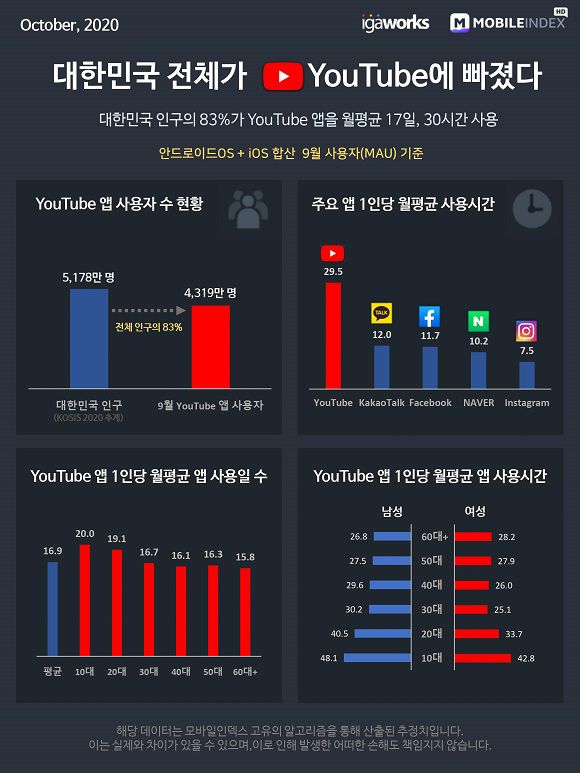
|  |
| --- |
| **Text Rank 기법을 활용한 유튜브 자막 요약**  2015104194 이규호  2014104112 신주민  **요 약**  최근 세계적으로 유튜브에 대한 관심도가 급증하고 있다. 단순히 자극적인 동영상만 올라오던 초창기와는 달리 현재는 유명 인사들의 강연, 정보 소개 영상, 수준 높은 강의 등 유튜브 영상을 통해 습득할 수 있는 정보들이 굉장히 많아지고 있다. 동영상이라는 매개체는 글과는 다르게 시각적 정보를 제공해주기에 몰입도가 높지만, 빠르게 볼 수가 없다. 본 연구는 이러한 상황에서 선택지를 제공해주기 위해 시작되었다. 본 프로젝트에서는 유튜브 영상의 자막을 추출하고 Graph centrality를 텍스트에 적용한 Text Rank 알고리즘을 활용하여, 해당 영상의 내용을 원하는 압축률로 요약하여 사용자에게 제공해준다. |

**1. 서론**

**1.1. 연구배경**

[그림1] 대한민국 국민들의 Youtube 사용률

유튜브 시장은 이미 동영상을 넘어 음악, 뉴스, 검색 등 다양한 기능과 콘텐츠를 제공하는 다면 플랫폼으로 진화하고 있다. 단순히 여가 생활을 위한 콘텐츠만을 제공하는 것이 아니라 강연, 정보 제공, 강의 등 교육적인 목표를 가진 채널과 이를 사용하는 시청자들도 굉장히 많아졌다.

유튜브의 월 사용자 수는 19억명에 달하며 전 세계인이 매일 유튜브에서 10억 시간 이상의 분량의 동영상을 시청하고 있다. 국내의 경우, 대한민국 인구의 83%가 유튜브 앱을 월평균 17일, 30시간 이상을 사용하고 있다.([그림1] 참조)

이러한 추세에서 유튜브가 갖는 파급 효과는 매우 높다. 하지만 교육을 목적으로 하는 유튜브 영상들의 경우에는 기본적으로 영상의 길이가 다른 콘텐츠에 비해 긴 편이다. 인터넷 강의와 마찬가지로 이를 긴 시간동안 집중해서 보는 것은 힘들다. 특히 특정 정보만을 얻고 싶음에도 불구하고 긴 시간동안 영상을 시청하는 경우도 잦다. 사용자가 원하는 정보를 얻는데 소요되는 시간을 보다 효과적으로 줄이기 위한 방법이 필요하다.

**1.2. 연구목표**

본 연구에서는 동영상 매체인 유튜브 영상을 텍스트로 요약한다. 이를 위해 유튜브에서 제공하는 xml형태의 자막을 추출하여 Text Rank 알고리즘에 활용한다.

