TF-IDF 와 워드 임베딩을 이용한 뉴스 기사 키워드 추출

이재욱
국민대학교 소프트웨어학부
e-mail: swljw@kookmin.ac.kr

Extracting Keywords from News Articles using TF-IDF and Word Embedding Technique

Jaewook Lee
Kookmin University

요 약

TF-IDF 를 통한 키워드 추출은 자주 쓰이는 방법이지만 동의어에 대한 처리가 부족하다는 단점이 있다. 따라서 본 보고서에서는 TF-IDF 와 워드 임베딩을 같이 사용해 기사 텍스트로부터 키워드를 추출하는 방법을 제안한다. 이를 위해, 'KCC150' 말뭉치를 학습한 Word2Vec 모델을 구성한다. 이후, 네이버 뉴스 기사에 대해 형태소 분석을 진행한 후 학습한 모델과 매핑의 과정을 거친다. 그리고 DBSCAN 클러스터링 알고리즘을 통해 클러스터 수 만큼의 키워드를 추출한다. 이를 통해 TF-IDF 만을 사용하는 방법보다 더 정확한 키워드 추출이 이루어질 것이라 기대한다.

1. 프로젝트 목적 및 필요성

인터넷과 스마트폰의 보급으로 인해 뉴스를 손쉽게 접할 수 있는 시대다. 그러나 수많은 뉴스 기사가 생산되고 있어 사용자들이 원하는 정보를 빠르게 찾기가 어려워졌다. 따라서 뉴스 키워드 추출은 매우 중요한 기술로 자리 잡았다.

TF-IDF 는 단어의 빈도수와 문서 내의 중요도를 고려해 가중치를 부여하여 키워드를 추출하는 방식이다. 하지만, 이 방법은 동의어나 유의어를 구분하지 못해 동일한 의미의 단어에 다른 가중치를 부여하는 문제점을 가지고 있다. 또한 단어의 빈도수를 계산에 사용하기 때문에 희귀한 단어에 적절한 가중치를 부여하지 못 하거나, 불용어에 가까운 단어에 많은 가중치를 부여하는 문제점을 가지고 있다. 이러한 문제점들을 극복하기 위해 본 프로젝트에서는 TF-IDF 와 워드 임베딩을 함께 사용하여 뉴스 키워드 추출의 정확도를 높이고자 한다. 이를 통해 유의어나 동의어를 구분할 수 있으며, 이를 TF-IDF 와 결합함으로써 더욱 정확한 키워드 추출이 가능해질 것으로 예상된다.

2. 프로젝트 관련 연구

TF-IDF 는 단어 빈도-역 문서 빈도로, 텍스트에서 각 단어의 중요도를 계산하는 데 사용되는 가중치다. 이 가중치를 계산하기 위해 TF 가중치와 IDF 가중치를 사용하고 이 가중치들의 곱으로 TF-IDF 가중치를 정의한다. TF 가중치는 특정 문서에서 특정 단어가 얼마나 등장했는지 나타내는 가중치로 단어의 빈도수를 나타내는 값이다. IDF 가중치는 DF 가중치의 역수를 취한 값에 log 를 씌운 값이다. DF 가중치는 특정 단어가 몇 개의 문서에서 등장했는지 나타내는 가중치로 등장한 문서의 수를 나타내는 값이다[1].

워드 임베딩은 단어를 벡터 공간에 표현하는 방식으로, 단어의 의미를 밀집 벡터 형태로 표현한다. 비슷한 의미를 가지는 단어들은 벡터 공간에서 서로 가까이 위치하게 된다. 따라서 벡터 간의 유사도 계산이 단어 간의 유사도를 계산하는 것이라 할 수 있다. 이를 통해 앞에서 지적한 동의어나 유의어를 구분하지 못하는 TF-IDF 의 문제점을 해결할 수 있다. 워드 임베딩 방법론으로 LSA, Word2Vec, FastText, Glove 등이 있다[2]. 본 프로젝트에서는 실습한 경험이 있는 Word2Vec 방법론을 채택한다.

