**캡스톤 디자인 I**

**종합설계 프로젝트**

|  |  |
| --- | --- |
| 프로젝트 명 | *COPYDET* |
| 팀 명 | *Detective* |
| 문서 제목 | 1차 중간보고서 |

|  |  |
| --- | --- |
| **Version** | 3.0 |
| **Date** | 03 |

|  |  |
| --- | --- |
| **팀원** | 강 유진 (조장) |
| 김 승환 |
| 이 아르미 |
| 이 현영 |
| 전 승철 |
| **지도교수** | 강 승식 교수님 |

|  |
| --- |
| **CONFIDENTIALITY/SECURITY WARNING**  이 문서에 포함되어 있는 정보는 국민대학교 전자정보통신대학 컴퓨터공학부 및 컴퓨터공학부 개설 교과목 캡스톤 디자인I 수강 학생 중 프로젝트 “ COPYDET”를 수행하는 팀 “Detective”의 팀원들의 자산입니다. 국민대학교 컴퓨터공학부 및 팀 “Detective”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다. |

**문서 정보 / 수정 내역**

|  |  |
| --- | --- |
| **Filename** | 중간보고서1차-COPYDET.docx |
| **원안작성자** | 강유진, 김승환, 이아르미, 이현영, 전승철 |
| **수정작업자** | 강유진, 김승환, 이아르미, 이현영, 전승철 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
| 2014-03-25 | 김승환 | 1.0 | 최초 작성 | 프로젝트 목표, 수행 내용 |
| 2014-03-27 | 김승환 | 1.1 | 내용 수정 | 프로젝트 개발 배경 및 목표, 수행 내용 추가 및 보충 |
| 2014-03-29 | 이현영 | 1.2 | 내용 수정 | 수정사항 및 향후 진행 계획 설정 |
| 2014-04-01 | 전승철 | 1.3 | 수행 내용 추가 | Source-Retrieval 개발 내용 보충 |
| 2014-04-03 | 강유진 | 1.4 | 수행 내용 추가 | Text-Alignment 개발 내용 보충 |
| 2014-04-03 | 이현영 | 1.5 | 수행 내용 수정/ 추가 | Text-Algorithm 개발 내용 보충 및 최종 수정 |
|  |  |  |  |  |

**목 차**

[1 프로젝트 목표 4](#_Toc384312324)

[1.1 개발 배경 4](#_Toc384312325)

[1.2 프로젝트 목표 5](#_Toc384312326)

[2 수행 내용 및 중간결과 6](#_Toc384312327)

[2.1 계획서 상의 연구내용 6](#_Toc384312328)

[2.1.1 연구 및 개발 내용 6](#_Toc384312329)

[2.1.2 개발에 쓰인 알고리즘 및 기법 7](#_Toc384312330)

[2.2 수행내용 11](#_Toc384312331)

[2.2.1 Source-Retrieval Module 개발 11](#_Toc384312332)

[2.2.2 Text-Alignment Module 개발 20](#_Toc384312333)

[3 수정된 연구내용 및 추진 방향 25](#_Toc384312334)

[3.1 수정사항 25](#_Toc384312335)

[4 향후 추진계획 27](#_Toc384312336)

[4.1 향후 계획의 세부 내용 27](#_Toc384312337)

[4.1.1 Source Retrieval 향후 계획 27](#_Toc384312338)

[4.1.2 Text Alignment 향후 계획 28](#_Toc384312339)

[4.1.3 user Interface 구현 29](#_Toc384312340)

# 프로젝트 목표

## 개발 배경

얼마 전, 교육부 장관의 논문 표절 의혹이 불거졌다. 문화계•정치계•연예계 등 다양한 분야에서 표절 문제는 끊임없이 흘러나오고 있다. 이는 비단 개인의 문제는 아닐 것이다. 표절은 심각한 정보 윤리 위반이며 우리나라뿐만 아니라 세계적으로 해결해야 하는 문제이다.

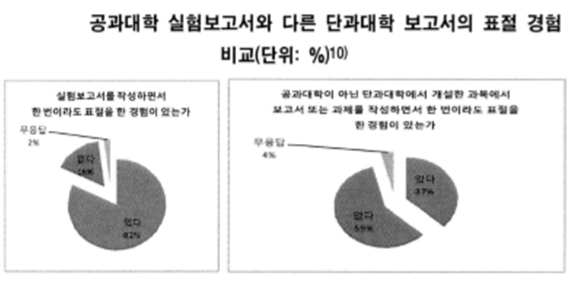


그림 1.1-1 서울대 공과대학 학생들을 대상으로 한 표절경험 비교

대학에서 가장 빈번하게 발생하는 비윤리적인 행동 중 한 가지가 표절(Plagiarism)이다. 인터넷의 발달로 학생들이 과제를 제출하거나 보고서를 제출할 때 웹사이트에서 쉽게 자료를 복사해서 제출할 수 있고 실제로 많은 학생들이 이러한 방식으로 과제를 하고 있기 때문에 학생들은 습관적으로 표절행위를 인식 없이 행하고 있다.

외국의 경우 표절을 연구 부정으로 간주하여 엄격히 금지하며, 표절에 대한 교육을 실시하고 있다. 이러한 표절행위는 학문의 창의성을 감소시키며, 연구의 정직성을 훼손시키게 된다.

정보화 시대에서 인터넷은 항상 빠르고 쉽게 만족할만한 정보를 접근할 수 있는 개념의 장으로 인식되어왔다. 하지만 이와는 반대로 신종 학문 범죄가 일어나는 장으로 변질 되고 있으며, 이렇게 얻어진 정보가 무분별하게 도용되는 사례가 증가하고 있다. 이처럼 인터넷과 같이 전자 매체가 발달한 현재에 수많은 문서를 사람이 직접 비교하여 표절여부를 판별하는 것은 시간적, 인적자원에 드는 비용은 거대하다. 그러므로 사실상 사람이 직접 표절검사를 수행한다는 것은 효율적이지 못하다. 따라서 컴퓨터를 이용해 문서의 특징을 파악하고 이를 토대로 표절 검사를 자동으로 수행하는 시스템이 필요하다.

## 프로젝트 목표

본 프로젝트의 목표는 학교 및 여러 교육기관들과 공공기관들에서 문서 표절 여부를 쉽게 찾아낼 수 있는 프로그램을 만드는 것이다. 사용자들은 이 프로그램을 통해서 손쉽게 문서들의 표절여부를 찾아낼 수 있으며 또한 사용자가 문서표절에 소비하는 시간과 비용을 절감할 수 있다. 그리고 자기소개서의 표절을 예방 할 수 있다. 또한 이 프로그램을 통해 사회 전반적으로 만연해 있는 표절에 대한 경각심을 일으키고, 표절 근절에 앞장설 수 있다.

