

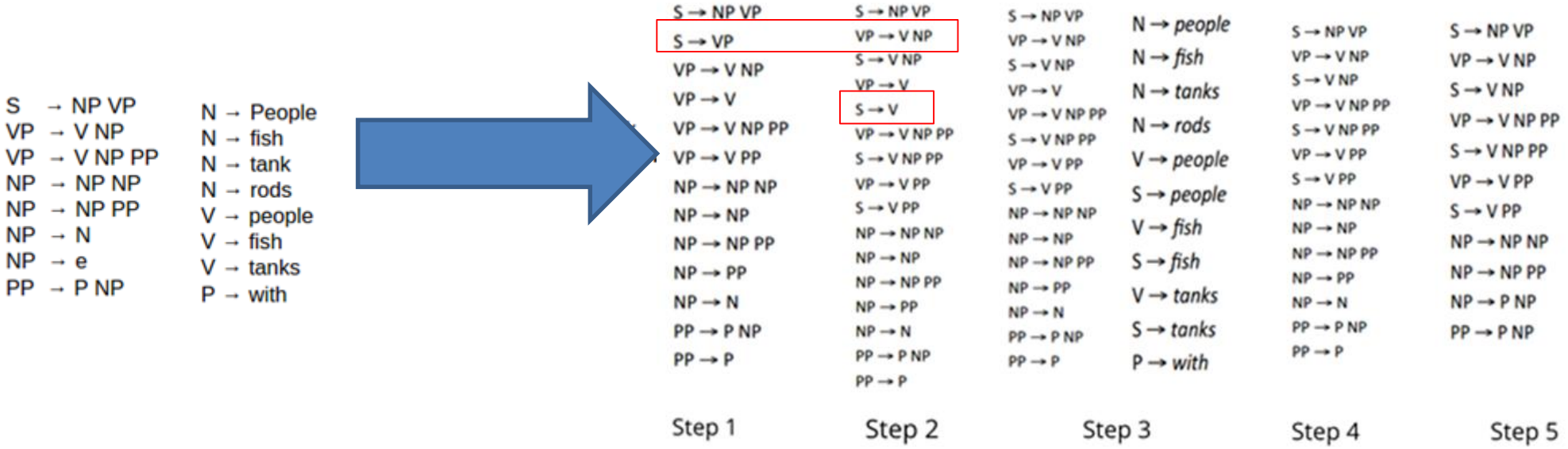
- 문자열의 확률 계산은 트리와 트리 확률을 모두 고려하고 이 값을 더함.
- P(S) = P(t1) +P(t2)
- = 0.0008232 + 0.00024696
- = 0.00107016

□ 문법 변형

- 문법 변형은 문법을 좀 더 제한적으로 만드는 기술로 파싱 과정을 효율적으로 만듬.
- 촘스키 표준형 (CNF, Chomsky Normal Form)을 사용함.
- CNF 규칙
  - $X \rightarrow Y Z$ or $X \rightarrow w$ where $X, Y, Z \varepsilon N$ and $w \varepsilon T$
  - 문법 규칙의 오른쪽 2개를 초과한 비터미널이 있어서는 안 됨. (단일 터미널 포함 가능)
- 기존 문법을 CNF로 변환 절차
  - 빈 규칙과 단항 규칙은 재귀함수를 사용해 제거
  - N항 규칙은 문법 규칙에 새로운 비터미널을 도입해 분리

□ 문법 변형

- 앞의 문법 규칙을 CNF 적용
- 먼저 빈 규칙을 제거
  - 오른쪽에 NP가 있으면 S -> NP VP 같은 두 가지 규칙 존재
  - NP에 빈 값을 입력하면 S-> VP 가 되고 이 방법을 재귀적으로 적용
- 단항 규칙 제거 시도
  - 첫번째 단항 규칙 S -> NP를 제거하려고 왼쪽에 VP가 있는 모든 규칙을 고려
  - S가 즉시 VP로 바뀌므로 새 규칙을 도입 필요하여 S -> V NP 도입
  - S -> V 같은 단항 규칙을 모두 제거하면 어휘 엔트리 변경 필요.



