

DU-도전학기 결과보고서

과제명	효율 및 정확도 향상을 위한 표절 검사 프로그램 개발		
참여자	성명	소속	학번
	박OO	컴퓨터공학전공	
	안OO	컴퓨터소프트웨어전공	
	강OO	컴퓨터공학전공	
지도교수 의견	상기 학생들은 주 2-3회 이상 함께 모여 도전학기를 성실히 수행한 결과, 텍스트 마이닝 기반의 고도화된 표절 탐지 기술을 개발하였습니다. 학생들이 개발한 기술은 대표적인 표절 검사 프로그램인 카피킬러가 탐지하지 못하는 고도화된 표절 행위를 탐지할 수 있는 고유 기술로서 해당 기술을 논문으로 작성하여 지난 5월 개최된 대한임베디드공학회 춘계학술대회에서 발표하였습니다. 학생들이 계획한 도전학기 목표를 모두 성공적으로 달성하였다고 생각하고, 도전학기 활동을 통해 인공지능, 파이썬프로그래밍 등 전공 교과목을 심화학습하는 유익한 시간이 되었길 바랍니다. (소속) 컴퓨터공학전공 (성명) 김지연 (서명 또는 날인) 		

1. 도전 과제 내용

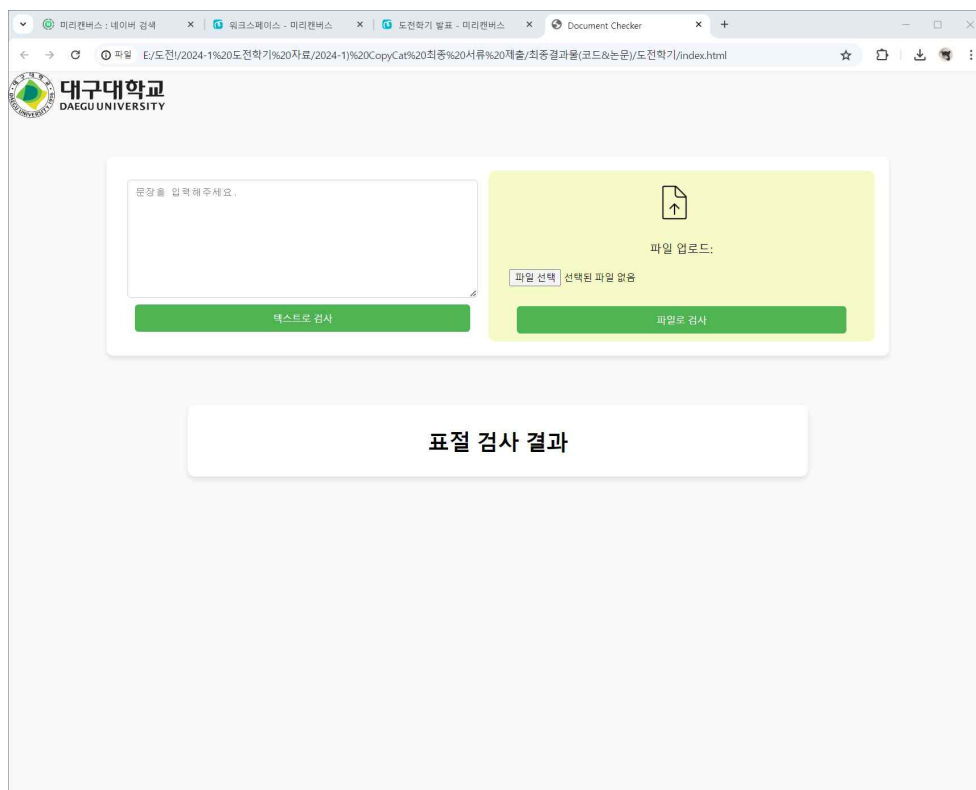
정보와 지식의 공유가 쉽게 이루어지고 디지털 기술의 발달로 정보에 대한 접근성이 증가함에 따라 지식 재산권을 침해하는 표절 행위가 더욱 심각해지고 있는 추세이다. 이러한 이유로 학문적 부정행위에 대한 경각심이 높아지고 있으며, 이러한 표절 행위로부터 지적 재산권을 보호하기 위한 고도화된 표절 검사 시스템이 필요하다. 본 팀의 주제는 효율 및 정확도 향상을 위한 표절 검사 프로그램 개발로 기존 제품보다 우수한 성능을 보이는 것에 대한 부분만을 강조하는 기술이라고 생각하였다. 본 팀은 이러한 기술들에서 학술적 가치를 탐구해보기 위해 텍스트마이닝을 중점으론 표절 검사 시스템을 제시하게 되었고, 다양한 선행 연구를 찾아보았을 때, WMD(Word Mover's Distance) 특성에 맞게 변형 표절을 탐지하는 내용은 연구가 미비하다는 것을 확인하였다. 따라서 본팀은 '텍스트마이닝 기반의 고도화된 표절 검사 시스템'을 제시한다는 점에서 학술적 가치가 있다고 판단되었기에 이러한 내용을 기반해 의의를 두고 도전학기 프로젝트를 진행했다.