최근 들어 유명인사의 학력 위조, 학계의 논문 표절 등이 사회적 이슈로 대두됨에 따라, 표절에 대한 관심도 급증하고 있다. 특히 그 중에서도 논문, 기사 등 한글문서의 표절이 논란이 되고 있는데, 특허청에서는 표절을 “다른 사람의 저작물의 전부나 일부를 그대로, 또는 그 형태나 내용에 다소 변경을 기하여 자신의 것으로 제공 또는 제시하는 행위”라고 정의하고 있다. 이러한 문제들이 사회에서 이슈가 되듯이, 문서 간 표절여부, 즉 문서간의 유사성을 검사하는 시스템의 연구 개발이 상당히 활발히 진행이 되고 있는 상태이다. 따라서, 현재의 유사 문서 탐색 시스템에서는 유사 문서 판정의 정확도는 물론 다량의 문서에 대한 탐색 속도 또한 중요한 척도가 되고 있다. 우리는 이 프로젝트를 통하여 기존 유사 프로그램들보다 표절 검사 속도와 정확도를 향상시키고자 한다. 일반적으로 유사 문서 탐색 속도는 대상 문서의 개수에 좌우되므로 모든 검사 대상에 대해 검사를 수행하는 것은 비효율적이다. 유사 문서 검사를 하기 전에 전처리 과정을 통하여 유사 가능성이 높은 문서만을 추출한다면 유사 문서 탐색 속도의 향상을 기대할 수 있다.

본 프로젝트에서는 전처리과정으로 의심문서와 유사 문서인 소스문서를 추출을 과정을 통하여 표절문서의 탐색시간과 정확도를 줄이고자, 수많은 문서군 사이에서 Source-Retrieval을 통해 표절의심문서와 유사도가 높은 문서들을 선별한 뒤 Text-Alignment로 각 문서간의 표절구간을 정밀하게 탐색한다.

# 수행 내용 및 중간결과

## 계획서 상의 연구내용

### 연구 및 개발 내용

문서들의 표절여부를 검사를 하기 위하여 표절의심문서와 소스문서군들 간의 유사도 검색 후 유사도가 일정치 이상의 문서들 간의 표절 구간을 찾는 프로그램을 만든다. 주요 모듈로는 Source-Retrieval과 Text-Alignment가 있다. 언어 처리에 탁월한 기능을 가지고 있으며 현재 대중적으로 많이 쓰이고 있는 Python을 개발 언어로 채택하였다.

Source-Retrieval의 역할은 자세한 표절검사를 하기 전에 표절 의심문서에서 단어를 추출하여 표절 의심 문서가 웹 상의 어떤 문서를 표절하였는지 찾아주는 것이다. 웹 상에서 표절 소스 문서가 발견되면 표절 의심 문서와 쌍을 지어 text로 출력한다.

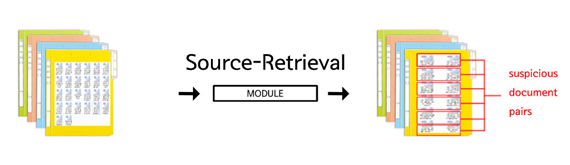


그림 2.2-1 Source-Retrieval 모듈의 역할

Text-Alignment모듈은 시그니쳐 기법과 지역정렬을 이용하여 Source-Retrieval문서에서 넘겨준 문서 쌍의 표절률과 표절구간을 찾아주는 역할을 하는 모듈이다..

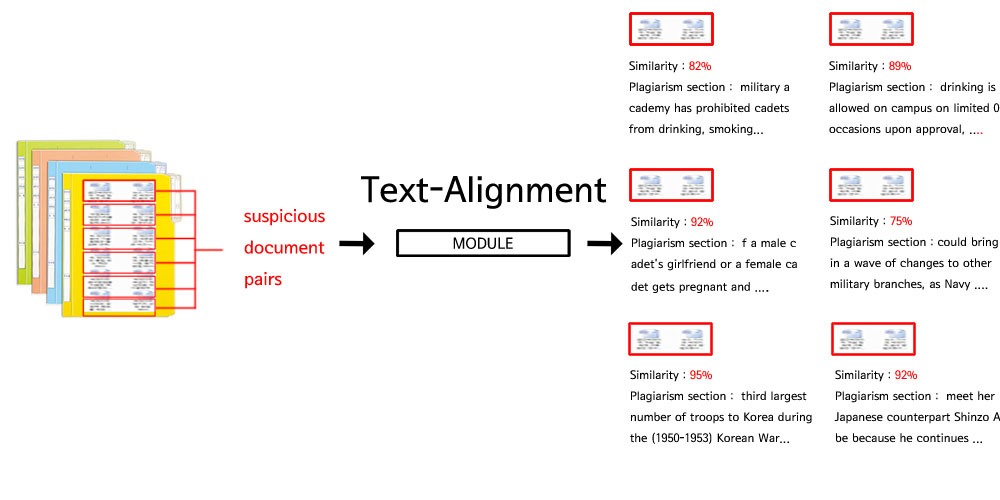


그림 2.2-2 Text-Alignment 모듈의 역할

### 개발에 쓰인 알고리즘 및 기법

#### TF-IDF

TF(Term Frequency): 단어 빈도수는 말 그대로 "단어가 그 문서에서 나타난 횟수"를 나타낸다. 단어 빈도수는 1이 아닌 그 문서에서 나타난 횟수를 나타냄으로써, 그 문서에서 그 단어가 얼마나 중요한지를 표현 할 수 있다.

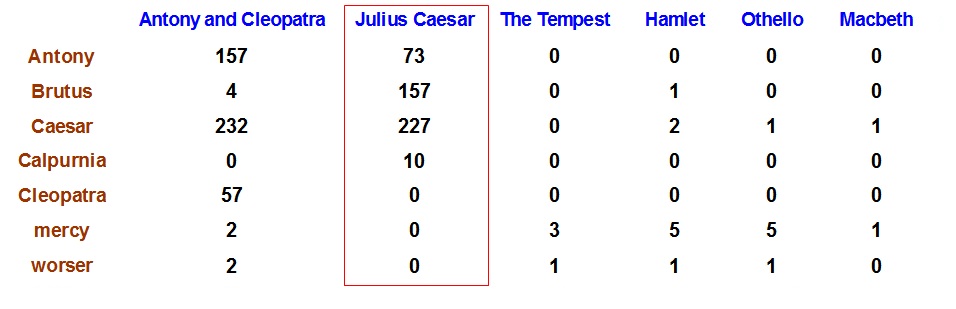


그림 2.1.2-1 TF의 예시

DF(Document Frequency) : DF는 문서 빈도수는 "해당 단어가 나타난 문서의 수" 이다. DF 값이 높은 단어는 많은 문서에서 나타나는 것이므로, 그 단어는 일반적으로 이 문서군에서는 많이 쓰이는 단어이므로 중요도가 떨어진다. 그리하여 검색에서 별로 중요한 단어가 아니라는 것을 나타낸다.

