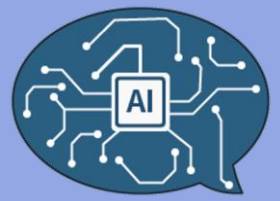
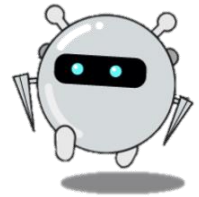


기본 언어 모델



멘토 황주훈

Let’s Start

1. 자연어 처리

자연어 처리(NLP)

- 사람의 언어를 컴퓨터가 이해, 조작, 생성 할 수 있게 해주는 인공지능

자연어 처리(NLP)

- 사람의 언어를 컴퓨터가 이해, 조작, 생성 할 수 있게 해주는 인공지능
- 일상 속 NLP

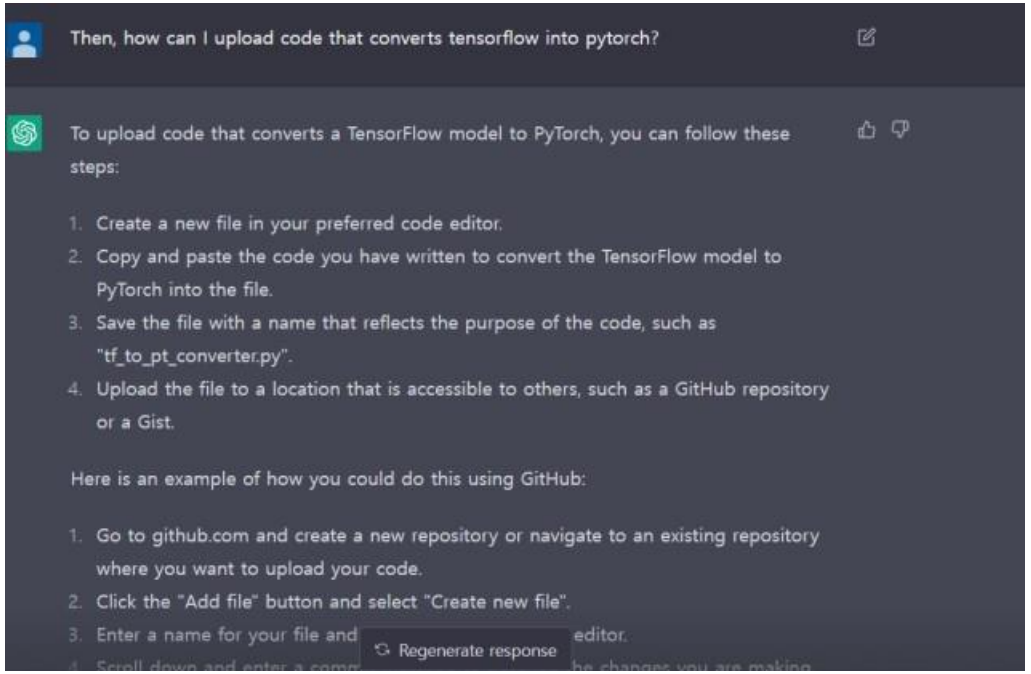
  챗봇

자연어 처리(NLP)

