1. Tokenization

Tokenization?

Tokenization이란 무엇인가?

What is tokenization?

주어진 Text를 특정 단위로 구분해 Token으로 만드는 것.

- Exapmle text: "Time is an illusion. Lunchtime double so!”
- Example tokenization: [Time, is, an, illustion, Lunchtime, doudble, so]

Why tokenization is need?

Tokenziation의 목적은, 복잡한 Text가 주어지더라도 특정 단위로 구분함으로써, 의미있는 정보를 추출하거나 전체 Text를 해석 가능하게 하는 것임.

이 단위는 추출하고자하는 정보의 성격에 따라 달라질 수 있음. (Word, Sub-word, Sentecne, etc.)

즉, Tokenizatoin의 성패에 따라서 추출되는 정보나 Text에 대한 해석이 완전히 달라질 수 있음.

Why tokenization is difficult?

영어의 경우 띄어쓰기를 기반으로 한 Word tokenization이 꽤나 잘 동작하나, ’와 같은 특수문자 처리에 민감함.

- Example text: “Don't be fooled by the dark sounding name, Mr. Jone's Orphanage is as cheery as cheery goes for a pastry shop.”

한국어의 경우 조사 등에 의해 띄어쓰기를 기반으로 한 Word tokeniziation이 잘 동작하지 않음.

- Exapmle text: "이현규는 조경대와 함께 운영체제를 공부한다.”
- Example tokenization: [이현규는, 조경대와, 함께, 운영체제를, 공부한다]

더 나아가, 띄어쓰기가 정상적이지 않은 경우 의미가 크게 변화할 수 있음.

- Example text: “아버지가방에들어가신다.”
- Example tokenization: [아버지, 가방에, 들어, 가신다]

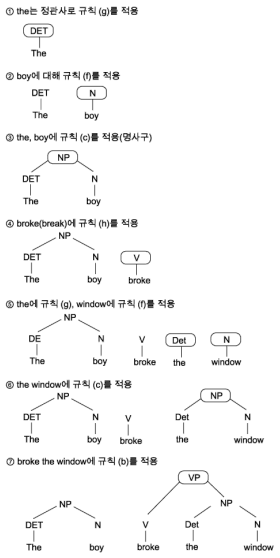
Word tokenization

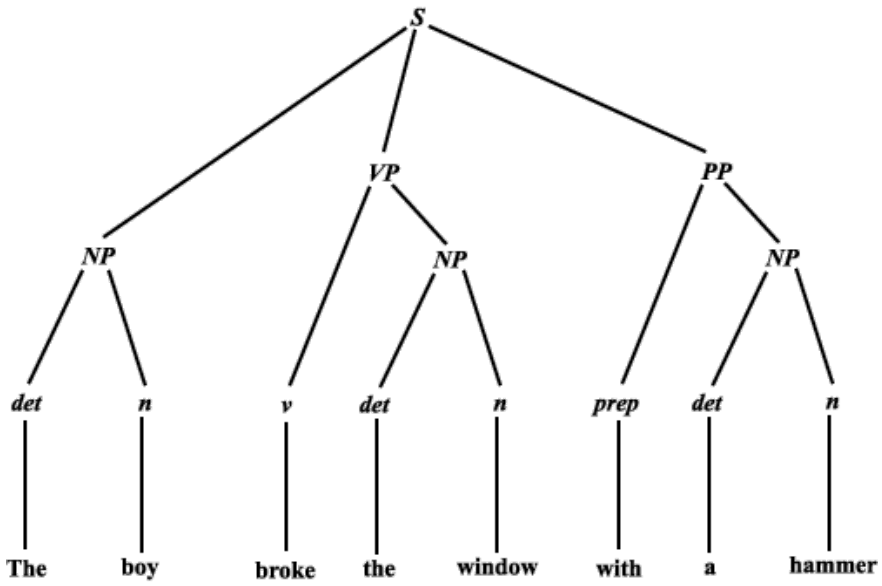
단어(Word) 기반의 tokenization에 대해 살펴본다.