본 팀은 초기 도전학기 계획서 내용과 같게 데이터셋 구축을 위해 크롤러를 개발하여 국내 학술 DB인 'DBpia'에서 논문을 수집하여 데이터셋을 구축하였으며, 이를 기반으로 표절 검사 프로그램 개발을 진행할 수 있도록 하였다. 또한, 본 팀에서는 의도적으로 표절 검사 규칙을 우회하기 위해 문장 내 단어를 대체 단어로 변경하고, 문장 내의 단어 순서를 변경하는 우회 행위를 탐지하여 표절 여부를 검사하기 위해 WMD(Word Mover's Distance)를 사용하고, 뿐만 아니라 SHA-256 해시 알고리즘, 교집합 비율을 통해 문장과 문장간의 비교를 수행하였고, Doc2Vec 계산을 통해 문장-문장 비교를 수행하였다. 또한, 웹사이트를 제작하여 본 팀에서 개발한 표절 검사 시스템을 활용할 수 있도록 진행하였다.

2. 도전 과제 수행 결과 및 성과

- 팀 공통 과제 수행 결과

본 팀은 도전학기 초기 계획대로 크롤러 개발을 통해 논문 데이터셋 구축을 진행하였고 목표했던 표절 검사 시스템 개발을 성공적으로 수행하였다. 표절 검사 시스템에서는 문장의 비교를 빠르게 수행하기 위한 해시 비교 알고리즘을 SHA-256으로 선정하여 진행하였고, WMD(Word Mover's Distance)를 통해 문장 내의 변형하여 표절한 부분도 탐지할 수 있도록 개발을 진행했다. 다음으로는 문서간의 비교를 위해 Doc2Vec을 활용하여 시스템 개발을 진행했고, 개발한 시스템의 코드를 통합하여 시각화 및 시스템 제공을 위한 웹사이트를 제작하였으며 문서 업로드 기능과 검사 기능 및 표절률을 나타내는 기능까지 성공적으로 구현했다.



3. 자기 평가

- 팀원 개별 과제 수행 결과

- 박OO : 도전 학기 초기에 계획했던대로 표절 검사 프로그램에서 비교 분석을 수행하기 위해 바른API를 사용하고 불용어 처리를 통해 단어간의 밴다이어그램을 제작하였다. 추가로 교집합 비율을 계산하여 특정값 이상일 경우 표절로 판단하는 알고리즘 및 해시값 비교를 위한 SHA-256 해시 알고리즘을 추가하였다. 또한 '카피킬러'와의 성능 비교를 진행하였으며 팀원과 함께 문서-문서 비교를 위한 Doc2Vec을 추가로 사용하여 개발을 성공적으로 진행했다.
- 안OO : 표절 검사 프로그램에서 비교 분석에서 관련 기술을 조사하고 데이터셋을 구축하기 위해 C# WinForm 애플리케이션 기반의 크롤러를 개발하고, 학술 DB에서 논문 데이터를

