**친지인 Final Report**

텍스트이(가) 표시된 사진

자동 생성된 설명**(Chin-ji-in, 櫬地人)**

|  |  |
| --- | --- |
| 오픈소스SW프로젝트 1분반 | 9조 |
| 20170454 | 이창민 |
| 20171218 | 임우섭 |
| 20172848 | 정석우 |
| 20170364 | 한태균 |
| 20172864 | 서정현 |

# 목차

1. 서론
   1. 프로젝트 소개 및 필요성
      1. 친지인이란?
      2. 천지인 키보드 특징
      3. 천지인 오타 특징
   2. 기존 기술 조사 및 차별성
      1. 자동완성 기능
      2. Peter Norvig
      3. Symspell
      4. 한글 오타 교정
2. 본론
   1. 구현
      1. 아키텍처
      2. 알고리즘
   2. 개선
   3. 평가
      1. Demo
      2. 선행 기술과 비교
3. 결론
   1. 협업
   2. 향후 계획

1-1. 프로젝트 소개 및 필요성

## **친지인(櫬地人)**

천지인 키보드에서 오타를 줄이기 위해 고안된 프로젝트이다. 단어 ‘천지인’에서 아래아(ㆍ)가 탈락된 오타로 이름이 지어졌으며, 한자는 무궁화 나무 친 ‘櫬’을 사용하여 한글 창제 원리를 기반으로 한 천지인 키보드가 우리나라 사람들에게 보다 쉽게 사용될 수 있길 바라는 마음을 담았다.

## **천지인 키보드 특징**

가장 흔히 사용되는 두벌식(qwerty) 키보드와 비교해보면 천지인 키보드는 기본적으로 필요한 자판이 문자 자판 10개와 공백 자판으로 총 11개인 반면, 두벌식 키보드는 기본적으로 문자 자판 26개와 공백 자판, Shift 자판이 필요하므로 총 28개의 자판을 가진다. 따라서 천지인 키보드는 자판이 넓지만 타수가 많아지고, 두벌식 키보드는 자판은 좁지만 타수는 적어진다. 이에 천지인은 두벌식 보다 한 손으로 입력하기 용이하다는 장점을 가진다.

많은 타수를 입력해야 하는 천지인 키보드 사용자는 두벌식 키보드 사용자보다 빠른 입력이 요구되기 때문에 넓은 자판에도 불구하고 자판을 잘못 누르는 경우가 종종 발생한다. 특히 자판 간의 순서 바뀜, 자판 하나 안눌림, 자판 하나 더눌림과 같은 오타는 자판의 크기와 크게 관련없이 발생하기 때문에 **천지인에서의 오타가 두벌식에서보다 빈번히 발생**하는 것으로 조사되었다.

## **천지인 오타 특징**

실제로 팀원들이 천지인 키보드를 사용하여 오타 수정없이 노래 총 12곡의 가사들을 입력해 보고 통계를 내보았는데, 총 1592단어에서 260개의 단어에서 오타가 발생했으며, 이는 16%의 비율이다. 오타의 원인으로는 Insert(자판 한개 추가 입력)이 30%로 가장 빈번했으며, Replace(자판 한개 다른 자판 입력)이 22%, Deletion(자판 한개 입력 안함)이 20%, Transpose(자판 한 쌍 순서 바뀜)가 7%를 차지했으며, 이외에 한 단어에서 두가지 이상의 오타가 일어난 경우는 20%였다.

이 데이터셋에서 중요한 것은 오타의 내용이었다. 천지인 키보드의 오타는 두벌식 키보드에서는 발생하기 힘든 오타가 종종 발견되기도 했으며, 두벌식 키보드에서 쉽게 발생할 법한 오타가 천지인 키보드에서는 가능성이 적은 경우도 많았다.