IDF(Inverse Document Frequency) : 역문서 빈도수는 DF를 역수 취한 것이다. DF는 값이 클수록 중요하지 않은 단어를 나타내는 것인데, IDF의 값이 클수록 DF란 값은 작아진다.

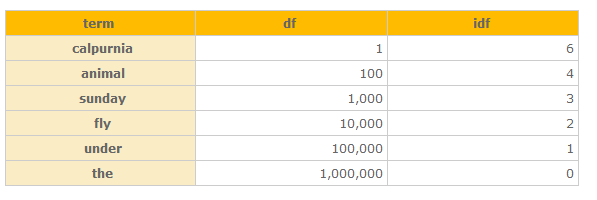


그림 2.1.2-2 DF와 IDF의 예시

TF-IDF : TF-IDF는 단어의 중요도를 수치로 표현 한 것으로써, TF \* IDF으로 이루어 진다.

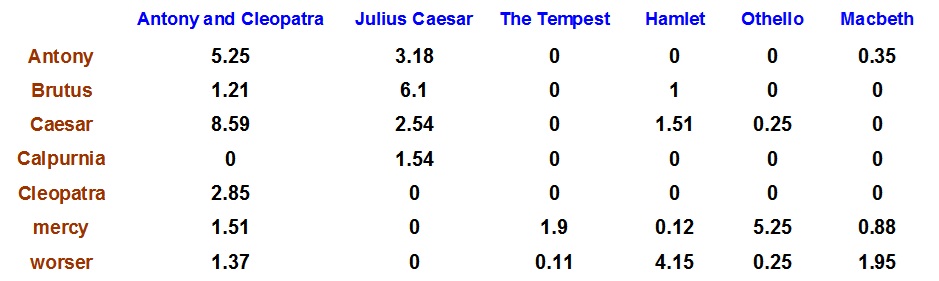


그림 2.1.2-3 TF-IDF 예시

#### IR Vector Space Model : Cosine Similarity

벡터 공간 모델에서 문서(document)는 벡터로 표현되며, 각각의 차원(dimension)은 개별 단어(term)에 대응된다. 만약, 문서 내에 특정 단어가 포함되어 있다면, 벡터 내에서 해당 차원은 0이 아닌 값을 갖게 된다. 단어 가중치(term weight)라고 하는 이 값을 계산하는 데에는 여러 가지 방법이 있으며, 가장 잘 알려진 방법 중의 하나는 TF-IDF 가중치를 구하는 방법이다.

벡터 공간 모델은 키워드 검색 시, 문서의 유사도 순위를 매기는 데에 응용될 수 있다. 문서 유사도 이론의 가정을 이용하여, 각각의 문서 벡터와 (쿼리를 문서들과 같은 종류의 벡터로 표현한) 쿼리 벡터 사이의 각도를 비교해서 유사도 순위를 매긴다. 코사인 값이 0인 경우는 쿼리와 문서 벡터가 직각인 경우로, 둘 간의 유사성이 전혀 없음을 의미한다. 그리고 코사인 값이 1에 가까울수록 두 문서간의 유사성이 높다는 것을 의미한다.

http://cfs8.tistory.com/original/8/tistory/2008/06/04/12/57/484612a3dee2b

그림 2.1.2-4 코사인 계산법

문서 d의 가중치 벡터 http://cfs7.tistory.com/original/17/tistory/2008/06/04/12/58/484612cad9e31이고http://cfs7.tistory.com/original/23/tistory/2008/06/04/12/58/484612ebb21c8이다.

여기서, tf는 문서 d에서 단어 t가 등장하는 단어 빈도(term frequency)이며 (지역 매개 변수),

http://cfs8.tistory.com/original/12/tistory/2008/06/04/13/00/48461346977a7 는 는 역 문서 빈도(inverse document frequency)이다. (전역 매개 변수)

| D |는 전체 문서 수이며, http://cfs8.tistory.com/original/17/tistory/2008/06/04/13/00/4846135fb6ec4는 은 단어 t를 포함하고 있는 문서들의 숫자이다

#### Stanford POS(Part –Of-Speech) tagger

Stanford NLP연구소에서 개발한 Python용 모듈로써 문서를 Token화 하여 각 Token마다 품사태그를 표시하는 방식이다. 태그 종류는 다음과 같다.

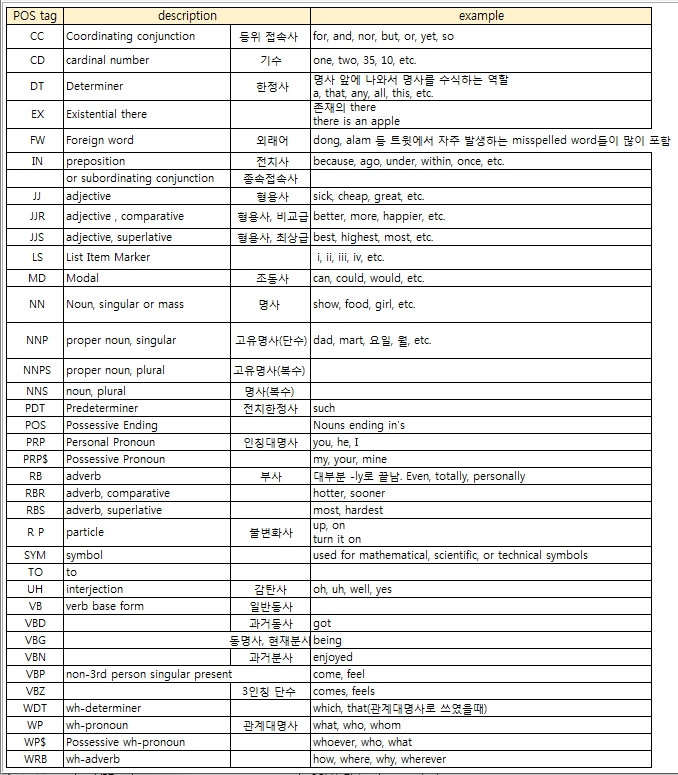


그림 2.1.2-5 POS tagger의 종류

## 수행내용

### Source-Retrieval Module 개발

#### Source-Retrieval Module 개발 개요

문서들간의 유사도를 검색하여 유서도가 일정한 수치를 넘은 문서들끼리 Text-Alignment를 하도록 TF-IDF기법으로 유사도가 높은 문서 쌍을 검출한다. Python기반으로 만들어졌다.