- 사람의 언어를 컴퓨터가 이해, 조작, 생성 할 수 있게 해주는 인공지능

- 일상 속 NLP

챗봇

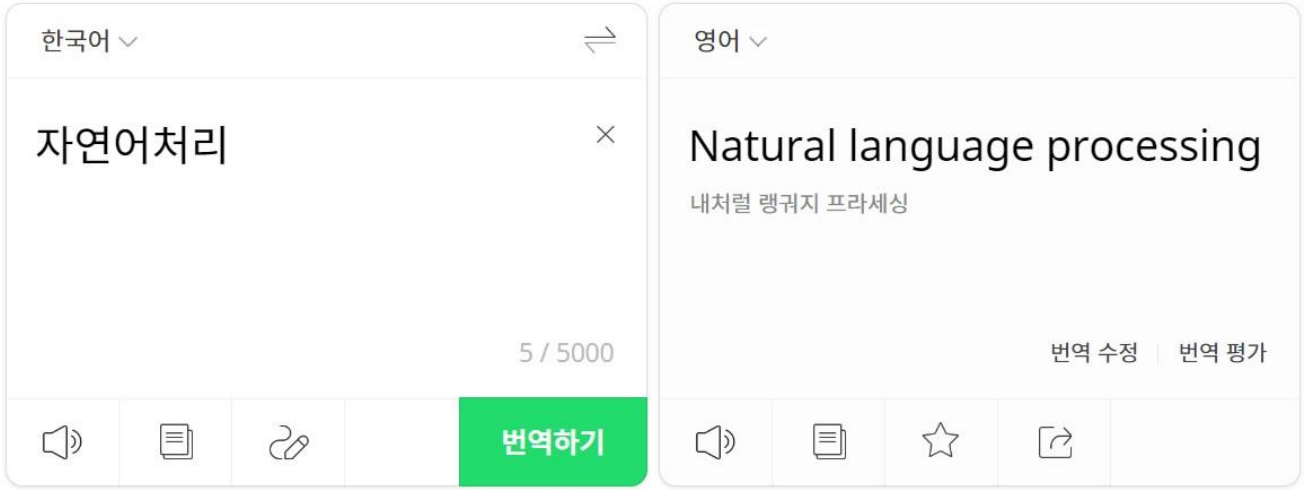


자연어 처리(NLP)

- 사람의 언어를 컴퓨터가 이해, 조작, 생성 할 수 있게 해주는 인공지능
- 일상 속 NLP

  챗봇

  번역



자연어 처리(NLP)

- 사람의 언어를 컴퓨터가 이해, 조작, 생성 할 수 있게 해주는 인공지능

- 일상 속 NLP

  챗봇

  번역

  AI 상담사

기존에 고객상담센터의 일평균 처리량인 4~8만 콜 중 약 50%를 AI가 상담하고, 전체 콜의 약 25%가량을 상담사 연결 없이 AI가 자체적으로 해결하는 성과

아웃바운드 업무(은행에서 고객에게 전화를 거는 통지성 업무)의 95%를 AI 상담으로 대체

출처 : 인공지능신문(https://www.aitimes.kr)

자연어 처리(NLP)

- 사람의 언어를 컴퓨터가 이해, 조작, 생성 할 수 있게 해주는 인공지능

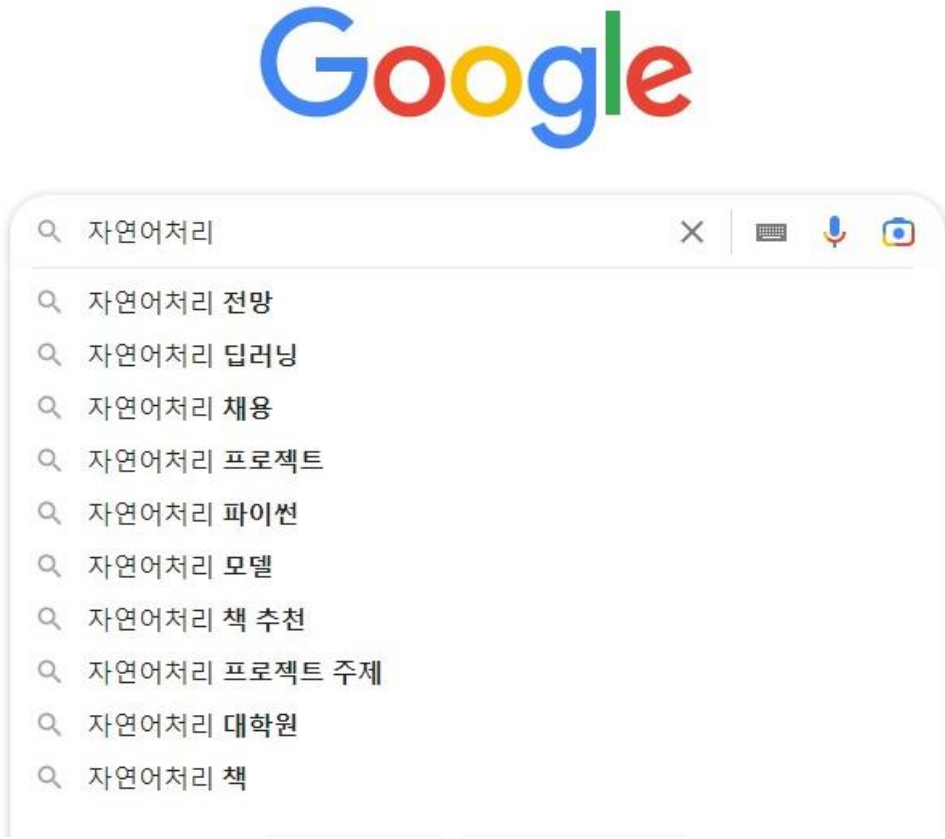
- 일상 속 NLP

  챗봇

  번역

  AI 상담사

  검색



2. 텍스트 분석과 토큰화

말뭉치(Corpus)