Traditional approach

전통적으로는, 문장에 대한 구문분석을 통해 규칙을 정립하고 이를 활용하고자 했음.

- Example text: “The boy broke the window with a hammer”





아무리 규칙을 잘 정의하더라도, 명확한 한계가 존재했음.

- 새로 생겨난 유행어나 약어
- 완료되지 않은 형태의 Text
- 잘못 된 띄어쓰기, 오탈자 등이 포함된 형태의 Text

Cleansing and Normalization

토큰화 작업 전 노이즈 데이터를 정제함으로써 Tokenization과 결과물의 성과를 크게 향상시킬 수 있음.

다양한 방법이 있을 수 있지만, 대표적으로는 다음과 같은 방법이 있음.

- 특정 단어들의 통합
  - USA, USE
  - uh-huh, uhhuh
  - Automobile, automobile
- 불필요한 단어의 제거
  - 등장 빈도가 매우 적은 단어
  - 길이가 매우 짧은 단어

Lemmatization, Stemming

표제어(Lemma)란 단어의 뿌리를 의미하는 것으로, 사전적 원형 정도로 이해할 수 있음.

- Example words: [am, are, is]
- Example lemma: “be”

단어를 형태학적으로 파싱하게 되면, 어간(stem)과 접사(affix)로 분류할 수 있음. 여기서 어간(stem)을 추출하는 것을 Stemming이라고함.

- Example words: [formalize, allowance, electricical]
- Example stems: [formal, allow, electric]

한국어의 경우, 5언9품사의 구조를 가지고 있으며 이를 활용해 위 작업을 수행할 수 있음.

언	품사
체언	명사, 대명사, 수사
수식언	관형사, 부사
관계언	조사
독립언	감탄사
용언	**동사, 형용사**

Stopword

불용어(Stopword)란, Text 상에서 자주 등장하지만 의미론적으로나 분석에 필요하지 않은 단어를 의미함.

- Exmple ko stempwords: [은, 는, 이, 가, 을, 를, 다]
- Example en stopwords: [I, my, me, is, over]

불용어를 제거하는 것을 통해 유의미한 Token만을 활용할 수 있음.

Limitation

위와 같은 방법을 활용하면, Word level의 token을 비교적 잘 얻을 수 있게 되었음. 허나, Word tokenization 방식은 다음과 같은 한계를 가짐.

- Word의 Stem만 활용할 경우, Affix가 포함하고 있는 세부적인 정보를 누락함.
- Affix를 포함하는 Word token은 무한정 늘어날 수 있음.
- 다른 단어와 합성되어 별개의 의미를 갖는 단어를 표현할 수 없음.
- 신규 단어에 대해서 강건하지 못함.

때문에 최근의 NLP에서는 대체로 Sub-word tokenization의 방법을 활용함.

- refs.
  - https://wikidocs.net/86649
  - https://velog.io/@gypsi12/토크나이저-정리BPEWordPieceSentencePiece

Encoding

컴퓨터가 이해할 수 있고, 연산이 가능한 숫자의 형태로 자연어를 표현하는 방법을 알아본다.

Integer Encoding

정수 인코딩이란, 각 단어를 고유한 정수(Integer, Index)에 매핑시키는 방법을 의미함.

- Example mapping: {1: a, 2: barber, 3: is, 4: person, 5: good}
- Example encoded text 1: “a barber is a person” == [1,2,3,1,4]
- Example encoded text 2: “a barber is a person” == [1,2,3,5,4]

Bag of words (vocabs)

위와 같이 특정 단어를 하나의 정수로 매핑하기 위해서는, 먼저 단어 집합(Vocabulary, Vocabs)를 구성해야 함.

이는 주어진 Text의 집합(Corpus)로부터, 주어진 Token에 대해 구성할 수 있음.