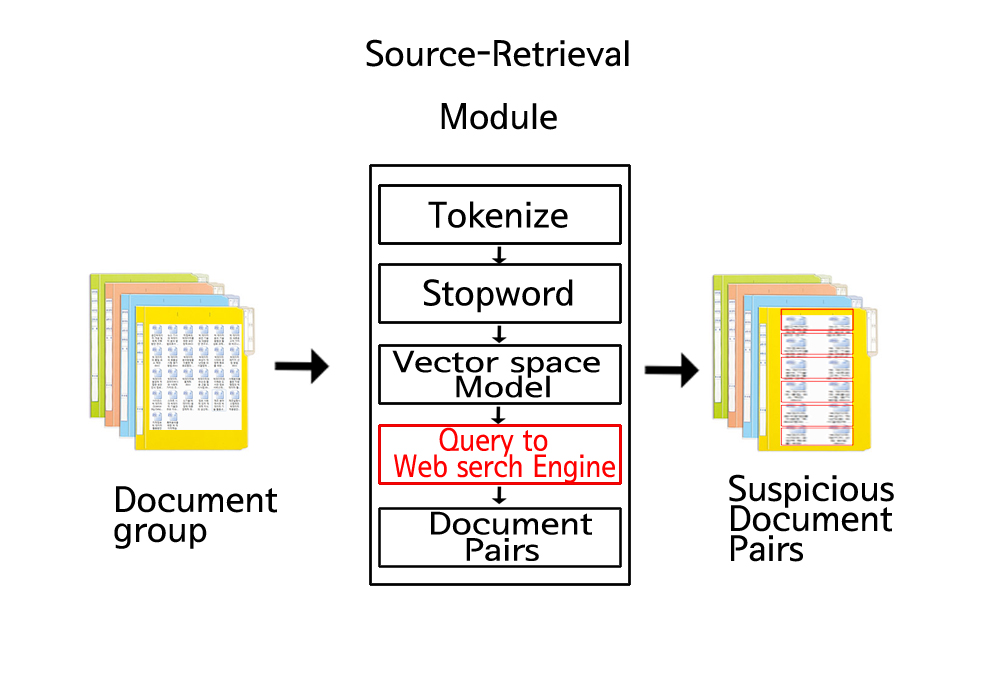


그림 2.2.1-1 Source-Retrival 개념도

* Tokenize(토큰화) : 문서의 내용을 리스트형태로 토큰화 시킨다.
* POS Tagger : 리스트의 각 token에 품사 tag를 표시한다.
* Stopword(불용어 제거) : 토큰화한 리스트에서 불용어를 제거해준다.
* Vector space Model(유사도 추출) : 벡터 스페이스 모델을 이용하여 문서의 유사도 순위를 매긴 후 유사도가 비슷한 문서들끼리 쌍을 묶어 준다.
* Query to Web serch Engine(웹 문서 비교) : 문서에서 특정 키워드를 이용하여 Web search Engine에 쿼리를 보낸다.
* Document Pairs : Vector space Model에서 추출한 유사도로 비슷한 문서들끼리 쌍을 만들어 주고 그 이후에 Text-Alignment로 넘겨준다.

#### Source-Retrieval Module 개발 수행사항

* Web Search API

현재 공개되어있는 Web Search API를 이용하여 표절문서인지 아닌지 판단하는 Source-Retrieval프로그램을 만든다. 공개되어 있는 Web Search API인 Char-Noir를 이용한다.

* 불용어 처리 함수

토큰화한 문서파일들에서 불용어들을 제거하기 위하여 만든 함수다. 기본적으로 텍스트파일에서 라인 단위로 구분을 하며 손쉽게 추가 제거가 가능하다.

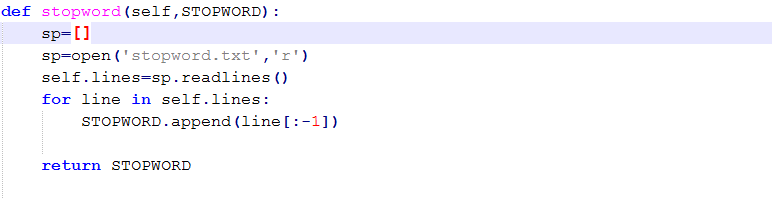


그림 2.2.1-2 불용어 처리 함수

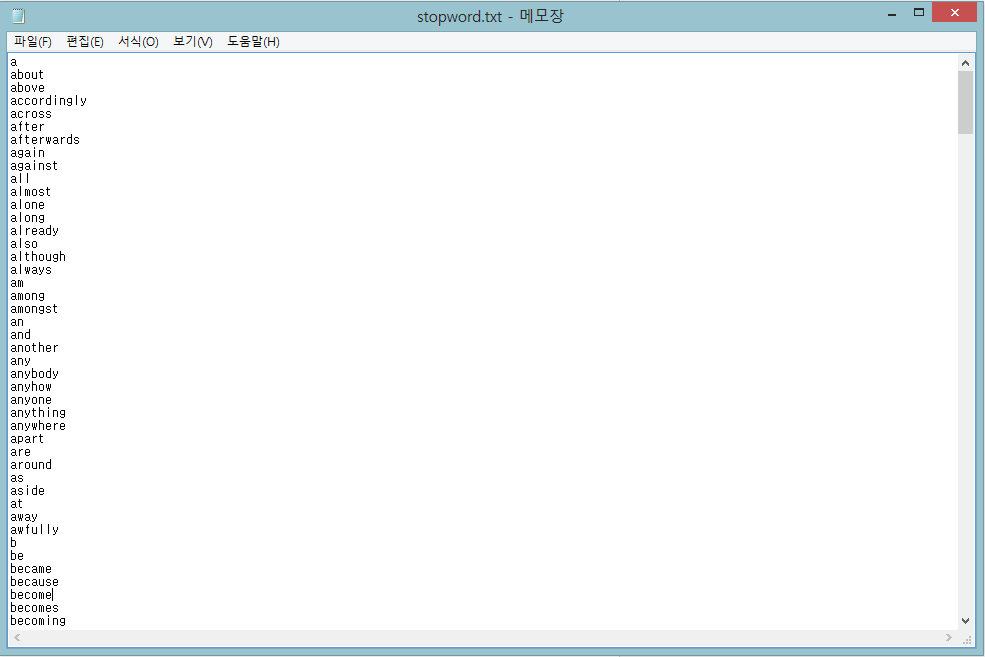


그림 2.2.1-3 불용어 목록

* 토큰화 함수

문서를 토큰화 해주는 함수. 토큰화 할 때 불용어 처리를 하여 불용어 목록에 있는 토큰들을 제거하고 리스트형식으로 만든다. 그리고 완성된 리스트에서 Stanford-PosTagger를 이용하여 각각의 토큰에 품사 tag를 달아준다.