□ 문법 변형

- CNF 적용이 실환경에서 필요 없을 수 있음.
- 단항 규칙을 문법 규칙으로 유지가 필요 => 이유는 단어가 동사, 또는 명사 뿐만 아니라 비터미널 기호 정보를 알려줌
- 파서의 모든 기본 개념과 파상을 결합해 파서를 개발하는 알고리즘을 배워보자.

S → NP VP
VP → V NP
S → V NP
VP → V @VP_V
@VP_V → NP PP
S → V @S_V
@S_V → NP PP
VP → V PP
S → V PP
NP → NP NP
NP → NP PP
NP → P NP
PP → P NP

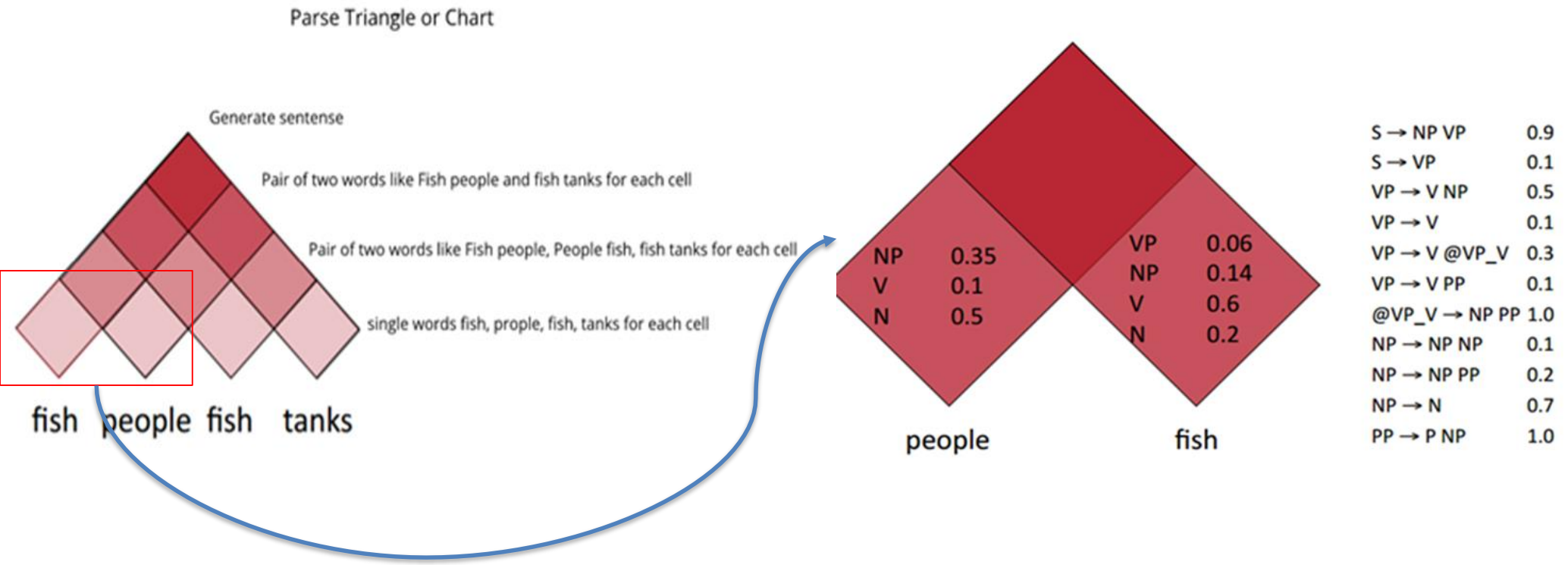
최종 결과

NP → *people*
NP → *fish*
NP → *tanks*
NP → *rods*
V → *people*
S → *people*
VP → *people*
V → *fish*
S → *fish*
VP → *fish*
V → *tanks*
S → *tanks*
VP → *tanks*
P → *with*
PP → *with*

