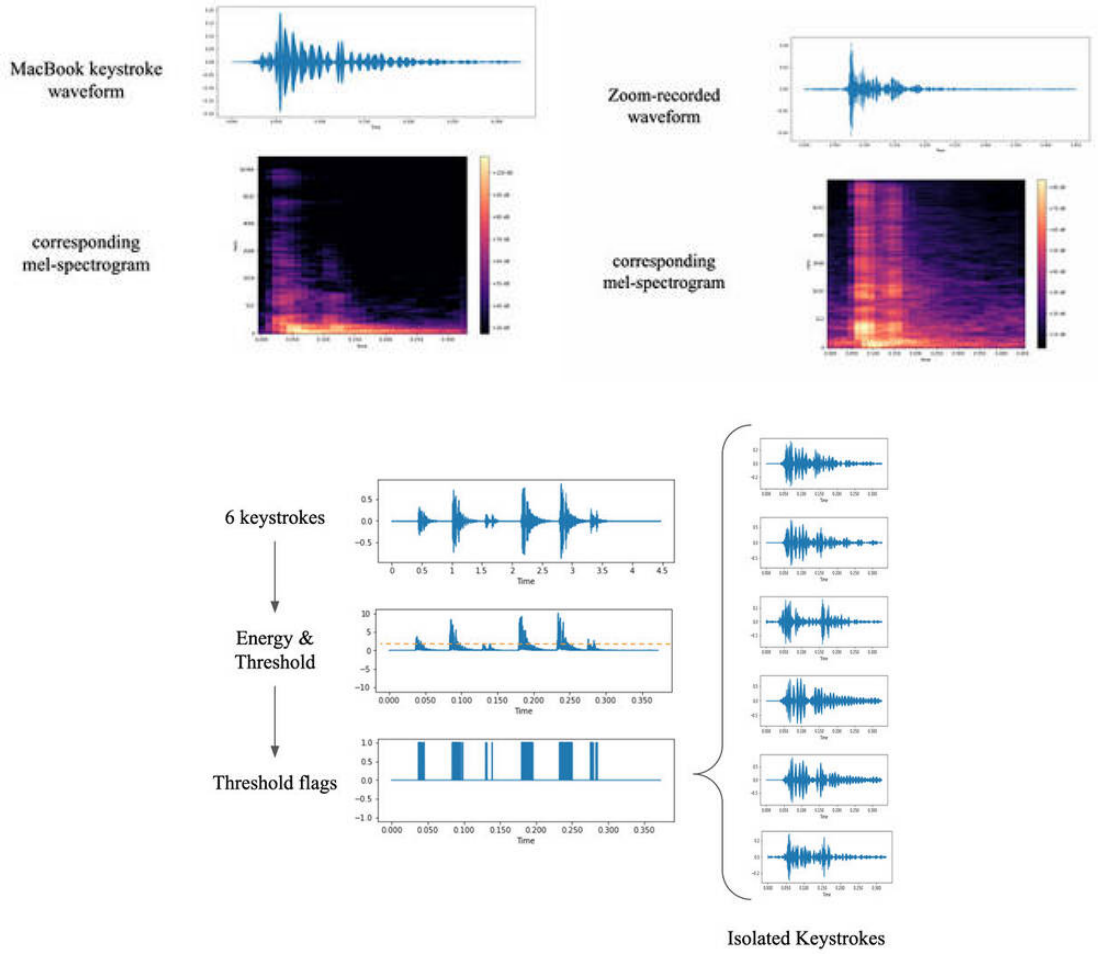
KeySpyder

키보드 소리를 통한 입력값 인식

Team 2. 김성학, 김준식, 박종익

프로젝트의 시작부터, 현재까지 진행된 내역을 누적 기재

Background



키보드 소리만으로도 해킹이 가능하다는 기존의 연구 결과 및 프로젝트에 착안

해당 주제를 한국어 기반으로도 가능한지를 증명하기 위한 프로젝트

Github Repository

https://github.com/DummyBlue/Capstone_2023F_Team2

- Readme 파일 갱신

자체 설정한 제약 조건

1. 쌍자음 제외
2. 문장이 아닌 단어만 이식
3. 영어를 사용하지 않는, 한글 사용만을 가정
4. 주변 소음은 차단된 상태
5. Zoom을 타고 들어오는 소리를 감청해 사용
6. 타자연습 프로그램을 이용, 일정한 입력값을 받도록 제약 설정