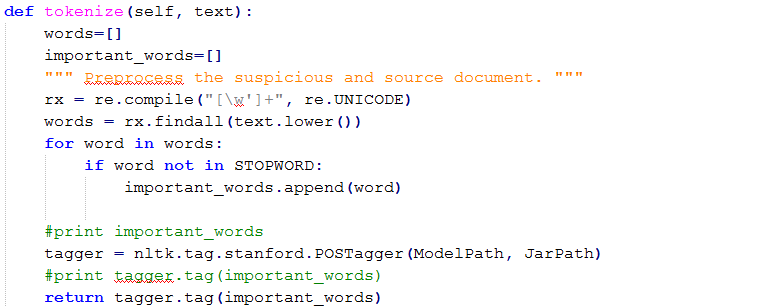


그림 2.2.1-4 토큰화 함수



그림 2.2.1-5 토큰화된 문서의 리스트

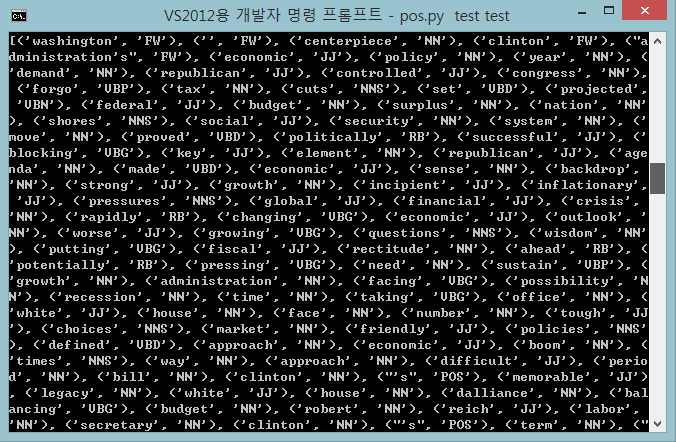


그림 2.2.1-6 각각의 토큰에 달린 품사태그

* 쿼리 추출 함수

토큰화 함수를 이용해 각각의 토큰에 tag를 달은 후에 쿼리를 추출하는 함수. 현재는 명사(NN, NNS)나 동사(VB)가 연속으로 3개가 있는 경우에 쿼리로 추출하는 방식을 사용하고 있으며 여러 실험들을 통해 효과적인 쿼리를 전송하는 방식을 찾고 있다.

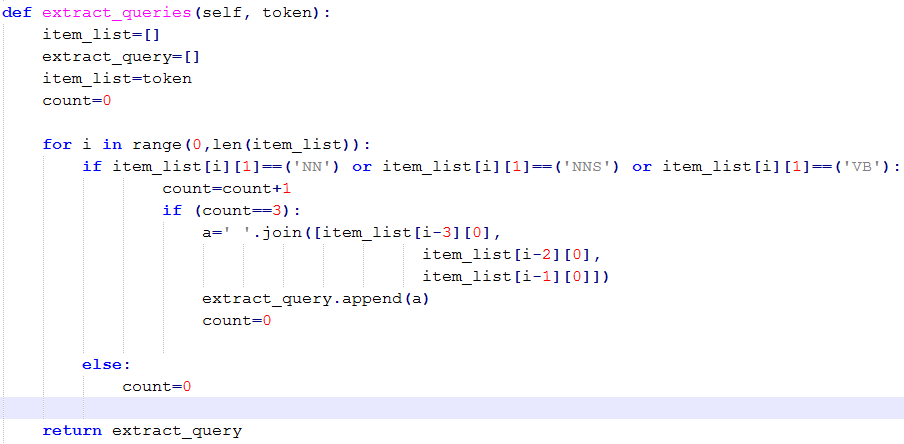


그림 2.2.1-7 쿼리 추출 함수

* 로그 저장 함수

쿼리를 보낸 시간, 쿼리, 검색 API에서 검색되는 URL을 기록하는 로그파일

총 2번 실행되며 첫 번째에 쿼리를 보낸 시간, 쿼리가 기록되며 두 번째에 검색 API에서 검색되는 URL을 기록한다.

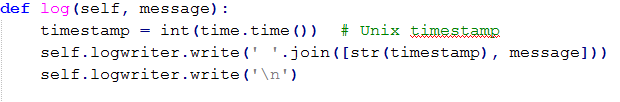


그림 2.2.1-8 보낸 쿼리를 로그로 저장하는 함수

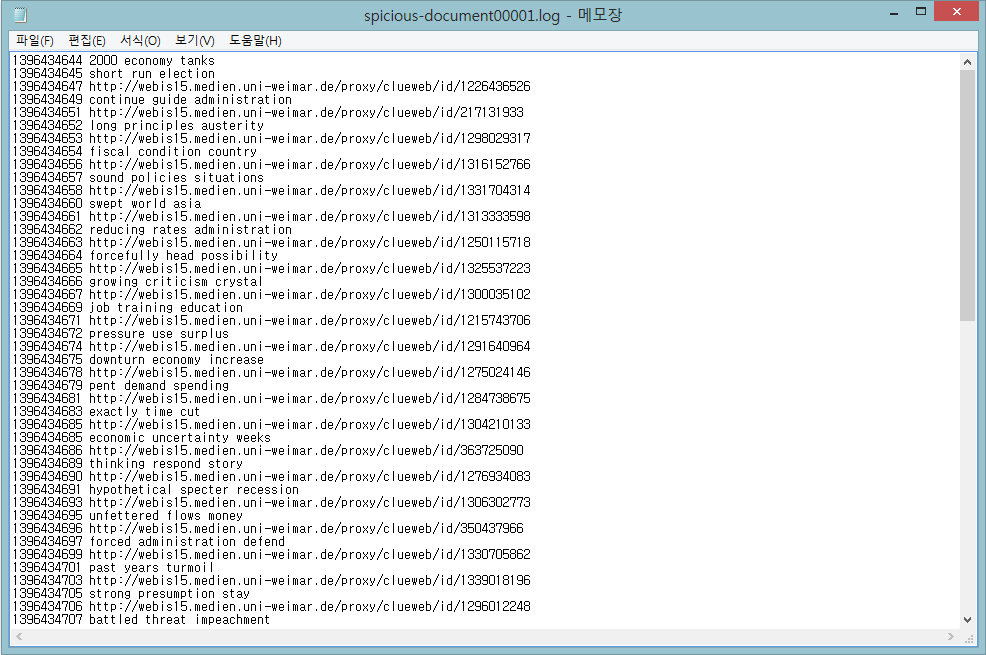


그림 2.2.1-9 로그파일

* 결과 함수

쿼리를 날린 후 검색 API에서 보내주는 결과.

결과가 없을 경우 다음 쿼리를 검색 API에 전송해주며 결과가 있을 경우 json형식의 결과를 리턴한다.