□ 파서와 파싱

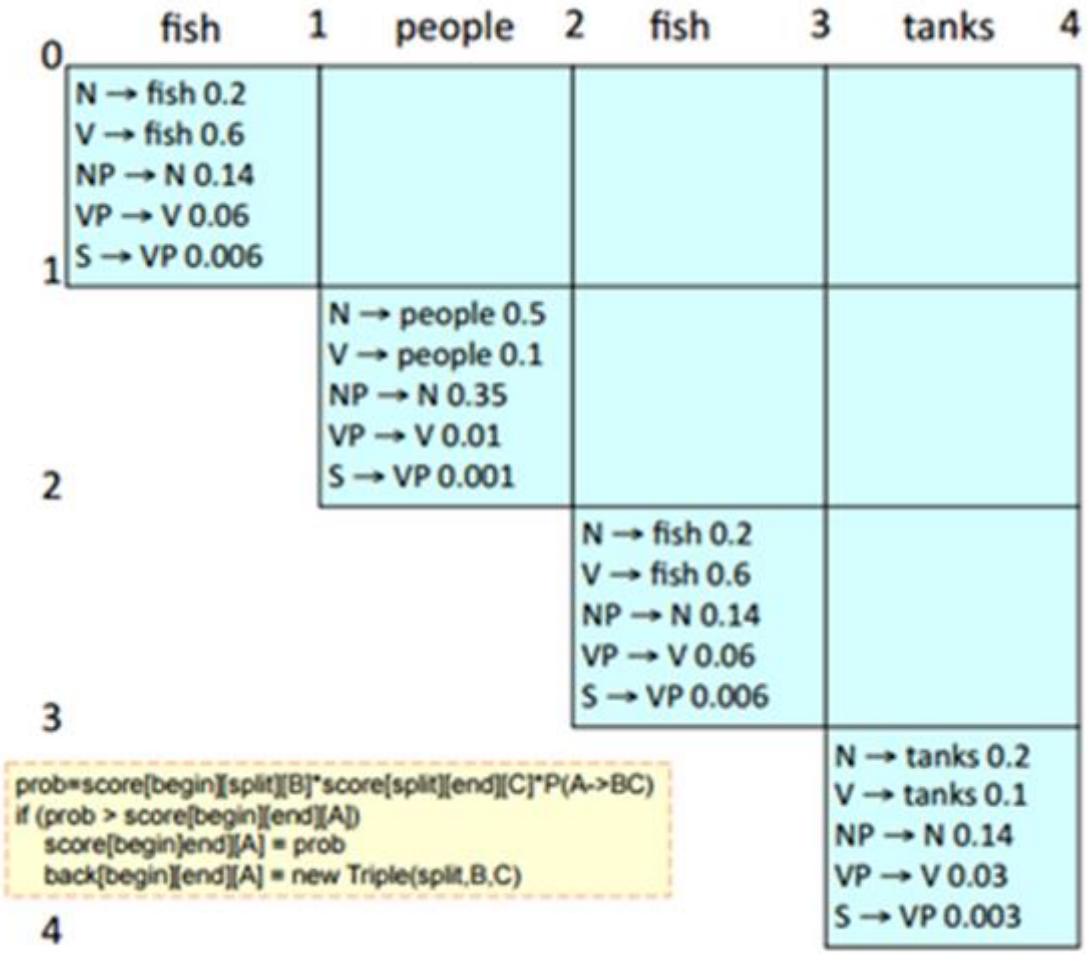
□ CKY 알고리즘으로 파서 개발

- CKY(Cocke-Kasami-Yunger) 알고리즘은 문장에서 단어를 가져와서 상향 파싱을 사용해 파스 트리를 생성
- 파스 삼각형 또는 차트 라는 데이터 구조를 정의



