**캡스톤 디자인 I**

**종합설계 프로젝트**

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

|  |  |
| --- | --- |
| **Version** | 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 |  | 1.0 | 최초 작성 |  |
| 2014-03-27 |  |  |  |  |
| 2014-03-29 |  |  |  |  |
| 2014-04-01 |  |  |  |  |
| 2014-04-03 |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**목 차**

[1 프로젝트 목표 4](#_Toc384244060)

[1.1 개발 배경 4](#_Toc384244061)

[1.2 프로젝트 목표 5](#_Toc384244062)

[2 수행 내용 및 중간결과 6](#_Toc384244063)

[2.1 계획서 상의 연구내용 6](#_Toc384244064)

[2.1.1 연구 및 개발 내용 6](#_Toc384244065)

[2.1.2 개발에 쓰인 알고리즘 및 기법 7](#_Toc384244066)

[2.2 수행내용 9](#_Toc384244067)

[2.2.1 관련 서비스 분석 9](#_Toc384244068)

[2.2.2 관련 논문 분석 13](#_Toc384244069)

[2.2.3 Source-Retrieval Module 개발 21](#_Toc384244070)

[2.2.4 Text-Alignment Module 개발 22](#_Toc384244071)

[3 수정된 연구내용 및 추진 방향 23](#_Toc384244072)

[3.1 수정사항 23](#_Toc384244073)

[4 향후 추진계획 24](#_Toc384244074)

[4.1 향후 계획의 세부 내용 24](#_Toc384244075)

[5 애로 및 건의사항 25](#_Toc384244076)

# 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.

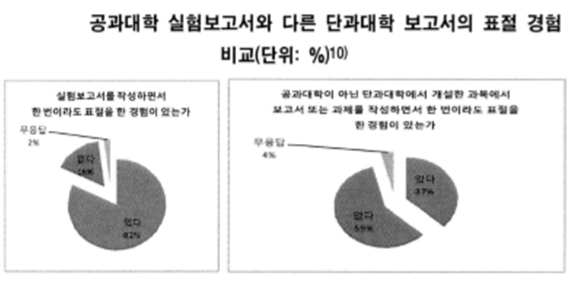


그림 1.1-1 표절경험 비교

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 표절이 비윤리적인 행위라는 인식 없이 행하고 있다.

서울대 공대생 100명을 대상으로 설문 조사를 한 결과 “실험 보고서 작성 시 표절을 한 번 이상 한 경험이 있다.”고 답한 학생이 82%, “공대가 아닌 다른 단과 대학의 보고서 작성 시 표절을 한 번 이상 한 경험이 있다”고 답한 학생이 37%였다. 또 현재의 표절 대응책의 효과가 거의 없거나 전혀 없다고 한 학생이 77%였다. 이를 바탕으로 다른 학문 분야의 과제물에서보다는 공대에서 표절이 더 빈번하다는 것을 알 수 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 사실상 불가능하다. 따라서 컴퓨터를 이용해 문서의 특징을 파악하고 이를 토대로 표절 검사를 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. 또한 이 프로그램을 통해서 아래 그림들과 같이 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 문제를 사회가 반영하듯, 문서 간 표절여부, 즉 유사성을 검사하는 시스템의 연구 개발이 상당히 보편화되어 있는 상황이다. 따라서 현재의 유사 문서 탐색 시스템에서는 유사 문서 판정의 정확도는 물론 다량의 문서에 대한 탐색 속도 또한 중요한 척도가 되고 있다. 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 유사 문서 추출 과정을 통해 유사 문서 탐색에 소요되는 시간을 줄이고자, 수많은 문서군 사이에서 Source-Retrieval을 통해 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 문서들 간의 유사도 검색 후 유사도가 비슷한 문서들의 표절 구간을 찾는 프로그램을 만든다. 주요 모듈로는 Source-Retrieval과 Text-Alignment가 있으며 언어 처리에 탁월한 기능을 가지고 있으며 현재 조원들이 쓰기 가장 편한 프로그램언어인 Python을 이용해서 만든다.

각각의 모듈의 역할 중 Source-Retrieval의 역할은 전처리과정으로 표절검사를 하기 전에 유사도를 측정을 한다. 프로그램에서 불러온 문서들을 TF-IDF기법과 코사인 유사도를 이용하여 각 문서간의 유사도를 구한 후에 각각의 문서의 유사도가 일정치 이상인 경우에 표절 의심 문서로 판단되어 Text-Alignment로 넘겨주는 역할을 하는 모듈이다.

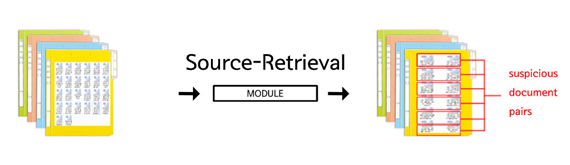


그림 2.2-1 Source-Retrieval 모듈의 역할

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

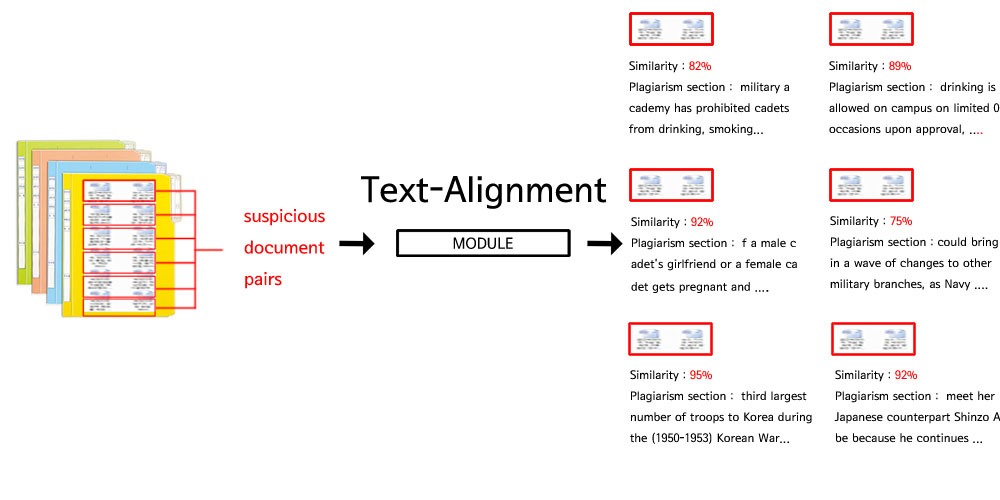


그림 2.2-2 Text-Alignment 모듈의 역할

### 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.

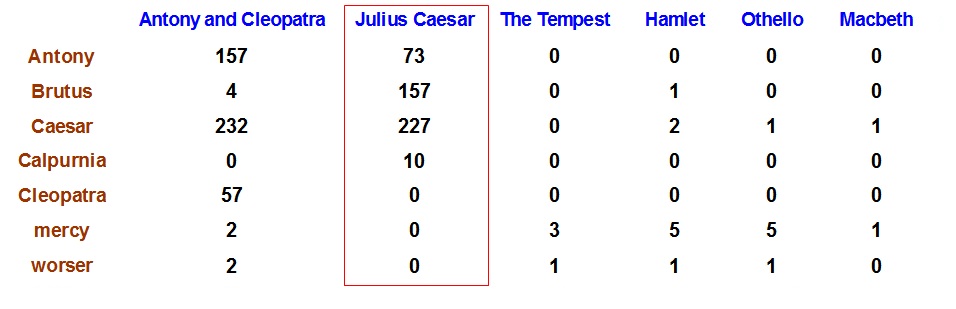


그림 2.2.2-1 TF의 예시

DF(Document Frequency) : DF는 문서 빈도수는 "해당 단어가 나타난 문서의 수" 이다. DF 값이 높은 단어는 많은 문서에서 나타나는 것이므로, 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.

