

RS: find similar tag (word2vec)

20.08.21

컴퓨터과학전공 1715237 이혜승

Word2vec?

Word2vec 개념 정리

- 단어 임베딩의 방법론 중 하나.
즉, 단어들을 고정된 차원의 vector space에 유의미하게 배치해주는 단어 embedding model들 중 대표적인 모델이다.
- Word embedding(워드 임베딩): 단어를 벡터로 표현하는 것. (단어 → 밀집 표현으로 변환)
- Dense Representation(밀집 표현) ↔ sparse representation(one-hot 인코딩)
: 하나의 차원이 하나의 속성을 명시적으로 표현하지 않고, 여러 차원들이 조합되어 나타내고자 하는 속성들을 표현한다.
: 벡터의 값들이 더 이상 0과 1의 값이 아닌 **실수값**을 갖는다.
: 임베딩 벡터(embedding vector)
- 대상 단어와 주변 단어들을 바탕으로, 다음 단어를 예측하는 모형(두 가지 방식: CBOW / Skip-gram)

PLAN

Metadata[0:9]의 태그 정보만 가지고 진행.

tag

	magazine_id	user_id	title	article_tag_list
0	8982.0	@bookdb	사진으로 옮기 기에도 아까운, 리치필드 국립 공원	['여행', '호주', '국립공원']
1	12081.0	@kohwang56	[시] 서러운 봄	['목련꽃', '아지랑 이', '동행']
2	0.0	@hannahajink	무엇을 위해	[]
3	16315.0	@bryceandjuli	싫다	['감정', '마음', '위로']
4	29363.0	@mijeongpark	Dubliner#7	['유럽여행', '더블 린', '아일랜드']
5	17646.0	@jordan777	지식·기술이 가 장 경쟁력 있는 자원이다 "	['석유에너지', '베 네수엘라', '경제']
6	7164.0	@m-life	유치권 심문기 일	['입찰', '유치권', '부동산경매변호 사']
7	9963.0	@hohogg	(過去)Rain	['사랑', '연애', 'rain']
8	28849.0	@elang8151	출간제의 받았 습니다.	['메일', '출판사']
9	0.0	@purnbd	동시쟁가게	['도시애벌레', '공 부', '동시']

tag.article_tag_list.tolist()
이중 리스트 형태

[['여행', '호주', '국립공원'],
['목련꽃', '아지랑이', '동행'],
[],
['감정', '마음', '위로'],
['유럽여행', '더블린', '아일랜드'],
['석유에너지', '베네수엘라', '경제'],
['입찰', '유치권', '부동산경매변호사'],
['사랑', '연애', 'rain'],
['메일', '출판사'],
['도시애벌레', '공부', '동시']]

Word embedding using
word2vec

```
ko_model.wv[tags_after[0]].shape  
>> (3, 200)
```

태그 3개 각각에 대한 200차원의 벡터를 이중리스트
형태로 갖게 됨.

tag하나 하나 → vector
한 item의 tag들 모음(리스트) → cluster로 보고,
Cluster간 유사도를 계산하여 태그 기반 유사한 글을
찾고자 한다.

Pretrained model

사전 훈련된 Word2Vec 임베딩 (Pre-trained Word2Vecembedding)

- 한국어로 사전 훈련된 word2vec 모델.
- 출처: [Kyubyong / wordvectors](#)
- Word2vec의 한계점: “기존에 학습한 text data에 특정 단어가 포함되어 있지 않거나, 유효하지 않은 경우(min_count에 걸려서 고려하지 않는 경우)”일 때는 해당 word에 대해서 vector로 표현불가
- 따라서, 해당 word가 model.wv.vocab내에 존재하는 지 확인 필요
- 이 것이 매우 문제가 된다면, FastText 모델을 사용을 추천!(한다고 하나, 우선 word2vec으로 진행했음)

진행상황

- 1) 태그-> vector로 만들어주는 vector extractor
- 2) Vector set -> cluster
- 3) Cluster간 연결 법 고민 (중심 연결법 – centroid 구함, 거리 계산법 – cosine # 아직 이유는x)

유사도(similarity): 값이 클수록 비슷 / 거리(distance): 값이 작을수록 비슷

유사도/거리 계산법>

1. 맨하탄 거리(Manhattan distance)
2. 마할라노비스 거리(Mahalanobis distance)
3. 코사인 유사도(cosine similarity)
4. 자카드 유사도(Jaccard similarity)