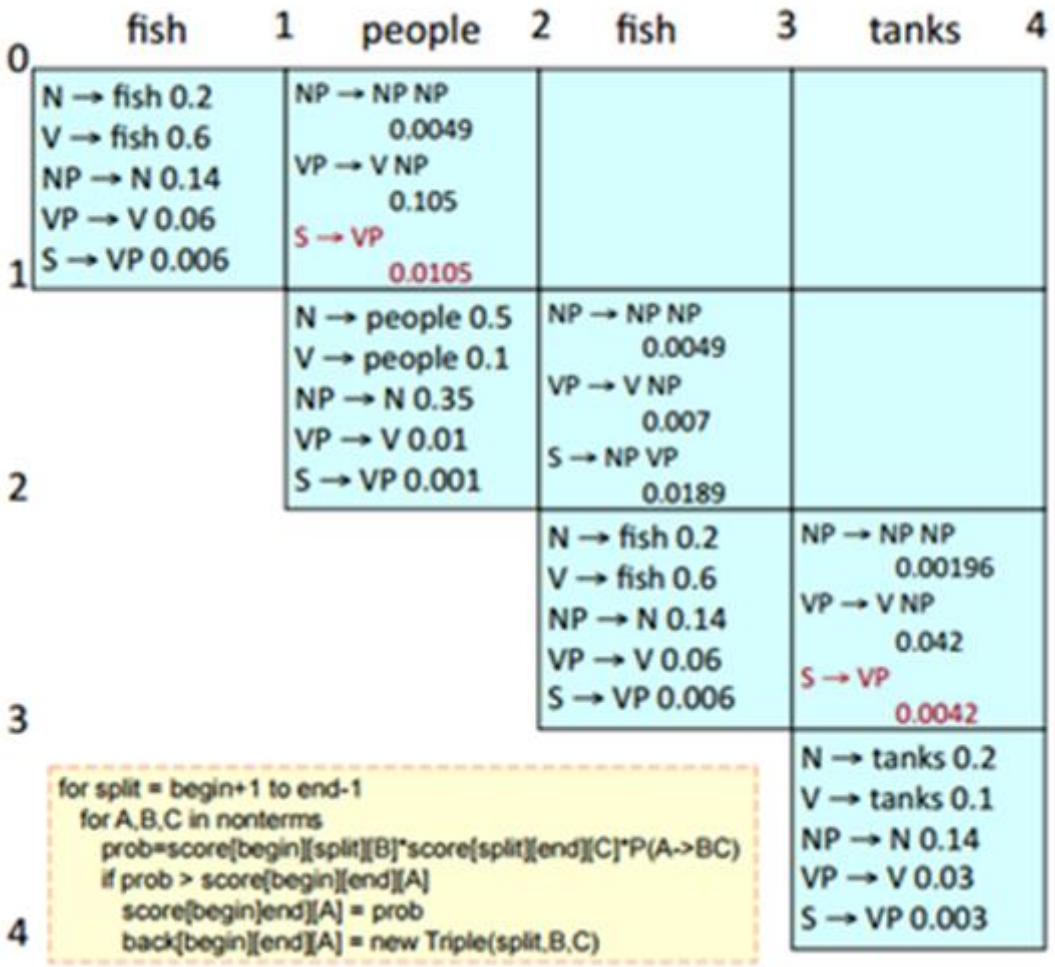
□ CKY 알고리즘으로 파서 개발

S → NP VP	0.9
S → VP	0.1
VP → V NP	0.5
VP → V	0.1
VP → V @VP_V	0.3
VP → V PP	0.1
@VP_V → NP PP	1.0
NP → NP NP	0.1
NP → NP PP	0.2
NP → N	0.7
PP → P NP	1.0
N → *people*	0.5
N → *fish*	0.2
N → *tanks*	0.2
N → *rods*	0.1
V → *people*	0.1
V → *fish*	0.6
V → *tanks*	0.3
P → *with*	1.0



□ CKY 알고리즘으로 파서 개발

S → NP VP	0.9
S → VP	0.1
VP → V NP	0.5
VP → V	0.1
VP → V @VP_V	0.3
VP → V PP	0.1
@VP_V → NP PP	1.0
NP → NP NP	0.1
NP → NP PP	0.2
NP → N	0.7
PP → P NP	1.0
N → *people*	0.5
N → *fish*	0.2
N → *tanks*	0.2
N → *rods*	0.1
V → *people*	0.1
V → *fish*	0.6
V → *tanks*	0.3
P → *with*	1.0