- 강OO : 표절 검사 프로그램 개발을 위해 Word2Vec 모델 알고리즘에 대한 조사 및 공부하였으며 조사를 진행하며 예시 코드 작성에서 문제를 발견하고 기존 계획인 중심성 계수에서 WMD로 변경하여 프로젝트를 진행하였다. 표절 탐지 시스템 성능 확인을 위해 단순히 문장을 복사한 ‘완전표절’과 문장의 단어를 변형시켜 표절한 ‘변형표절’과 같은 시나리오를 구성하여 팀원과 함께 성능 검증을 진행하였다. 나아가 자신의 관심 분야를 살려 개발한 시스템의 최적화 및 표절률 시각화를 위한 웹 사이트 제작을 위한 가상환경 설정과 사용자 인터페이스 구축을 성공적으로 진행했다.

4. 최종 결과물

- 박OO : 표절 검사 시스템 개발 코드를 작성하며 시각화 진행, 팀원과 함께 시나리오 기반 카피 킬러와의 성능 비교 진행

| 개인 역할_박은영

import hashlib

해시 값 비교

```
def get_hash(text):
    sha256 = hashlib.sha256(text)
    sha256.update(text.encode('utf-8'))
    return sha256.hexdigest()
```

교집합 비율 계산 → 벤다이어그램 시각화

```
def get_intersection_ratio(path1, path2):
    file_paths = os.listdir(path1)
    for file_path in file_paths:
        file1 = os.path.join(path1, file_path)
        count1 = FileCounter(file1)
        count2 = FileCounter(path2)
        intersection = count1.intersection(count2)

        # 교집합 비율, 두 폴더 각각의 파일 수를 곱한 값에 나눠서 계산
        ratio = len(intersection) / (count1 * count2)

        # 두 폴더의 교집합, 차집합, 합집합
        venn2(subsets=(set(count1) & intersection), set(count2) & intersection, len(intersection))
        set_label1 = os.path.basename(path1).split('.')[0]
        set_label2 = os.path.basename(path2).split('.')[0]

        plt.title(f'Venn Diagram of {set_label1} and {set_label2}')

        # 그림에 표시
        intersection_text = f'Intersection: {len(intersection)}'
        plt.text(0.5, 0.5, intersection_text, transform=plt.gca().transfrom_axes,
              verticalalignment='center', fontweight='bold')
        plt.show()
```

Document Similarity Matrix

	0	0.06	0.25	0.03	0.16	0.34	0.2	0.28	0.21	0.38
문헌1_정책_해석_해	0.06	0	0.45	0.60	0.53	0.5	0.38	0.4	0.37	0.51
정책_해석_해	0.25	0.45	0	0.33	0.43	0.27	0.27	0.26	0.32	0.36
자료_및_1000원	0.03	0.60	0.33	0	0.47	0.35	0.48	0.45	0.4	0.32
문헌2_정책_해	0.16	0.53	0.43	0.47	0	0.38	0.48	0.45	0.45	0.32
문헌3_정책_해	0.34	0.5	0.27	0.35	0.38	0	0.35	0.29	0.31	0.25
문헌4_정책_해	0.2	0.38	0.27	0.48	0.48	0.35	0	0.33	0.26	0.47
문헌5_정책_해	0.28	0.4	0.26	0.45	0.45	0.29	0.33	0	0.3	0.26
문헌6_정책_해	0.21	0.37	0.32	0.4	0.45	0.31	0.26	0.3	0	0.23
자료_및_1000원	0.38	0.51	0.36	0.32	0.32	0.25	0.47	0.38	0.23	0