- 말뭉치(Corpus) : 언어 연구를 위해 특정 목적을 가지고 언어의 표본을 추출한 집합


```
    "id": "MDRW1900000002.1.1.32",
    "form": "ㅋㅋㅋㅋㅋㅋ진짜로??????",
    "original_form": "ㅋㅋㅋㅋㅋㅋ진짜로??????",
    "speaker_id": "1",
    "time": "20191128 15
},
{
    "id": "MDRW1900000002.1.1.33",
    "form": "작년에 8개월 나갔다왔자나 ㅎㅎ",
    "original_form": "작년에 8개월 나갔다왔자나 ㅎㅎ",
    "speaker_id": "2",
    "time": "20191128 15
},
{
    "id": "MDRW1900000002.1.1.34",
    "form": "진짜???어디어디????",
    "original_form": "진짜???어디어디????",
    "speaker_id": "1",
    "time": "20191128 15
```

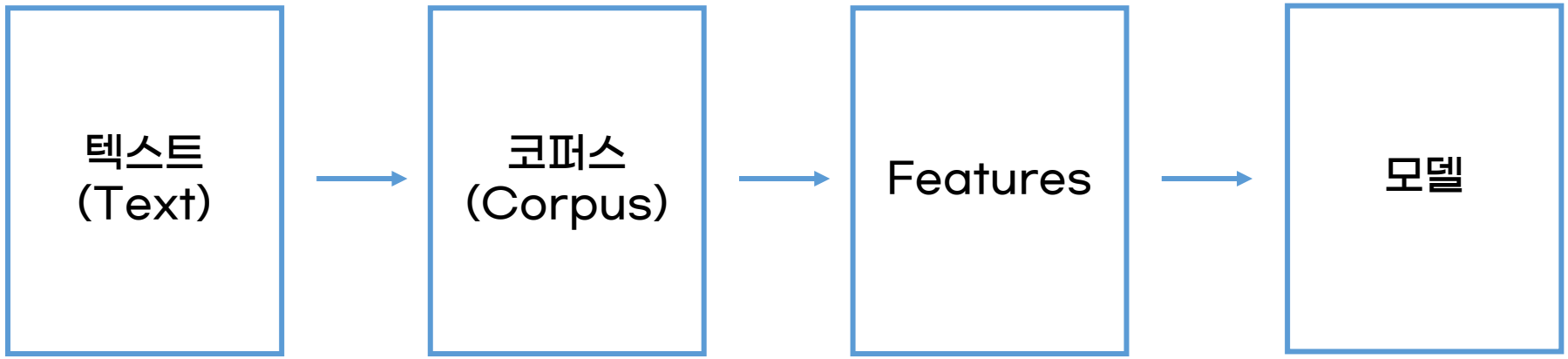
토큰화(Tokenization)

- 토큰화 : 주어진 말뭉치(Corpus)를 토큰(token)이란 단위로 나누는 작업

  토큰 -> 보통 의미가 있는 단위로 정의한다(단어, 구, 형태소 등)

- NLP에서 일반적으로 사용하는 전처리 과정.

  (RNN , LSTM 같은 신경망 모델에서 많이 사용한다)