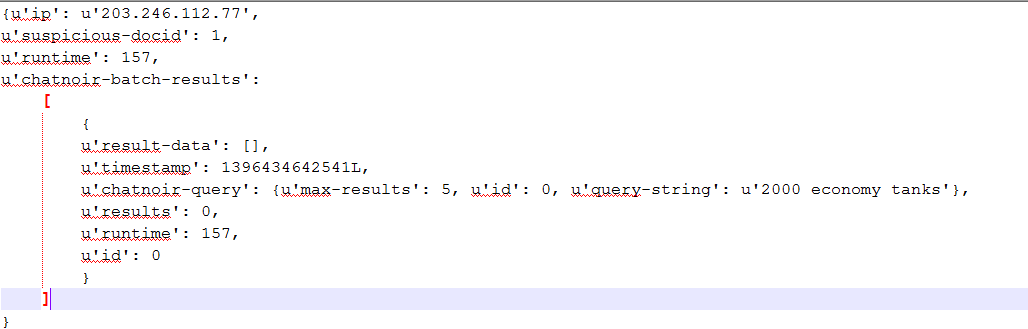


그림 2.2.1-10 전송한 쿼리에서 결과가 없을 경우



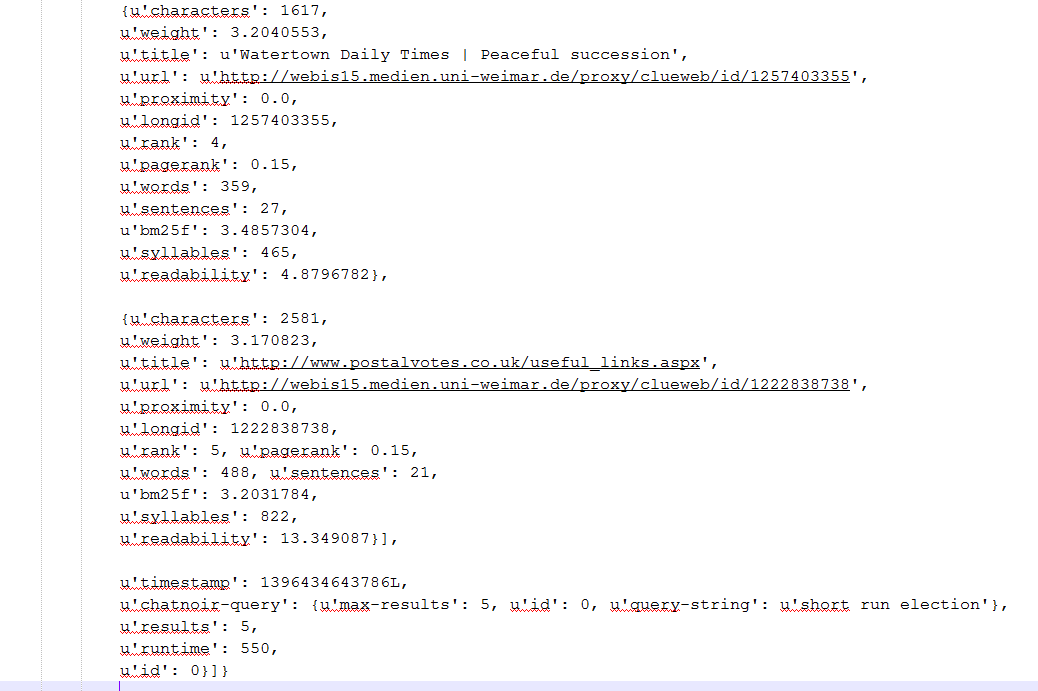


그림 2.2.1-11 전송한 쿼리에 결과가 있는 경우

* URL 다운 함수

결과에서 가져온 Json형식의 파일을 이용하여 로그파일에 기록 후에 결과에 저장되어있는 첫 번째 URL을 다운로드 받아오는 함수. 차후에 첫 번째URL만 다운받는 방식이 아니라 좀더 효율적인 방식으로 바꿀 예정이다.

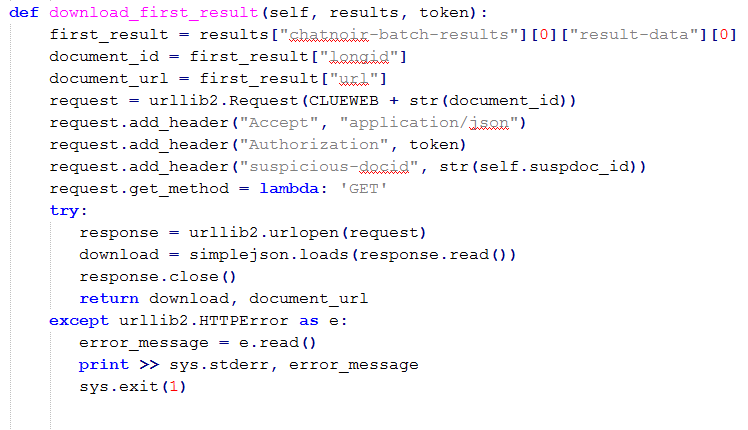


그림 2.2.1-12 URL 다운

* URL 비교 함수

다운로드 받은 URL이 표절한 문서인지 서버API의 ORACLE을 이용하여 판별하는 함수.

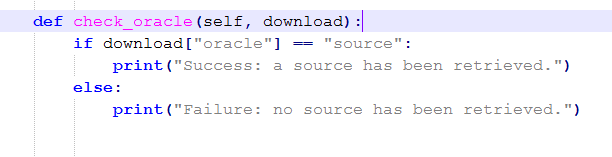


그림 2.2.1-13 URL 비교 함수

### Text-Alignment Module 개발

#### Text Alignment 개요

Source-Retrieval를 통하여 표절이 의심되는 문서가 웹 상의 소스문서를 찾을 수 있고, 표절의심와 소스문서간의 유사도를 측정을 할 수 있다. 그리하여 Text Alignment 과정에서는 웹 상의 문서 중 표절 의심 문서와의 유사도가 가장 높은 문서를 다운로드하여 표절이 의심되는 문서와 정밀한 검사를 진행을 하거나 현재 가지고 있는 소스 문서군에서 유사도가 높은 것을 추출하여 정밀검사를 진행한다. 표절이 의심되는 문서를 의심 문서, 의심 문서가 표절한 것으로 생각되는 문서를 소스 문서라고 칭하겠다.