최종 Solution

- 키보드 음성 처리
- 음성 유사도 측정
- 글자/단어 자동 완성

Process

녹음 - 처리 - 비교 - 추론

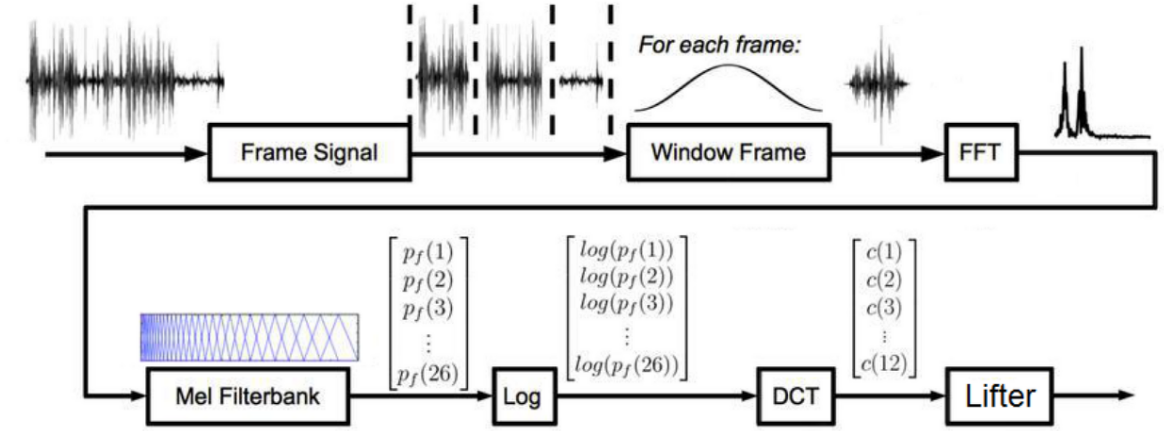
Detailed Process

녹음

- 녹음 수행
    - Zoom 이용
- Background Sound 차단
    - 통제된 환경 사용
    - 백색 소음에 해당하는 진폭 부분을 제거

- 마이크 위치에 따른 차이 확인
    - 노트북의 내장 마이크 위치 확인
    - 타이핑 과정에서 발생하는 소음의 양상 확인

오디오 데이터 처리_MFCC(Mel-Frequency Cepstral Coefficients)



Mel-Scale로 음성 데이터 구간을 정하고, 그 구간에 대한 Spectrum을 분석하여 푸리에 변환을 한 특징 추출 방법 (자세한 내용은 뒤에서 설명하기로 한다)

비교

keytap 프로젝트는 기본적으로 acoustic keylogging에 관한 연구를 바탕으로 키보드 입력의 소리를 분석하여 키 입력을 예측. 주요 과정은 몇 가지로 분류 가능.

1. 데이터 수집:
첫 번째 단계는 키보드의 각 키를 누를 때 발생하는 소리를 녹음하여 데이터베이스를 만드는 것이다. 이 데이터베이스는 나중에 소리의 패턴을 인식하는 데 사용된다.

2. 특징 추출 (Feature Extraction):
녹음된 데이터에서 유용한 정보, 즉 "특징"을 추출한다. 이 프로세스에서는 주로 시간 및 주파수 도메인에서 데이터의 특성을 분석하여 관련된 정보를 찾아낸다. FFT (Fast Fourier Transform)와 같은 기법들이 이 단계에서 사용될 수 있다.

3. 분류 (Classification):

특징 추출 단계를 거친 데이터는 분류기에 입력되어 어떤 키의 소리인지를 결정한다. keytap은 k-최근접 이웃 (k-NN) 알고리즘을 사용하여 분류를 수행한다. k-NN은 분류하려는 데이터 포인트와 가장 가까운 k개의 학습 데이터 포인트를 찾아 해당 데이터 포인트의 분류를 결정하는 알고리즘이다.