□ CKY 알고리즘으로 파서 개발

S → NP VP	0.9
S → VP	0.1
VP → V NP	0.5
VP → V	0.1
VP → V @VP_V	0.3
VP → V PP	0.1
@VP_V → NP PP	1.0
NP → NP NP	0.1
NP → NP PP	0.2
NP → N	0.7
PP → P NP	1.0
N → *people*	0.5
N → *fish*	0.2
N → *tanks*	0.2
N → *rods*	0.1
V → *people*	0.1
V → *fish*	0.6
V → *tanks*	0.3
P → *with*	1.0

	0 fish 1	1 people 2	2 fish 3	3 tanks 4
0	N → fish 0.2 V → fish 0.6 NP → N 0.14 VP → V 0.06 S → VP 0.006	NP → NP NP 0.0049 VP → V NP 0.105 S → VP 0.0105	NP → NP NP 0.0000686 VP → V NP 0.00147 S → NP VP 0.000882	NP → NP NP 0.0000009604 VP → V NP 0.00002058 S → NP VP 0.00018522
1		N → people 0.5 V → people 0.1 NP → N 0.35 VP → V 0.01 S → VP 0.001	NP → NP NP 0.0049 VP → V NP 0.007 S → NP VP 0.0189	NP → NP NP 0.0000686 VP → V NP 0.000098 S → NP VP 0.01323
2			N → fish 0.2 V → fish 0.6 NP → N 0.14 VP → V 0.06 S → VP 0.006	NP → NP NP 0.00196 VP → V NP 0.042 S → VP 0.0042
3				N → tanks 0.2 V → tanks 0.1 NP → N 0.14 VP → V 0.03 S → VP 0.003
4				

Call buildTree(score, back) to get the best parse

□ 파서와 파싱

□ 기존 파서 도구

- 스탠포드 파서
  - https://stanfordnlp.github.io/CoreNLP/ 에서 다운로드
  - $ cd stanford-corenlp-full-2016-10-31/
  - $ java -mx4g -cp "*" edu.stanford.nlp.pipeline.StanfordCoreNLPServer
- https://github.com/jalajthanaki/NLPython/tree/master/ch5/parserexample 예제
- $ conda create -n py27 python=2.7 anaconda
- $ conda activate py27
- $ pip install pycorenlp nltk

```
nlp = StanfordCoreNLP('http://localhost:9000')
def stanfordparserdemo(sentnece):
    text = (sentnece)

    output = nlp.annotate(text, properties={
        'annotators': 'tokenize,ssplit,pos,depparse,parse',
        'outputFormat': 'json'
    })

    print  "\n------------Stanford Parser Parseing Result------------"
    parsetree = output['sentences'][0]['parse']
    print "\n------parsing------\n"
    print parsetree
    print  "\n------ Words inside NP ------\n"
    for i in Tree.fromstring(parsetree).subtrees():
        if i.label() == 'NP':
            print i.leaves(),i.label()
    print  "\n------ Words inside NP with POS tags ------\n"
    for i in Tree.fromstring(parsetree).subtrees():
        if i.label() == 'NP':
            print i

def NLTKparserfordependancies(sentnece):

    path_to_jar = '/home/jalaj/stanford-corenlp-full-2016-10-31/stanford-corenlp-3.7.0.jar'
    path_to_models_jar = '/home/jalaj/stanford-corenlp-full-2016-10-31/stanford-corenlp-3.7.0-models.jar'
    dependency_parser = StanfordDependencyParser(path_to_jar=path_to_jar, path_to_models_jar=path_to_models_jar)
    result = dependency_parser.raw_parse(sentnece)
    dep = result.next()
    print "\n------Dependencies------\n"
    print list(dep.triples())

if __name__ == "__main__":
    stanfordparserdemo('The boy put tortoise on the rug.')
    NLTKparserfordependancies('The boy put tortoise on the rug.')
```