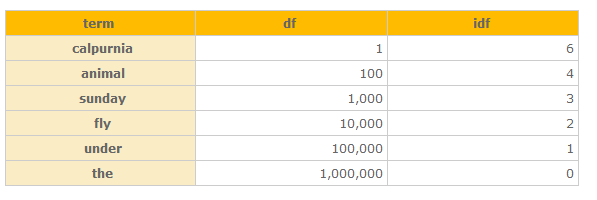


그림 2.2.2-2 DF와 IDF의 예시

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

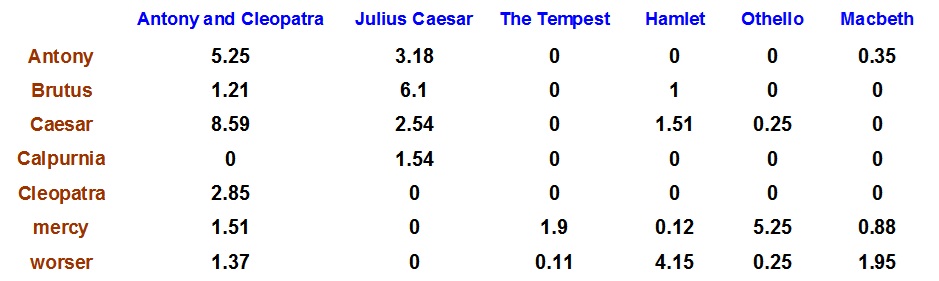


그림 2.2.2-3 TF-IDF 예시

#### 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.

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

그림 2.2.2-4 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

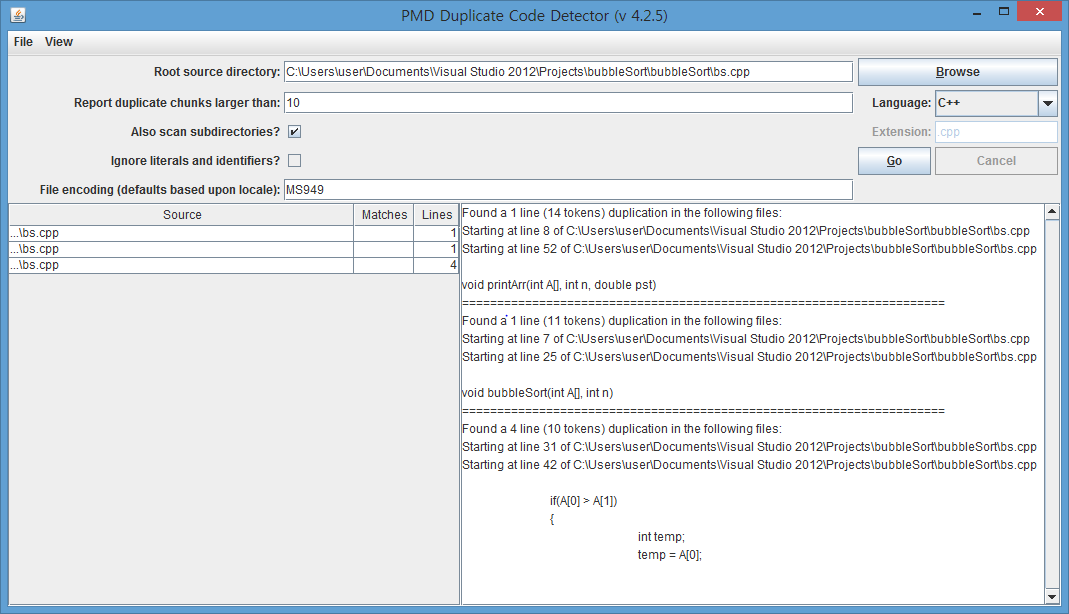
## 수행내용

### 관련 서비스 분석

#### CPD(Copy/Paste Detector)

PMD는 코드가 실제로 동작하기 전에 검토하여주는 정적 분석도구이다. 효율적이고 간결한 프로그램이 되려면 중복되는 코드를 최대한 없애야 하므로 중복코드를 찾기 위해 Copy/Paste Detector(CPD)를 제공한다. CPD의 기본 알고리즘은 Karp-Rabin 알고리즘이다.

Karp-Rabin은 문자열 패턴을 수치로 바꾸어 문자열의 비교를 수치 비교로 대신한다.



System.out.println("Hello");

System.out.println("World");

PMD의 Java-cc로 토큰을 인식하여 소스코드를 토큰화한다(System, . , out, . , println).

공백, ; , import, package 등의 필요 없는 토큰은 제거한다.

system . out . println ( “ hello ” ) system . out . println ( “ world ” )

위의 코드로 occurrence table을 만든다면,

토큰들을 다음 토큰에 이어 붙임(expand)으로써 TABLE을 간소화시고 out은 다른 토큰 덩어리(tile)의 영역을 침범하므로 일단 무시한다.

|  |  |
| --- | --- |
| **Title** | **occurrences** |
| system | 1, 11 |
| . | 2, 4, 12, 14 |
| out | 3, 13 |
| println | 5, 15 |
| ( | 6, 16 |
| “ | 7, 9, 17, 19 |
| ) | 10, 20 |

|  |  |
| --- | --- |
| **Title** | **occurrences** |
| system. | 1, 11 |
| .println | 4, 14 |
| (“ | 6, 16 |
| “) | 10, 20 |

여기서 “)는 마지막 토큰 덩어리이므로 제거한다.

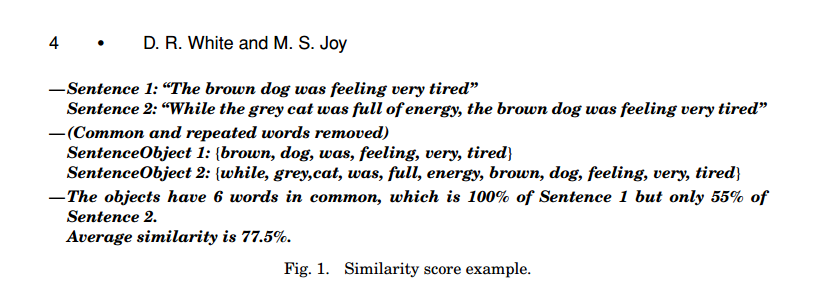
|  |  |
| --- | --- |
| **Title** | **occurrences** |
| system.out.println(“ | 1, 11 |

|  |  |
| --- | --- |
| **Title** | **occurrences** |
| system.out | 1, 11 |
| .println(“ | 4, 14 |

중복되는 소스코드는 system.out.println(“ 임을 알 수 있다.

#### The Sherlock Plagiarism Detector

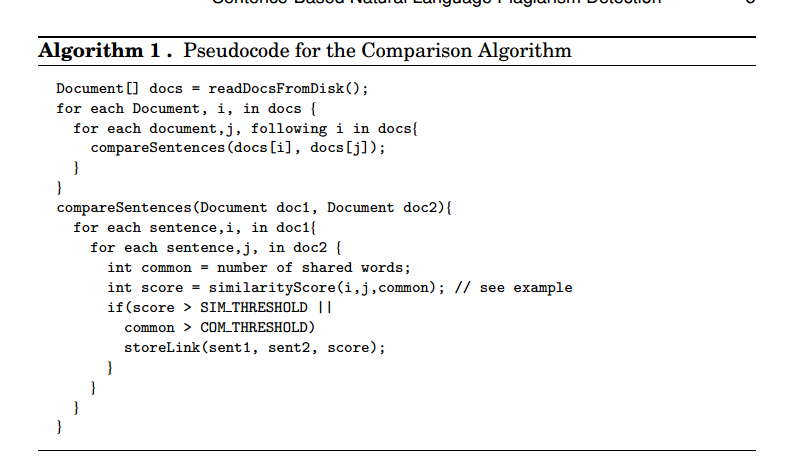
* Preprocessing Documents(Preprocessing stage)



Document objects안에 Sentence objects를 넣어서 비교를 한다.

Sentence objects안에 있는 단어 리스트는 비교 시 시간 절약을 위해 모든 단어들은 소문자로 전환이 되고, 일반적으로 필터에서 의미가 없는 단어 “the”, “a”, “that”. 을 제거한 후 반복되는 어구를 제거한다.

* Comparing Documents



Document by Document에서 모든 문장들이 비교를 한다. 비교도 검사는 Similarity score를 가지고 유사도를 분석을 한다. Similarity score는 일치하는 단어 수/전체구간 단어 수, Threshold common은 두 문장이 사이에 공통인 문자의 수를 나타낸다.