예를 들면, ‘고양이’라는 단어에서 양과 이 사이에 스페이스바를 누르지 않으면 ‘고야미’라는 결과가 나오고, 양에서 아래아(ㆍ) 하나를 ‘ㅣ’보다 먼저 누르면 ‘고어ㆍㅇ이’ 라는 결과가 나오게 되는데 이는 모두 두벌식에서는 나타나기 힘든 오타이다.

또한, 천지인 키보드에는 두 번 이상 눌러야 입력할 수 있는 자음과 모음(ㅋ, ㄲ, ㅁ, ㅏ, ㅠ 등)이 많기 때문에 이와 같은 음소가 실수로 추가되었다고 보는 것은 우선 순위가 낮다. 예를 들어 ‘맴수’라는 입력이 있으면 ‘매수’보다는 ‘맹수’로 교정해주는 것이 옳다는 것이다.

위와 같은 특징들이 존재하기 때문에 **천지인 전용의 오타 교정 알고리즘이 필요**하다.

1-2. 기존 기술 조사 및 차별성

스마트폰에 내장되어있는 자동 완성 기능과, 영어 오타 교정 알고리즘으로 유명한 **Peter-Norvig Algorithm**, **symspell** 알고리즘을 조사하였으며, 기존에 개발되어 있는 한글 오타교정을 찾아보았다.

## **자동 완성 기능**

기존에 알고 있던 단어와 사용자가 자주 입력했던 단어를 기반으로, 현재 입력중인 단어를 완성 또는 대체할 단어들을 추천해준다.

사용자가 입력한 단어를 기억하기 때문에 사전에 없는 단어도 수정해주는 것이 장점이지만, 오타교정 목적으로 사용하기에는 한계가 존재한다. 가장 먼저, 위에서 언급했던 특수한 천지인 오타(‘고야미’, ‘고어ㆍㅇ이’ 등)를 잘 잡아내지 못했다. 그리고 ‘노양이’와 같은 오타를 못 고쳐주는 것으로 보아 이 기능은 오타교정보다는 자동완성을 목적으로 구현되었기에 단어 초반의 오타를 잘 잡아주지 못하는 것으로 보인다. 또한 이 기능은 여러 교정이 가능한 상황에서는 **단어의 빈도수만을 기반으로 우선순위를 두기 때문에 보다 가능성 높은 오타를 추천해주지 않는다.**

위와 같은 한계가 존재하므로 **‘친지인’ 프로젝트는 자동완성기능과 차별성을 지닌다.**

**Peter Norvig 알고리즘**

Peter Norvig 알고리즘은 미국의 컴퓨터 과학자 Peter Norvig에 의해 고안된 가장 대표적인 영어 오타 교정 알고리즘이다.

이 알고리즘은 우선 교정 사전으로 활용할 텍스트 파일을 구축하는데, 텍스트 파일은 단어와 그 단어의 빈도 수를 저장한다. 이 빈도 수는 교정 단어 후보 간의 우선순위를 정할 때 사용된다.

어떤 단어를 입력 받았을 때, 단어가 사전에 있다면 그대로 반환한다. 사전에서 찾지 못한 경우, 입력 단어에서 delete, insert, transpose, replace 네가지 유형에 대하여 편집 거리가 1인 교정후보들을 모두 구한다. 예를 들어 ofen이라는 단어가 있다고 하면 delete군은 [fen, oen, ofn, ofe], insert군은 [aofen, bofen, cofen, … ofenz], transpose군은 [foen, oefn, ofne], replace군은 [afen, bfen, … , open, … opez]를 만들 수 있다. 위의 후보들 중에서 사전에 존재하는 단어만을 모아서 최종 후보를 정한 뒤, 사전에 저장되어 있는 단어의 빈도 수를 기반으로 입력 단어를 교정해준다.

만약 편집 거리가 1인 교정후보들 중 사전에 존재하는 단어가 없을 경우 이 교정후보들로부터 각각 편집 거리가 1인 교정후보들을 생성한다. 즉, 이는 입력 단어에 대해 편집 거리가 2인 교정후보들이 된다. 마찬가지로 이 후보들 중 사전에 존재하는 단어를 빈도 수 기반으로 입력 단어를 교정해주며, 거리가 2인 교정후보도 존재하지 않을 경우엔 입력 단어를 그대로 반환해준다.