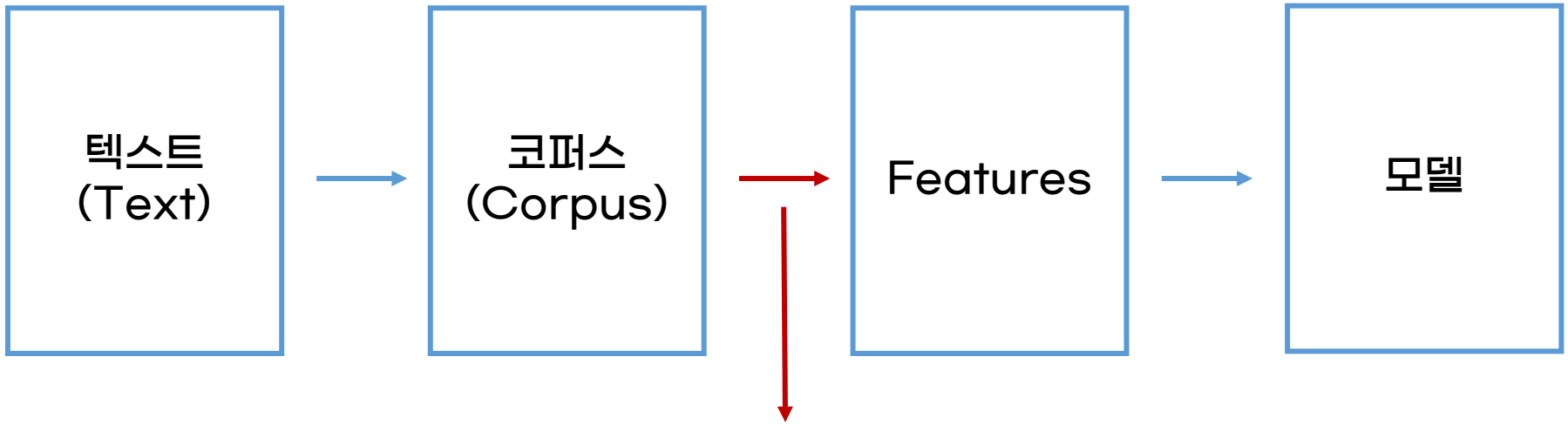
토큰화(Tokenization)

- 토큰화 : 주어진 말뭉치(Corpus)를 토큰(token)이란 단위로 나누는 작업

  토큰 -> 보통 의미가 있는 단위로 정의한다(단어, 구, 형태소 등)

- NLP에서 일반적으로 사용하는 전처리 과정.

  (RNN , LSTM 같은 신경망 모델에서 많이 사용한다)



전처리 과정 중 한 단계이다

토큰화(Tokenization)

토큰화 예시

단어 토큰화

“A dog is chasing a boy on the playground”

글자 토큰화

“Tokenization”

토큰화(Tokenization)

토큰화 예시

단어 토큰화

“A dog is chasing a boy on the playground”

“A”, “dog” , “is” , “chasing” , “a” , “boy” , “on” , “the” , “playground”

글자 토큰화

“Tokenization”

토큰화(Tokenization)

토큰화 예시

단어 토큰화

“A dog is chasing a boy on the playground”

“A”, “dog” , “is” , “chasing” , “a” , “boy” , “on” , “the” , “playground”

글자 토큰화

“Tokenization”

“T” , “o” , “k” , “e” , “n” , “i” , “z” , “a” , “t” , “i” , “o” , “n”

토큰화(Tokenization)

왜 토큰화가 필요한가?

- 품사를 수월하게 매핑 가능

  ('코딩', 'Noun'), ('하느라', 'Verb'), ('고생', 'Noun')

- 원하지 않는 토큰 제거

  욕설, 비속어 단어 제거 등

- 단어 사전(토큰의 리스트) 생성 가능

  Corpus에서 나오는 데이터를 중복 제거 후 나온 토큰으로 단어 사전을 만든다

  (Bow, TF-IDF에서 다시 등장)

불용어 (Stop words)

문장 내에서 자주 등장하면서 중요한 문법적 기능을 수행하지만,

언어 분석 시 의미가 없는 단어

문서에 출현한 용어를 빈도 별로 리스트 후 stop word list 작성
- the, of, is, a / 을, 를, 는, …
- 상황에 따라 사용자가 직접 불용어 사전을 정의하기도함

해당 분야와 의미상 관련된 용어는 포함시켜야 함

인코딩 (Encodings)

텍스트를 숫자로 표현하는 작업

- 정수 인코딩, 원-핫 인코딩 등

인코딩이 필요한 이유

- 컴퓨터가 읽을 수 있는 형식은 텍스트가 아닌 숫자

- 문서를 정량적으로 분석한 후 정성적인 해석 가능

인코딩 (Encodings)

단어 토큰화

“A”, “dog” , “is” , “chasing” , “a” , “boy” , “on” , “the” , “playground”

정수 인코딩

원-핫 인코딩

인코딩 (Encodings)

단어 토큰화

"A", "dog", "is", "chasing", "a", "boy", "on", "the", "playground"

정수 인코딩

"A" : 0, "dog" : 1 , "is" : 2 , "chasing" : 3 , "boy" :4 , "on" : 5 , "the" : 6 , "playground" : 7

원-핫 인코딩

인코딩 (Encodings)

단어 토큰화

"A", "dog" , "is" , "chasing" , "a" , "boy" , "on" , "the" , "playground"

정수 인코딩

"A" : 0, "dog" : 1 , "is" : 2 , "chasing" : 3 , "boy" :4 , "on" : 5 , "the" : 6 , "playground" : 7