교집합 비율 계산 → 벤다이어그램 시각화

Venn Diagram of A and E

Venn Diagram of B and E

Venn Diagram of C and E

Venn Diagram of D and E

```
def calculate_similarity(doc1_id, doc2_id):
    doc1_content = doc1_id['content']
    doc2_content = doc2_id['content']

    # 토큰화
    doc1_tokens = doc1_content.split()
    doc2_tokens = doc2_content.split()

    # 모델 생성
    model = DocVecVectorizer(window=5, min_count=1, workers=1, epochs=100)
    model.fit(doc1_tokens + doc2_tokens)
    model.train(doc1_tokens, total_examples=model.corpus_count, epochs=model.epochs)

    # 유사도 계산
    doc1_vector = model.docvecs.get(doc1_id)
    doc2_vector = model.docvecs.get(doc2_id)

    # 두 문서의 유사도 계산
    similarity = calculate_similarity(doc1_vector, doc2_vector)

    return similarity

# 유사도 계산
for i in range(len(documents)):
    for j in range(i+1, len(documents)):
        doc1_id = documents[i]
        doc2_id = documents[j]
        similarity = calculate_similarity(doc1_id, doc2_id)
        print(f'Document {doc1_id["id"]} and {doc2_id["id"]} similarity: {similarity:.4f}')

# 유사도 행렬 생성
similarity_matrix = np.zeros((len(documents), len(documents)))
for i in range(len(documents)):
    for j in range(i+1, len(documents)):
        similarity_matrix[i][j] = calculate_similarity(documents[i], documents[j])
```

시나리오

	S	S	S	S	S
A	□	■	□	□	□
B	■	□	□	□	□
C	■	■	■	■	■
D	■	■	■	■	■

■=단어변경, □=단어순서변경, ▣=문장변경 X

- The figure is a composite image illustrating the WMD (Word Movement Distance) algorithm and its application. It is divided into three main sections:

 - Top Section: WMD 비교 코드 작성 (WMD Comparison Code Writing)**

This section shows a code editor with JavaScript code for calculating WMD. The code defines a function `wmd(sent1, sent2)` that calculates the minimum cost of moving words between two sentences. It uses a dynamic programming approach to find the optimal alignment between the words of the two sentences. The code also includes a function `ranking` that ranks sentences based on their WMD scores relative to a query sentence.
 - Middle Section: 크롤러 개발 및 데이터 수집 (Crawler Development and Data Collection)**

This section shows a diagram of the WMD algorithm. It illustrates the process of finding the minimum cost of moving words between two sentences. The diagram shows two sentences, `sent1` and `sent2`, and a table of costs for moving words between them. The minimum cost is found by comparing the costs of moving words from the beginning of one sentence to the beginning of the other, and the costs of moving words from the beginning of one sentence to the end of the other.
 - Bottom Section: WMD 비교 코드 작성 (WMD Comparison Code Writing)**

This section shows a screenshot of a web application interface. It displays a search bar and a list of search results. A red box highlights the "WMD 비교 코드 작성" (Write WMD Comparison Code) button, which is used to initiate the WMD calculation process.