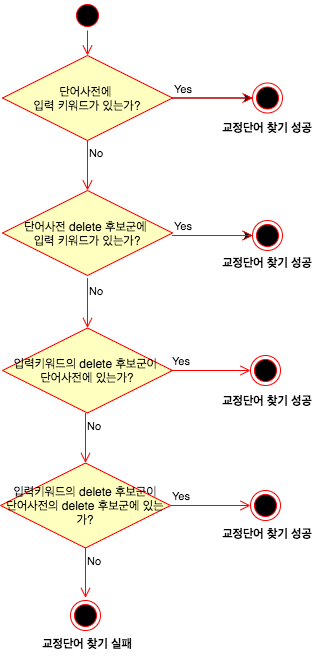
가장 직관적인 알고리즘이지만 동작이 느리다는 단점이 존재한다. 영어 기준으로 입력 길이가 n인 단어에 대하여 편집 거리가 1인 교정 후보의 개수는 54n + 25개이며, 편집 거리가 2인 교정후보의 개수는 대략 개나 되기 때문이다.

텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명

**Symspell 알고리즘**

<https://github.com/wolfgarbe/SymSpell>

[[1]](#footnote-1)Peter Norvig의 높은 연산비용 문제를 해결한 알고리즘이 Symspell 알고리즘이다. 이 알고리즘은 우선 단어사전 파일을 읽어서 모든 단어에 대한 delete후보군을 계산해 놓는다.

그 후 단어를 입력받았을 때 단어 사전에 있는 단어면 그대로 반환한 다. 없다면 단어 사전으로부터 미리 계산해놓은 단어사전 delete 후보군에 입력 단어가 있는지 확인한다. 있다면 사전에서 해당되는 단어를 반환한다. 단어사전 delete 후보군에서 찾지 못했다면 입력 키워드의 delete 후보군을 구한다. 이 각각의 후보군을 단어사전에서 찾아보고 찾았다면 해당 단어를 반환한다. 여기서도 찾지 못했다면 입력 키워드의 delete 후보군 각각을 단어사전 delete 후보군에서 찾아보며, 찾았다면 해당 단어를 반환한다. 못 찾았다면 교정에 실패한 것으로 원래 입력 키워드를 반환한다.

Peter Norvig 알고리즘과 달리 Delete 후보군만을 사용하며 미리 사전 Delete 후보군을 계산해 놓기 때문에 수행 속도가 훨씬 개선되었다.

**기존의 한글 오타 교정 프로그램**

우선 천지인이 고려된 한글 오타 교정 프로그램은 없었기 때문에 일반적인 한글 오타 교정 프로그램을 찾아보았다.

**● spell-checker**

<https://github.com/pirate/spellchecker>

Peter Novig’s 알고리즘의 확장 버전으로 splits, deletes, transposes, replaces, insserts에 단락옵션을 추가하고, 해밍 거리를 기반으로 단어 선택을 하도록 하였다. 다중 오타를 더 감지하기 쉽도록 이중 단어 변형도 검사하고, 모음끼리의 교환을 감지하는 기능, 또한 한 단어에 대한 메모리 효율성을 다중 리스트(리스트 안 리스트)로 높혔다.

**● star.js**

<https://github.com/alexyorke/star.js>

사용자가 직접 고칠 단어를 입력하면 자동으로 고칠 부분을 찾아 변환해주는 프로그램이다. ayoxk -> quick 과 같이 기존 맞춤법 검사기에서는 제안할 수 없는(예측 불가능한) 단어를 수정할 수 있다. 키보드의 물리적 거리와 “levenshtein distance”에 중요도를 부여한다는 점에서 우리 프로젝트와 유사한 부분이 있다.