원-핫 인코딩

"A" : [1, 0, 0, 0, 0, 0, 0, 0], "dog" : [0, 1, 0, 0, 0, 0, 0, 0]

"is" : [0, 0, 1, 0, 0, 0, 0, 0] , "chasing" : [0, 0, 0, 1, 0, 0, 0, 0]

"boy" : [0, 0, 0, 0, 1, 0, 0, 0] , "on" : [0, 0, 0, 0, 0, 1, 0, 0]

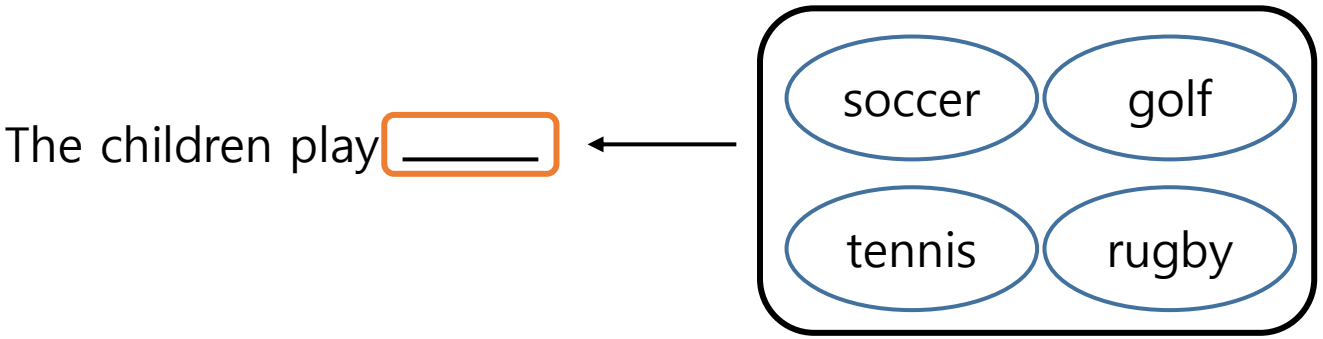
"the" : [0, 0, 0, 0, 0, 0, 1, 0] , "playground" : [0, 0, 0, 0, 0, 0, 0, 1]

3. 언어 모델

언어 모델(Language Model)

- 통계적 언어 모델 / 인공 신경망 언어 모델

- 학습을 통해 다음에 어떤 단어가 나올지 예측하는 작업



통계적 언어 모델(Statistical Language Model, SLM)

1. 조건부 확률

사건 A가 일어났을 때 사건 B가 일어날 조건부 확률

$$P(B \mid A) = \frac{P(A \cap B)}{P(A)}$$

$$P(A, B) = P(A)P(B \mid A)$$
$$= P(A \cap B)$$

통계적 언어 모델(Statistical Language Model, SLM)

1. 조건부 확률

$$P(B \mid A) = \frac{P(A \cap B)}{P(A)}$$

사건 B가 일어났을 때 사건 A가 일어날 조건부 확률

사건(A,B)이 2개

$P(A, B) = P(A)\,P(B \mid A) = P(A \cap B)$

사건(A,B,C)이 3개

사건(E1, E2,⋯ , En)이 n개

통계적 언어 모델(Statistical Language Model, SLM)

1. 조건부 확률

$$P(B \mid A) = \frac{P(A \cap B)}{P(A)}$$

사건 B가 일어났을 때 사건 A가 일어날 조건부 확률

사건(A,B)이 2개

$$P(A, B) = P(A)\,P(B \mid A) = P(A \cap B)$$

사건(A,B,C)이 3개

$$P(A, B, C) = P(A)\,P(B \mid A)\,P(C|A, B) = P(A \cap B \cap C)$$

사건(E1, E2,⋯ , En)이 n개

통계적 언어 모델(Statistical Language Model, SLM)

1. 조건부 확률

$$P(B \mid A) = \frac{P(A \cap B)}{P(A)}$$

사건 B가 일어났을 때 사건 A가 일어날 조건부 확률

사건(A,B)이 2개

$$P(A, B) = P(A)\,P(B \mid A) = P(A \cap B)$$

사건(A,B,C)이 3개

$$P(A, B, C) = P(A)\,P(B \mid A)\,P(C|A, B) = P(A \cap B \cap C)$$

사건(E1, E2,⋯ , En)이 n개

$$P\,(E1,E2,E3, \ldots ,En) = P(E1)P(E2|E1)P(E3|E1, E2)\cdots P(En|E1, \ldots , En-1)$$

이를 조건부 확률의 연쇄 법칙 이라고 한다!