진행상황

- 1) 태그-> vector로 만들어주는 vector extractor
- 2) Vector set -> cluster
- 3) Cluster간 연결 법 고민 (중심 연결법 – centroid 구함, 거리 계산법 – cosine # 아직 이유는x)

결론적으로, 선택한 것은

```
"""
클러스터 centroid간 거리 계산
cosine 유사도 모듈 사용
"""
tag_cluster_similarity = cosine_similarity(cluster_centroid, cluster_centroid)
tag_cluster_similarity # pair한 유사도 결과값
```

```
array([[ 1.          ,  0.15173459,  0.          , -0.05007893,  0.32018477,
        0.09498899,  0.0513012 ,  0.07778219,  0.13442078,  0.09296269],
       [ 0.15173459,  1.0000001 ,  0.          ,  0.15025163,  0.06357107,
       -0.15914805,  0.1245089 ,  0.21953331,  0.02336979,  0.34965456],
       [ 0.          ,  0.          ,  0.          ,  0.          ,  0.          ,
        0.          ,  0.          ,  0.          ,  0.          ,  0.          ],
       [-0.05007893,  0.15025163,  0.          ,  1.0000001 , -0.10380967,
       -0.03304074,  0.05057949,  0.56366986,  0.00167014,  0.22441882],
       [ 0.32018477,  0.06357107,  0.          , -0.10380967,  1.          ,
        0.02247831, -0.13866454,  0.03195077,  0.04824585,  0.06821413],
       [ 0.09498899, -0.15914805,  0.          , -0.03304074,  0.02247831,
        1.          ,  0.11584206,  0.01028934, -0.04597205,  0.09042774],
       [ 0.0513012 ,  0.1245089 ,  0.          ,  0.05057949, -0.13866454,
        0.11584206,  1.          , -0.08108482,  0.15682852,  0.08890989],
       [ 0.07778219,  0.21953331,  0.          ,  0.56366986,  0.03195077,
        0.01028934, -0.08108482,  1.0000001 ,  0.14143258,  0.22775096],
       [ 0.1344208 ,  0.02336978,  0.          ,  0.00167013,  0.04824585,
       -0.04597205,  0.15682852,  0.14143257,  0.9999999 ,  0.11435215],
       [ 0.0929627 ,  0.3496545 ,  0.          ,  0.22441882,  0.06821413,
        0.09042774,  0.08890989,  0.22775096,  0.11435215,  0.9999999 ]],
      dtype=float32)
```

진행상황

```
[537] ▶ MI
# 타겟item 0번과 제일 유사도 높은 글의 태그
print(tags[(np.where(tag_sort[-2] == tag_cluster_similarity[0])[0][0])])

['유럽여행', '더블린', '아일랜드']

[544] ▶ MI
tag[np.where(tag_sort[-2] == tag_cluster_similarity[0])[0][0]:np.where(tag_sort[-2] == tag_cluster_similarity[0])[0][0]+1]
```

magazine_id	user_id	title	article_tag_list	display_url	sub_title	reg_ts	article_id	id	date	hour	weekday	view	recent_view	
4	29363.0	@mijeongpark	Dubliner#7	[유럽여행, 더블린, 아일랜드]	https://brunch.co.kr/@mijeongpark/34	#7. 내 친구의 집은 어디인가	1523292942000.0	34	@mijeongpark_34	2018-04-10	01:55:42	Tue	1.0	0.0

함수화, 모듈화는 아직 하지 않았음.

['여행', '호주', '국립공원']라는 태그를 가진 글과 가장 유사한 글로 4번 글을 추천할 수 있다. (태그: [유럽여행, 더블린, 아일랜드])

→ 여행이라는 테마는 비슷하나.. 과연 궁금해할 내용인지는 잘 ...

issues

1) 태그 목록에 있는 단어 중 'word is not in vocab'

I. Rain, study, love : 영어 태그

→ 자동 번역 or 삭제(def isHangul(text): 함수 구현하여, 우선 삭제하고 진행 중)

II. 유럽여행, 부동산경매변호사, 목련꽃: 유럽/여행, 부동산/경매/변호사, 목련/꽃 형태소 분석이 추가적으로 필요한 경우

→ 추가적인 형태소 분석 or 삭제(konlpy 사용이 안되어서 우선 삭제 했음)

III. 젠트리피케이션: 영어를 한국어 발음으로 명명하는 단어들 중 일부

IV. 송중기, 내셔널지오그래픽, 도시에벌레: 사람 이름, 브랜드명, 책이름 등의 고유명사

→ 그냥 삭제?