4. 결과 출력:

분류 결과를 사용하여 어떤 키가 눌렸는지를 예측하고 결과를 출력한다.

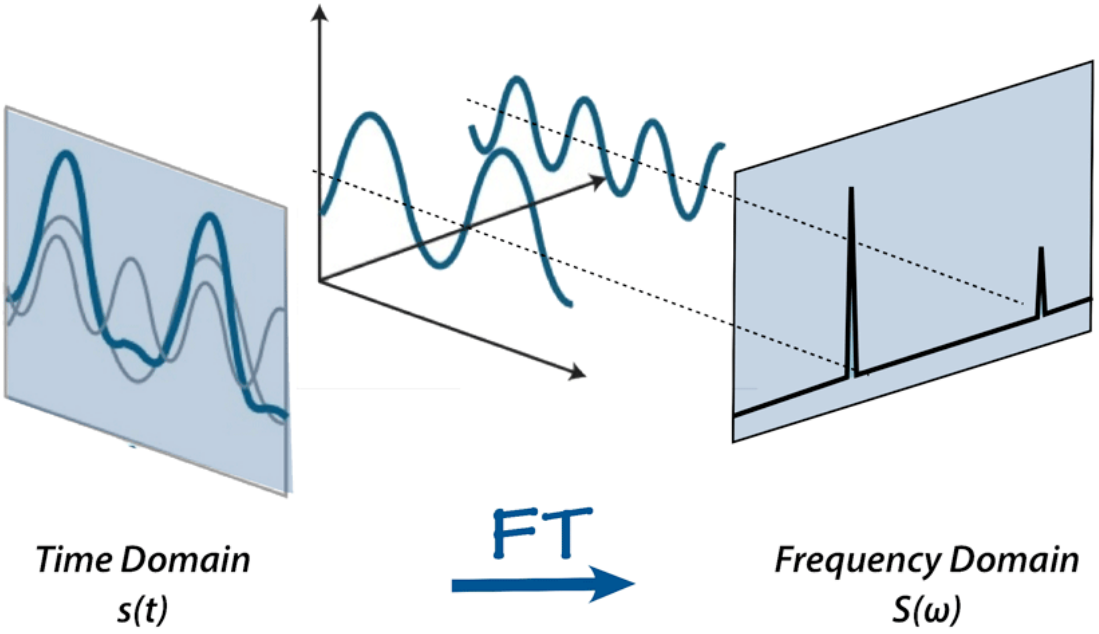
핵심은 각 키의 소리가 약간씩 다르다는 것을 이용하는 것이다.

키보드의 각 키는 서로 다른 위치와 구조를 가지므로 약간 다른 소리를 만들 수 있다. keytap은 이러한 소리의 차이를 분석하여 특정 키를 예측한다.

[Background Knowledge]

FT (Fourier Transform)

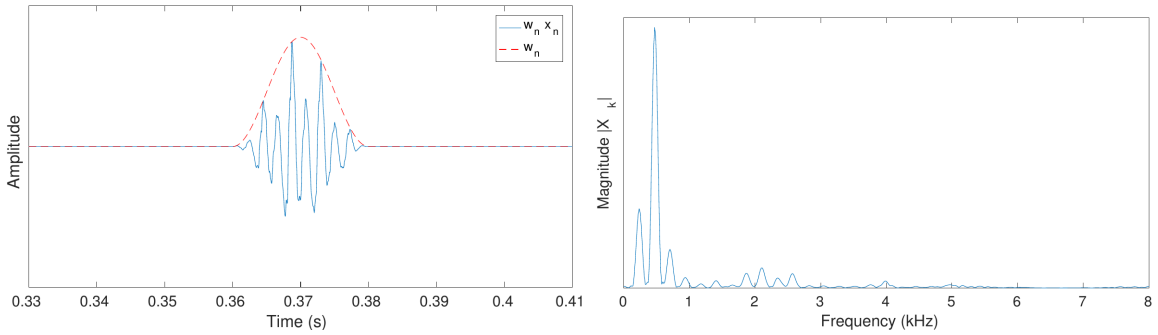
Fourier Transform (FT)은 시간 영역에서의 신호를 주파수 영역으로 변환하는 수학적 연산이다. 즉, 어떤 신호가 어떤 주파수 성분으로 구성되어 있는지 분석할 수 있게 해준다.



