import 형태소 분석

Updated on 2019-11-01

변영인

---

1. Intro

Data Scientist
1200억 매출 /500명 / 클라이언트 1200사 이상
서울 강남구

간편하게 지원하세요

수집 및 분석(크롤링, **형태소**분석, TA등) 4. Tensorflow를 활용한 Deep Learning Model 구현 5. AWS 클라우드와 상용 분석 툴(SAS 등), 그리고 메가존의 경험 기반 기술을 결합한 새로운 **형태**...

7일 전 · 스크랩 · 더보기

인공지능(AI)·빅데이터 정규직 - 경력무관
한국스코어링
서울 마포구

연봉 28,000,000원 - 30,000,000원

채봇 자연어 처리 언어 - 언어학, 언어자원구축, 언어분류체계 구축, . 분류체계 표준화 경험 우대 . **형태소**분석, 개체명 인식, 구문분석 등 . 학원 전산과정을 이수한 분 영어 의사소통 가능자 우대 ・모든 것이 처음이지만...

인크루트 · 23일 전 · 스크랩 · 더보기

자연어처리전문가 금융챗봇 만들어 갈 엔지니어
파운트
서울 충정로역

발하고 Python으로 구현 및 성능 검증을 진행 - 자연어 전처리 (**형태소** 분석, 개체명인식), word vectorizer (TFIDF, word... 검증을 진행 - 자연어 전처리 (**형태소** 분석, 개체명인식), word...

(출처)

import 형태소 분석

2. '형태소 분석'이란? (참고자료)

a. 형태소(morpheme)

- 뜻을 가진 가장 작은 말의 단위
- **자립 형태소** : 접사, 어미, 조사와 상관없이 자립하여 사용할 수 있는 형태소. 그 자체로 단어
  - 체언(명사, 대명사, 수사), 수식언(관형사, 부사), 감탄사 등
- **의존 형태소** : 다른 형태소와 결합하여 사용되는 형태소
  - 접사, 어미, 조사, 어간
- 예) 변영인이 빅데이터를 배웠다
  - 자립 형태소 : 변영인, 빅데이터
  - 의존 형태소 : -이, -를, 배-, -웠, -다

자연어처리 (출처)

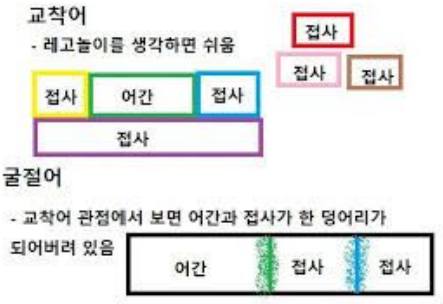
b. 자연어 처리

- SNS(트위터, 인스타그램), 채팅(카카오톡), STT(SIRI, Clova) 등의 텍스트 비정형 데이터를 처리하기 위해서는 '**자연어 처리**'를 통해 *문장의 요소와 의미를 분석*하는 과정이 필요.
  - **토큰화**(tokenization) : 주어진 코퍼스(corpus, 언어를 분석하기 위한 표본들, a.k.a 말뭉치)에서 토큰(token)이라 불리는 단위로 나누는 작업, 대개 토큰은 최소 의미 단위로 나눔

import 형태소 분석

c. 한국어에서의 자연어처리의 어려움

- 자연어 처리에 있어서, **한국어는 영어와는 달리 띄어쓰기만으로는 문장을 분석할 수 없음.**
  - 한국어의 띄어쓰기 단위가 되는 단위를 '어절'이라고 하는데 즉, 어절 토큰화와 단어 토큰화가 같지 않음
  - 근본적인 이유는 한국어가 영어와는 다른 형태를 가지는 언어인 교착어라는 점에서 기인.
  - **교착어** : 조사, 어미 등을 붙여서 말을 만드는 언어



(출처)

