자연어처리 텀프로젝트 보고서

파이썬 문서 검색기

2팀

컴퓨터학과 2013210043 권민규  
컴퓨터학과 2014210035 전수혁  
컴퓨터학과 2014210064 변지석

# 서론

오늘날 정보통신 기술과 검색엔진 기술의 발달로 인터넷에 공개된 많은 자료들을 손쉽게 찾아볼 수 있게 되었지만, 프로그래머들이 자주 찾게 되는 언어 레퍼런스 문서들은 아직 손쉬운 문서검색 방법이 나오지 않았다. 그래서 프로그래머들은 불편함을 감수하고서라도 자신이 원하는 언어상 기능이 나올 때까지 일일히 찾아보거나, stackoverflow (https://stackoverflow.com) 같은 질문-대답 커뮤니티에 질문을 올려 아는 사람에게 대답을 듣는 수밖에 없다. 따라서 이런 어려움을 고치기 위해 사용자에 질의에 가장 답이 될 프로그래밍 언어 문서를 찾아주는 문서 검색기를 만들어 보았다.

지금까지 많은 종류의 프로그래밍 언어가 공개되었기 때문에 모든 프로그래밍 언어 문서들에 대한 검색기를 만드는 일은 단기 프로젝트에 적합하지 않았다. 따라서 공식 파이썬 문서 중 라이브러리와 레퍼런스 부분(docs.python.org/3/library/, docs.python.org/3/reference/) 로 시작하는 321개의 html 파일들에 대해 검색하는 검색기를 만들어보았다. 공식 파이썬 문서에 없는 단어까지 질의문에 포함될 수 있기 때문에, 의미를 잘 반영하기 위해 Google Word2Vec Pretrained Word Embedding을 사용했고, 공식 파이썬 문서라는 특정 도메인에서 뭉쳐진 문서 표현 벡터를 덜 뭉치도록 만들기 위해 Term Frequency, Document Frequency값을 사용했다. 실제 stackoverflow의 질문-대답 자료 약 45000개를 사용하여 검색기의 성능을 평가한 결과 약 2%의 정밀도(precision)와 20%의 재현율(recall)을 얻을 수 있었다.

우리는 이 검색기를 쉽게 웹 서비스로 사용할 수 있도록 제작했다. 우리가 공개한 소스코드에서 Django 웹 프레임워크를 설치하고 PyMaker 소스코드에서 약간의 사전설정을 한 후 <https://github.com/cushionbadak/PyMaker/tree/master/WebService> 의 설명을 따라 서버를 실행하면 웹으로 쉽게 검색기를 실행해볼 수 있다.

# 문서 검색 방법

이를 논의하기 전에, 321개의 파이썬 문서들 D = {D1, D2, … , D321} 와 사용자 질의 Q에 대해 문서 검색을 위해서 다음 값들을 사용하였다.

* 어떤 단어 w를 받아 고정된 크기의 실수 벡터를 반환하는 함수 *W2V*(w)
* 어떤 문서 d에서 단어 w가 출현하는 빈도 *TF*(d, w)
* 어떤 단어 w가 D에 있는 문서들에서의 출현 빈도 *DF*(w)

위의 값들을 조합하여 임의의 자연어 문자열을 받아 고정된 크기의 실수 벡터를 반환하는 함수 *Embed =* {*Embed\_1, Embed\_2, … , Embed\_n*}들을 만들 수 있었고, 미리 정해진 임베딩 *Embed\_x, Embed\_y*에 대해 파이썬 문서 검색기가 검색결과로 내 놓는 파이썬 문서들 D\_result(Q)는 다음과 같이 구했다.

Max\_cosine(Q) = Argmax (d ∈ D) (*cos\_sim*(*Embed\_x*(d), *Embed\_y*(Q)))  
 D\_result(Q) = { d ∈ D | *cos\_sim*(*Embed\_x*(d), *Embed\_y*(Q)) > Max\_cosine(Q) – 0.05 }

위에서 cos\_sim은 두 같은 크기의 실수 벡터를 받아 두 벡터가 이루는 각도의 코사인 값을 반환하는 함수이다.

## 파이썬 문서 표현

파이썬 문서 Dk 에 대한 표현법은 다음 세 방법으로 구현하였다.

* *Embed\_naive* (Dk) = (w ∈ Dk) ∑ (*W2V* (w))
* *Embed\_tfidf* (Dk) = (w ∈ s) ∑ (*W2V* (w) \* log(*TF* (Dk, w) / *DF*(w)))
* *Embed\_tfidf2* (Dk) = (w ∈ s) ∑ (*W2V* (w) \* log(*TF* (Dk, w) / *DF*(w)) / *DF*(w))

## 질의 문장 표현

질의 문장 Q는 다음 두 방법으로 구현하였다.

* *Embed\_naive* (Q) = (w ∈ Dk) ∑ (*W2V* (w))
* *Embed\_df* (Q) = ((w ∈ Dk) ∑ (*W2V* (w) / *DF2*(w))

여기서 *DF2*는 *DF* 와 비슷한 함수로, D에 나타나지 않은 단어에 대한 빈도를 0으로 표현할 때 0으로 나누는 문제가 생기지 않도록 새로 정의한 함수이다. D의 크기가 321이므로 DF(w)의 출력값이 0부터 321 사이라고 가정할 때, 정확한 정의는 다음과 같다.