큰 학습 데이터를 사용하여 정밀한 모델링이 필요한 abstractive approaches 기법이 아닌 extractive approaches 기법을 사용하여 요약 알고리즘을 설계한다. 주어진 자막 내에서 이를 대표할 수 있는 단어들이나 문장들을 선택하여 결과물을 만들어낸다. 이는 모델 특성상 학습도 굉장히 빠르다. 핵심 단어를 선택하기 위해서 단어 간의 co-occurrence graph를 만든다. 이 후 각각 그래프에 PageRank를 학습하여 각 마디의 rank를 계산한다. 높은 rank를 갖는 키워드를 핵심 문장으로 채택하고 이를 기반으로 텍스트 요약을 진행한다.

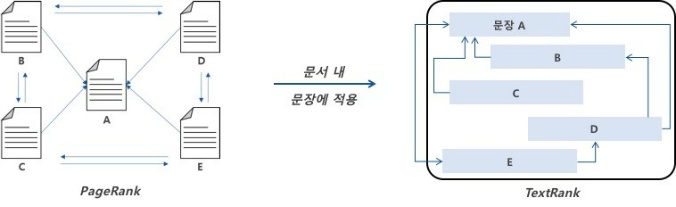
유튜브 영상이라는 한정된 콘텐츠에 대한 요약이기에 이를 활용한다. 일반적으로 강의 영상의 제목에는 무엇을 가르쳐 줄 것인가를 표기하는 경우가 대부분이다. 따라서 영상의 제목에 나온 텍스트들의 중요도를 높여서 더 주제에 근접한 텍스트들을 반환하는 작업을 거칠 것이다.

원하는 영상의 URL과 원하는 압축률 두 가지의 정보를 갖고 서버내에서 알고리즘을 적용하여 요약된 문장들을 반환할 수 있는 프로그램을 개발하는 것이 목적이다.

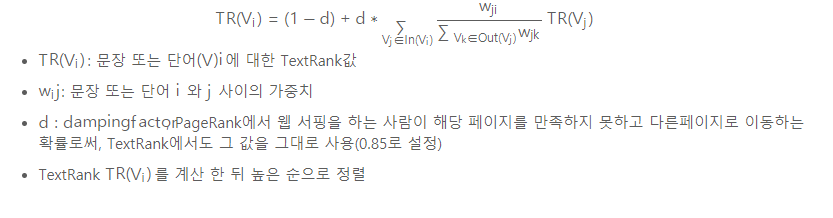
**2. 관련연구**

**2.1. Text Rank Algorithm**

글의 키워드와 핵심문장을 골라내기 위해 가장 많이 사용되는 알고리즘 중 하나인 Text Rank 알고리즘이다. Text Rank 알고리즘은 단어 graph나 문장 graph를 구축하고 구글의 Page Rank 알고리즘을 이용하여 키워드와 핵심문장을 골라낸다. 그리고 이러한 방식으로 골라낸 키워드와 핵심문장을 이용하여 글을 요약하는 방식이다.



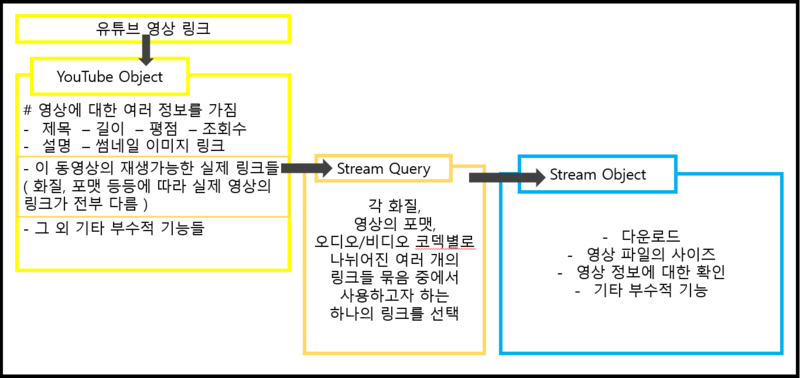
[그림2] PageRank -> TextRank 적용

TextRank을 위한 식은 다음과 같다.

