나뭇가지의 나뭇잎을 추천해주는

NAMU-Branch

정 지 민

NAMU-Branch

INDEX

목차

목차1 개요

목차2 목표와 구현방법

목차3 흐름도

목차4 구현 결과

목차5 문제점과 해결방법

목차6 느낀점과 마무리

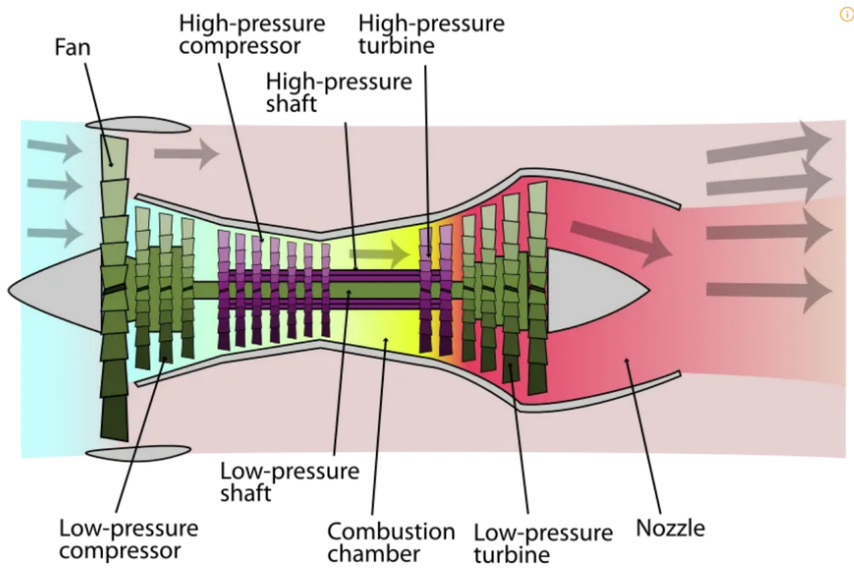
01

프로젝트 개요

- 나무위키 대부분의 문서에는 '관련 문서' 문단이 있지만, 일부 문서에는 존재하지 않음.

- 비슷한 주제의 글을 쉽게 찾을 수 있도록 추천 시스템이 필요했던 경험을 함.

- TF-IDF 실습을 바탕으로 이 경험을 개선하는 프로그램을 개발하는 프로젝트를 진행함.



터보 팬 엔진은 엔진 직경보다 큰 팬을[9] 연소실의 터빈들과 샤프트로 연결하여 함께 회전하게 하는데, 이 때, 엔진 주변의 공기를 추가로 뒤로 밀어내 추진력을 만들어 내게 된다. 이 공기들은 엔진의 연소실을 통과하지 않기 때문에 바이패스 에어(Bypass air)라고 한다. 터보제트 엔진에 비해서 연료 효율이 훨씬 좋고(아음속에서는 프로펠러의 추진 효율이 더 좋은데, 바이패스를 지나는 공기의 경우 프로펠러의 추진과 동일하게 볼 수 있다.) 큰 추력을 만들어 낼 수 있어 현대 여객기와 전투기들에게 사랑받는 엔진이다. 다만 고속에서 효율이 터보제트엔진에 비해 떨어지기 때문에, 전투기의 경우 엔진 앞에 관(덕트)을 만들어 엔진에 도달하는 공기의 속도를 늦춰준다. 여객기와 다르게 전투기 엔진의 블레이드가 밖으로 노출되지 않은 이유이며, 고속으로 비행하는 콩코드 같은 경우 전투기와 비슷하게 엔진 앞에 관이 달려있다.

물론 저기서 앞에 달린 팬 하나만 뚝 떼어내면 그냥 터보제트 엔진이 된다. 또한 바이패스 비중이 커진다고 제트 추진의 세기를 무시하면 안 되는게, 바이패스비는 어디까지나 팬의 추력과 제트 추력의 비율만을 나타낸 것이기 때문이다. 저 팬을 구동하고있는 건 결국 터보제트 엔진이고 그 엔진의 힘은 절대 약하지 않다.