**● Hunspell-dict-ko**  
<https://github.com/spellcheck-ko/hunspell-dict-ko>

오픈소스 데스크톱에 널리 사용되는 hunspell 맞춤법 검사 프로그램에서 동작하는 유일한 한국어 사전이다.

**● Atom-korean-spell-daum**  
<https://github.com/yomybaby/atom-korean-spell-daum>

포털 사이트 daum에서 교정사항을 확인 후 적용시키면 원문이 교정글로 바뀌는 Atom 패키지이다.

**● WordTypoCorrector-Korean**<https://github.com/JeaminRhee/WordTypoCorrector-Korean->

워드 파일에 대한 한글 오타 교정 프로그램이다.

2-1. 구현

프로젝트 특성상 문자열을 많이 다루어야 하므로 문자열에 대한 편리한 기능들이 많은 Python3를 개발 언어로 선택했다.

## **아키텍처**

* **word\_fixer.py**

각converter와 edit\_distance\_calculater를 사용하여 단어 단위의 오타를 교정한다. 가장 가능성 높은 교정 후보로 바로 교정할 수도 있고, 가능성 순으로 모든 교정후보를 보여줄 수도 있다.

* **chinjiin.py**

word\_fixer를 이용하여 문장/파일/폴더 단위의 오타 교정을 지원한다.

* **/converter**
  + **cji\_converter.py**

한글 문자열을 천지인 자판 단위의 문자열로 변환한다.

* + **del\_converter.py**

문자열에 대한 delete 후보군을 계산한다.

* + **han\_converter.py**

천지인 문자열을 읽을 수 있는 한글 문자열로 변환한다.

* + **/dict**

사전 데이터

* **/measurer**
  + **edit\_distance\_calculater.py**

기준 문자열에 대하여 어떤 문자열의 물리적 편집거리를 계산한다.

## **알고리즘 // 수정예정**

**1. 음소분리**

친지인 프로젝트의 오타 교정 알고리즘에서 가장 중요하고 기초가 되는 부분은 음소분리이다. 천지인 키보드의 음소분리는 기존의 두벌식 키보드 음소분리와 차이가 크다. 다시 한 번, ‘고양이’라는 단어로 예를 들자면, 두벌식에서는 우리가 직관적으로 생각할 수 있듯이, ㄱ, ㅗ, ㅇ, ㅑ, ㅇ, ㅇ, l로 음소를 분리되며, 이런 방식으로 음소를 분리하는 오픈소스는 이미 많이 존재한다. 하지만, 천지인 키보드에서는 ‘고양이’가 ㄱ, ㆍ, ㅡ, ㅇ, ㅣ, ㆍ, ㆍ ,ㅇ , space, ㅇ, l로 분리된다. 그렇기 때문에 음소 분리 알고리즘부터 직접 설계해야 한다.

**2. 사전 구축**

사전은 여러 데이터들로 실험해 볼 생각이다. 국립 국어원에서 제공하는 국어사전을 사용하면 대부분의 한글 단어들을 찾아낼 수 있지만, 잘 쓰지 않는 단어들이 지나치게 많기 때문에 올바른 교정이 이루어지지 않을 가능성이 크기 때문에 실험이 필요하다고 생각했다. 다른 사전으로는 위키에서 제공하는 ‘자주 쓰이는 한국어 낱말 5800’과 같이 자주 쓰이는 단어만으로 이루어진 데이터를 써볼 예정이다. 마지막으로 국립국어원에 요청 시 받을 수 있는 ‘말뭉치’가 있다. 이 말뭉치는 실제 대화에서 등장한 문장들로 이루어진 Json파일인데 이 파일로 사전을 구축해서 사용해 볼 생각이다.

위와 같은 방법으로 사전을 선택한 뒤, 입력 키워드와 쉽게 비교할 수 있도록 모든 단어에 대해 음소분리를 사전에 해놓을 것이다.