#### Plagiarism.org

Plagiarism.org은 온라인 기반의 서비스로

교수가 사이트에 등록을 하면 학생들이 과제를 올리는 방식으로 운영된다. 다른 대학교들과의 과제와 비교 할 수 있으며, 과제의 결과들이 교수님들에게 e-mail로 보내진다.

온라인 과제 및 리포트 사이트들과 학생들의 이전과제를 포함하고 있는 멀티-기가바이트 급의 데이터베이스를 유지함으로써 운영된다.

사용 알고리즘으로는 핑거프린트 알고리즘(Finger-print algorithm)을 이용한 iThenticate© 알고리즘을 사용한다.

#### *Glatt* Plagiarism Services

Screening 기능을 이용해서 학생들이 과제를 내면 글귀들의 모든 5번째 단어들을 삭제하고 빈칸으로 만들고 빈칸에 대해 학생들이 다시 입력함으로써 정확한 단어의수 채우는 시간과 다른 다양한 요인에 따른 표절점수를 부여한다. Self-Detection 기능은 표절을 피할 수 있도록 돕는 역할을 한다.

사용 알고리즘으로는 Close procedure기술을 사용하였다.

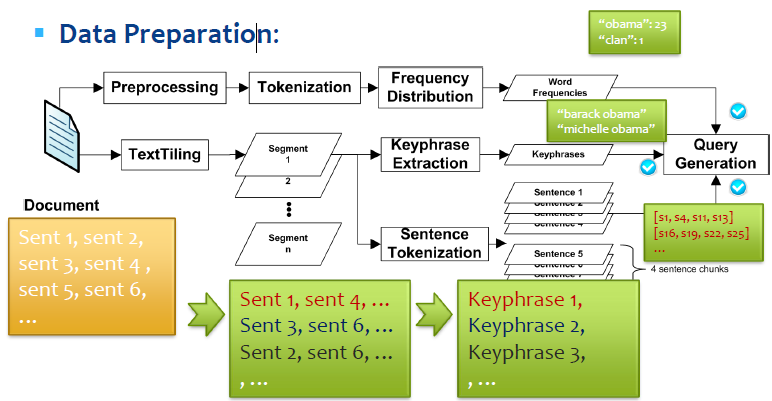
출판물의 독이성(readability)을 검토하는 데 사용하는 공식. 출판효과를 극대화하기 위해 미국 등지에서는 일찍부터 이에 관한 연구와 개발이 활발히 진행되어 왔는데, 그 대표적 공식들로는 플레시의 독이성측정공식(Flesch Reading Ease Formula), 데일-찰의 독이성측정공식(Dale-Chall Readability Formula), 거닝(Gunning)의 공식(The Fog Index), 테일러(Tayler)의 클로즈 프로시저 공식(cloze procedure) 등이 있다. 그러나 우리나라에서는 아직 한글문장의 독이성을 측정할 수 있는 공식들이 개발되어 있지 않은 실정이다.

### 관련 논문 분석

#### Plagiarism Candidate Retrieval Using Selective Query Formulation and Discriminative Query Scoring

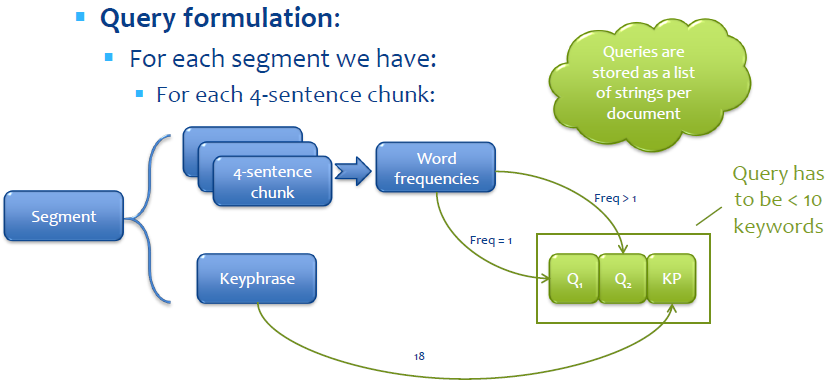
논문에서는 표절 의심 문서에서 차별 쿼리를 생성하는 방법을 제시한다. queries는 의심스러운 문서에서 possible sources를 검색하는데 이용된다.

주어진 문서를 관련 subtopics으로 나누면서 keyphrases로 구분하고, 관련 없는 문서들을 찾는 것을 피하기 위해 미리 aforementioned subtopics의 가능한 것을 찾는 것에 중점을 둔다.



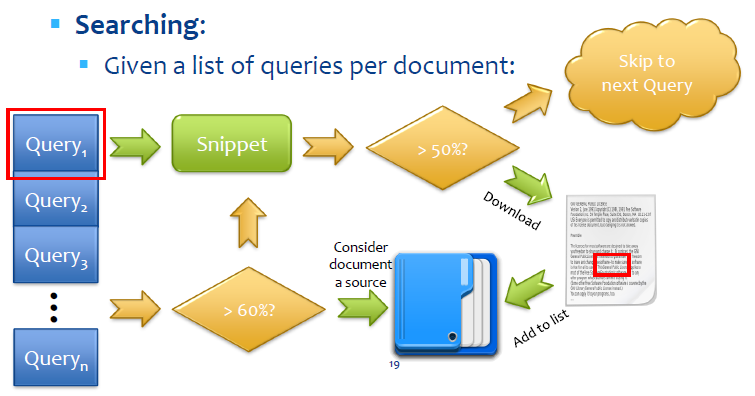
주어진 suspect plagiarized document가 입력되면, 영어가 아닌 문자(including numbers, symbols, punctuation, etc)들을 모두 제거 하는 regular expressions(정규표현식)으로 전처리된다.

그리고 공백으로 tokenized되고, 문서의 도수분포(frequency distribution)를 계산된다. 그리고 TextTiling algorithm(provided by Python’s NLTK[10] platform)을 사용한다. 문서는 주제별(topically) 관련된 부분/하위문서 (segments/subdocuments)로 분할 한다. 큰 segments를 keyphrases로 구분한다.



Keyphrases(주요구문)은 각 문서의 the KPMiner(keyphrase extraction system)을 이용해 생성된 부분(segments)로 추출된다. 이것은 segment의 topmost keyphrase를 반환한다. keyphrase는 segment의 특성을 대변한다. segment는 문장(sentences)으로 나누어지고, 모든 4문장들은 chunk로 그룹화 된다. query는 다음 chunk에 의해 생성된다.