코드 수정 필요

□ 피처 추출과 이해

- 문장에서 NP를 얻는 코드

```
print  "\n------------Stanford Parser Parseing Result------------"
parsetree = output['sentences'][0]['parse']
print "\n------parsing------\n"
print parsetree
print  "\n------ Words inside NP ------\n"
for i in Tree.fromstring(parsetree).subtrees():
    if i.label() == 'NP':
        print i.leaves(),i.label()
print  "\n------ Words inside NP with POS tags ------\n"
for i in Tree.fromstring(parsetree).subtrees():
    if i.label() == 'NP':
        print i
```

```
------ Words inside NP with POS tags ------

(NP (DT The) (NN boy))
(NP (NN tortoise))
(NP (DT the) (NN rug))
```

□ POS 태깅과 POS 태거

□ 스탠포트 POS 태거 예제

- https://github.com/jalajthanaki/NLPython/blob/master/ch5/POStagdemo.

```
from pycorenlp import StanfordCoreNLP
nlp = StanfordCoreNLP('http://localhost:9000')


def stnfordpostagdemofunction(text):
    output = nlp.annotate(text, properties={
        'annotators': 'pos',
        'outputFormat': 'json'
    })
    for s in output["sentences"]:
        for t in s["tokens"]:
            print str(t["word"])+ " --- postag --"+ str(t["pos"])


if __name__ == "__main__":
    stnfordpostagdemofunction("This is a car.")
```

```
This --- postag --DT
is --- postag --VBZ
a --- postag --DT
car --- postag --NN
. --- postag --.
```

□ POS 태깅과 POS 태거

□ POS 태그를 피처로 사용하기

- POS 태그는 머신러닝 알고리즘으로 챗봇을 제작할때 중요.
- POS 태그 시퀀스는 머신이 다양한 문장 구조를 이해할때 유용

- 다중단어 표현(MWE, multiword express)을 식별하는 시스템 구축에도 유용
- 감성 분석에서 POS 태그 활용

□ 개체명 인식

□ NER 클래스

- 개체명 인식(name entity recognition , NER)은 위치명, 사람이름, 조직이름 등을 식별
- https://github.com/jalajthanaki/NLPython/tree/master/ch5/NERtooldemo
- https://nlp.stanford.edu/software/CRF-NER.shtml#Download.

```
from nltk.tag import StanfordNERTagger
from nltk.tokenize import word_tokenize

st = StanfordNERTagger('/home/jalaj/stanford-ner-2016-10-31/classifiers'
                       '/english.muc.7class.distsim.crf.ser.gz',
                       '/home/jalaj/stanford-ner-2016-10-31/stanford-ner-3.7.0.jar',
                       encoding='utf-8')

text = 'While in France, Christine Lagarde discussed short-term ' \
       'stimulus efforts in a recent interview at 5:00 P.M with the Wall Street Journal.'

tokenized_text = word_tokenize(text)
classified_text = st.tag(tokenized_text)
print(classified_text)
```

```
Resource punkt not found.
Please use the NLTK Downloader to obtain the resource:

>>> import nltk
>>> nltk.download('punkt')

For more information see: https://www.nltk.org/data.html
```

```
[(u'While', u'O'), (u'in', u'O'), (u'France', u'LOCATION'),
(u',', u'O'), (u'Christine', u'PERSON'), (u'Lagarde', u'PERSON'),
(u'discussed', u'O'), (u'short-term', u'O'), (u'stimulus', u'O'),
(u'efforts', u'O'), (u'in', u'O'), (u'a', u'O'), (u'recent', u'O'),
(u'interview', u'O'), (u'at', u'O'), (u'5:00', u'O'), (u'P.M', u'O'),
(u'with', u'O'), (u'the', u'O'), (u'Wall', u'O'), (u'Street', u'O'),
(u'Journal', u'O'), (u'.', u'O')]
```

□ N 그램

- N 그램은 텍스트 데이터 또는 음성 데이터의 주어진 시퀀스에서 있는 n개 항목의 연속 시퀀스

			1-gram
Name of domain	**items**	**Sample sequence of the data**	**unigram**
Computational biology (DNA sequence)	base pair	...AGCTTCGA...	..., A,G,C,T,T,C,G,A ,...
Computational biology (Protine sequence)	Amino acid	...Cys-Gly-Leu-Ser-Trp ...	..., Cys, Gly, Leu, Ser, Trp, ...
NLP	character	...this_is_a_pen...	..., t,h,i,s,_,i,s,_,a,p,e,n ,...
NLP	words	...This is a pen...	..., this,is,a,pen ,...