**3. Symspell 적용**

위에서 설명한 Symspell 알고리즘을 적용할 계획이다.

**4. 물리적 거리**

차별성을 더하기 위해, 친지인 오타 교정 알고리즘은 Symspell 알고리즘을 기반으로 하되, 키보드 자판간의 거리로 교정 단어간의 우선순위를 정할 계획이다. 참고할 자료로 영어 Qwerty키보드 자판의 물리적 거리를 Symspell에 적용한 Customized-Symspell이라는 프로젝트를 찾아볼 수 있었다.  
<https://github.com/MighTguY/customized-symspell>

예를 들면, ‘낭아지’라는 오타 단어를 입력받으면, ‘강아지’, ‘망아지’ 등의 교정 후보가 존재할 텐데 ‘ㄱ’자판이 ‘ㅁ’자판보다 물리적 거리가 가깝기 때문에 ‘강아지’를 우선적으로 추천해 줄 것이다.

2-2. 개선

중간 발표 이후 개선된 내용은 다음과 같다.

## **사전 로딩 속도 개선 (Pickle)**

파이썬은 객체를 파일로 저장하게 해주는 pickle을 제공하고 있다. 또한 pickle은 파일을 binary형태로 저장해주기 때문에 적은 용량으로 운용이 가능하다. 기존 txt 형식의 사전 파일을 프로그램 내에서 사용하는 딕셔너리로 pickle 저장하여 사전 로드 시간을 3배 가량 눈에 띠게 줄일 수 있었다.

## **코딩 스타일 통일 (PEP8)**

원활한 협업을 위해서 파이썬 PEP8 스타일 가이드를 채택하였다.

## **사전 최적화**

국립국어원의 2020말뭉치ver 1.0 json파일을 파싱한 기존 사전은 빈도수를 기반으로 정렬되어 있었다. 그 중 사용 활용도가 낮은 빈도수가 1인 단어를 삭제하고, 다른 단어 데이터인 챗봇 데이터와 한국어 학습용 단어를 추가하였다. 이를 통해 사전의 용량은 줄이고, 실효성은 높일 수 있었다.

## **편집거리 알고리즘 개선**

기존의 damerau-levenshtein-distance를 이용한 편집거리 계산 알고리즘에서는 deletion 오타에 대해서도 물리적 거리가 적용되었다. 예를 들어 ‘운동자’라는 오타에 대해 ‘눈동자’와 ‘운동장’ 두가지 교정 후보가 존재한다고 하면, ‘눈동자’(replace)에서는 ‘ㅇ’과 ‘ㄴ’의 거리를 적용하면 된다. (두 칸 떨어져있기 때문에 가능성 낮은 오타) 그에 반해 ‘운동장’(deletion)에서는 거리 계산이 적용되면 안되는데 ‘ㆍ’와 ‘ㅇ’의 거리(세 칸 떨어져있음)가 적용되는 문제가 있었기 때문에 ‘눈동자’가 먼저 추천되었다. 따라서 deletion 오타에 대해서는 물리적 거리를 1이라 가정하고 편집거리를 계산하도록 고쳐주었으며 이제는 ‘운동장’이 추천된다.

## **단어 분절 적용 검색 (Bigram)**

블루베리스무디와 같은 복합명사는 사전에 등재가 안 되어있다. 이런 단어의 오타를 잡기 위해 Bigram을 적용하여, [블/루베리스무디, 블루/베리스무디 … 블루베리스무/디] 와 같이 입력을 두 부분으로 나눈 뒤, 양쪽의 교정 후보군의 최저 키값의 합이 키가 된다. 이 때 나누지 않고 잡을 수 있는 경우를 우선시하기 위해, penalty로 키값에 1을 더해주었다. 이와 같은 방식으로 좀더 많은 오타를 잡아낼 수 있게 되었다. 물론 Trigram을 적용하여 세 부분으로 나누면 더 정확도가 높아질 수 있기에 차후 적용해볼 생각이다.