DBSCAN 알고리즘은 데이터 포인트들의 밀도를 기반으로 하는 클러스터링 알고리즘이다. 이 알고리즘은 데이터 포인트들이 서로 가까이 위치한 경우를 하나의 클러스터로 인식하고, 이를 기반으로 데이터를 그룹화하는 방식으로 동작하는 알고리즘이다[3]. 클러스터에 포함되지 못 한 데이터 포인트들은 노이즈 값으로 판단한다. 다른 대표적인 클러스터링 알고리즘으로 K-Means 알고리즘이 있다. K-Means 알고리즘은 데이터 포인트들을 k 개의 클러스터로 묶는

알고리즘으로 각 클러스터와 거리 차이의 분산을 최소화하는 방식으로 동작하는 알고리즘이다[4]. 이 프로젝트에서는 뉴스 기사의 단어 분포를 특정할 수 없기 때문에 클러스터의 개수를 특정하지 않는 DBSCAN 알고리즘을 클러스터링 알고리즘으로 채택한다.

이전에도 TF-IDF와 워드 임베딩을 같이 사용해서 뉴스 기사에서 키워드를 추출하는 사례가 있다[5]. 이 논문에서는 다양한 장애인 관련 기사들을 크롤링해서 형태소 분석을 진행한 후, TF-IDF 방법을 활용해 가중치를 부여했다. 그리고 워드 임베딩 벡터를 구성하기 위해 네이버 신문 기사 100만 건을 사용했다. 다음으로 TF-IDF 가중치가 높은 상위 N개의 데이터에 대해 사전에 구성한 단어 벡터와 매핑의 과정을 거치고 클러스터링을 진행한다.

3. 방법론

말뭉치를 이용한 Word2Vec 모델을 생성하는 과정이 필요하다. 이를 위해 사용하는 말뭉치는 'KMU KCC 한국어 뉴스 말뭉치'(이하 KCC150)를 사용한다. 이 말뭉치는 1천만개 이상의 문장 수를 가지고 있다. 전부 모델 훈련에 사용할 때 훈련 진행 중 터미널이 죽어버리는 문제가 발생했다. 따라서 전체 문장 수의 50%를 모델 생성에 사용했다. 표 1은 KCC150에 있는 원시 데이터 문장 수와 어절 수, 절반을 샘플링 했을 때의 문장 수와 어절 수를 나타낸 것이다. 그리고 표 2는 모델을 학습할 때 사용한 파라미터다.

표 1. KCC150 말뭉치 정보

	문장 수	어절 수
KCC150(전체)	11,961,347	150,705,457
KCC150(50%)	5,980,671	75,334,470

표 2. Word2Vec 모델 학습 파라미터

vector size	window size	min count	epochs
300	5	2	30

그 다음, 분석하려는 뉴스 기사에 대해 형태소 분석을 진행한다. 형태소 분석은 모델의 원활한 학습을 위해서도 진행되어야 한다. 형태소 분석을 위한 도구로 국민대학교 소프트웨어학부 강승식 교수님이 제공하는 'KLT2000'을 사용한다. 파이썬 패키지 'konlpy'에서도 5가지의 형태소 분석기들(Kkma, Okt, Mecab, Komoran, Hannanum)을 제공한다. 하지만 이전의 학부 과제를 통해 KLT2000의 실행 속도가 Mecab과 비슷했고 빠른 때도 있는 것이 확인됐다. Mecab 분석기는 이전의 과제를 통해 속도가 다른 분석기들 보다 빠름을 확인했으므로 KLT2000 또한 빠른 속도를 가지고 있다는 것이 확인됐다. 또한 임의의 문장에 대한 형태소 분석 성능은 대체적으로 비슷했고, 일부 문장에 대해서는 KLT2000이 우수한 측면을 보였다.

뉴스 기사에 형태소 분석을 적용한 결과에 TF-IDF를 적용한다. TF-IDF 과정은 파이썬 패키지 'sklearn'을 사용했다. 표 3은 TF-IDF 만을 적용했을 때, 선택되는 뉴스 키워드들을 나타낸 것이다. 이는 후에 프로젝트 결과와 비교할 결과다. 뉴스는 네이버 IT/과학 뉴스 중 임의로 선택[1]했다.