통계적 언어 모델(Statistical Language Model, SLM)

2. 문장의 확률

문장이 나올 확률

$$P(x_1, x_2, \ldots, x_n) = \prod_{n=1}^{n} P(x_n | x_1, x_2, \ldots x_{n-1})$$

문장 "나는 배가 고파서"

P(나는 배가 고파서) = ??

P(나는) x P(배가|나는) x P(고파서 | 나는 배가)

통계적 언어 모델(Statistical Language Model, SLM)

2. 문장의 확률

문장이 나올 확률

$$P(x_1, x_2, \ldots, x_n) = \prod_{n=1}^{n} P(x_n | x_1, x_2, \ldots x_{n-1})$$

P(나는 배가 고파서 밥을 먹었다) =

P(나는) x P(배가|나는) x P(고파서 | 나는 배가) x … x P(먹었다 | 나는 배가 고파서 밥을)

통계적 언어 모델(Statistical Language Model, SLM)

3. 카운트 기반 접근

확률 계산식

"나는 배가 고파서" 다음 "밥을" 이 나올 확률인

P(밥을 | 나는 배가 고파서) 을 구하는 법은?

$$P(\text{밥을} \mid \text{나는 배가 고파서}) = \frac{\text{count}(\text{나는 배가 고파서 밥을})}{\text{count}(\text{나는 배가 고파서})}$$

기계가 학습한 말뭉치 안에서 각 부분이 나온 횟수를 count

통계적 언어 모델(Statistical Language Model, SLM)

3. 카운트 기반 접근

확률 계산식

"나는 배가 고파서" 다음 "**밥을**" 이 나올 확률인

P(밥을 | 나는 배가 고파서) 을 구하는 법은?

$$P(\text{밥을} \mid \text{나는 배가 고파서}) = \frac{\text{count}(\text{나는 배가 고파서 밥을})}{\text{count}(\text{나는 배가 고파서})}$$

기계가 학습한 말뭉치 안에서 각 부분이 나온 횟수를 count

count(나는 배가 고파서 밥을) = 25번
count(나는 배가 고파서) = 100번
P(밥을 | 나는 배가 고파서) = 25%

통계적 언어 모델(Statistical Language Model, SLM)

통계적 언어 모델의 한계점

- 희소 문제

$$P(\text{밥을} \mid \text{나는 배가 고파서}) = \frac{\text{count}(\text{나는 배가 고파서 밥을})}{\text{count}(\text{나는 배가 고파서})}$$

= 0번
= 0번
...?

N - gram 언어 모델

- SLM 의 한계 해결 하고자 등장

  SLM은 문장이 길어질수록 Corpus에 문장이 존재하지 않을 가능성 높다

- N – gram : 주어진 말뭉치에서 연속된 n글자/단어의 집합

- 전체 문장을 참고하는게 아닌 N개의 단어만 참고한다

N - gram 언어 모델

Example 1. “The black cat eats a muffin.”

- Unigram (N=1)

The / black / cat / eats / a / muffin

- Bigram (N=2)

The black / black cat / cat eats / eats a / a muffin

- Trigram (N=3)

The black cat / black cat eats / cat eats a / eats a muffin

- 4-gram (N=4)

The black cat eats / black cat eats a / cat eats a muffin

N - gram 언어 모델

언어 모델 적용

무시됨 !

Example. "~~An adorable little~~ boy is spreading _____ "

N-1개의 단어

N = 4 인 4-gram 모델이라 가정하자!

$$P(w \mid \text{boy is spreading}) = \frac{\text{count(boy is spreading w)}}{\text{count(boy is spreading)}}$$

- n-1 : n-gram 모델에서 특정 단어를 예측하는 데 살펴볼 앞 단어의 개수

N - gram 언어 모델

언어 모델 적용

무시됨 !

Example. "~~An adorable little~~ boy is spreading _____"

N-1개의 단어

N = 4 인 4-gram 모델이라 가정하자!

$$P(w \mid \text{boy is spreading}) = \frac{\text{count(boy is spreading w)}}{\text{count(boy is spreading)}}$$

- n-1 : n-gram 모델에서 특정 단어를 예측하는 데 살펴볼 앞 단어의 개수

Corpus에서 "boy is spreading"이 1000번

"boy is spreading insults"가 400번,

"boy is spreading smiles"가 200번 나왔다면…

$P(\text{insults} \mid \text{boy is spreading}) = 0.4$

$P(\text{smiles} \mid \text{boy is spreading}) = 0.2$

N - gram 언어 모델

N – gram 의 한계

무시됨 !