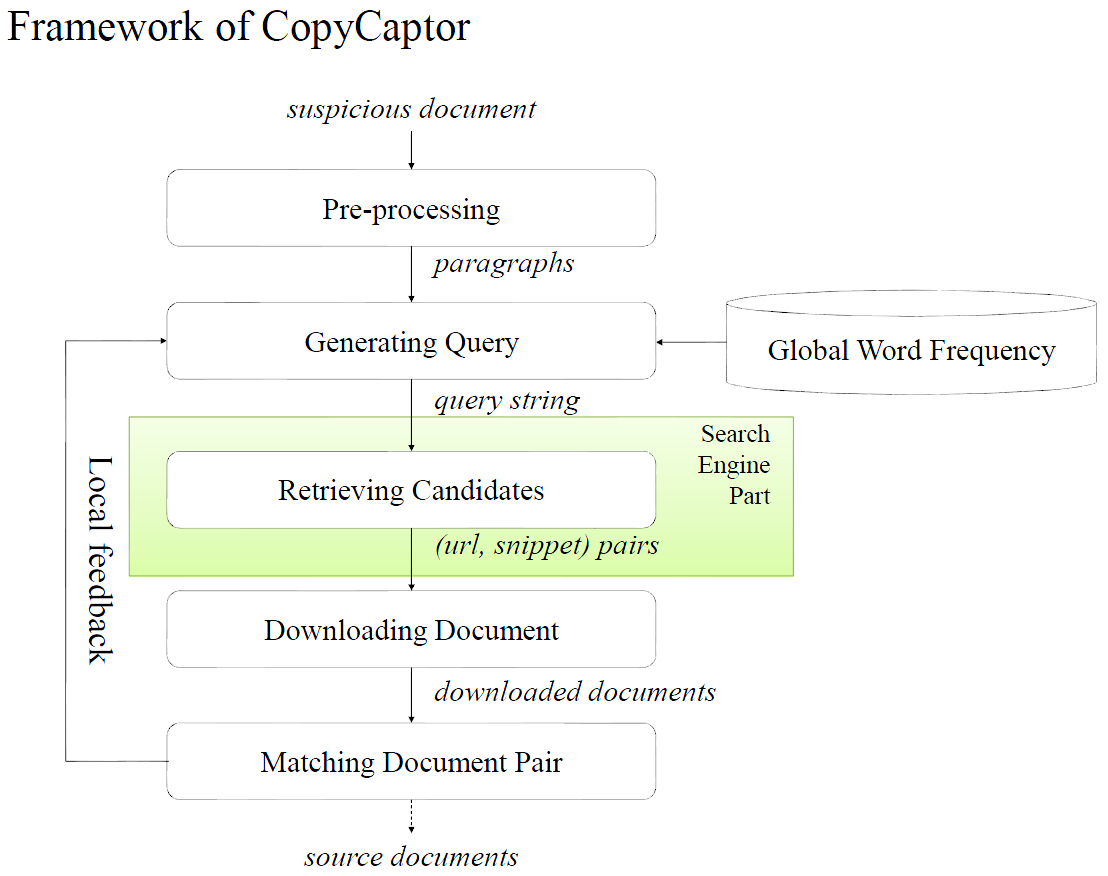
각 chunk는 정확하게 전처리 후 공백으로 tokenized되고 stopwords는 제거한 unique words를 identified한다. unique words는 chunk에서 지워지고, chunk’s query에 추가되고 chunk의 남아있는 non-unique words는 빈도에 따라 오름차순으로 정렬되고, 정렬된 순서로 query로 이동된다. query는 이제 unique words가 처음에 오고 오름차순빈도(ascending frequency)의 나머지 단어가 이런 방법으로 organized된다. 만약 query가 keyphrase’s terms에 충족되지 않는 경우 segment의 keyphrase는 query의 마지막 n words를 추가하게 된다. queries는 각 문서당 문자열의 목록으로 정렬된다.



Query는 검색결과에 대한 snippet request를 활용하여 제출된 query의 검색결과로부터 주변 글자 수를 반환한다. 첫 번째 query는 snippet을 요청하기위해 제출되고 simple token matching으로 snippet에 대한 query는 scored된다. snippet의 tokens에서 query의 token의 50%나 그이상 발견된 경우, search result는 candidate source로 간주한다. 이 후보는 다운로드 되고 다운된 문서들의 목록에 추가된다. 각각의 후속 query는 다운로드된 문서의 목록에 대하여 simple token matching을 통하여 확인한다.

다운로드 된 문서의 token에서 the query’s tokens이 60%나 그 이상 발견된 경우에는, 이 문서는 이 query에 대한 source로 간주된다. 그리고 이 query는 검색을 위해 사용되지 않는다. query가 모든 다운로드 된 문서와 60%의 일치에 실패 할 경우, 새로운 snippet request을 위해 이용되고, 새로운 후보 소스를 찾는다.

#### CopyCaptor : Plagiarized Source Retrieval System using Global word frequency and Local feedback



Pre-processing을 위해 의심 문서를 문단 단위로 나눈다. 문단 안의 단어들을 모두 tokenize하고 keyword가 아닌 단어를 빼서 stemming기법을 사용한다.

Query 생성은 연속적인 k개의 단어들을 이어 query를 만든다. 여기서는 k를 8로 두고 8개의 단어들을 이어 query를 만들었다. query를 만드는 방법은 직관적인 방법을 사용하였다.

첫 번째 직관은 unique query가 best query라는 것이다. 여기서 unique란 일반적으로 잘 쓰이지 않는 단어를 말한다. 일반적으로 잘 쓰이지 않는 단어를 중심으로 query를 만들면 표절을 찾을 수 있을 것이라는 직관이다. 단어의 일반적인 사용 빈도수는 google의 n-gram을 이용하여 구했으며, global word frequency라고 칭한다.

Retrieving Candidates을 위하여 Search Engine을 사용하여 download할 documents를 선정한다. Indri search engine을 사용하였으며, ChatNoir API를 사용하여 후보 문서들의 snippet을 검색을 이용한다. 이전 snippet과 똑같은 snippet이 보이거나 snippet에 아무 단어도 없으면 그 URL은 사용하지 않는다.

Downloading Document를 위해 후보 문서 중에 상위 k개의 URL을 다운받고, 또한 다른 query string으로부터 자주 등장하는 후보 문서 URL을 다운 받는다.

Matching Document Pair는 의심 문서와 관련된 문서가 다운되었을 수도 있고 아닐 수도 있다. 우리는 같은 문서를 비교하는 것을 막기 위해 의심 문서와 다운 문서를 document pair로 묶어 저장한다. 만약 matching ratio가 threshold보다 높으면 source document라고 받아들인다.

문단의 수보다 query의 수가 많아지면 query를 만드는 것을 중단하고 source document를 반환한다. 아니면, local feedback을 통해 다시 query를 만든다.

local feedback은 예전에 실행했던 query로부터 local feedback을 얻는다. 의심 문서는 여러 개의 문서를 표절했을 수도 있고, 그 문서들은 다른 단어들을 가지고 있으므로, 이전에 쓰였던 query는 쓰지 않으며, Matching Document Pair 과정을 통해 다운받았던 문서는 다시 보지 않는다.

frequency를 조사 했음에도 불구하고 똑같은 단어가 다른 문서에서 비교되면 matched penalty를 준다. 쿼리를 이용해서 다운로드 한 문서와 비교하는 과정에서 source document에서 쓰였던 단어이면 GWF를 2배로 늘린다.

#### 한국어 문장 표절 유형을 고려한 유사 문장 판별

N-gram이란, 인접한 N개의 음절을 말하며, N-gram 방식은 두 문장 내에 존재하는 N-gram을 추출하고 그것들 중에서 얼마나 많은 N-gram이 일치하느냐에 따라서 문장의 유사 여부를 판단하는 방법이다[13]. 다음은 N-gram 추출 방법에 대하여 설명한다. 문서에 있는 문장을 먼저 빈칸, 마침표, 쉼표 등을 구분 자로 하여 모든 어절을 추출하고, 추출한 어절들에 대해 N-gram을 추출한다. 예를 들면 “표절검색”이라는 단어의 bi-gram은 “표절”, “절검”, “검색”이며, tri-gram은 “표절검”, “절검색” 이다.

어절의 음절 수가 N보다 큰 경우에는 여러 개의 N-gram으로 분리되고, 작은 경우에는 하나의 N-gram으로 취한다. 따라서 문서에서 철자 오류가 있더라도 문장은 유사한 것으로 검색될 가능성이 높다. 그러나 이 방식은 문장의 양이 많아질수록 많은 수의 N-gram이 생성되기 때문에 많은 저장 공간이 필요하며, 전혀 다른 문장임에도 불구하고 생성된 N-gram에 의해서 일치하게 되는 부적합 현상이 생길 수 있다.