- 한국어에서 영어에서의 단어 토큰화와 유사한 형태를 얻으려면 어절 토큰화가 아니라 *형태소 토큰화를 수행해야함*
  - 한국어는 어절이 독립적인 단어로 구성되는 것이 아니라 조사 등의 무언가가 붙어있는 경우가 많아서 이를 전부 분리해줘야 함
- 이 문서에서는 **(텍스트 전처리를 위한) 형태소 토큰화 및 품사 태깅**을 형태소 분석이라고 지칭
  - 언어학 또는 국문학과에게 이런 소리 했다간 ~~쳐맞음~~ 혼남

이런 걸 만들기 위해 문장을 쪼개는 과정이 (일단) 형태소 분석 (출처)

import 형태소 분석

3. 오픈소스 한국어 처리기 (OKT)

a. 설치 및 Docs

- 테스트 : https://openkoreantext.org/
- 깃허브 (링크) ⇐ 자세한 사용법 및 예제 참고
  - 제공 기능

    정규화 normalization (입니닼ㅋㅋ -> 입니다 ㅋㅋ, 샤릉해 -> 사랑해)

    - 한국어를 처리하는 예시입니닼ㅋㅋㅋㅋㅋ -> 한국어를 처리하는 예시입니다 ㅋㅋ

    토큰화 tokenization

    - 한국어를 처리하는 예시입니다 ㅋㅋ -> 한국어Noun, 를Josa, 처리Noun, 하는Verb, 예시Noun, 입니다Adjective(이다), ㅋㅋKoreanParticle

    어근화 stemming (입니다 -> 이다)

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

    어구 추출 phrase extraction

    - 한국어를 처리하는 예시입니다 ㅋㅋ -> 한국어, 처리, 예시, 처리하는 예시

  - 예제

- Maven

```
<dependency>
   <groupId>org.openkoreantext</groupId>
   <artifactId>open-korean-text</artifactId>
   <version>2.1.0</version>
 </dependency>
```

import 형태소 분석

b. 예제

- Example (링크)
- JAVA 인스타 크롤링 및 해시태그 추출, 명사 빈도 분석 예제 (링크)
  - 인스타 크롤링 데이터 정규화

```
public static List<String> getInstaPosts(String tag) throws IOException {
    return Jsoup.connect("https://www.instazu.com/tag/" + tag).get()
            .select(".box-photo").stream()
            .map(v -> OpenKoreanTextProcessorJava.normalize(
                    v.selectFirst(".photo-description").text()).toString())
            .collect(Collectors.toList());
```

  - 태그 리스트 만들기

```
public static List<List<String>> makeListByTag(String tag, List<String> list) {
    return list.stream()
            .map(v -> OpenKoreanTextProcessorJava.tokenize(v))
            .map(v -> OpenKoreanTextProcessorJava.tokensToJavaKoreanTokenList(v))
            .map(v -> v.stream().filter(v2 -> v2.getPos().toString().equals(tag))
                    .map(v2 -> v2.getText())
                    .collect(Collectors.toList()))
            .collect(Collectors.toList());
```

토큰화 -> 토큰리스트받기 -> Pos(품사) 값 바탕으로 분류

```
// Hashtag
[#첫줄, #백기짬뽕, #탕수육, #바나나킥, #일상, #행복]
[#바나나, #새우덮밥, #수제비, #탕수육, #점심, #단체급식
[#일상, #데일리, #instadaily, #instagood, #오랜
[#점심, #짜장면, #짬짜면, #탕수육, #일상생활]
[#제주도, #제주도여행, #오드랑베이커리, #마농바게트, #짬
```

(해시태그 추출)

```
// Noun
[동네, 맛집, 짬뽕, 최애, 꽈자, 둥이, 때, 공부, 꽈]
[오늘, 새우, 밥, 매콤, 수제비, 국, 탕수육]
[탕수육, 한국, 관]
```

(명사 추출)

  - 빈도 분석하기

```
public static Stream<Entry<String, Long>> makeFreqTable(int num, List<List<String>> list) {
    List<String> all = new ArrayList<>();
    for(List<String> v : list) {
        all.addAll(v);
    }
    return all.stream().collect(Collectors.groupingBy(String::toString, Collectors.counting()))
            .entrySet().stream()
            .sorted(Comparator.comparing((Entry<String, Long>::getValue)).reversed())
            .limit(num);
}
```