2) 그렇다면 추가적 형태소 분석을 해서, 의미 변동이 생기는 경우는 어떡하지?

→ Jtbc뉴스룸 -> jtbc/뉴스/룸으로 쪼개지면 원래의 단어의 의미가 사라질 것 같다.(뉴스를 제외하고는...)

3) 유사도 기반으로 글을 추천했는데 너무 옛날 글이면 어떡하지?

4) 추천해야 하는 100개 중 몇 개 혹은 몇 %를 태그기반으로 추천할 것인지?

Konlpy

해결 할 문제: Konlpy설치

```
from konlpy.tag import Okt
```

```
okt = Okt()
```

```
print(okt.morphs(u'단독입찰보다 복수입찰의 경우'))
```

```
>> ['단독', '입찰', '보다', '복수', '입찰', '의', '경우']
```

```
tag[0:10].article_tag_list.tolist() = [['여행', '호주', '국립공원'],  
    ['목련꽃', '아지랑이', '동행'],  
    [],  
    ['감정', '마음', '위로'],  
    ['유럽여행', '더블린', '아일랜드'],  
    ['석유에너지', '베네수엘라', '경제'],  
    ['입찰', '유치권', '부동산경매변호사'],  
    ['사랑', '연애', 'rain'],  
    ['메일', '출판사'],  
    ['도시애벌레', '공부', '동시']]
```

2개 이상의 단어로 더 쪼개질 수 있는 단어를 한 번 더 쪼개서 저장하고자 함

→ (+) 형태소 분석을 안하면, 임의 삭제해야하므로 데이터를 잘 활용할 수 있음

→ (-) 단어를 쪼갰을 때, 그 의미를 잃는 경우는 어떻게 할 지? (하나하나 파악하고 임의로 삭제해주기에는 데이터양이 너무 많다.)

Word2vec?

해결 할 문제: Konlpy설치

```
from konlpy.tag import Okt
okt = Okt()
print(okt.morphs(u'단독입찰보다 복수입찰의 경우'))
>> ['단독', '입찰', '보다', '복수', '입찰', '의', '경우']
```

FileNotFoundError: [Errno 2] No such file or directory: '/usr/lib/jvm'

```
$ sudo apt update
```

```
$ sudo apt install default-jre
```

```
$ sudo apt install default-jdk
```

```
$ pip3 install konlpy
```

```
# 권한때문에 접근 불가
```

Future work

목표

1. 타겟 사용자가 좋아할 만 한 글 추천 2. 그동안 잘 추천되지 않았던 / 잘 읽히지않았던 글 추천

① 브런치 사용을 잘 하는 user 및 조회수가 높은 item

: 그대로 item-based CF적용

② Long-tailed item: 잘 읽히지 않았던 글 (즉, 조회수가 낮은 글)

: tag정보 → word2vec → 유사도 계산 → 비슷한 글 추천

: contents-based filtering(작가, 매거진 등의 정보를 feature로 활용해서 글 자체의 특성 비교) → 유사도 계산해서 타겟 사용자가 그동안 봤던 글과 가장 유사한 글을 추천해준다.

③ Cold-start item: 추천기간에 새로 작성된 글

: 브런치 특성상 구독하는 작가 글 읽음 → 구독작가 기반 추천

: 자주 읽었던 매거진 글 읽음 → 매거진 기반 추천

: 관심있는 글(그동안 읽었던 글들과 유사한 글이 관심있을 것이라고 가정.) – 참고한 코드의 경우에는 똑같은 태그가 2개이상일 때, 해당 태그를 관심사로 판단했음. 만약에 태그가 ‘유럽’, ‘여행’ / ‘유럽여행’이라면 비슷한 글임에도 불구하고 유사한 글로 분류해주질 못한다. 따라서 word2vec을 활용해서 유사도를 비교하여 가장 비슷한 태그를 가진 글을 추천해주고자 함.

④ Cold-start user / 최근or전체 이용이 적은 user

: 앞에서 소개한 CF기법과 사회연결망 기법의 중심성을 결합한 분석

reference

- [유사도,거리관련 글](#)[브런치, gimmesilver]
- 저널: 추천시스템 기법 연구동향 분석 - 손지은(고려대학교), 김성범(고려대학교), 김현중(서울대학교), 조성준(서울대학교), 2015.04