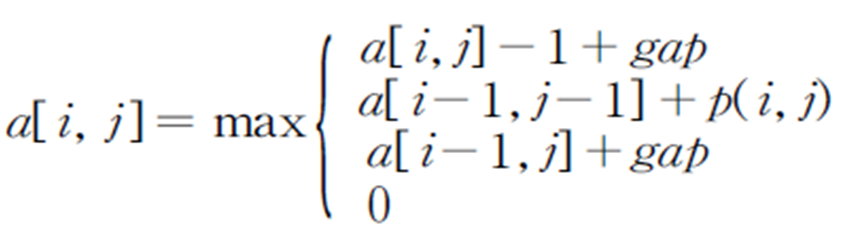
문자열 비교 방식은 문자열 비교를 이용한 연구는 기존에 영어권에서 만들어진 시스템에서 널리 사용되고 있는 방법으로서, 비교하고자 하는 두 문장과 문장 간에 정확한 문자열 비교가 아닌 문장의 변형을 고려하여 두 문장 간에 차이를 허용하는 문자열 비교를 통해서 문장과 문장 간을 비교하고 두 문장 사이에서 일치하는 결과가 문장 내에 포함되는 정도를 가지고 문장의 유사여부를 판단한다. 문자열 비교는 기존의 많은 시스템에서 사용되는 것에서 알 수 있듯이, 어순의 변경이나 단어의 삽입이 적게 이루어진 경우에서는 높은 신뢰도를 가지지만, 영어에 비하여 어순이 자유로운 한국어에 대해서는 그 어순 변경을 통한 변형의 경우 견고하지 못한 면이 있다. 또한, 문장 검사할 때 최소 매칭 길이의 결정에 있어서, 최소 매칭 길이를 작게 설정해 주면 문장의 변형이 많이 이루어진 경우에도 문장의 유사여부를 판단할 수 있으나, 그 판단의 확실성은 떨어지게 된다. 반면에 최소 매칭 길이를 크게 설정해 주었어도 문장의 두 문장의 유사여부 판정의 확실성은 보장할 수 있으나 , 문장의 변형에 견고하지 못한 단점이 있다. 또한 효율성 면에서 문장이 n개가 존재하는 경우 약 n\*(n-1)번 문자열 비교를 수행해야 하므로 문서 집합이 증가하는 경우 시간이 오래 걸리는 단점을 가지고 있다.

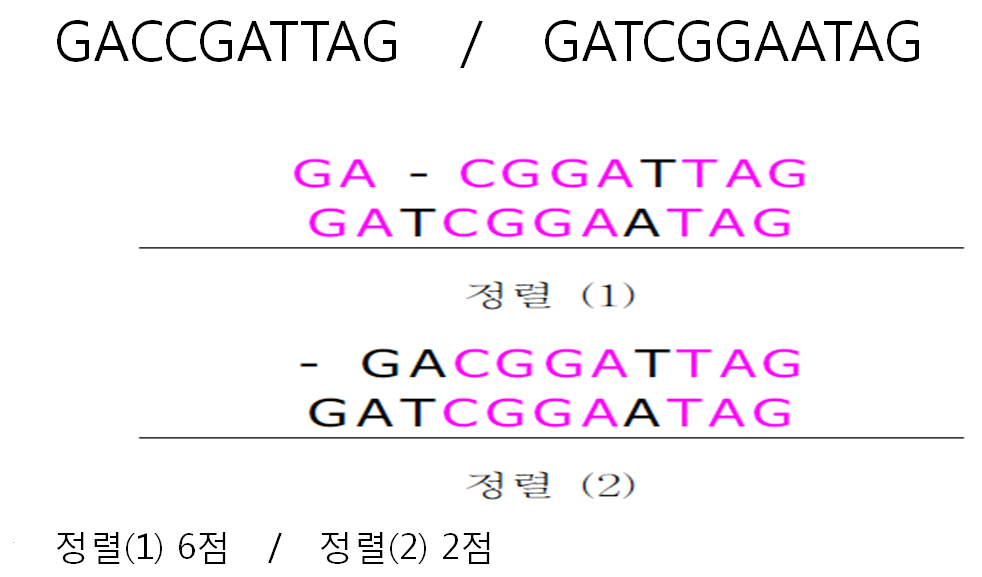
벡터 공간 모델 방식은 정보검색 모델의 한 종류인 벡터 공간 모델을 응용하여 문장의 유사도 값을 계산하여 유사 여부를 판단하는 방법이다. 각 문장을 이루는 색인어를 추출하여 벡터 공간 상의 벡터로 표현하여 문장과 문장 사이의 유사도를 계산한다. 만약 문장 내에 색인어가 포함되어 있으면, 벡터 내의 해당차원은 0이 아닌 가중치의 값을 가지게 된다. 가중치 값은 많은 방법이 있으나 주로 가장 잘 알려진 TF-IDF 가중치 방법이 주로 많이 사용된다. 가중치로 인하여 표현된 문서를 N차원의 벡터 값으로 나타낸 후, 문장과 문장 간의 유사도를 계산한다. 유사도 계산은 다이스 유사계수, 쟈카드 유사계수, 내적 계수, 코사인 유사계수 등이 이용되는데 주로 코사인 유사 계수가 가장 많이 사용된다. 벡터 공간 모델은 계산이 비교적 간단하며 정규화 된 유사도 값을 얻을 수 있다는 장점이 있지만 색인어로 추출되는 키워드가 정확히 일치되어야 하며, 비슷한 의미를 지니고 있더라도 그것을 유추해 낼 수 없다는 단점이 있다.

#### 염기서열 분석기법을 이용한 알고리즘

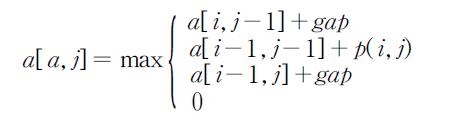
생물정보 학에서 많은 연구가 되어 있는 분야 중 하나가 서열들을 비교하여 유사한 영역을 찾아내는 서열 분석이다. 서열 분석 연구는 유전자 서열이나 단백질 서열들을 비교해서 유사한 영역을 찾아내는 것인데, 이것은 동일한 서열을 가지는 유전자는 같은 기능을 가진다는 가정 하에 동종이나 타종의 유전체 서열의 기능 분석에 사용된다. 유전체 서열의 정렬에 있어 정렬이란 서열에 공백을 넣어 길이가 같은 서열로 만드는 것을 말한다.

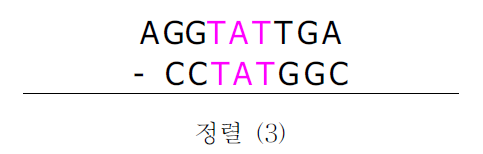
global alignment는 전체 서열을 비교하는 전체 정렬로써 서열의 비교 과정인 점수를 저장하는 행렬을 사용한다. 일치 +1, 불일치 -1, 공백 -2로 점수화 하여 판단 한다.





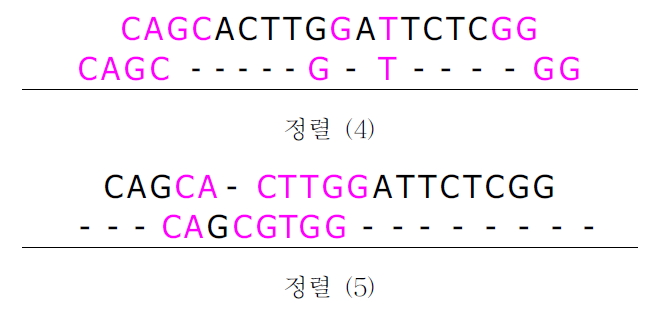
local alignment는 부분열(substring)의 일치에 초점을 두는 지역정렬이다.





공백을 사용해서 강제 정렬하지 않으며 서열의 부분 열이 가장 길게 일치하도록 하는데 초점을 두고 점수화 한다.

semi-global alignment는 길이의 차이가 나는 서열의 비교에 많이 사용되는 정렬로 전체 정렬 방법은 공백을 넣어 두 서열을 정렬시키기 때문에 비교하는 서열의 길이가 다르면 결과가 좋지 않다. 전체 정렬과 같이 일치는 되지만 서열의 앞 뒤의 붙어있는 공백을 무시하면서 정렬한다.



정렬(4)의 경우 중간의 너무 많은 공백으로 점수가 좋지 않다.

