**텍스트 마이닝 강의자료**

1. 텍스트 마이닝이란?

비정형 텍스트 데이터에서 가치와 의미가 있는 정보를 찾아내는(Mining) 기술

2. 웹 크롤링 ( Web Crawling)

웹 문서 내에서 원하는 데이터를 자동으로 긁어 오는 것

텍스트 마이닝의 기반이 되는 텍스트를 마련

■ 웹 스크래핑 (Web Scraping)

<다양한 웹 스크래핑 도구들>

1) 리퀘스트 (requests)

: 특정 url에 정보를 요청하여 결과를 받는 모듈

설치 방법 : pip install request

2) 뷰티풀숲 (beautifulsoup)

: 받아온 html string 정보를 분석해주는 도구

설치 방법 : pip install bs4

3) 셀레니움 (Selenium)

: 웹 앱을 테스트하는데 사용하는 프레임워크인데 강력한 크롤링 도구로 사용됨

설치 방법 : pip install selenium

Webdriver라는 API를 통해 Crome, Firefox 등의 다양한 브라우저를 제어

브라우저를 직접 동작시킴으로써, Javascript를 이용해 비동기적[[1]](#footnote-1)인 컨텐츠를 가져올 수 있음

Requests 라이브러리와는 다른 동적페이지의 결과 또한 접근가능

Ex) 로그인을 해야만 볼 수 있는 화면에 대한 크롤링을 가능하게 해줌

<가능성>

* 범용성 : URL을 획득할 수 있다면 어떤 웹사이트에서든 활용 가능
* 직관, 편리성 : 필요한 Data를 직접 확인하면서 수집 가능

<한계>

* Not Polite : 비공식적이고 무리한 요청은 서버 과부하 가능성
* 제한 : 특정 Page들은 URL을 숨기거나 요청을 차단하는 경우

■ API 이용

API (Application Programming Interface)

: 라이브러리에 접근하기 위한 규칙들을 정의한 것

프로그래머가 라이브러리가 제공하는 여러 함수를 이용하여 프로그램을 작성할 때 해당 함수의 내부 구조는 알 필요없이 단순히 API에 정의된 입력 값을 주고 결과 값을 사용할 수 있게 해줍니다.

* 웹사이트에서 제공하는 API를 이용해 공식적으로 데이터를 요청해서 받아오는 방법

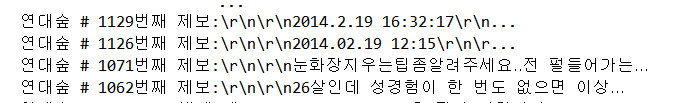
API를 제공하는 대표적인 웹사이트 : 페이스북, 네이버

<한계>

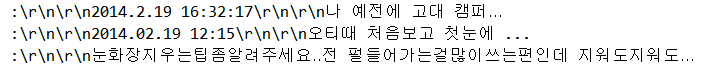
제한된 형태의 데이터 밖에 얻을 수 없음.

3. 전처리 (preprocessing)

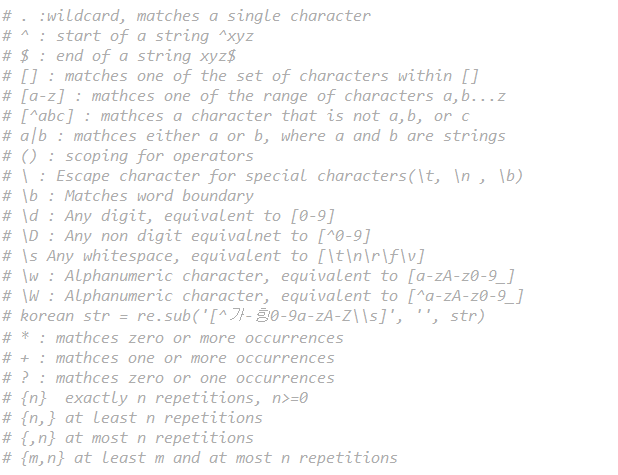
1.정규표현식을 통한 기본적인 전처리

예시) 

-> Str.replace(r’^(연대숲 # [~0-9가-힝\s#]+ 제보)’, ‘’) (\는 역/ 입니다.)