리스트를 합쳐 줌 -> 단어 기준으로 그룹핑 -> 그룹별로 카운트 -> Sorting 및 확인하기 쉽게 Entry화 -> 내림차순 정렬 및 최빈 n개

```
// Noun Frequency
애월=28
탕수육=19
짬짜미=17
제주시=10
해안로=10
```

import 형태소 분석

■ 하둡 적용 예제

- Mapper에 적용

```
public void map(Object key, Text value, Context context)
throws IOException, InterruptedException {
                        // Hadoop에 Okt 명사 분석 적용
                        for(String v :
OpenKoreanTextProcessorJava
                                .tokensToJavaKoreanTokenList(

OpenKoreanTextProcessorJava.tokenize(value.toString()))
                                .stream()
                                        .filter(v ->
v.getPos().toString().equals("Noun"))
                                        .map(v ->
v.getText()).collect(Collectors.toList())) {
                                word.set(v);
                                context.write(word, one);
                        }
                }
```

- Maven Assembly Plugin 설치

```
<plugin>
<groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-assembly-plugin</artifactId>
        <configuration>
                <descriptorRefs>

<descriptorRef>jar-with-dependencies</descriptorRef>
                        </descriptorRefs>
                </configuration>
<executions>
        <execution>
        <phase>package</phase>
        <goals>
                        <goal>single</goal>
                </goals>
        </execution>
</executions>
</plugin>
```

- maven install로 dependencies 포함된 jar 생성
- 결과

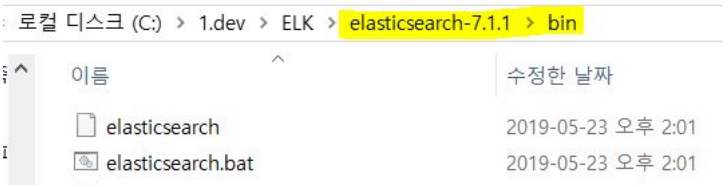
```
[hadoop@dn01 ~]$ hdfs dfs -cat /output6/part-r-00000
가기      14
가능성    7
가드      7
가장      35
가정폭력          7
가지      28
가현      7
가형      7
각광      7
각인      7
각주      14
간        21
간접      14
간주      14
간함      7
```

import 형태소 분석

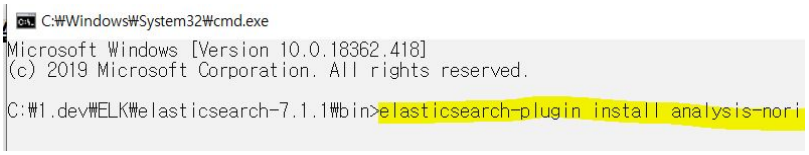
4. 노리 플러그인

a. 설치 및 Docs

- 공식 소개 (링크)
- Install (링크)
  - ~₩(엘라스틱서치폴더)₩bin 에서 주소표시줄에 cmd 입력



  - elasticsearch-plugin install analysis-nori (cmd창에)



b. Test

- Analyzer 실행 (참고)

```
post _analyze
{
"analyzer":"nori",
"text":"교대역 14번 출구는 어디에 있죠?"
}
```

import 형태소 분석

■ **인덱스 맵핑** (참고1) (참고2)

delete test

put test
{"settings":{"number_of_shards":2,"number_of_replicas":0,"index":{"analysis":{"tokenizer":{"nori_tokenizer_mixed":{"type":"nori_tokenizer","decompound_mode":"mixed"}},"analyzer":{"korean":{"type":"custom","tokenizer":"nori_tokenizer_mixed","filter":["nori_readingform","lowercase","nori_part_of_speech_basic"]}},"filter":{"nori_part_of_speech_basic":{"type":"nori_part_of_speech","stoptags":["E","IC","J","MAG","MAJ","MM","SP","SSC","SSO","SC","SE","XPN","XSA","XSN","XSV","UNA","NA","VSV"]}}}},"mappings":{"properties":{"message":{"type":"text","analyzer":"korean"}}}}