### 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.

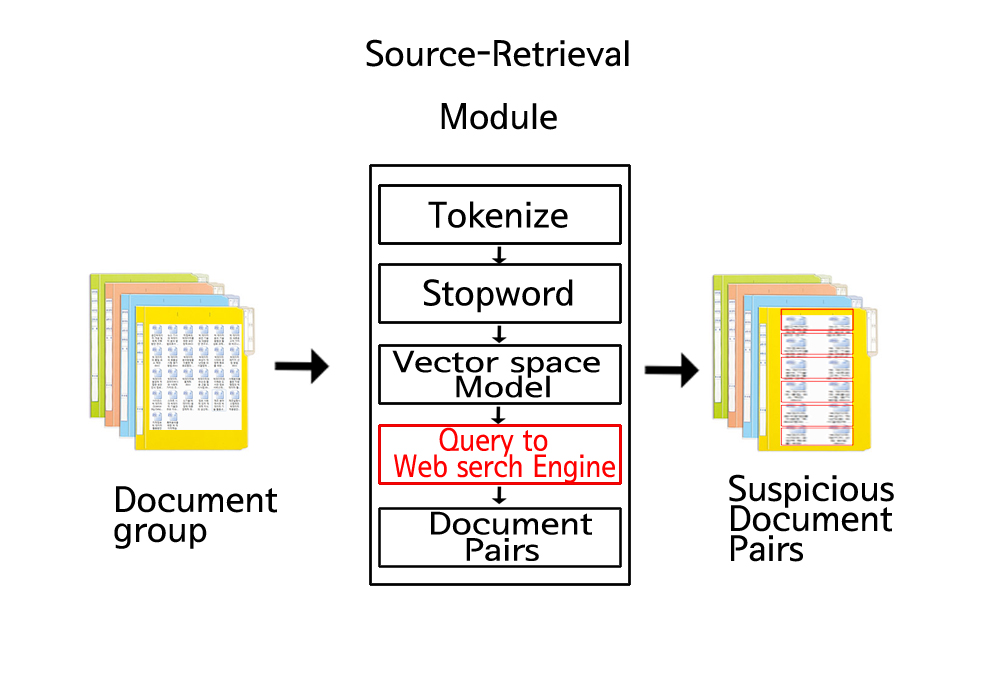


그림 2.3.2-1 Source-Retrival 개념도

* Tokenize(토큰화) : 문서의 내용을 리스트형태로 토큰화 시킨다.
* 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.

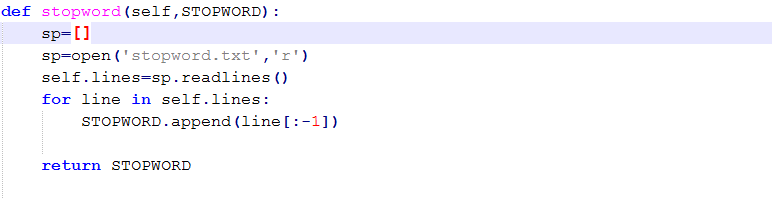


그림1. 불용어 처리 함수

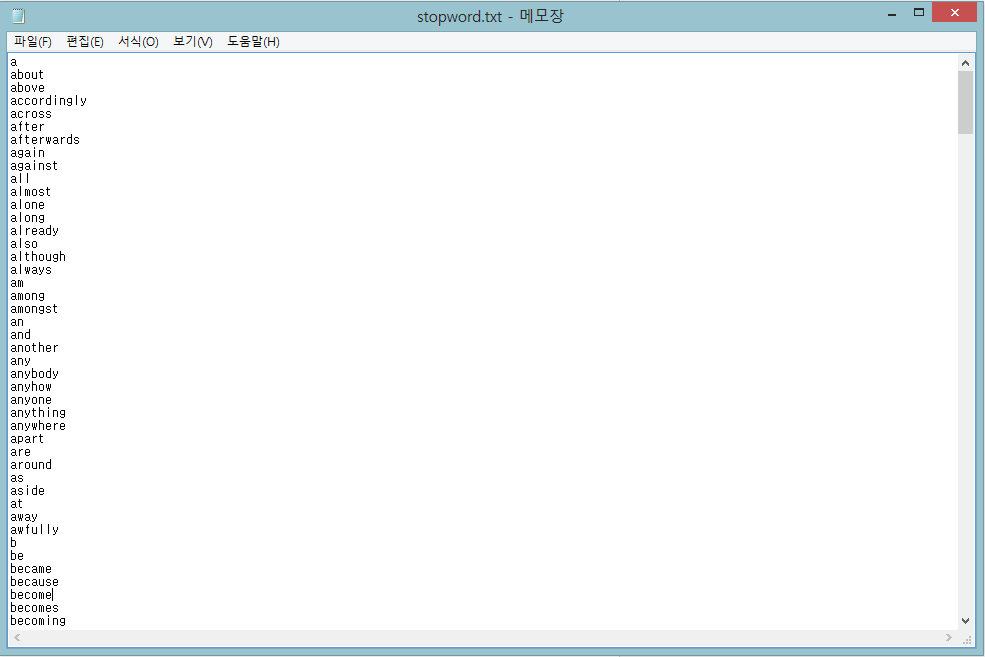


그림2. 불용어 목록

* 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.

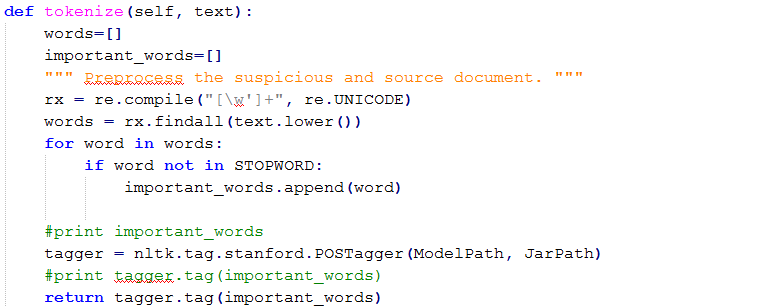


그림1. 토큰화 함수



그림2. 토큰화된 문서의 리스트

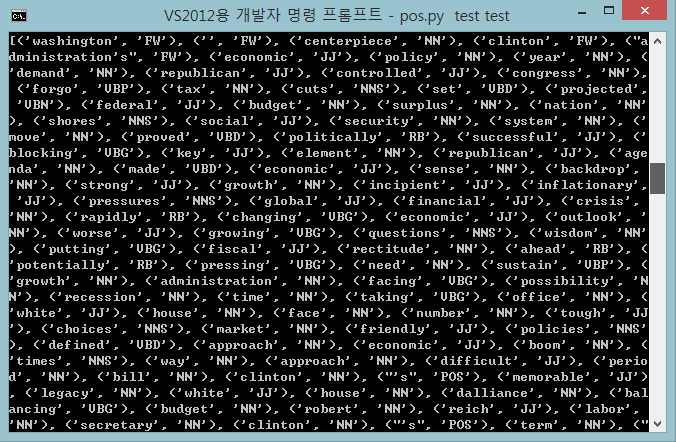


그림3. 각각의 토큰에 달린 품사태그

* 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.