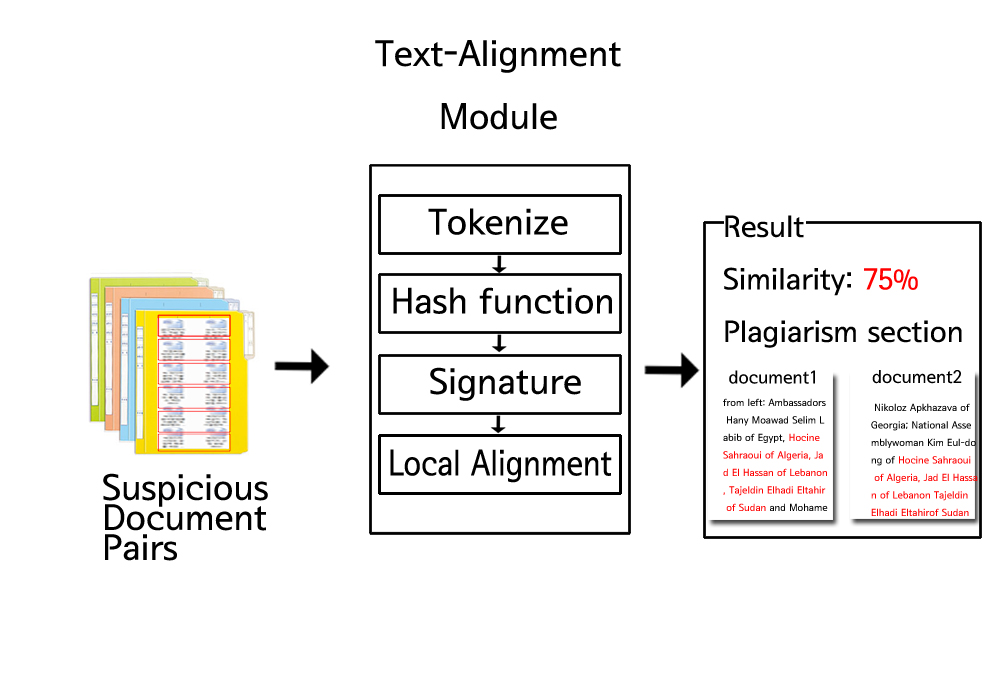


그림 2.2.2-1 Text-Alignment 1차 개념도

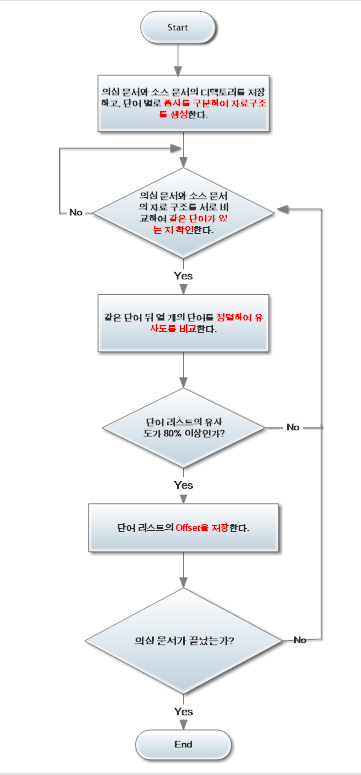


그림 2.2.2-2 Text Alignment 흐름도

#### Text Alignment 알고리즘

Text Alignment의 핵심 알고리즘은 다음과 같다.

1. 의심 문서와 소스 문서의 디렉토리를 저장한다.
2. 의심 문서와 소스 문서에 대한 자료 구조를 생성한다.
3. 각각의 자료구조를 비교해 동일 단어를 찾아낸다.
4. 동일 단어부터 각각 10개의 단어를 추출하여 리스트를 만든다.
5. 리스트를 각각 정렬하여 유사도를 비교한다.
6. 일정 유사도를 초과하는 경우 이 구간이 표절이라고 가정한다.
7. 표절 구간의 Offset을 저장한다.

#### Text Alignment 알고리즘 구현

1. 사용 라이브러리

NLTK (Natural Language Tool Kit) : NLTK 라이브러리 중 Stanford의 POS Tagger를 이용하였다. 이를 이용하여 단어의 품사를 분류하였으며, 형용사, 동사, 명사만을 중요 단어라고 간주하고 불용어를 제거하였다.

NUMPY : NUMPY를 이용하여 벡터를 표현할 수 있다. 문서 간 유사도를 구할 때 cosine 유사도를 이용하게 되는데, 이 때 사용하였다.

SCITIT LEARN : TF-IDF와 cosine 유사도를 구할 때 사용하였다. TF-IDF를 구해 벡터 모델로 만들어주며, 이를 이용한 cosine 유사도 추출 함수도 사용하였다.

1. 실제 구현 소스코드

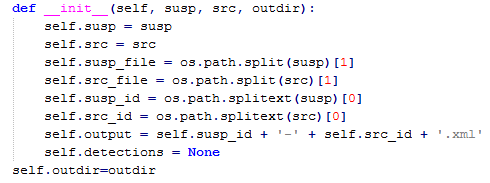


그림 2.2.2-3 Text Alignment 디렉토리 저장 함수

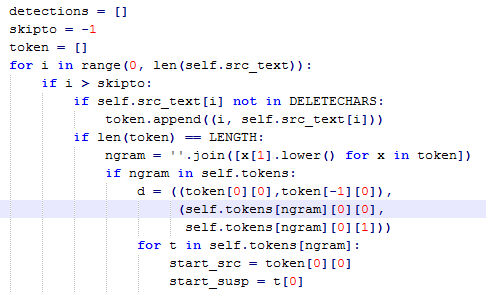
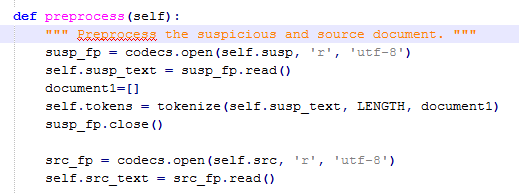


그림 2.2.2-4 두 문서에서 동일 단어를 추출하는 함수



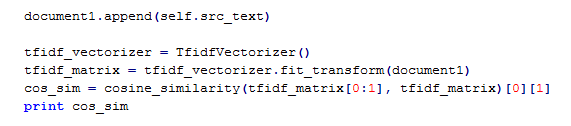


그림 2.2.2-5 Text Alignment 문서 유사도 추출 함수

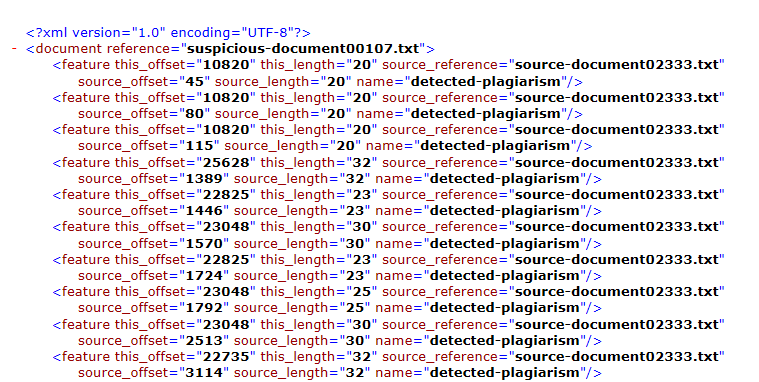


그림 2.2.2-6 Text Alignment 표절 구간 Offset 출력 XML

# 수정된 연구내용 및 추진 방향

## 수정사항

#### Source Retrieval 수정 사항

1. 웹 검색을 위한 Web Search API의 추가

Web Search API인 Chat-Noir는 약 5억개의 웹 페이지가 저장되어있는 12TB의 데이터를 소유하고 있다. 이를 이용하여 웹 페이지상의 표절된 문서들을 손쉽게 찾을 수 있는 API를 제공한다. 프로그램들을 이용하여 쿼리를 보내면 쿼리를 Web Search Engine을 이용하여 검색하고 검색결과 URL을 제공한다. 사용자는 이를 이용하여 표절 문서를 손쉽게 찾을 수 있을 뿐만 아니라 자기가 원하는 표절 문서를 쉽게 찾을 수 있다.