[그림3] TextRank Algorithm

**2.2. PyTube**

유튜브 동영상의 링크를 가져와서 해당 영상에 대한 정보를 얻을 수 있는 Python에서 제공하는 라이브러리다. 유튜브의 URL을 입력하면, 그에 상응하는 Youtube의 Object를 얻을 수 있다. 상세한 Object의 정보는 [그림4]에서 확인할 수 있다.



[그림4] pytube의 YouTube 객체 정보

**2.3. Komoran**

Shineware의 기술로만 연구 및 개발된 한국어 형태소 분석기이다. 기존 Java Libraray 형태로만 제공되던 방식에서, 최근 PyKomoran을 활용하여 Python에서도 Komoran을 이용할 수 잇다. 이는 여러 어절을 하나의 품사로 분석 가능함으로써, 다른 형태소 분석기와는 달리 공백이 포함된 고유명사를 더 정확하게 분석할 수 있다.

한국어를 5언 9품사로 구분되는 형태소로 태깅 해준다. Hash 기반의 파싱 방법을 적용하여 한글 분석 속도가 뛰어나며, TRIE 기반의 사전 구성으로 사전의 탐색 속도도 뛰어나다. 사용자 사전을 추가하여 신조어, 고유명사 등을 분석할 수 있어 개발자가 원하는 방향대로 모델링을 새로 구성할 수 있다.

**2.4. numpy**

Numarray와 Numeric이라는 오래된 Python 패키지를 계승해서 나온 수학 및 과학 연산을 위한 Python 패키지이다. Python에서 편리한 수치해석을 제공한다. 내부는 상당부분이 C나 포트란으로 작성되어 있어 실행 속도도 빠른 편이다. 기본적으로 array라는 자료를 생성하고 이를 바탕으로 색인, 처리, 연산 등을 하는 기능을 수행한다. Python에서 수치해석, 통계 관련 기능을 구현하고자 할 때 기본이 되는 모듈이다.

**2.4. SciPy**

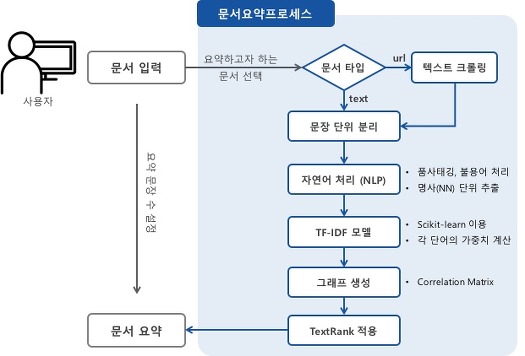
Python을 기반으로 하여 과학, 분석, 그리고 엔지니어링을 위한 과학적 컴퓨팅 영역의 여러 기본적인 작업을 위한 라이브러리이다. 기본적으로 Numpy와 함께 동작한다. 수치적분 루틴과 미분방정식 해석기, 방정식의 근, 표준 연속/이산 확률분포와 다양한 통계관련 도구들을 제공한다. Numpy와 Scipy를 함께 사용하면 확장 애드온을 포함한 MATLAB을 완벽하게 대체할 수 있다.

**2.5. scikit-learn**

sckit-learn은 2007년 구글 썸머 코드에서 처음 구현됐으며, Python으로 구현된 기계 학습 오픈 소스 라이브러리다. 이는 크게 지도/비지도 학습, 모델 선택 및 평가, 데이터 변환으로 나눌 수 있다. 지도 학습에서는 서포트 벡터 머신, Naïve Bayes, 결정 트리 등을 제공하며, 비지도 학습에서는 군집화, 이상치 검출 등이 있다. 모델 선택 및 평가에는 교차 검증, 파이프라인 등이 있으며, 마지막으로 데이터 변환에서는 속성 추출, 전처리 등이 있다. Python에서 기계 학습에 대한 여러가지 기법을 쉽게 사용할 수 있다.