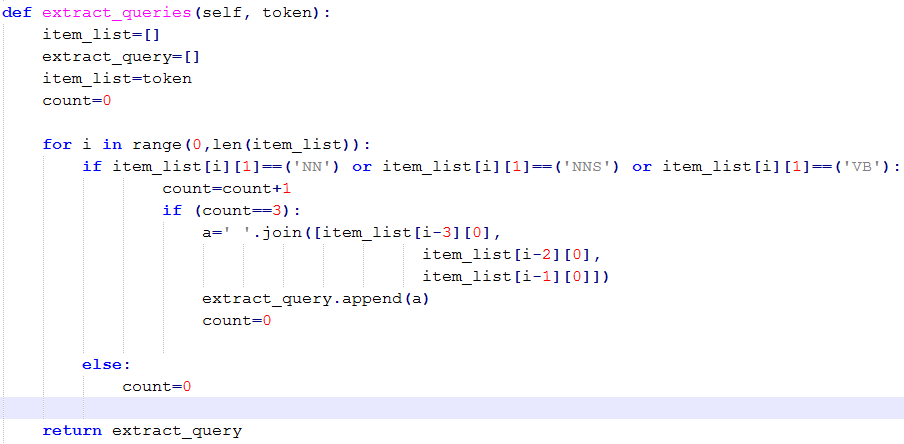


그림1. 쿼리 추출 함수

* 로그 저장 함수

쿼리를 보낸 시간, 쿼리, 검색api에서 검색되는 url을 기록하는 로그파일

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

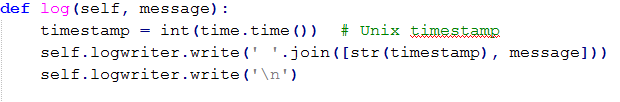


그림1. 보낸 쿼리를 로그로 저장하는 함수

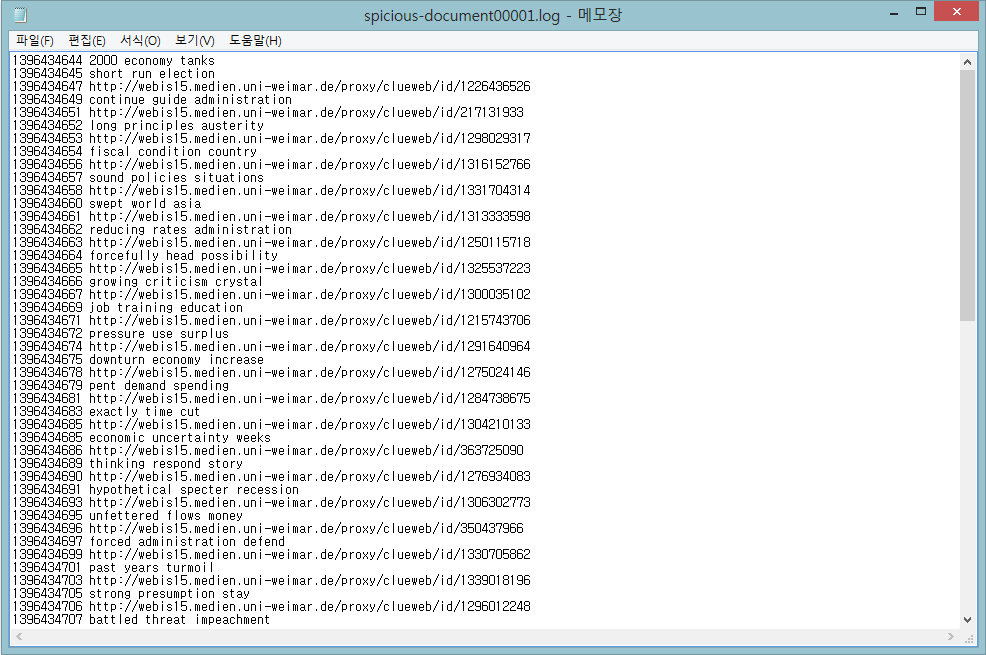


그림2. 로그파일

* 결과 함수

쿼리를 날린 후 검색 API에서 보내주는 결과.

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

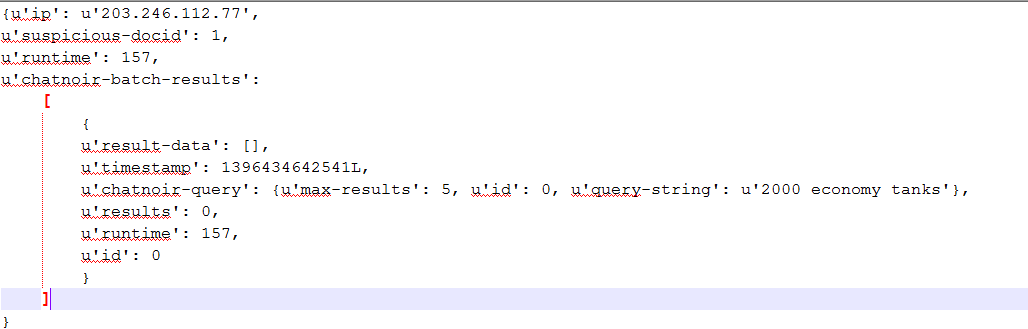


그림1. 전송한 쿼리에서 결과가 없을 경우



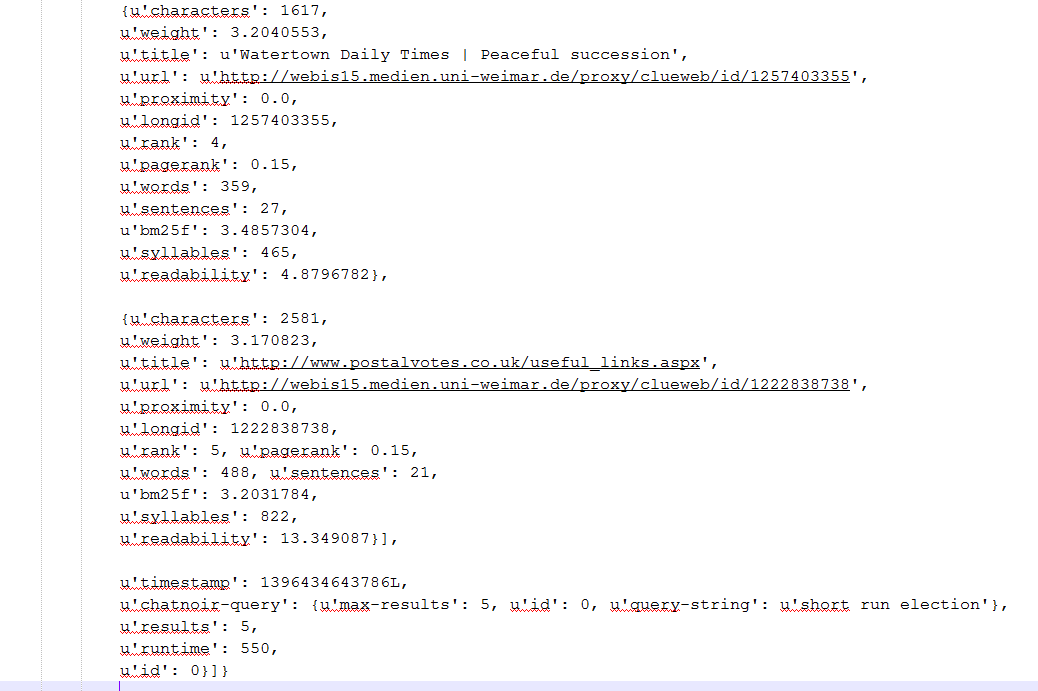


그림2. 전송한 쿼리에 결과가 있는 경우

* URL 다운 함수

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

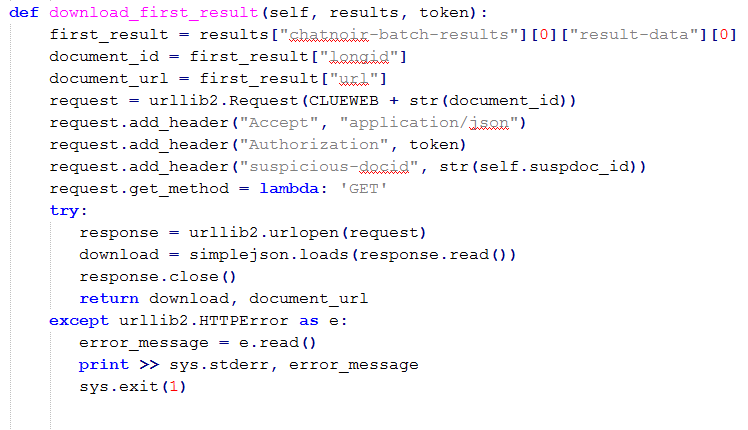


그림1. URL 다운

* URL 비교 함수

다운로드 받은 URL이 표절한 문서인지 서버API의 ORACLE을 이용하여 판별하는 함수.

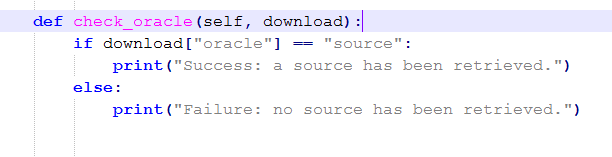


그림1. URL 비교 함수

### Text-Alignment Module 개발

#### Text Alignment 개요

Source-Retrieval를 통하여 표절이 의심되는 문서가 웹 상의 문서를 표절했음을 알 수 있다. Text Alignment 과정에서는 웹 상의 문서 중 표절 의심 문서와의 유사도가 가장 높은 문서를 다운로드하여 표절이 의심되는 문서와 정밀한 검사를 진행을 한다. 표절이 의심되는 문서를 의심 문서, 의심 문서가 표절한 것으로 생각되는 문서를 소스 문서라고 칭하겠다.