Example. "~~An adorable little~~ boy is spreading _____"

N-1개의 단어

P(insult | boy is spreading) = 0.4

P(smiles | boy is spreading) = 0.2

- 앞에 무시된 수식어로 인해 다소 맞지 않는 단어가 채택됨! -> n이 커질수록 성능이 좋아짐
- 하지만 n을 너무 키우면 앞서 봤던 말뭉치에 구문이 존재하지 않는 문제점이 발생
- n은 최대 5를 넘어가지 않도록 권장 (길어질수록 희소 문제 발생 확률 커짐)

희소 문제

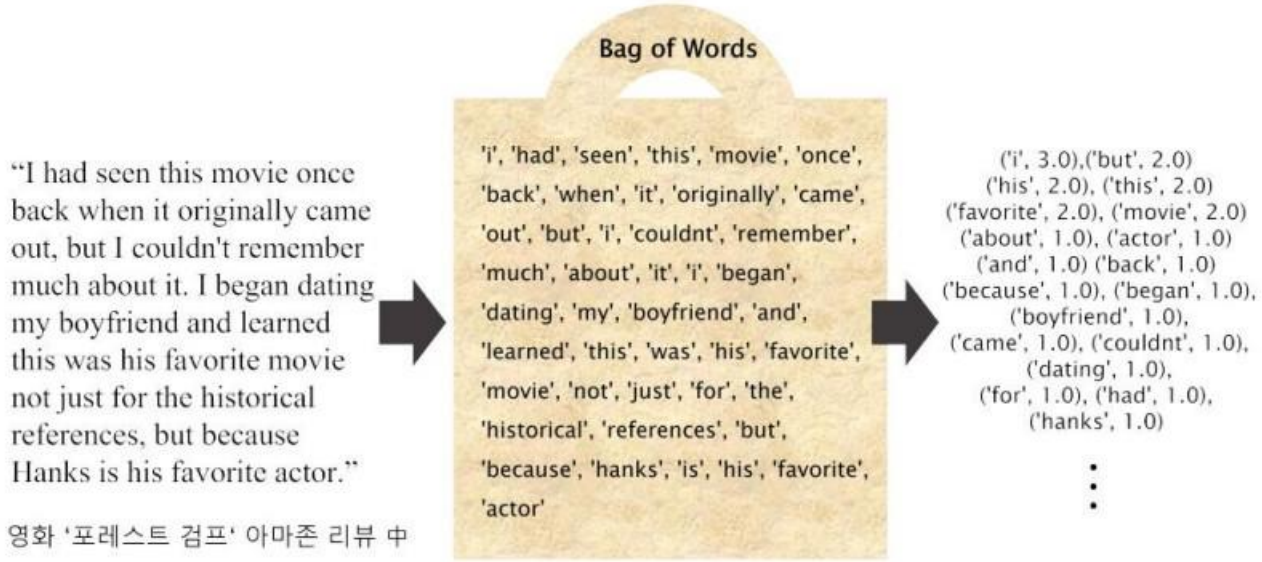
SLM에 비해 희소 문제 발생 확률이 현저히 줄긴 했지만 완전히 해결 하진 못함

단어 가방 모델 (Bag of Words, BoW)

- BoW 란?

  문자를 숫자로 표현하는 방법 중 하나.

  문장의 순서, 문맥 등 을 고려하지 않고 오직 등장 횟수에만 집중해 텍스트를 수치화



단어 가방 모델 (Bag of Words, BoW)

- BoW 만드는 과정
  1. 각 단어에 고유한 인덱스(Index) 부여
  2. 각 인덱스 위치에 단어 토큰의 등장 횟수를 기록한 벡터(vector)를 만든다

문서-단어 행렬(Document - Term matrix, DTM)

Example1: 'My wife likes to watch baseball games and my daughter likes to watch baseball games too'

Example2: 'My wife likes to play baseball

'and':0, 'baseball':1, 'daughter':2, 'games':3, 'likes':4, 'my':5, 'play':6, 'to':7, 'too':8, 'watch':9, 'wife':10