**3. 프로젝트 내용**

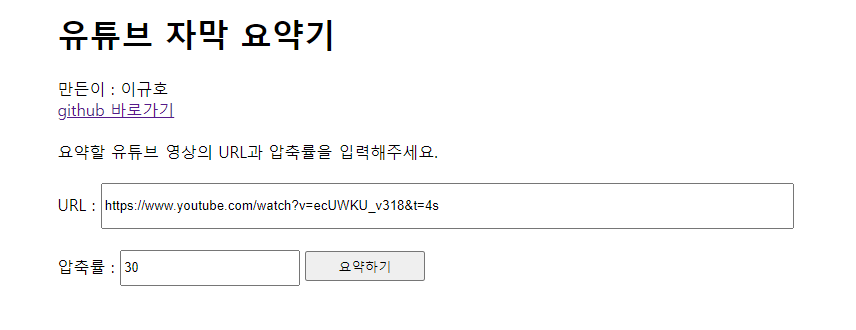
**3.1. 자막 요약 프로세스**



[그림5] 문서요약 프로세스

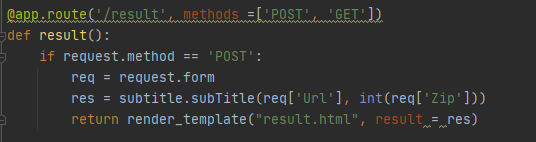
**3.2. 시나리오**

**3.2.1 Web Front-End, Back-End**

****

[그림6] URL을 입력 받는 Front-End

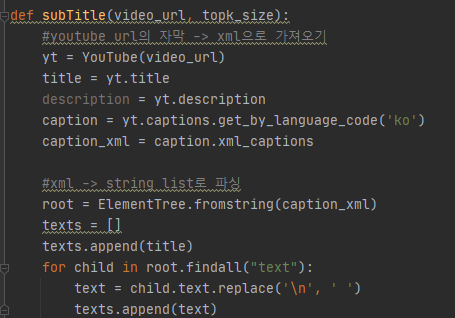
사용자는 [그림6]에 구현된 웹 페이지에 요약을 원하는 YouTube영상의 URL과 압축률을 입력한다.



[그림7] BackEnd에서의 데이터 처리

그 후 [그림7]과 같이Flask로 구현된 로컬 서버에서 입력 받은 URL과 압축률을 기반으로 subtitle.py에서 반환된 결과물을 렌더링해준다.

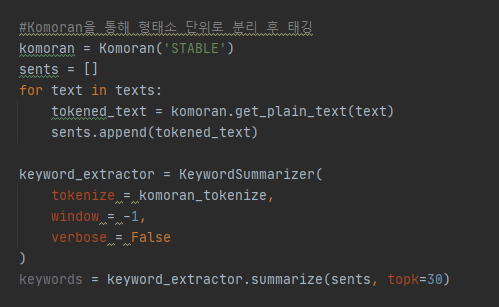
**3.2.2 유튜브 자막 추출**



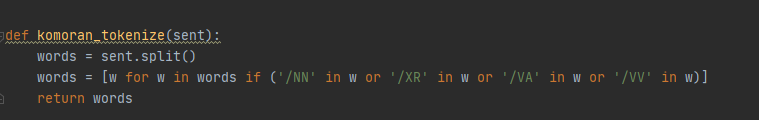
[그림8] 유튜브 자막 추출

URL이 입력되면, 제일 처음 [그림8]의 코드에서는 이를 데이터화 하기 위해서 string list로 파싱한다. 이 때, 한국어 자막이 있어야만 알고리즘이 동작 가능하다. 처음에 PyTube에서 xml을 통해 제공된 자막을 문장별로 나누어 text라는 list에 저장한다.

**3.2.3 형태소 분석**



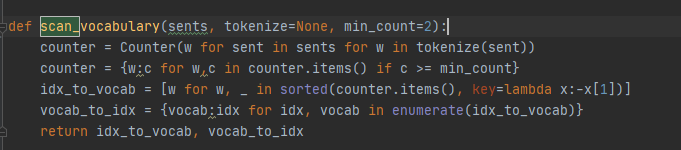
[그림9] 형태소 분석



[그림10] tokenize 함수

이 후 [그림9]에서 komoran을 통해 각 문장들을 형태소 단위로 나누어 품사를 붙이는 태깅 작업을 수행한다. 명사, 어근, 형용사, 동사를 제외한 품사들은 문장의 뜻과 관련이 없으므로 삭제하고([그림10] 참조), 필요한 품사들을 이용하여 유사도를 측정한다.