## **문장/파일/폴더 교정 기능 추가**

단어 하나를 고쳐주는 word\_fixer를 import하는 chinjiin에서는 fix, fix\_file, fix\_dir과 같은 메서드를 지원하는데 각각 문장, 파일, 폴더 내 모든 파일에 대한 교정이다. 이는 교정 후보군 중 하나하나 고르기가 어렵기 때문에 가장 가능성 높은 교정 후보로 바꿔주는 direct\_fix를 사용한다.

2-3. 평가

## **DEMO**

pass

## **선행 기술과 비교**

pass

3-1. 협업

// 추가해야함

## **역할 분담**

역할은 다 같이 하되, 중심점을 맡는 사람은 각각 다음과 같다.

|  |  |
| --- | --- |
| **역할** | **인원** |
| 총괄 팀장 | 서정현 |
| 개발 팀장 | 임우섭 |
| 테스트 팀장 | 정석우 |
| 발표 팀장 | 이창민 |
| 문서화 팀장 | 한태균 |

# 3-2. 향후 계획

pass

**Usage Scenario // 어디로 들어가야 되지**

이러한 이유들로 개발된 라이브러리 개념의 '친지인'은 검색 엔진 속에 탑재되어서, 기존 빈도 수 기반 오타 교정이나, 쿼티 자판 기준 오타 분석에 더해 '천지인' 키보드의 관점에서 단어를 추천할 수 있게 된다. 또한, 스마트폰 조작이 익숙하지 않은 사람들이 '천지인' 키보드를 쓰는 경우가 많은데, 검색 엔진이 아닌 실제 천지인 키보드 프로그램에 탑재되어서, 사용자들에게 더욱 질 좋은 오타교정을 해 주어서 스마트폰의 진입장벽을 낮추는 데에도 기여를 할 것이다.

**• 프로젝트 의의 //**

한글 교정에 대한 내용, 그 중 맞춤법 검사에 대한 프로젝트가 주를 이루고 있는 것을 확인할 수 있었다. (3) 한글 오타 교정에 대한 프로그램도 있었지만 Microsoft word 파일에 대해서만 교정이 이루어졌고, 현재 깃에 업로드 된 사전 파일에는 영단어만 존재해 제대로 작동하는지에 대한 의문도 들었다.

맞춤법과 오타는 ‘사용자의 인지 여부’가 차이를 만든다. 예를 들어 타이핑을 하고 있는 사용자가 틀린 단어를 입력한 후 단어가 틀렸다는 것을 알고 나서 고치려 한다면 오타, 틀렸는지를 모른다면 맞춤법의 영역이다. 현재 공개되어 있는 한글 교정 프로그램은 맞춤법 교정에 중심을 두고 있었고 실제로 많은 모바일 키보드에서 맞춤법 교정 기능을 제공하고 있기도 하다.

하지만 그 중 천지인이라는 특정한 한글 키보드에 대해 그 특징을 고려한 오타 교정에 대해 논의된 프로젝트가 없다는 점이 이 프로젝트의 필요성을 대두시킨다. 2010년 기준 국내 제조사의 한글 자판 시장 점유율은 천지인 54%, 나랏글 22%, SKY 16%, 기타 8%로[[2]](#footnote-2) 천지인 입력 방식은 스마트폰의 본격적인 보급이 시작되던 시기 가장 널리 사용되었던 키보드였음을 통계에서 확인할 수 있었다. 천지인 입력 방식에 대한 오타 교정 알고리즘 연구가 오픈소스화 되어 공개된다면 현재도 천지인 방식을 사용하고 있는 사용자들의 편의성을 크게 증가시켜줄 것이기 때문에 이 오픈소스 프로젝트가 필요하다.

1. <https://americanopeople.tistory.com/349> 그림 출처 [↑](#footnote-ref-1)
2. <https://scienceon.kisti.re.kr/srch/selectPORSrchTrend.do?cn=SCTM00092613> [↑](#footnote-ref-2)