* *DF2*(w) = if (*DF*(w) is 0) then (1) else (*DF*(w))

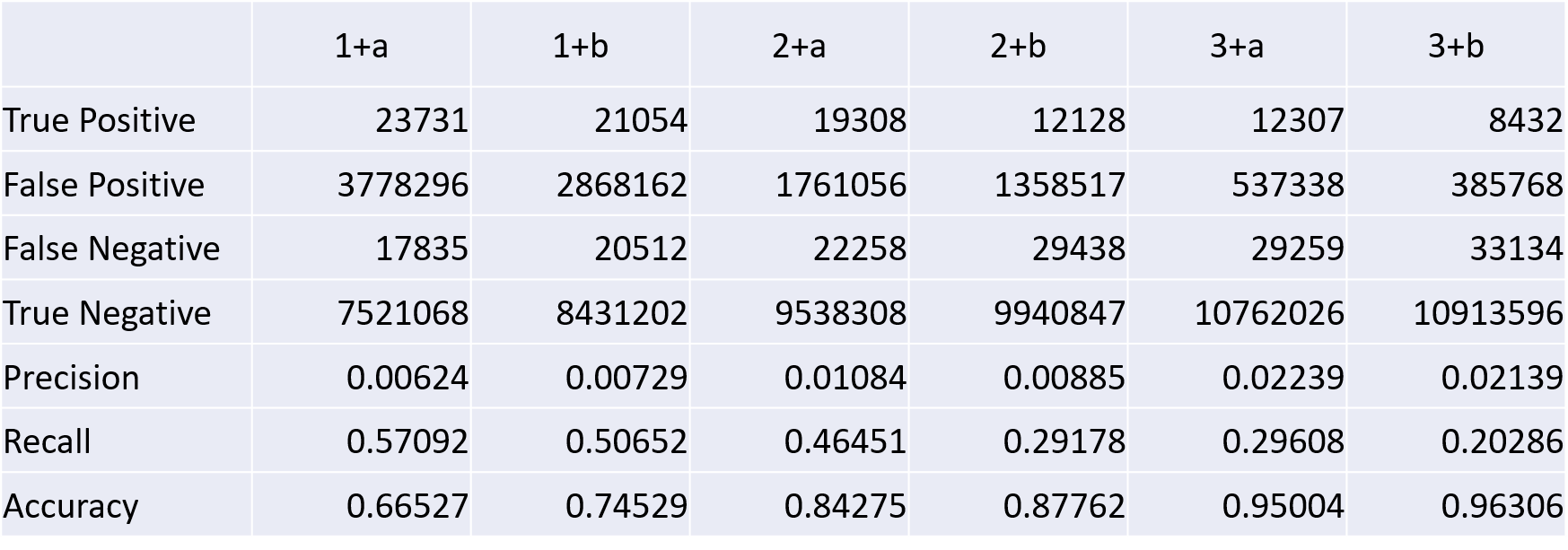
다시 말하자면, 위 *DF2* 정의에서 1은 0이 아닌 가장 작은 *DF* 출력값이라고 해석할 수 있다.

# Stackoverflow 평가 데이터

어떤 프로그래밍 언어 질문에 연관된 파이썬 공식 문서 대답은 지금껏 만들어지지 않았다. 따라서 우리 팀은 stackoverflow.com 에서 ‘python’ 태그가 달린 질문-대답 문서들을 모아, 만약 어떤 질문-대답 문서에 파이썬 공식 문서 링크가 달려 있다면 그 연결된 파이썬 공식 문서가 질문에 대한 답이 된다고 가정하여 테스트 데이터를 만들었다. 자료를 모은 결과 약 45000개의 테스트 데이터가 만들어졌으며, <https://github.com/cushionbadak/PyMaker/tree/master/PyMaker/datas/object3/> 에서 그 데이터를 볼 수 있다. 테스트 데이터의 조건을 만족하는 질문-대답 문서들의 정제되지 않은 데이터는 <https://github.com/cushionbadak/PyMaker/tree/master/PyMaker/datas/object2> 에서 볼 수 있다.

# 평가

평가 결과는 다음과 같다.



1, 2, 3, a, b는 다음과 같다.

1: *Embed\_naive*

2: *Embed\_tfidf*

3: *Embed\_tfidf2*

a: *Embed\_naive*

b: *Embed\_df*

평가 결과에서 False Positive 값에 주목해야 한다. 1+a 의 경우, 질의 개수를 45000개로 보았을 때 하나의 질의에 대해 평균 84개의 잘못된 검색 결과를 보여준다 볼 수 있다. 최대 검색 결과 개수가 321개임을 감안하면 이는 매우 큰 숫자이며 사용자에게 유용한 결과를 보여준다고도 볼 수 없다. 실제 결과값을 살펴보았을 때 대부분 200개 이상 또는 0개의 검색결과를 보여준다는 점에서 각 파이썬 공식 문서의 벡터 표현 방법이 벡터들을 비슷한 곳에 모아둔다고 볼 수 있다. Term frequency, Document Frequency 값의 활용은 이러한 뭉침 현상을 현저히 줄여 False Positive 수를 많이 줄였고, True Positive 수를 줄이는 비율보다 False Positive 수를 더 많이 줄일 수 있어 Precision이 세 배 이상 늘었다.

# 결론

범용적인 미리 학습된 단어 임베딩 값을 가지고 파이썬 공식 문서라는 특수한 도메인의 문서들을 임베딩하는 과정에서 Term Frequency, Document Frequency 값을 사용하면 좀 더 구분이 용이한 문서 표현 벡터를 얻을 수 있다. Precision을 높이고 짧은 Query 에서 좀 더 나은 벡터 표현을 만드는 방법은 좀 더 연구해야할 주제가 될 것이다.

# 참고 자료

* + Project Repository  
    <https://github.com/cushionbadak/PyMaker>
  + Stack Overflow  
    <https://stackoverflow.com/questions/tagged/python>
  + Python Documentation  
    <https://docs.python.org/3/>
  + Google Word Vector  
    <https://code.google.com/archive/p/word2vec/>