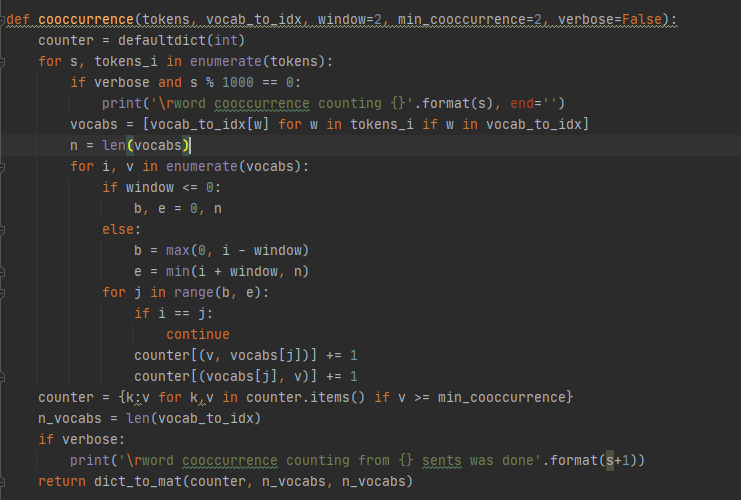
**3.2.4 키워드 추출**



[그림11] 키워드 추출 함수

키워드를 추출하기 위해서 먼저 단어 그래프를 만들었다. 마디인 단어는 주어진 문서 집합에서 최소 빈도수(min\_count) 이상 등장한 단어들이다. Sents는 string list 형식의 문장들이며, tokenize는 string 형식의 문장을 list of string 형식의 단어열로 나누는 토크나이저이다.

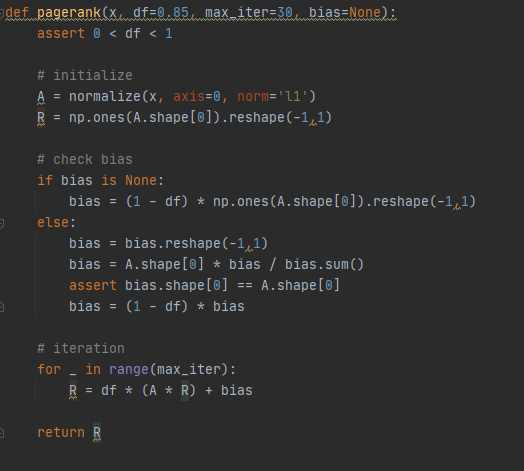
**3.2.5 유사도 측정**

****

[그림12] 유사도 측정 함수

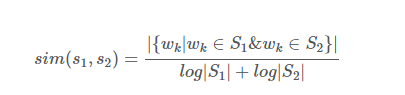
TextRank에서 두 단어 간의 유사도를 정의하기 위해서는, 두 단어의 co-occurrence를 계산해야 한다. 이는 문장 내에서의 두 단어의 간격이 일정한 정수인 횟수이다. 여기서는 문장 내에 함께 등장한 모든 경우를 co-occurrence로 정의하였다. 이 후 dict\_to\_mat 함수를 통해 dictionary of dictionary 형식의 그래프를 scipy의 sparse matrix 형태로 변환하여 반환했다.

**3.2.6 PageRank**



[그림13] PageRank 함수

[그림13]은 이렇게 만들어진 그래프에 PageRank를 학습하는 함수이다. 입력되는 x는 co-occurrence 그래프일 수 있으니, column sum이 1이 되도록 L1 normalization을 한다. 각 rank가 전달되는 값에 df를 곱하고, 모든 마디에 1-df를 더한다. 이를 max\_iter만큼 반복한다.