10. 기타 [편집]

자동차용 터보차저를 이용해서 제트 엔진을 만들 수 있다. 특이점으로는 용접을 하지 않고도 만들 수 있다는 것이다.[10]

11. 참고 링크 [편집]

- (영문 위키백과) Turbojet

02

NAMU-Branch

열람중인 문서와
유사한 문서를 추천해주는
프로그램

- 열람중인 문서와 유사한 문서 추천
- 키워드 검색으로 문서 추천
- 열람중인 문서 개요 출력

+

TF-IDF ⇒ "문서간 벡터 비교"

- 문서 전체를 형태소로 분리 후 TF-IDF로 벡터화
- 문서의 TF-IDF 벡터로 다른 문서와 유사도 비교
- 같은 주제의 문서 추천

Word2Vec ⇒ "키워드와 유사한 단어 추출"

- 키워드 기반 검색을 위해 문서에 나오는 단어 임베딩
- 입력 단어와 유사한 단어를 Word2Vec으로 찾은 후, TF-IDF 벡터화하여 비교
- 연관 키워드 기반 문서 추천

03 프로젝트 흐름도

1

데이터 축소

데이터 크기가
다루기에 너무 커
축소 후 진행

2

데이터 전처리

추천 정확도를 올리기 위한
데이터 전처리

3

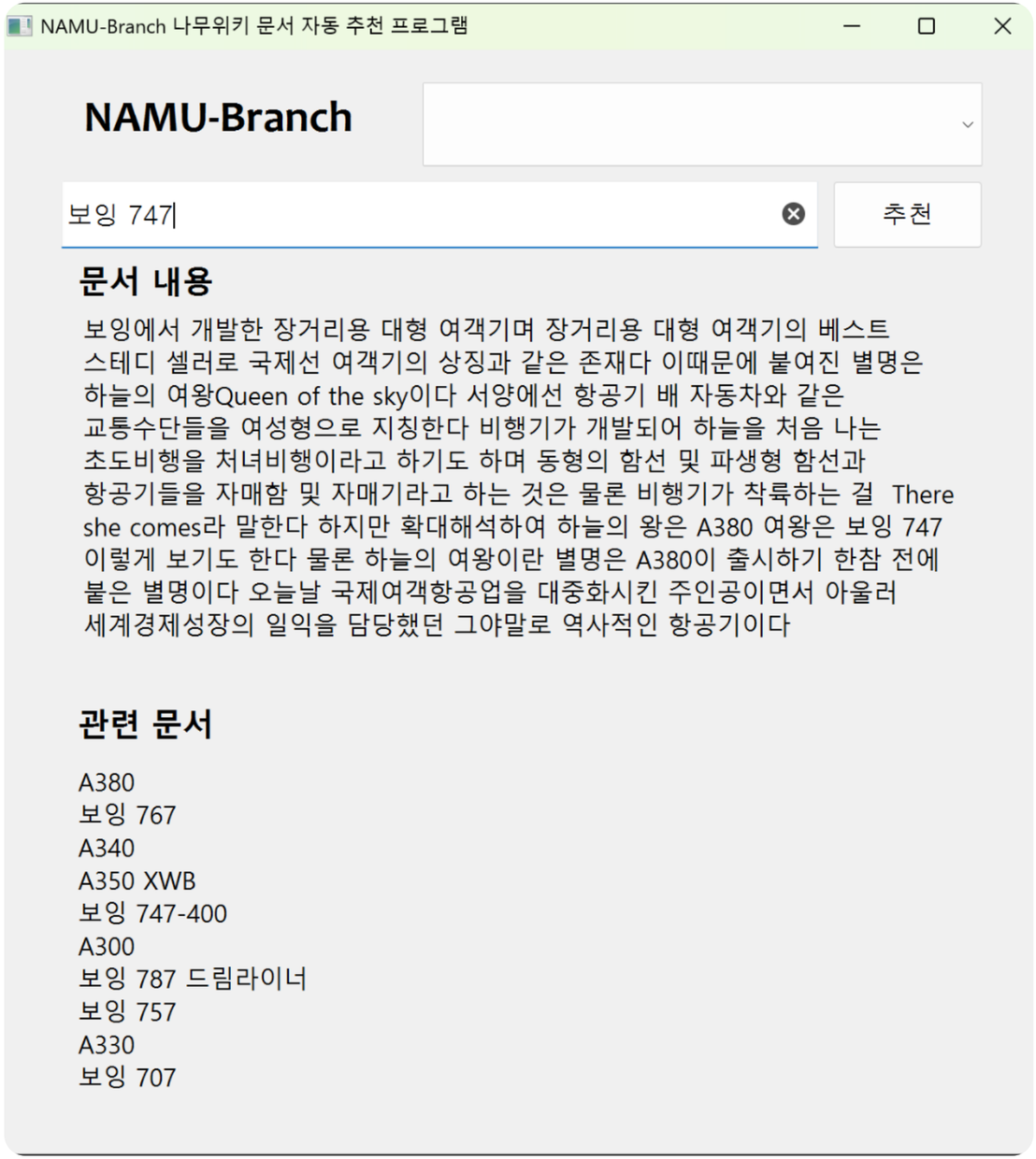
TF-IDF, Word2Vec

데이터의 단어를 벡터화
문서 의미 부여

4

UI 구현

사용자를 위한 UI 구축
문서 개요와
추천 문서 출력



NAMU-Branch

04

프로젝트 구현 결과

문서를 고르거나 키워드로 검색하면 추천 문서 출력 - 구현 완료

TF-IDF로 문서를 벡터화 결과 유사도가 높은 문서를 자동적으로 추천해준다

키워드 검색도 유사한 문서를 추천해준다

PyQt5를 이용하여 시각화,
개요 출력 및 관련 문서를 출력하는 UI를 제작하였다.

NAMU-Branch

05

문제점과 해결방안 - 1

원본 데이터의 크기가 너무 크다.

나무위키 덤프 데이터는 약 8.7GB
⇒ 전체 데이터의 2%만 추출하여 사용

나무위키에서 제공하는 덤프 데이터의 크기가 너무 커서
현재 작업환경에 맞게 축소 후 사용

검증은 전체 데이터의 0.1%데이터로
구현 가능한지 진행

검증 후 데이터의 크기를 2%로 늘려서 진행
데이터의 크기 = 약 17,000개, 220mb

데이터 축소는 32GB 메모리를 가진 작업환경에서 진행
16GB 메모리의 작업환경에선 원본 데이터 로드 불가능

NAMU-Branch