■ **테스트 데이터**

post test/_doc/1
{ "message" : "정부가 신도시 교통문제를 해결하기 위해 광역철도망을 대폭 확충하는 방안을 발표했다. 운정·검단 등 서울 서북부 '2기 신도시'를 중심으로 지하철을 연장하고 또다른 광역급행철도(GTX)를 착공하는 방안도 거론된다. 지역주민의 민원이 제기됐던 3호선, 9호선 연장과 인천2호선 신안산선 연결 방안 등도 본격 추진된다. 정부는 2020년까지 광역철도 노선을 현재의 2배 수준으로 확충해 경기권 주요 신도시에서 30분 내 서울 진입이 가능하도록 한다는 방침이다."}

post test/_doc/2
{ "message" : "31일 국토교통부 대도시권광역교통위원회는 이 같은 내용 '광역교통2030' 비전을 발표했다. 핵심은 1·2기 신도시의 광역철도 확충이다. 지하철 3호선 대화~운정 구간을 연장하고 9호선 강일~미사 구간을 연장하는 방안이 본격 추진된다. 각각 경기 고양·파주와 하남 쪽으로 철도망이 확충되는 것이다. 대화~운정 연장은 현재 사업재기획 용역이 진행되고 있다. 대광위는 이를 서울시 도시철도망구축계획에 반영하는 방안을 검토 중이다. 인천2호선을 신안산선(안산·시흥~여의도)으로 연결하는 방안도 추진된다. 김포한강선은 검단으로 연결된다. 현재 김포한강선 방화~양곡 구간 사업에 대한 사전타당성 조사 용역 중으로, 이를 통해 김포한강선이 검단 지역으로 이어지게 된다."}

import 형태소 분석

■ Search API (참고)

- 기존에 whitespace 기준으로 부정확하게 나뉘던 text 계열의 분석을 좀 더 정확하고 세밀하게 가능
- 일치도에 따른 scoring도 정확성을 높일 수 있음

```
get test/_search?
{"query":{"match":{"message":"광역"}}}

get test/_search?
{"query":{"match":{"message":"김포"}}}
```

import 형태소 분석

■ Term Vector API (참고)

- Analyze된 내용을 단어 단위로 확인할 수 있음

```
get test/_termvectors/1?fields=message

get test/_termvectors/2?fields=message

get test/_termvectors/2?fields=message
{"offsets":false,"payloads":false,"positions":false,"term_statistics":true,"field_statistics":false}
```


```
"_index" : "test",
"_type" : "_doc",
"_id" : "1",
"_version" : 1,
"found" : true,
"took" : 43,
"term_vectors" : {
  "message" : {
    "field_statistics" : {
      "sum_doc_freq" : 142,
      "doc_count" : 2,
      "sum_ttf" : 190
    },
    "terms" : {
      "2020년까지" : {
        "term_freq" : 1,
        "tokens" : [
          {
            "position" : 86,
            "start_offset" : 180,
            "end_offset" : 187
          }
        ]
      },
      "2기" : {
        "term_freq" : 1,
        "tokens" : [
          {
            "position" : 32,
            "start_offset" : 63,
            "end_offset" : 65
          }
        ]
```

```
"_index" : "test",
"_type" : "_doc",
"_id" : "2",
"_version" : 1,
"found" : true,
"took" : 23,
"term_vectors" : {
  "message" : {
    "field_statistics" : {
      "sum_doc_freq" : 142,
      "doc_count" : 2,
      "sum_ttf" : 190
    },
    "terms" : {
      "1" : {
        "term_freq" : 1,
        "tokens" : [
          {
            "position" : 27,
            "start_offset" : 56,
            "end_offset" : 57
          }
        ]
      },
      "2030" : {
        "term_freq" : 1,
        "tokens" : [
          {
            "position" : 18,
            "start_offset" : 36,
            "end_offset" : 40
          }
        ]
```

```
"term_vectors" : {
  "message" : {
    "terms" : {
      "1" : {
        "doc_freq" : 1,
        "ttf" : 1,
        "term_freq" : 1
      },
      "2030" : {
        "doc_freq" : 1,
        "ttf" : 1,
        "term_freq" : 1
      },
      "2기" : {
        "doc_freq" : 2,
        "ttf" : 2,
        "term_freq" : 1
      },
```