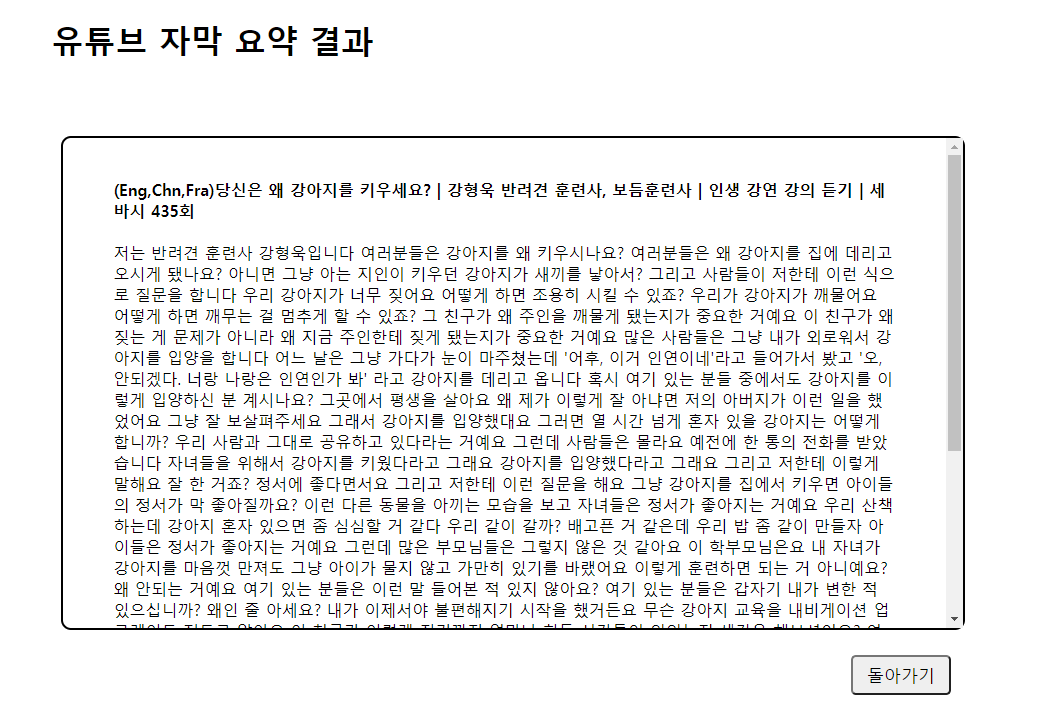


[그림14] 유사도 공식

TextRank를 이용하여 핵심 문장을 추출하기 위해서는 문장 그래프를 만들어야 한다. 각 문장이 마디가 되며, edge weight는 문장간 유사도이다. [그림14]는 문장 간 유사도를 측정하기 위한 척도이다. 두 문장에 공통으로 등장한 단어의 개수를 각 문장의 단어 개수의 log 값의 합으로 나눈 것이다. 이는 문장의 길이가 길수록 높은 유사도를 지닌다. 자세한 코드는 하단 링크에서 확인할 수 있다.

<http://khuhub.khu.ac.kr/2020-2-capstone-design1/HCG_project/tree/master/src>

**3.2.7 유튜브 요약 결과 FrontEnd**



[그림15] 결과 화면

최종적으로 [그림15]와 같이 웹 페이지에서 유튜브 자막의 요약 결과를 보여준다. 이는 유튜브의 영상 길이와 압축률에 따라 길이가 상이할 수도 있기 때문에 스크롤 뷰로 구현하였다.

**4. 프로젝트 결과**

**4.1. 연구 결과**

본 프로젝트에서 3.2의 시나리오와 같이 사용자가 유튜브 영상의 URL과 압축률을 입력하면 해당되는 자막의 요약본을 제공하는 웹 페이지를 개발하였다.

서버에는 유튜브 영상의 URL을 기반으로 자막 XML을 추출하여 파싱하고, 이를 NLP의 형태소 분석을 통해서 품사를 각 형태소에 태깅한다. 이 후 각 형태소들의 중요도를 파악하여 키워드를 추출한다. 그리고 각 문장들에 포함된 단어와 문장 사이의 거리를 기반으로 문장들의 유사도를 측정한다. 이렇게 측정된 데이터들을 통해서 Text Rank 알고리즘을 적용한다. 총 문서 내에서 각 문장들의 중요도가 측정되면, 이를 순회하면서 결과값에 추가한다.