#### Text Alignment 수정 사항

1. 알고리즘의 수정

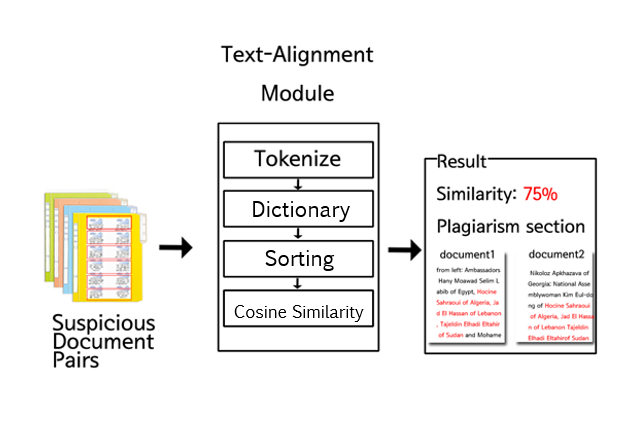


그림 3.1-1 Text Alignment 2차 개념도

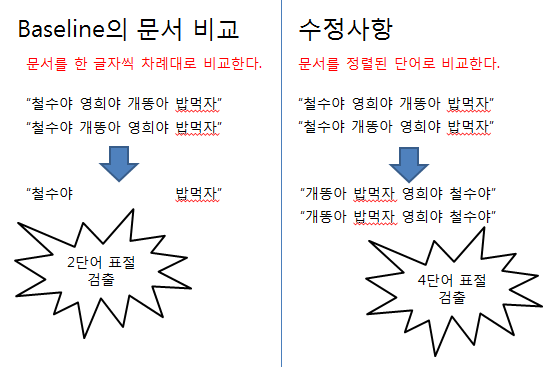
1. 구체적 알고리즘 수정 사항

그림 3.1-2 알고리즘 수정 사항

기존 알고리즘의 Hash Function 사용과 시그니쳐 생성 기법을 사용하지 않고 자료형으로 파이썬의 사전 자료형을 채택하였다. 단어의 순서가 뒤바뀌면 표절 구간을 추출하지 못하는 문제점이 있어 연속된 단어를 일정 단위의 리스트로 추출하여 정렬한 뒤 비교하였다. 비교 기법으로는 코사인 유사도 비교를 사용하였다.

# 향후 추진계획

## 향후 계획의 세부 내용

Source Retrieval과 Text Alignment 모듈을 최적화 시키는 작업을 진행할 예정이다. 실험 과정을 통해 최적화된 변수 값을 찾고 2차 중간 보고에는 실험 결과를 토대로 최적화된 모듈을 완성할 예정이다. 현재 진행중인 Algorithm을 개선을 시키거나 다른 방식으로 구현을 하면서 각 구현 모듈마다 성능 측정을 할 것이다. 또한, User Interface에 대한 부분을 추진할 계획이다

### Source Retrieval 향후 계획

#### Text Alignment 모듈과의 연동 문제

Source Retrieval 과정을 거치면 표절 의심 문서가 표절한 웹 상의 소스 문서 후보군을 알 수 있다. 하지만 정확히 어떤 문서를 표절했는가는 Text Alignment 모듈과의 적절한 연동을 통해 알 수 있다. 후보군의 모든 문서에 대해 Text Alignment 모듈을 적용할 것인가 혹은 몇 개의 문서에 대해서만 적용할 것인가에 대한 실험이 필요하다.

#### Query 최적화 문제

표절 의심 문서의 단어 중 어떤 단어를 Query로 생성할 것인가 하는 문제이다. 몇 개의 단어를 하나의 쿼리로 만들 것인가, 어떤 단어를 쿼리로 만들어야 정확도가 높아지는가에 대한 다양한 실험이 필요하다.

#### 웹 상의 문서를 다운로드 하는 횟수 최적화 문제

표절 문서가 표절했다고 생각되는 소스 문서가 여러 개일 때 문서를 모두 받아 자세한 검사를 수행할 것인가에 대한 문제이다. 만약 쿼리를 날렸을 때 웹 상의 무수히 많은 문서가 검색되었다면 이는 프로그램 성능의 저하를 야기하기 때문에 최적의 실험값이 필요하다.

### Text Alignment 향후 계획

#### 한 번에 비교하는 단어의 수 최적화 문제

현재 Text Alignment는 소스 문서와 의심 문서에서 같은 단어부터 시작하여 10개의 단어씩 정렬하여 비교하고 있다. 이는 단어의 순서를 교묘히 바꿔 표절하였을 때 표절 여부를 가려내기 위함이다. 몇 개의 단어를 정렬하여 비교할 것인가에 대한 실험이 필요하다.

#### Cosine 유사도 함수를 통한 유사도 추출의 부정확성 개선

코사인 유사도를 이용한 문서 간 유사도 비교는 리스트를 벡터화하여 비교하는 것이므로 지역 정렬로 비교하였을 때보다 정확도가 떨어진다. 유사도가 어떤 상황에서 떨어지는가, 다른 비교 기법을 사용하면 성능이 올라가는가에 대한 실험이 필요하다.

#### Algorithm 개선 및 수정사항

현재 구현한 Python의 dictionary를 이용하는 방식과 문장 별 안의 token별 첫 단어인 색인어들을 비교 방식과 성능 측정을 통한 최적화 방식 구현, 더욱더 좋은 Algorithm 방식이 있다면 구현하여 성능측정을 통한 Algorithm을 개선한다.

### user Interface 구현

User Interface에 대한 수행 결과가 없으므로 이를 구현한다. 대략적인 User Interface 흐름도는 다음과 같다.

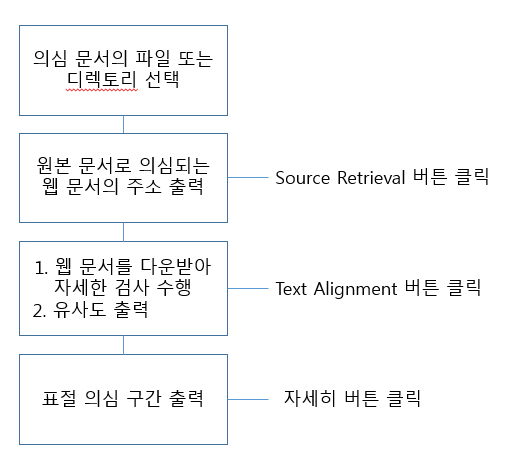


그림 4.1.3-1 User Interface 흐름도