- ## | 개인 역할_강동원

웹 사이트 프론트엔드 개발

WMD 학습 코드 작성

표절 탐지 알고리즘 제작

표절 검사 결과

```
def token_sentences(s):
    result = []
    for i, sentence in enumerate(s):
        tokenized = my_tokenizer.tokenize(sentence)
        print(s[i])
        nouns = tokenized.nouns()
        print(i + 1, nouns)
        result.append(nouns)
    result = list(filter(None, result))
    return result

[ ] model = Word2Vec(tokens)

[ ] sen1 = "최근 대중문물"
sen2 = "최근 문화 직:"

[ ] model.wv.similarity('
0.0

[ ] model.wv.similarity("최근 대중문물" and "최근 문화 직:" and "최근 문화 직:" and "최근 문화 직:")
0.0

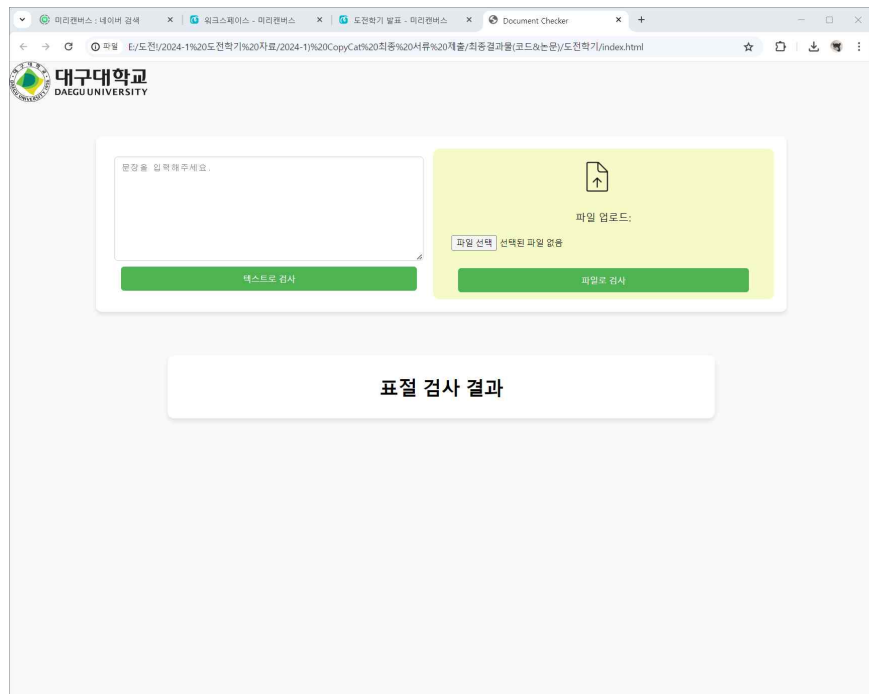
[ ] model.wv.similarity("최근 대중문물" and "최근 문화 직:" and "최근 문화 직:" and "최근 문화 직:")
0.0

[ ] model.wv.similarity("최근 대중문물" and "최근 문화 직:" and "최근 문화 직:" and "최근 문화 직:")
0.0
```

카피킬러와의 비교 진행

시나리오	카피킬러 표절률	결과	
		해시	WMD
A	98%	완전 표절	변형 표절
B	92%	완전 표절	변형 표절
C	32%	-	변형 표절
D	0%	-	변형 표절