			2-gram
Name of domain	**items**	**Sample sequence of the data**	**bigram**
Computational biology (DNA sequence)	base pair	...AGCTTCGA...	..., AG,GC,CT,TC,CG,GA ,...
Computational biology (Protine sequence)	Amino acid	...Cys-Gly-Leu-Ser-Trp ...	..., Cys-Gly, Gly-Leu, Leu-Ser, Ser-Trp, ...
NLP	character	...this_is_a_pen...	..., th,hi,is,s_,_i,is,s_,_a,a_,_p,pe,en ,...
NLP	words	...This is a pen...	..., this is, is a, a pen ,...

			3-gram
Name of domain	**items**	**Sample sequence of the data**	**trigram**
Computational biology (DNA sequence)	base pair	...AGCTTCGA...	..., AGC,GCT,CTT,TTC,TCG,CGA ,...
Computational biology (Protine sequence)	Amino acid	...Cys-Gly-Leu-Ser-Trp ...	..., Cys-Gly-Leu, Gly-Leu-Ser, Leu-Ser-Trp ,...
NLP	character	...this_is_a_pen...	..., thi,his,is_,s_i,_is,is_,s_a,_a_,a_p,_pe,pen ,...
NLP	words	...This is a pen...	..., this is a, is a pen ,...

□ BOW 이해하기

- 문서에서 포함된 단어의 리스트를 BOW(Bag of words)

Text document 1: John likes to watch cricket. Chris likes cricket too.

Text document 2: John also likes to watch movies.

BOW

```
List  of words= ["John", "likes", "to", "watch", "cricket", "Chris", "too",
"also", "movies"]
```

Frequency count for Document 1: [1, 2, 1, 1, 2, 1, 1, 0, 0]

Frequency count for Document 2: [1, 1, 1, 1, 0, 0, 0, 1, 1]

□ NLP에 대한 기본 통계 피처

□ TF-IDF

- TF-IDF는 term frequency-inverse document frequency(용어빈도 – 역문서 빈도)의 줄임말
- 문서내에서 어떤 단어(t)가 얼마나 중요한지에 대한 지표로 사용.
- TF(t) = (문서 안에서 용어 t가 나타나는 횟수) / (문서에 있는 용어의 총수)
- IDF(t) = log10 (문서의 총수 / 용어 t가 들어간 문서의 수) <= 단어의 가중치
- TF-IDF = TF(t) * IDF(t)

□ TF-IDF 계산

- Document 1: This is a a sample.
- Document 2: This is another example another example example .

Step 1 : Calculate TF

Step 1.1 : Term Count for each document

Document 1

Term	Term Count
this	1
is	1
a	2
sample	1

Document 2

Term	Term Count
this	1
is	1
another	2
example	3

Step 1.2 : Now calculate total nuber of words in each document

Document 1 : Total words are = 5
Document 2 : Total words are = 7

Step 1.3 : Now calculate TF

TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document)

$$\text{tf}(\text{"this"}, d_1) = \frac{1}{5} = 0.2$$

$$\text{tf}(\text{"this"}, d_2) = \frac{1}{7} \approx 0.14$$

□ TF-IDF 계산

Step 2 : Calculate IDF

Step 2.1 : IDF calculation

IDF(t) = log(Total number of documents / Number of documents with term t in it)

So here there are 2 document and term "this" appears in both of them
So IDF is given below.