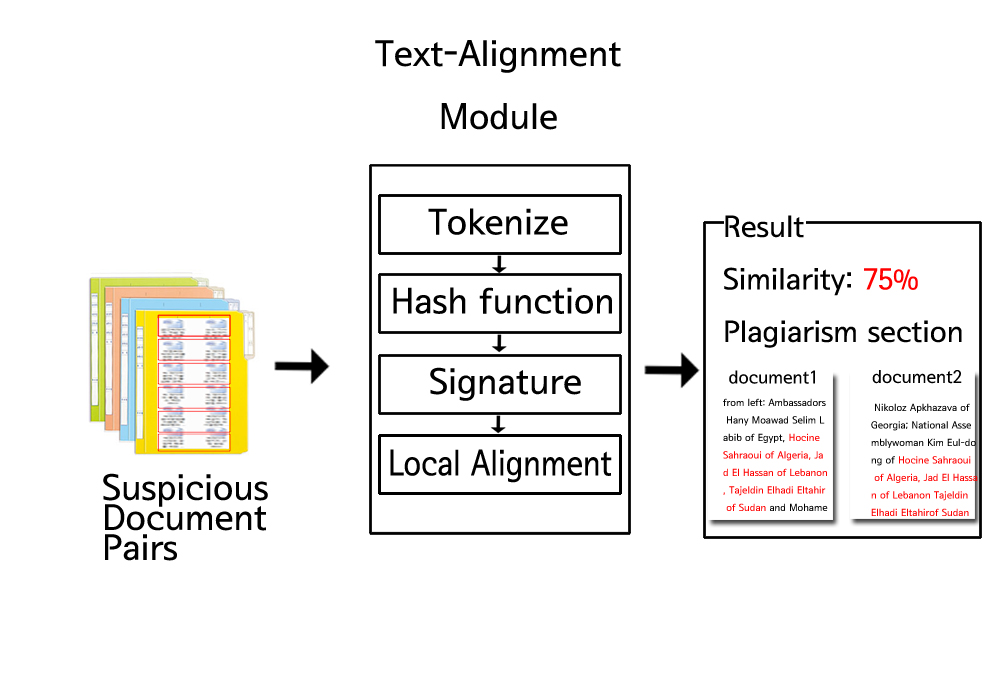
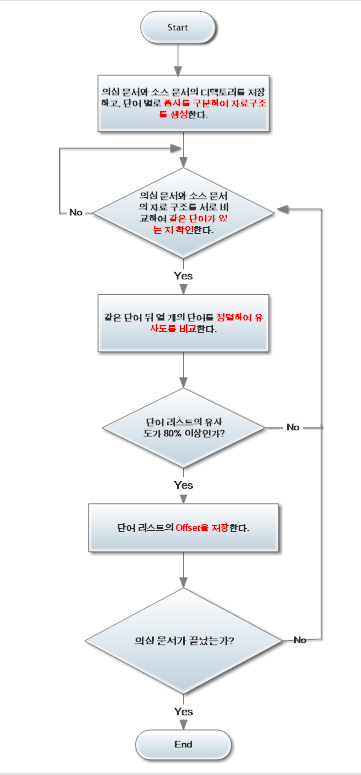


그림 1. Text-Alignment 1차 개념도

#### 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.

그림2. Text Alignment 흐름도

#### 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

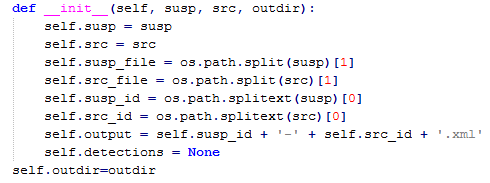


그림3. Text Alignment 디렉토리 저장 함수

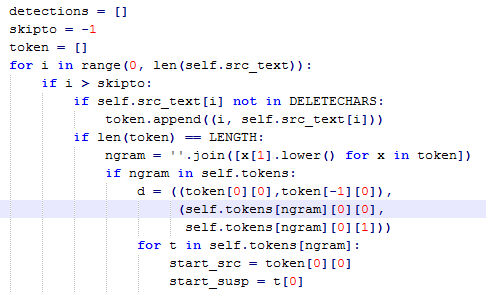
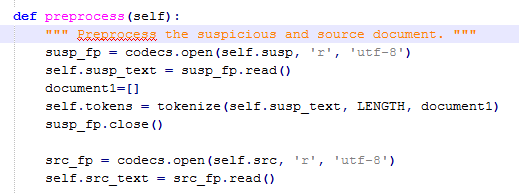


그림4. 두 문서에서 동일 단어를 추출하는 함수



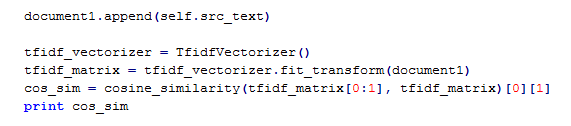


그림5. Text Alignment 문서 유사도 추출 함수

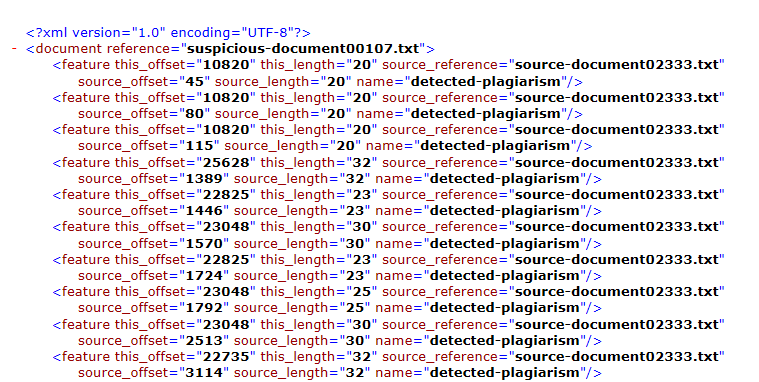


그림6. Text Alignment 표절 구간 Offset 출력 XML

# 수정된 연구내용 및 추진 방향

## 수정사항

#### Text Alignment 수정 사항

1. 알고리즘의 수정

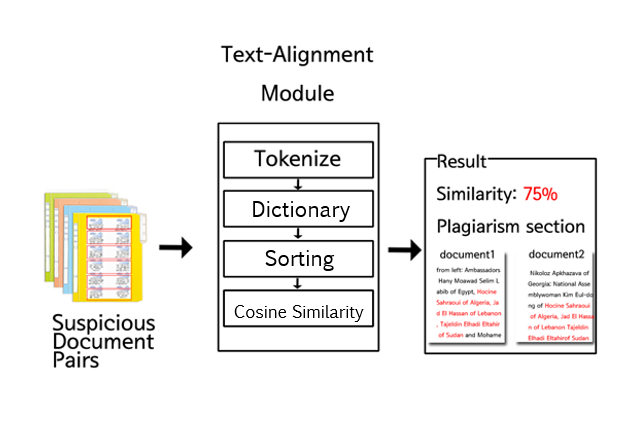
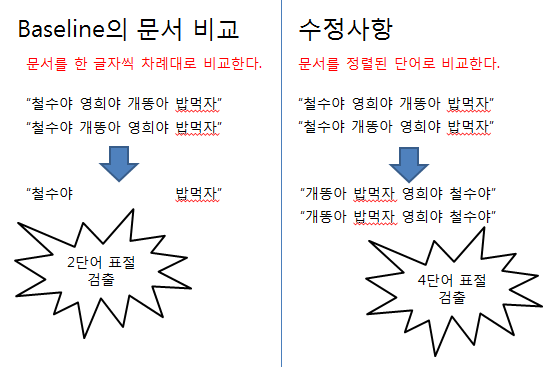


그림7. Text Alignment 2차 개념도

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.

1. 구체적 알고리즘 수정 사항



웹 검색을 위한 Web Search API의 추가

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

# 향후 추진계획

## 향후 계획의 세부 내용

# 애로 및 건의사항