- 팀 공통 결과물 : 표절 검사를 시행할 수 있도록 문서를 업로드하여 표절률을 나타내주는 웹 사이트(텍스트마이닝 기반), 학술대회 논문



```
challenge.py < X
E> 도전! > 2024-1 도전학기 자료 > 2024-1 CopyCat 최종 서류 제출 > 최종문과물(크드노는원) > 도전학기 > challenge.py > @ get
1 import sys
2 import argparse as brs
3 from hashing import Tokenizer
4 import re
5 import os
6 from gensim.models import Word2Vec
7 import pandas as pd
8 from sklearn.metrics.pairwise import cosine_similarity
9 import hashlib
10
11 def get_hash(text):
12     sha256 = hashlib.sha256()
13     sha256.update(text.encode('utf-8'))
14     return sha256.hexdigest()
15
16 def token_sentences(s):
17     result = []
18     for i, sentence in enumerate(s):
19         tokens = my_tokenizer.tokenize(sentence)
20         words = tokens[2:-1]
21         result.append(words)
22     result = list(filter(None, result))
23     return result
24
25 def split_sentences(text):
26     # 공백 구분을 기준으로 텍스트를 분리
27     sentences = re.split('([.?!-])', text)
28     print(sentences)
29     return sentences
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56 sentences = all_files_text
57
58 token_sentences = token_sentences(sentences)
59
60 model = Word2Vec(token_sentences, vector_size=100, window=1, min_count=1, workers=4)
61
62 from flask import Flask, request
63 from flask_cors import CORS
64 app = Flask(__name__)
65 CORS(app) # 모든 엔드포인트에 CORS를 적용
66
67 # POST 요청을 처리하는 엔드포인트
68 # POST 요청을 처리하는 엔드포인트
69 @app.route('/')
70 def index():
71     return '서버가 정상적으로 실행 중입니다!'
72
73 @app.route('/compare', methods=['POST'])
74 def compare_sentences():
75     data = request.json
76     if data is None or 'POST_INPUT' not in data:
77         return 'POST_INPUT을 전달해주세요.', 400
78     POST_INPUT = data['POST_INPUT']
```

```
challenge.py 4 # style.css X
E> 도전! > 2024-1 도전학기 자료 > 2024-1 CopyCat 최종 서류 제출 >
1 body {
2     font-family: Arial, sans-serif;
3     margin: 0;
4     padding: 0;
5     background-color: #f5f5f5;
6 }
7
8 .container {
9     width: 100%;
10    max-width: 1200px;
11    margin: 0 auto;
12    padding: 20px;
13 }
14
15 .header {
16     display: flex;
17     align-items: center;
18     padding: 20px;
19     background-color: #e8f5e9;
20 }
21
22 .header img {
23     height: 50px;
24     margin-right: 20px;
25 }
26
27 .header h1 {
28     margin: 0;
29     font-size: 24px;
30 }
31
32 .content {
33     display: flex;
34     justify-content: space-between;
35     background-color: #ffffff;
36     padding: 20px;
37     border-radius: 10px;
38     box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
39 }
40
41 .upload-section,
42 .result-section,
43 .percentage-section {
44     width: 30%;
45     text-align: center;
46     background-color: #f0f4c3;
47     padding: 20px;
48     border-radius: 10px;
49 }
50
51 .upload-section img {
52     width: 50px;
53     height: 50px;
54 }
55
56 .upload-section p,
57 .percentage-section p {
58     font-size: 18px;
59     margin: 10px 0;
60 }
61
62 .result-section h2 {
63     margin-bottom: 20px;
64     font-size: 20px;
65 }
66
67 .result-section p {
68     font-size: 16px;
69     margin: 5px 0;
70 }
71
72 .percentage-section h2 {
73     font-size: 20px;
74     margin-bottom: 10px;
75 }
76
77 .percentage-section p {
78     font-size: 36px;
79     color: red;
80     margin: 10px 0;
81 }
82
83 .upload-section {
84     width: 47.5%;
85     text-align: center;
86     background-color: #f0f4c3;
87     padding: 20px;
88     border-radius: 10px;
89     height: 200px; /* 고정된 높이 추가 */
90 }
91
92 .text-input-section form,
93 .upload-section form {
94     display: flex;
95     align-items: center;
96     justify-content: center;
97     flex-direction: column;
98     height: 100%;
99 }
100
101 .text-input-section textarea,
102 .upload-section input[type='file'] {
103     width: calc(100% - 20px); /* 20px는 padding 값 */
104     margin-bottom: 10px;
105 }
106
107 .text-input-section input[type='submit'] {
108     width: calc(100% - 40px); /* 40px는 padding 값 */
109 }
110
111 .upload-section input[type='submit'] {
112     width: calc(100% - 40px); /* 40px는 padding 값 */
113     padding: 10px 20px;
114     background-color: #4caf50;
115     color: white;
116     border: none;
117     border-radius: 5px;
118     cursor: pointer;
119     margin-bottom: 20px; /* margin-top 대신 margin-bot
120 }
121
122 body {
123     font-family: Arial, sans-serif;
124     background-color: #f2f2f2;
125     margin: 0;
126     padding: 0;
127 }
```

텍스트마이닝 기반의 고도화된 표절 검사 시스템 개발

Development of an Advanced Plagiarism Inspection System based on Text Mining

†대구대학교 컴퓨터정보공학부

(Eun-Young Park, Dong-Hwi An, Dong-Won Kang, Jiyeon Kim)

(†Division of Computer and Information Engineering, Daegu University)