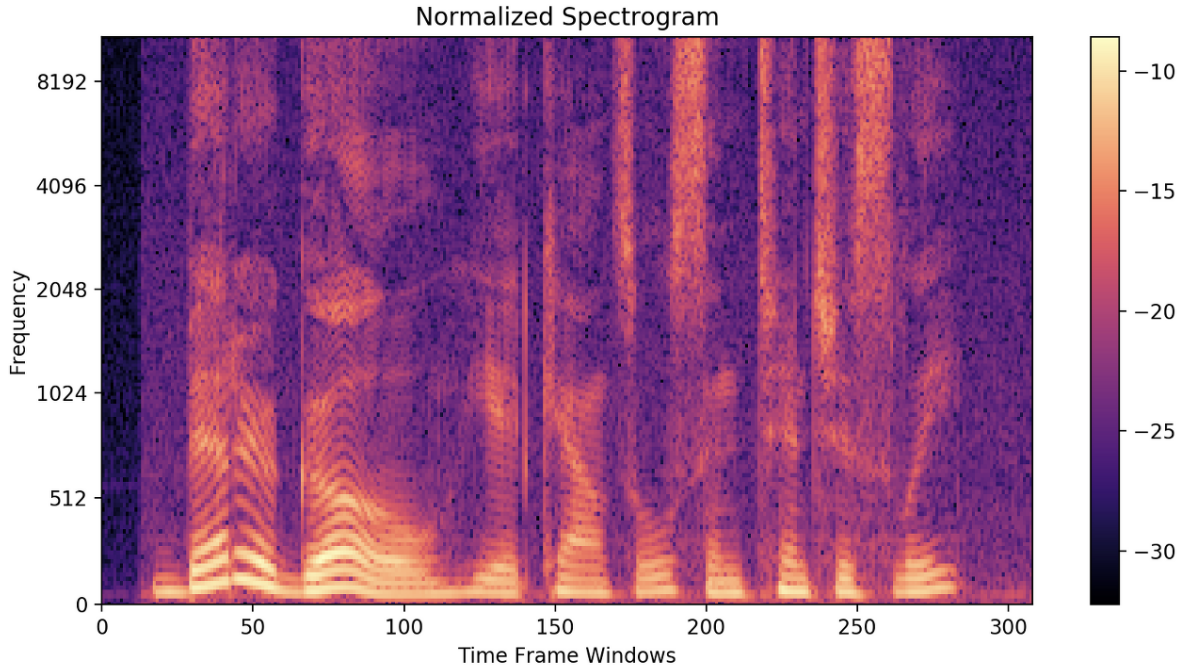
FFT(Fast Fourier Transform), STFT(Short-Time Fourier Transform):

Fast Fourier Transform은 Fourier Transform을 효과적으로 계산하는 알고리즘으로 FT를 직접 계산하는 것보다 훨씬 빠르게 주파수 영역 변환을 수행한다. 다만 time domain에서 frequency domain으로 mapping시켜주는 과정에서 time domain 정보를 완전히 잃어버린다는 단점이 있다. 이를 해결하기 위해 STFT(Short-Time Fourier Transform)이 time window를 움직이며 FFT를 수행시켜 local time area 진동수 성분을 파악한다. *(이번 프로젝트에서는 STFT를 사용하고자 한다.)*

Spectrum, Spectrogram



위와 같이 time domain에서 frequency domain 으로 바뀐 그래프를 spectrum이라고 한다. 여기서 Y축 magnitude를 제곱한 것을 power spectrum, log scale을 취해준 것을 log spectrum이라고 한다. 데시벨 변환 공식을 통해 log spectrum을 구하고 이를 세로로 세워서 frame마다 옆으로 쌓아 푸리에 변환으로 사라졌던 time domain을 복원한 결과를 spectrogram이라고 한다.