- Example sentences: [John really really loves this movie, Jane really likes this song]
- Example vocabs: {John, really, loves, this, movie, Jane, likes, song}
- Example vocab map: {1:John, 2:really, 3:loves, 4:this, 5:movie, 6:Jane, 7:likes, 8:song}

Text에서 추출된 모든 Token은 이 Vocabs를 기반으로 Mappling시킴.

Vocab에는 없는 Token이 들어올 수 있으며, 이를 Out of Vocabulary(OOV)문제로 칭한다.

- Example token: “picture”
- Example vocab map: {…, 9: OOV}

- Example encoded token: 9
- Example decoded token: “OOV”

OOV token의 경우, 해당 token에 대한 의미를 제대로 파악 할 수 없기 때문에 이를 최소화 하려는 방향으로 tokenizer를 구성하고자 함. (reated “sub-word tokenization”)

OOV를 줄이는 간단한 방법은, 새로운 Token이 들어올 때마다 이를 Vocab에 추가하는 방식이나, 이는 무한정 Vocab이 늘어날 수 있으며 Vector로 표현 시 차원의 크기 자체가 늘어나는 문제가 생김.

One-hot Encoding

One-hot Encoding은 Vocab을 기반으로 Token을 Vector로 표현하고자 하는 방식임.

Encoded vector는 Vocab만큼의 차원의 크기를 가지며, 각 Token은 해당하는 차원에서 1의 값을 갖는다.

- Example vocab map: {'나': 0, '는': 1, '자연어': 2, '처리': 3, '를': 4, '배운다': 5}
- Example word: "자연어”
- Example vector: [0, 0, 1, 0, 0, 0]

알고리즘의 순서와 예시는 다음과 같다.

1. 먼저, 문장에서 나타난 단어들의 집합을 구한다.
2. 각 단어를 One Hot Encoding하여 벡터로 표현한다.
3. 각 단어를 표현하는 벡터의 합을 통해 문장을 표현한다.

- **Step 1. Constructing the vocabulary containing unique words**
  - Example sentences: “John really really loves this movie“, “Jane really likes this song”
  - Vocabulary: {“John“, “really“, “loves“, “this“, “movie“, “Jane“, “likes“, “song”}

중복단어는 집합의 성질에 의해 1개만 남게 된다.

- **Step 2. Encoding unique words to one-hot vectors**
  - **Vocabulary: {"John", "really", "loves", "this", "movie", "Jane", "likes", "song"}**
    - John: [1 0 0 0 0 0 0 0]
    - really: [0 1 0 0 0 0 0 0]
    - loves: [0 0 1 0 0 0 0 0]
    - this: [0 0 0 1 0 0 0 0]
    - movie: [0 0 0 0 1 0 0 0]
    - Jane: [0 0 0 0 0 1 0 0]
    - likes: [0 0 0 0 0 0 1 0]
    - song: [0 0 0 0 0 0 0 1]

단어의 갯수만큼의 차원이 생기고, 각 차원에 해당하는 값만 1인 One Hot Vector로 표현된다.

- **A sentence/document can be represented as the sum of one-hot vectors**
  - Sentence 1: "John really really loves this movie"
    - John + really + really + loves + this + movie: [1 2 1 1 1 0 0 0]
  - Sentence 2: "Jane really likes this song"
    - Jane + really + likes + this + song: [0 1 0 1 0 1 1 1]

각 문장은 단어의 합으로 표현되나, 순서에 대한 표현은 불가능하다.

Limitation

One-hot encoding은 다음과 같은 한계점을 가짐.

- Sparse vector이므로 저장 공간이 비효율적임.
- 표현된 모든 Token은 다른 Token 간의 거리가 모두 $\sqrt{2}$이며, Cosine 유사도는 0임.

위와 같은 문제를 해결하기 위해 워드 임베딩이라는 방식이 도입됨.

- refs.
  - https://wikidocs.net/22644