흔히 알려진 토큰화, Stemming, Lematizaiong을 하기 이전에 정규표현식을 통한 기본적인 전처리가 필요합니다. 정규표현식을 사용하기 위해서는 아래와 같은 내용들을 숙지해야 합니다. 이는 흔히 저희가 사용하는 ‘노가다’의 범주에 속하고 많은 시간이 필요합니다.



2. 띄어쓰기

토큰화, 그리고 Stemming이나 Lematizationg을 하기이전에는 Text자료의 띄어쓰기가 어느정도는 되어 있어야 합니다. 띄어쓰기가 되어 있지 않은 상황에서 바로 어간을 추출하거나 단어를 기본형으로 만들어 줄 수는 없습니다. 정규 표현식을 잘 활용하여 띄어쓰기를 어느 정도 해줄 순 있지만(기본적으로 띄어쓰기가 잘 된 자료의 경우는 가능합니다) 이러한 방식에는 한계가 있기 때문에 딥러닝을 활용하여 띄어쓰기를 합니다. 하지만 사실상 저희가 띄어쓰기가 된 글들을 학습시켜 저희가 분석하고자 하는 글에 띄어쓰기를 적용하는 데에는 무리가 있습니다.

그래서 뉴스 1억건을 Training data로 학습시켜 띄어쓰기를 가능하게 해주는 Package가 공유되고 있습니다. 파이썬, R모두 사용이 가능합니다.

예시)



library(KoSpacing)

spacing("김형호영화시장분석가는'1987'의네이버영화정보네티즌10점평에서언급된단어들을지난해12월27일부터올해1월10일까지통계프로그램R과KoNLP패키지로텍스트마이닝하여분석했다.")

[1] "김형호 영화시장 분석가는 '1987'의 네이버 영화 정보 네티즌 10점 평에서 언급된 단어들을 지난해 12월 27일부터 올해 1월 10일까지 통계 프로그램 R과 KoNLP 패키지로 텍스트마이닝하여 분석했다."

정규표현식을 통해 기본적인 전처리를 하고, 띄어쓰기가 되어 잘 되어 있다면, 이후에 정규화, 토큰화, 불용어 제거, 어간 추출(Stemming)을 무리없이 할 수 있습니다.

3.정규화

아직 한단계가 남아있습니다. 정규화입니다. 정규화란

* 한국어를 처리하는 예시입니닼ㅋㅋㅋㅋㅋ -> 한국어를 처리하는 예시입니다 ㅋㅋ

위와 같이 단어를 정규화 해주는 것입니다.

4.토큰화

토큰화란 분석의 단위로 단어들을 분리시키는 것을 의미합니다. 주로 단위 또는 형태소 단위로 구분하게 됩니다. 그렇기 때문에 띄어쓰기가 기본적으로 이루어져 있어야 이 작업이 잘 이루어 질 수 있습니다. 토큰화가 이루어진 후에 해당 단어들에 대해 불용어를 제거하거나 어간을 추출할 수 있습니다.

* 한국어를 처리하는 예시입니다 ㅋㅋ -> 한국어Noun, 를Josa, 처리Noun, 하는Verb, 예시Noun, 입Adjective, 니다Eomi ㅋㅋKoreanParticle

문장을 형태소 쪼개는 동시에 해당 단어의 품사를 찾아내줍니다. 품사가 있으면 동음이의어를 구분할 수 있습니다.

5.어간추출(Stemming)

어간은 품사의 기본형을 찾아주는 것입니다. 토큰화과정에서 품사에 대한 정보가 있어야 가능한 과정입니다.

* 한국어를 처리하는 예시입니다 ㅋㅋ -> 한국어Noun, 를Josa, 처리Noun, 하다Verb, 예시Noun, 이다Adjective, ㅋㅋKoreanParticle

이러한 작업을 통해 TDM을 구성할 수 있습니다.

5. TDM

TDM은 Term Document Matrix(TDM)입니다. 텍스트를 단어 또는 형태소로 쪼개어 표 형태로 정리하는 것입니다.



위와 같이 표 형태로 정리하게 되면 다양한 통계적 기법을 적용할 수 있게 되고 이를 통해 LDA(문서의 주제를 뽑아내는 알고리즘)와 같은 다양한 기법들을 적용할 수 있게 됩니다.