Abstract : 정보와 지식의 공유가 쉽게 이루어지고 디지털 기술의 발달로 정보에 대한 접근성이 증가함에 따라 지식 재산권을 침해하는 표절 행위가 더욱 심각해지고 있다. 학교 내 과제 및 논문 작성 시 원문을 변경하여 자신이 작성한 것처럼 활용하는 행위가 증가하면서 이러한 학문적 부정행위에 대한 경각심이 높아지고 있으며, 이러한 표절 행위로부터 지식 재산권을 보호하기 위한 고도화된 표절 검사 시스템 개발이 필요하다. 본 논문에서는 해시(Hash) 및 WMD(Word Mover's Distance) 기반으로 표절을 탐지하는 2단계 표절 검사 시스템을 개발하고, 대표적인 표절 검사 프로그램인 '카피킬러'와의 성능 비교를 통해 개발된 시스템의 성능을 검증한다. 의도적으로 표절 검사 시스템을 우회하는 4개 유형의 시나리오를 개발하여 표절 검사를 수행한 결과, 본 논문에서 개발한 표절 검사 시스템이 모든 시나리오를 정확하게 표절로 판단하는 것을 확인하였고, 2개 시나리오만 명확하게 표절로 판단한 '카피킬러'보다 우수한 성능을 가지는 것을 실험을 통해 검증하였다.

Keywords : Plagiarism, Text Mining, Hash, WMD(Word Mover's Distance)

I. 서론

표절은 타인의 저작물로부터 출처를 밝히지 않고 인용하거나 모방하여 자신의 저작물인 것처럼 사용하는 행위를 뜻하며, 타인의 저작물 가치를 훼손함으로써 심각한 학문적, 사회적 문제를 야기한다. 2022년 대학 내 과제물 중, 약 46%가 과제물 30% 이상 표절된 '표절 위험' 군으로 조사되었다[1]. 이러한 학문적 부정행위를 방지하기 위해 다양한 표절 검사 프로그램이 등장하였지만, 표절 프로그램을 우회하는 다양한 표절 시도를 탐지하는 데에는 한계가 있다. 예를 들어, 대표적인 표절 검사 프로그램인 '카피킬러'의 경우, 한 문장에서 6어절 이상 일치하는 경우 표절이라고 판단한다[2]. 그러나 표절 프로그램을 우회하기 위하여 의도적으로 한 문장에서 6어절 미만으로 표절하거나, 한 문장 내의 단어 순서를 바꾸는 경우, 또는 비슷한 단어로 대체하여 작성하는 경우에는 표절로 탐지되지 않는다. 따라서 이러한 표절 우회 행위도 탐지할 수 있는 진화된 표절 탐지 기술 개발이 필요하다.

본 논문에서는 문장 단위의 표절 검사뿐 아니라, 다수의 문장에서 문맥을 파악하고, 단어 임베딩 공간에서 유사도를 계산하여 문장의 거리를 측정하는 표절 검사 기술을 제안한다. 문장 단위의 표절 검사를 위해서는 문장 전체의 해시(Hash)값 비교를 통해 단순히 문장을 표절한 완전 표절을 검출하고, WMD(Word Mover's Distance)를 활용하여 문장의 단어를 변형하거나 순서를 바꾸는 등의 변형 표절을 탐지한다. 또한, 본 논문에서는 제안된 기술의 설계 및 실행을 검증하기 위하여 표절 검사 시스템을 개발하고, 시스템의 성능을 평가하기 위하여 학술 데이터셋을 수집하는 크롤러(Crawler)를 개발한다.

본 논문의 구성은 다음과 같다. 2장에서는 표절 탐지 기술 동향을 살펴보고, 3장에서 본 논문에서 제안하는 표절 검사 시스템 및 학술 데이터 크롤러를 개발한다. 4장에서는 개발된 시스템의 성능을 실험을 통해 분석하고, 5장에서 결론 및 향후 연구를 제시한다.

II. 관련 연구

본 장에서는 기존에 수행된 표절 방지를 위한 관련 연구 동향을 살펴본다. 내부 DB만을 활용하는 기존의 표절 검사 기술을 개선하여 외부 웹 데이터를

*Corresponding Author (jyk@daegu.ac.kr)

김지연: 대구대학교 컴퓨터정보공학부