문제점과 해결방안 - 2

Word2Vec 모델생성 차질

문서 전체를 형태소 분리, 임베딩 모델 생성 단계에서 너무 많은 시간 소요

16,014개의 데이터, 139,827의 고유 단어(토큰)
임베딩 차원은 42,558개

차원이 너무 많고 토큰도 너무 많기에
데이터 전처리 다시 진행

문서당 최대 토큰수 2,000개 제한,
문서에서 5번 미만 언급되는 토큰 삭제,
전체 문서의 80%에 언급되는 토큰 삭제,
100토큰 미만의 데이터 삭제
임베딩 모델 학습 알고리즘 변경
(Skip-gram ⇒ CBOW)

NAMU-Branch

문제점과 해결방안 - 3

완벽하지 않았던 전처리

노이즈 데이터 토큰화

"~되었다" ⇒ "되어다" 로 토큰화

Okt로 토큰화를 진행해서
명사, 동사, 형용사만 추출했지만
stopword 리스트에 없는 쓰레기 토큰이 추가되었다.

노이즈 토큰은 문서의 벡터값을 망가트리기에
따로 삭제를 진행하였다.

순천
방송국

NAMU-Branch

문제점과 해결방안 - 4

토큰 수가 적은 데이터의 노이즈

토큰 수가 적은 데이터가 추천 알고리즘을 오염
100 토큰 미만 데이터 삭제

키워드 추천에서 키워드와
전혀 상관이 없는 문서를 추천해줌

워드 클라우드로 확인해보니 단 두개의 토큰
원 문서는 30자도 안되는 토막글

문서 벡터값을 망가트리고
전처리 데이터 연산 시간을 늘리기에
100 토큰 미만 문서 삭제

데이터 : 16,014 ⇒ 11,384

06 느낀점과 프로젝트 보완사항

느낀점

혼자 프로젝트를 진행하면서 시간적 한계를 느낌

AI 사용은 자신의 역량을 키우는데 부정적 효과를 불러올 것이라 생각했지만 AI를 얼마나 잘 다루는 스킬이 현대 개발자가 갖춰야할 덕목이라 느낌

단순한 데이터 처리에도 상당한 사양의 작업환경이 요구될 수 있다는 점이 놀라웠다.

프로젝트 보완사항

UI 보완 (스크롤바 삽입 등)

개요 데이터 컬럼 문자열 처리 문제
원본 데이터 = "== 개요 == [[asdf...."
우리가 읽기 쉬운 자연어로 처리 필요

처리한 개요를 한 CSV에 추가 작업
현재 구현된 프로그램은 두개의 CSV를 읽는 형식

NAMU-Branch

감사합니다!

Github

https://github.com/chipmunk-tail/NAMU-Branch-recommendation-for-namuwiki.git