4. Text Modeling

**[Bag of Words]**

머신러닝 알고리즘을 이용한 데이터마이닝 작업에서는 자연어 텍스트의 표현을 위해 여러가지 방식을 사용한다. 자연어 텍스트는 그냥 줄글이기 때문에 이를 분석하기 위해서는 어떠한 툴이 필요한데, 그 모델들 중 하나로 Bag of Words 모델이 있다.

Bag of Words 모델은, 텍스트를 단어의 집합으로 표현하는 것이다. 자연어 텍스트는 단어별로 쪼개져 섞인 뒤에 같은 단어끼리 뭉치게 된다. 이 과정에서, 단어의 순서나 문장 및 문법구조는 사라지고 최종적으로 단어 및 단어의 개수로 표현된다.

즉, Bag of Words = Representation in a Vector Space

궁극적으로 frequency of words를 보고 그 text의 contents를 추론하는 방법.

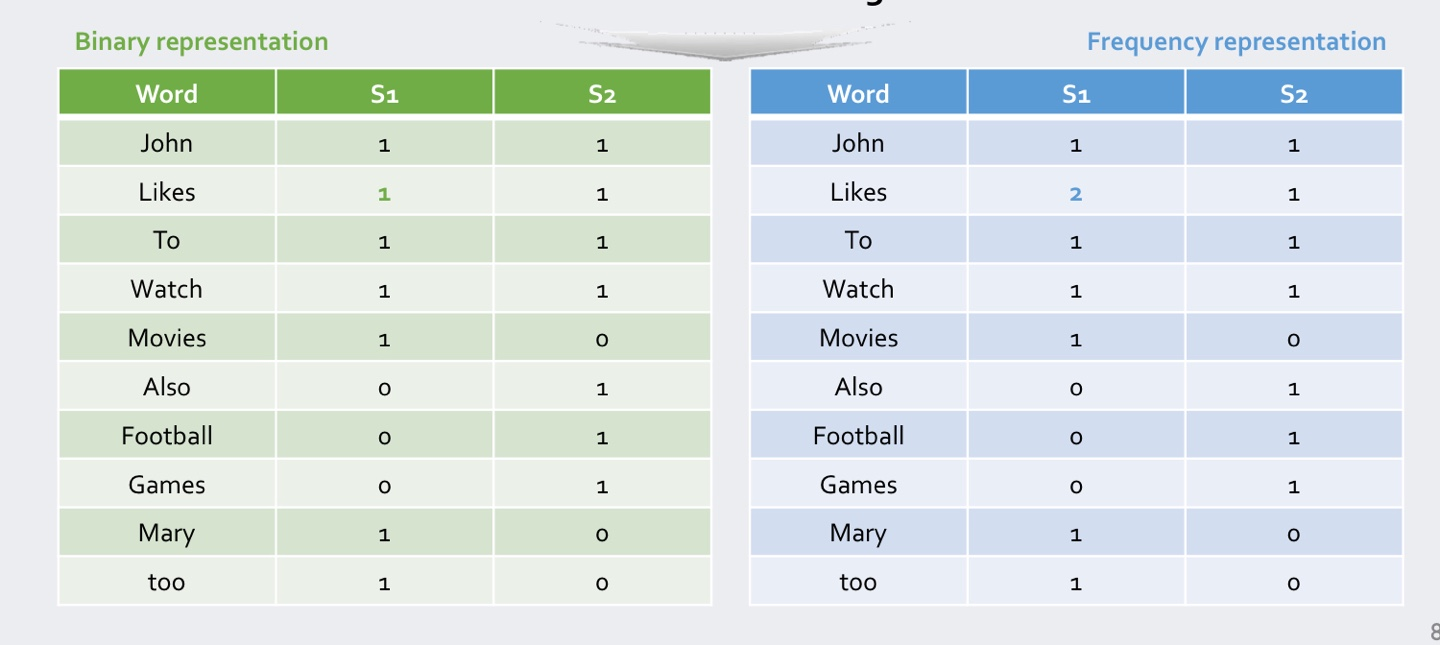
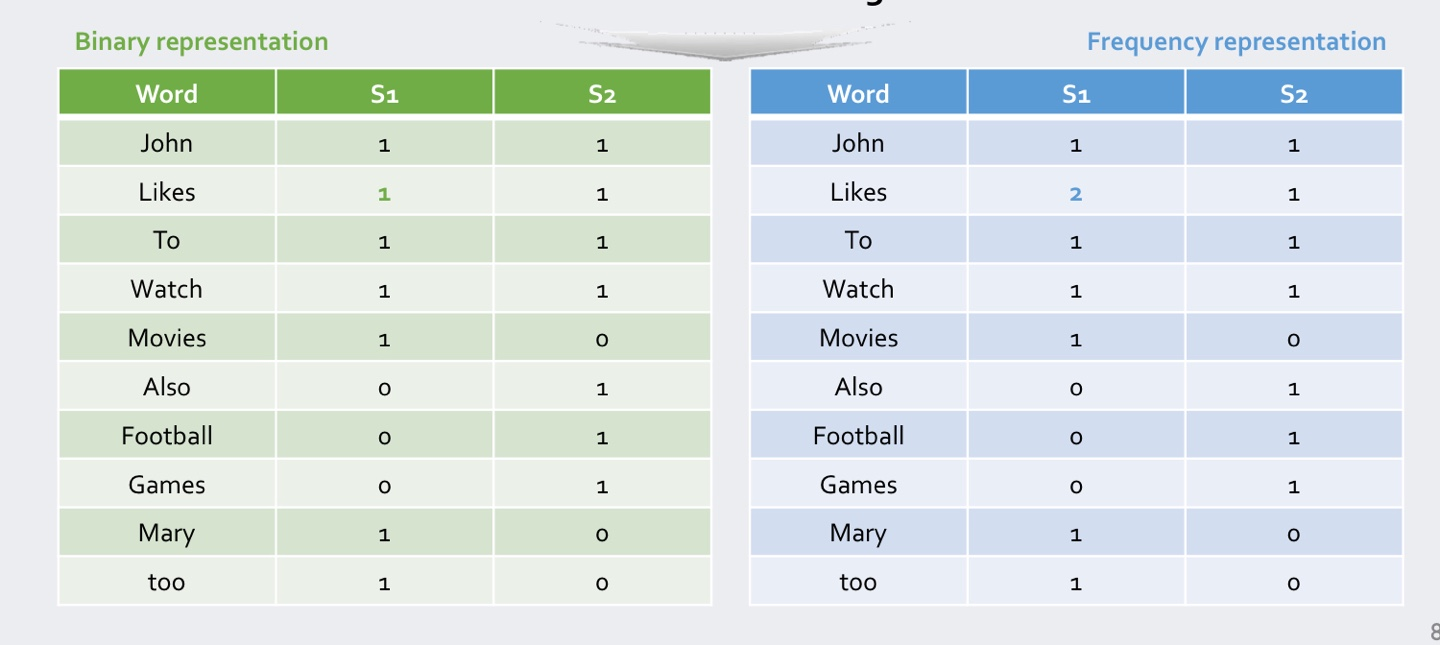
* 많이 사용된 words가 그 text의 핵심 단어일 가능성이 높다. 이렇게 문서에 포함된 단어들의 분포 및 frequency를 보고 이 문서의 종류를 판단하는 것이 Bag of Words의 궁극적인 목적이다.