표 3. TF-IDF로 확인한 뉴스 키워드 상위 5개

순위	키워드
1	홈페이지
2	제보
3	스미스
4	부회장
5	기술

키워드들에 군집화를 적용하기 전에 TF-IDF 결과 상위 30개의 토큰을 선택한다. 이후 Word2Vec 모델과 일대일 매핑의 과정을 거친다. 만약 키워드 중 모델에 없는 키워드가 있다면 그 토큰은 선택 대상에서 제외한다. 과정이 끝나면 각 토큰은 벡터 값을 가지게 된다. 그리고 이 벡터들을 가지고 DBSCAN 군집화를 진행한다. 표 4는 DBSCAN 군집화를 진행할 때 사용한 파라미터다.

표 4. DBSCAN 파라미터

eps	min samples	metric
26	4	Euclidean

이후에 각 클러스터 별로 평균 벡터를 계산한다. 이 평균 벡터와 가장 높은 유사도를 가진 토큰을 클러스터의 대표 키워드로 정의한다. 유사도는 벡터의 유사도를 계산할 때 자주 쓰이는 코사인 유사도 방법을 선택했다. 코사인 유사도 계산을 sklearn 패키지로 하려니 차원 수가 너무 커서 계산이 안 되는 문제가 발생한다. 따라서 벡터의 내적 값과 norm 값은 파이썬 패키지 'numpy'를 사용하고 코사인 유사도는 계산한 값들을 유사도 계산 식에 대입하여 계산했다.

4. 결과

프로젝트 실행 결과의 성능을 판단하는 방법은 기존의 TF-IDF 만을 사용했을 때와 얼만큼의 동일한 키워드를 선택했는지를 기준으로 삼는다.

프로젝트를 실행한 결과, 총 2개의 클러스터로 그룹화가 되었다. 하나의 클러스터는 그룹화 할 수 없는 값들(노이즈)들을 모았다. 노이즈가 아닌 나머지 하나의 클러스터로 모인 토큰들은 '스미스', '당신', '브래드', '여러분' 등 9개다. 이 중 평균 벡터와 가장 높은 유사도를 가진 토큰은 '밝혔습니다'이다. 이 토큰은 표 3의 결과와 전혀 일치하지 않는다.

[1][뉴스 사이트]
https://n.news.naver.com/mnews/article/374/0000332086?sid=105

4.1. 실패 이유 고찰 1 - 데이터 분포

임의의 네이버 뉴스 기사에 대해 예상과 다르게 클러스터는 2 개만 생성되었고, 이 중에 하나는 노이즈 값들만 모아둔 클러스터였다. 원인을 알아보기 위해 데이터들을 시각화했다. 데이터들은 300 차원의 벡터 값들을 가지고 있기에 PCA 방법을 적용해서 2 차원으로 차원을 낮췄다. 그림 1 은 파이썬 패키지 'matplotlib'을 사용해 시각화한 결과다. 클러스터라고 할만큼 데이터들이 한 곳에 모여있지 않고 고르게 흩어져 있다. 이는 특정 기사만 그런 것이 아니라 테스트한 여러 기사들에서 비슷한 분포가 나타났다.

그림 1. 뉴스 토큰들을 평면에 시각화

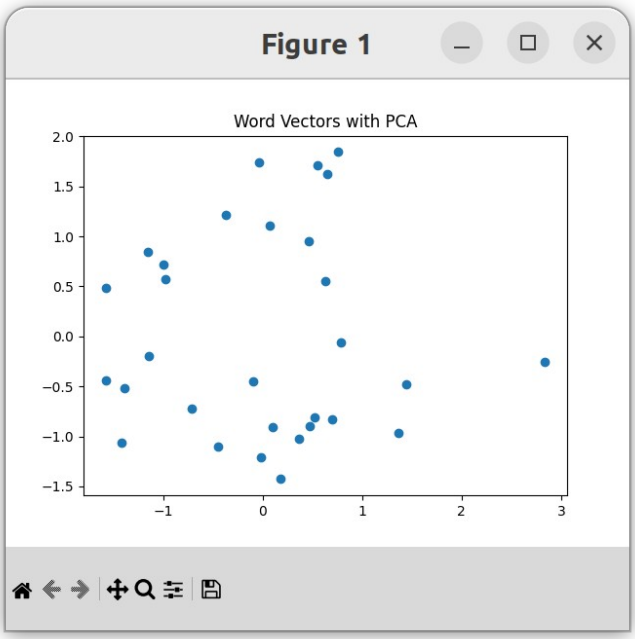
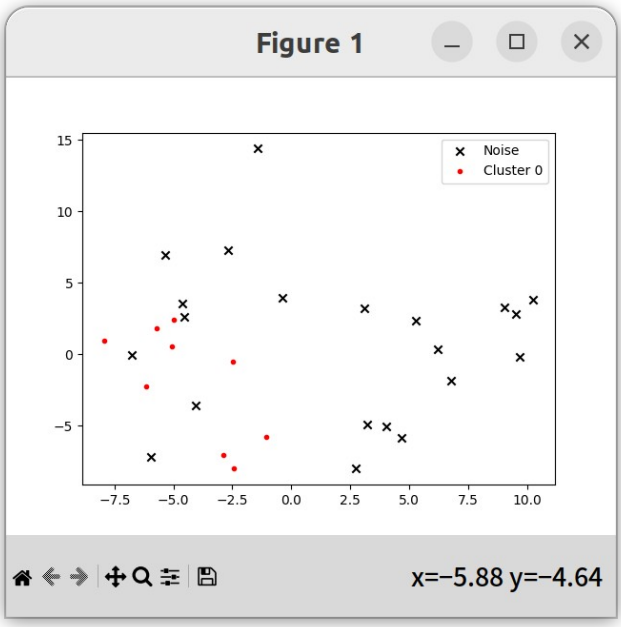