**4.2. 성능 평가**

1시간 길이의 영상의 요약본을 반환하는데 10초 이내의 시간이 걸렸고, 10분 길이의 영상의 결과는 2~3초 내에 처리되었다.

다만 요약된 결과에 대해서는 객관적인 성능을 평가하기가 힘들다. 제목에 포함된 주요 키워드와 영상에 가장 많이 나온 키워드를 기반으로 요약된 결과가 반환된다.

**4.3. 한계점**

영상의 소리가 아닌 자막을 기반으로 알고리즘을 수행하기 때문에, 한국어 자막이 지원되지 않는 영상은 요약이 지원되지 않는다. 또 영상 제목에 주요 키워드가 포함되지 않는 경우에 요약된 결과의 정확도가 조금 떨어지는 경향을 보인다.

extractive approaches 기법을 사용하였기에 자막 내에 있던 문장들만 종합하여 반환한다. 따라서 구어체로 구성된 영상의 경우에 문장의 flow가 어색한 구간이 존재한다.

또한 근본적인 알고리즘 측면에서 많이 나온 단어가 중요한 단어임은 보장되지만, 조금 나온 단어가 중요하지 않다는 전제는 거짓이다. 따라서 문장 단위가 짧고, 많이 나오지는 않았지만 흐름상 중요한 문장을 결과에서 놓치는 경우도 발생한다.

**5. 결론**

**5.1. 기대효과**

기존 텍스트 요약 알고리즘에 대한 연구는 활발히 진행되었으나, 이것이 실생활에 적용되어 사용된 경우는 찾기 힘들다. 본 프로젝트에서는 ‘강의 영상의 자막’이라는 특수한 상황에 텍스트 요약을 적용해보았다. 제목을 키워드로 설정하는 등 상황에 맞는 알고리즘을 추가로 구현해서 실생활에도 충분히 텍스트 요약 알고리즘이 적용될 수 있다는 가능성을 보여주었다. 모든 텍스트의 요약을 목표로 하지 않고, 목적을 분할하여 접근하면 더 정확도 높은 알고리즘을 구현할 수 있을 것이다.

**5.2. 추후 연구 방향**

우선적으로 현재의 프로젝트에서 텍스트 요약의 정확도를 더 증진시키는 로직이 필요하다. 예를 들어 강의에서 주요 내용을 설명하기 전에 자주 쓰는 “중요”, “기억”, “핵심”등의 단어를 포함하는 문장 혹은 이와 가까운 문장들의 중요도를 높이는 방법을 고안해 볼 수 있고, 사용자가 알고 싶은 단어를 입력하면 그 단어 위주로 요약해주는 경우도 고려해볼 수 있다.

또한 유튜브 영상의 자막만 국한되지 않고 분야를 넓혀서 더 다양한 텍스트를 요약할 때 어떤 것들을 고려해서 구현할 지 생각하여 범용성을 넓힐 수도 있겠다.

**6. 참고문헌**

[1] 유튜브 통계 :

<https://www.chosun.com/economy/tech_it/2020/10/08/NK7R4EVGR5GNDB45RYDLXJ5RUE/>

[2] 유튜브 추세 :

<https://spri.kr/posts/view/22578?code=industry_trend>

[3] TextRank :

<https://lovit.github.io/nlp/2019/04/30/textrank/>

[4] PyTube :

<https://m.blog.naver.com/PostView.nhn?blogId=skyshin0304&logNo=221620513883&proxyReferer=https:%2F%2Fwww.google.com%2F>

[5] Komoran : <https://shineware.tistory.com/entry/KOMORAN-10>

[6] Numpy : <https://numpy.org/>

[7] scipy :

<http://www.incodom.kr/%ED%8C%8C%EC%9D%B4%EC%8D%AC/%EB%9D%BC%EC%9D%B4%EB%B8%8C%EB%9F%AC%EB%A6%AC/SciPy>

[8] scikit-learn : <http://hleecaster.com/scikit-learn-cheat-sheet/>