Bag of Words 개념은 단어 순서를 고려하지 않았기 때문에 이를 보완하기 위해 N-Gram Model을 사용하기도 한다.

<Example>

S1: John likes to watch movies. Mary likes too.

S2: John also likes to watch football game.



(CF) N-Gram Model

N개의 단어를 하나로 묶어서, 하나로 취급한 후 그에 대한 frequency를 계산. 전 단어들을 고려한다는 맥락의 모델.

**[Porter Stemming & Lemmatizing]**

Bag of Words 모델을 좀 더 효율적으로 사용하려면 Porter Stemming과 Lemmatizing을 적용해야 한다. 참고로 이는 파이썬 NLTK 패키지를 이용하면 할 수 있다.

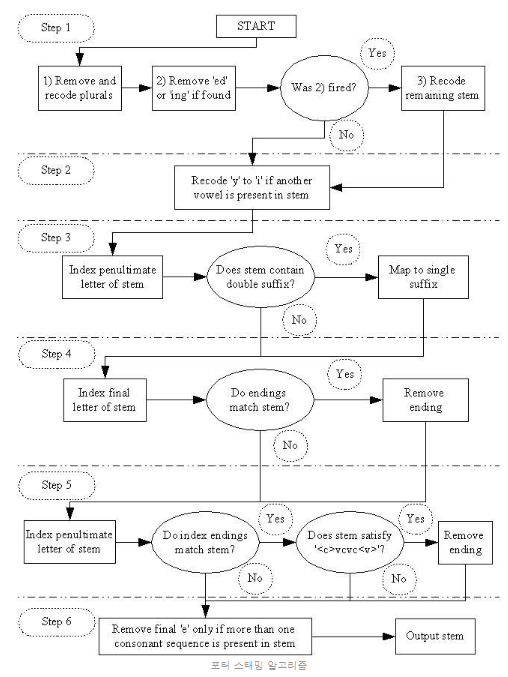
1. Porter Stemming

Porter 박사가 만든 Stemming Algorithm이다. 여기서 Stemming은 어간추출을 의미한다. 즉 형태가 변한 단어로부터 군더더기(접사, 어미, 분사형태 등)을 제거하고 그 단어의 원형을 추출해 내는 알고리즘이다.

Ex)

cats -> cat, going -> go 등

그런데 만약 going에서 ing를 제거하여 go를 만드는 로직을 계속 적용하다 보면 sing은 s만 남게 되는데, 이 Porter Stemming알고리즘은 이 문제점도 해결할 수 있다고 한다. (다만 이 알고리즘은 영어에 최적화)



1. Lemmatization

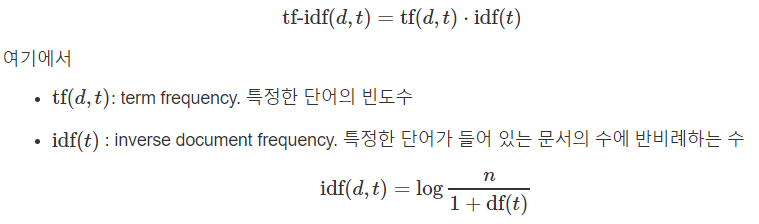
Stemming에서 약간 진화된 형태. Stemming과 마찬가지로 그 단어의 원형을 추출해내지만, Stemming과의 차이는 Lemmatization은 그 단어가 문장 속에서 어떤 품사(Part-of-Speech)로 쓰였는지까지 판단한다는 것이다.

그러므로 Stemming보다는 좀 더 복잡한 형태의 알고리즘이 필요하다.

**[TF-IDF 개념]**

텍스트를 단어로 split하고, porter stemming 혹은 lemmatization을 거친 후 단어별로 frequency를 계산해 내기만 하면, 중요도가 떨어지는 단어들의 frequency가 높게 나올 수 있다. 보통 우리는 frequency가 높은 단어가 텍스트 내에서 중요한 기능을 한다고 판단하는데, 예를 들어 “this”, “is”, “and”와 같은 지칭어, 접속사 등이 높은 frequency를 차지할 수 있다. 이는 분석에 좋지 않은 결과를 가져올 것이다.

그러므로 TF-IDF(Term Frequency – Inverse Document Frequency)라는 기법을 적용한다. 이는 중요도가 떨어지는 단어에 대해서 가중치를 적게 주는 방법이다. 또한 특정한 문서에서만 쓰이는 단어들, 의학 문서에만 들어갈 법한 의학 용어들, 철학 문서에만 쓰일 법 한 철학 용어들은 그 텍스트의 특성을 잘 나타내 주는 단어이므로 가중치를 높게 부여한다.



TF(단어 빈도) = 특정한 단어가 문서 내에 얼마나 자주 등장하는지를 나타내는 값

IDF(역문서 빈도) = DF(문서 빈도)의 역수정도의 값, 즉 다른 문서에서 자주 등장하면 단어의 중요도가 낮아짐.

TF-IDF = TF\*IDF 🡪 이 값이 높을수록, 다른 문서에 많이 등장하지 않고 해당 문서에서는 자주 등장하는 단어.

#참고 소스

<https://blog.naver.com/tmakdlfwotjd/221317362945>

<http://pureugong.tistory.com/119>

<https://nesoy.github.io/articles/2017-11/tf-idf>

등등..

1. 동기적 수행이란 요청하고 응답이 다 끝날 때까지 대기하고 응답이 오면 다음 업무를 처리하는 방식

   비동기적 수행이란 요청을 하고 바로 다음 일을 수행할 수 있다는 것 [↑](#footnote-ref-1)