문서-단어 행렬(Document - Term matrix, DTM)

index	0	1	2	3	4	5	6	7	8	9	10
	and	base ball	daug hter	game s	likes	my	play	to	too	watch	wife
예1	1	2	1	2	2	2	0	2	1	2	1
예2	0	1	0	0	1	1	1	1	0	0	1

문서-단어 행렬(Document - Term matrix, DTM)

Example1: 'My wife likes to watch baseball games and my daughter likes to watch baseball games too'

Example2: 'My wife likes to play baseball

'and':0, 'baseball':1, 'daughter':2, 'games':3, 'likes':4, 'my':5, 'play':6, 'to':7, 'too':8, 'watch':9, 'wife':10

문서-단어 행렬(Document - Term matrix, DTM)

index	0	1	2	3	4	5	6	7	8	9	10
	and	base ball	daug hter	game s	likes	my	play	to	too	watch	wife
예1	1	2	1	2	2	2	0	2	1	2	1
예2	0	1	0	0	1	1	1	1	0	0	1

단순히 단어의 빈도로 문서를 분석하기엔 한계가 있다

TF - IDF(Term Frequency - Inverse Document Frequency)

- TF : 특정 단어가 문서에서 등장하는 빈도

- IDF : 역 문서 빈도 – 불용어 처럼 문서와 연관성이 없음에도 자주 나오는 단어 들에게 페널티를 주기 위해 사용

- TF – IDF 특징

  해당 문서의 단어 출현 횟수 & 전체 문서의 단어 출현 횟수 동시 고려해 중요도 산출

  주로 문서의 유사도, 검색 결과의 중요도, 문서 내의 특정 단어의 중요도 측정에 사용됨

TF - IDF(Term Frequency - Inverse Document Frequency)

Document1 : “I am a dog”

Document2 : “I am a cat!”

Document3 : “I am not a dog?”

Document4 : “I am not a cat, am I !?!”

< TF >

	i	am	a	dog	cat	not
Doc 1	1	1	1	1	0	0
Doc 2	1	1	1	0	1	0
Doc 3	1	1	1	1	0	1
Doc 4	2	2	1	0	1	1

TF - IDF(Term Frequency - Inverse Document Frequency)

Document1 : "I am a dog"
Document2 : "I am a cat!"
Document3 : "I am not a dog?"
Document4 : "I am not a cat, am I !?!"

$$IDF = log_{10}\frac{N}{n_t}$$

N: 전체 문서 총 개수
n_t: 용어 t가 등장하는 문서의 총 개수

* 용어 t는 반드시 우리의 vocab에 포함되어야 함!

< IDF >

	i	am	a	dog	cat	not
Doc	0	0	0	0.301	0.301	0.301

TF와IDF 구하는 방법은 매우 다양함

분모가 0이 되는 것을 방지해주기 위해서 분모에 1을 더한 형태를 사용하기도 함!

TF - IDF(Term Frequency - Inverse Document Frequency)

< TF >

	i	am	a	dog	cat	not
Doc 1	1	1	1	1	0	0
Doc 2	1	1	1	0	1	0
Doc 3	1	1	1	1	0	1
Doc 4	2	2	1	0	1	1

각 단어 별로 IDF 값을 곱해준다!

< IDF >

	i	am	a	dog	cat	not
Doc	0	0	0	0.301	0.301	0.301

TF - IDF(Term Frequency - Inverse Document Frequency)

< TF >

	i	am	a	dog	cat	not
Doc 1	1	1	1	1	0	0
Doc 2	1	1	1	0	1	0
Doc 3	1	1	1	1	0	1
Doc 4	2	2	1	0	1	1

< IDF >

	i	am	a	dog	cat	not
Doc	0	0	0	0.301	0.301	0.301

< TF-IDF >

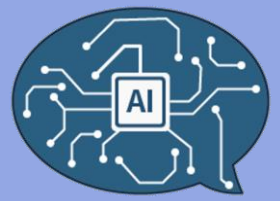
	i	am	a	dog	cat	not
Doc 1	0	0	0	0.301	0	0
Doc 2	0	0	0	0	0.301	0
Doc 3	0	0	0	0.301	0	0.301
Doc 4	0	0	0	0	0.301	0.301

TF - IDF(Term Frequency - Inverse Document Frequency)

- TF 와 IDF 를 결합하여 각 용어의 가중치 계산

- 적은 수의 문서에 용어 t가 많이 나타날 때 가장 높은 값을 갖는다
  IDF값이 크다 TF값이 크다

- 사실상 모든 문서 안에 그 용어가 나타날 경우 가장 낮은 값을 가진다



감사합니다