그림 2 는 DBSCAN 알고리즘을 적용한 후, 다시 한 번 시각화한 결과다. 빨간 점으로 표시한 데이터들이 하나의 클러스터에 속하는 값이다. 하지만 육안으로 확인할 때는 클러스터라고 보기 힘든 값들이 선택됐다.

그림 2. DBSCAN 알고리즘 적용 후 시각화



4.2. 실패 이유 고찰 2 - 파라미터

Word2Vec 모델을 생성할 때 사용한 파라미터는 [5] 에서 사용한 파라미터를 따랐다. 그리고 DBSCAN 을 실행할 때의 파라미터는 뉴스 토큰들에서 클러스터가 생기고 의미 있는 토큰들이 모이도록 여러 번 시행을 통해 찾았다. 그러나 이 파라미터 값들은 프로젝트에 최적화된 값이 아닐 수도 있다. 머신 러닝, 딥 러닝에서도 그렇듯, 파라미터 값을 어떻게 설정했는지에 따라 성능의 차이가 생긴다. 모델 훈련 시 벡터의 크기를 200 ~ 600 차원, DBSCAN 의 eps 값을 15 ~ 30 사이의 실수 값들로, min count 값을 2 ~ 5 사이로 여러 번 적용해본 결과 그나마 의미가 있는 결과가 나온 파라미터 값이 표 2 와 표 4 에 명시한 파라미터다. 하지만 범위 안에서 미처 입력하지 못한 값 또는 범위 밖의 파라미터 값이 최적의 결과였을 수도 있다.

Acknowledgement

1. 이 보고서는 국민대학교 소프트웨어학부 강승식 교수의 '빅데이터 최신기술' 강의의 과제로 제출하는 자료이다.
2. 이 프로젝트에서 사용한 말뭉치와 형태소 분석기는 다음의 사이트에서 얻을 수 있다. (http://cafe.naver.com/nlpkang)
3. 이 보고서의 1, 2 번 항목은 chatGPT 를 이용하여 초안을 작성하고 내용을 다듬는 방향으로 최종 보고서를 완성하였다.

참고자료

[1] 안상준, 유원준, "04-04 TF-IDF(Term Frequency-Inverse Document Frequency)", wikidocs 사이트, https://wikidocs.net/31698

[2] 안상준, 유원준, "09-01 워드 임베딩(Word Embedding)", wikidocs 사이트, https://wikidocs.net/33520

[3] "sklearn.cluster.DBSCAN", scikit-learn 공식 문서 사이트, https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html

[4] "k-평균 알고리즘", wikipedia 사이트, https://ko.wikipedia.org/wiki/K-%ED%8F%89%EA%B7%A0_%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98

[5] 최가람, 최성필, "단어 임베딩 기법을 적용한 키워드 중심의 사회적 이슈 도출 연구: 장애인 관련 뉴스 기사를 중심으로", 정보관리학회지, Vol.35 no.1, pp.231-250, 2018.