$$\text{idf}(''\text{this}'', D) = \log\left(\frac{2}{2}\right) = 0$$

Step 3 : TF x IDF calculation

$$\text{tfidf}(''\text{this}'', d_1) = 0.2 \times 0 = 0$$
$$\text{tfidf}(''\text{this}'', d_2) = 0.14 \times 0 = 0$$

zero implies that the word is not very informative
For other words is given below

$$\text{tf}(''\text{example}'', d_1) = \frac{0}{5} = 0$$
$$\text{tf}(''\text{example}'', d_2) = \frac{3}{7} \approx 0.429$$
$$\text{idf}(''\text{example}'', D) = \log\left(\frac{2}{1}\right) = 0.301$$

Step 4: TF X IDF for word example

$$\text{tfidf}(''\text{example}'', d_1) = \text{tf}(''\text{example}'', d_1) \times \text{idf}(''\text{example}'', D) = 0 \times 0.301 = 0$$
$$\text{tfidf}(''\text{example}'', d_2) = \text{tf}(''\text{example}'', d_2) \times \text{idf}(''\text{example}'', D) = 0.429 \times 0.301 \approx 0.13$$

□ NLP에 대한 기본 통계 피처

□ TF-IDF 실습

- https://github.com/jalajthanaki/NLPython/tree/master/ch5/TFIDFdemo.

```
for subdir, dirs, files in os.walk(path):...

# this can take some time
tfidf = TfidfVectorizer(tokenizer=tokenize, stop_words='english')
tfs = tfidf.fit_transform(token_dict.values())

str = 'this sentence has unseen text such as computer but also king lord juliet'
response = tfidf.transform([str])
#print response


feature_names = tfidf.get_feature_names()
for col in response.nonzero()[1]:
    print feature_names[col], ' - ', response[0, col]


feature_array = np.array(tfidf.get_feature_names())
tfidf_sorting = np.argsort(response.toarray()).flatten()[::-1]
n = 3
top_n = feature_array[tfidf_sorting][:n]
print top_n

n = 4
top_n = feature_array[tfidf_sorting][:n]
print top_n
```

```
(py27) braveji@DESKTOP-YGJI:~/NLPython/ch5/TFIDFdemo$ python tfidf_scikitlearn.py
/home/braveji/anaconda3/envs/py27/lib/python2.7/site-packages/sklearn/feature_extra
 stop words generated tokens [u'abov', u'afterward', u'alon', u'alreadi', u'alway',
', u'describ', u'dure', u'els', u'elsewher', u'empti', u'everi', u'everyon', u'ever
, u'hundr', u'inde', u'latterli', u'mani', u'meanwhil', u'moreov', u'mostli', u'nob
, u'sever', u'sinc', u'sincer', u'sixti', u'someon', u'someth', u'sometim', u'somew
wenti', u'veri', u'wa', u'whatev', u'whenc', u'whenev', u'wherea', u'whereaft', u'w
  'stop_words.' % sorted(inconsistent))
thi  -  0.34618161159873423
lord  -  0.6633846138519129
king  -  0.6633846138519129
[u'king' u'lord' u'thi']
[u'king' u'lord' u'thi' u'youth']
```

□ NLP에 대한 기본 통계 피처

□ 벡터화

- 텍스트를 벡터 형식으로 변형이 중요한 작업 (ML 알고리즘의 input)
- Scikit-learn에는 원핫 인코딩 (one-hot encoding)형식으로 변환하는 DictVectorizer가 있음.
- Word2vec를 사용할 수 있음 (6장에서 자세히 배움.)

□ 정규화

- 피처로 추출된 값을 0 과 1 사이 값으로 변환

$$z = \frac{x - \min(x)}{\max(x) - \min(x)}$$

$$x_{new} = \frac{x - \mu}{\sigma}$$

□ 확률모델

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-1})$$

texaco, rose, one, in, this, issue, is, pursuing, growth, in, a, boiler, house, said, mr., gurria, mexico, 's, motion, control, proposal, without, permission, from, five, hundred, fifty, five, yen

outside, new, car, parking, lot, of, the, agreement, reached

this, would, be, a, record, november

- 마코프 가정 적용

The Maximum Likelihood Estimate

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$