Mel filter Bank

사람이 받아들이기 어려운 Hertz scale 대신, 인간의 청각 시스템의 특성에 맞게 변환한 주파수인 mel scale로 spectrum을 변환한다. 변환된 spectrum에 mel filter bank를 적용하여 mel scale 로 나타낸 주파수 영역을 여러개의 밴드로 분할한다. 각 mel filter band의 power 평균값에 log scaling을 진행한다.

DCT(Discrete Cosine Transform)

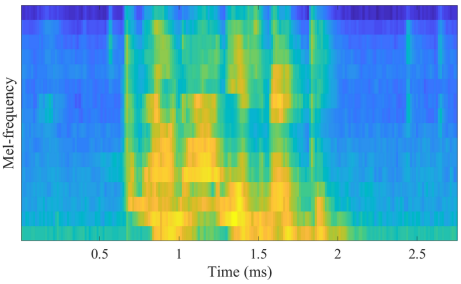
DCT는 특정 함수를 cosine 함수의 합으로 표현하는 변환이다. 이때 앞쪽(low) cosine 함수의 계수가 변환 전 데이터의 대부분의 정보를 가지고 있고 뒤쪽으로 갈수록 0에 근사해 데이터 압축의 효과를 보인다. 즉 낮은 주파수 쪽으로 에너지 집중현상이 일어난다.

MFCC는 이 계수(cepstrum coefficient) 중 주파수가 낮은, 정보와 에너지가 몰려있는 12개의 계수(cepstrum coefficient)를 선택해 이를 feature로 사용한다. 그리고 이 12개 계수에 해당하는 각 frame의 Energy를 13번째 feature로 사용한다. cepstrim coefficient는 다음과 같이 구해진다.

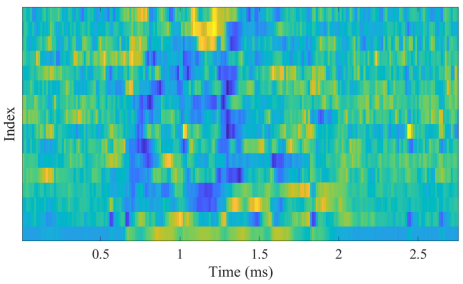
$$C_n = \sqrt{\frac{2.0}{\mathrm{fN}}} \times \sum_{j=1}^{\mathrm{fN}} \log_{10} \left(\sum_f W_j(f) \left| X(f) \right|^2 \right) \cos \left(\frac{\pi}{\mathrm{fN}} n \left(j - 0.5 \right) \right)$$

MFCC(Mel frequency cepstral coefficient)

Mel-spectrogram에 DCT를 취하고 낮은(정보가 많은) 계수(cepstrum coefficient) 12개와, 이들로 구해진 energy를 더해 한 frame 당 총 13개의 값이 feature 역할을 하게 되고 이를 MFCC(mel frequency cepstrum coefficient)라고 한다.



Mel-Spectrogram



MFCC

MFCC 13개 값의 각 frame 간 차이를 Deltas(1차 차분), 다시 이 Deltas 값의 frame 간 차이를 Delta-daltas(2차 차분)로 구해서 해당 값들을 추가적인 feature로 사용해 총 39개의 MFCC featrue를 최종적으로 사용한다.

k-NN (k-Nearest Neighbors)

기본 개념: k-NN은 간단한 지도 학습 알고리즘. 주어진 데이터 포인트와 가장 가까운 k개의 훈련 데이터 포인트를 찾아서, 그 중 가장 많이 나타나는 클래스 또는 평균값을 예측 결과로 사용하는 방식.

동작 원리:

새로운 데이터 포인트가 주어질 때, 훈련 데이터셋 내에서 이 포인트와 가장 가까운 k개의 데이터 포인트를 찾는다.

분류 문제의 경우, k개의 이웃 중 가장 많이 나타나는 클래스를 결과로 선택.

회귀 문제의 경우, k개의 이웃의 평균 값을 결과로 선택.

장점:

구현이 간단하고, 조정할 파라미터가 거의 없어 초기 모델링 시에 유용하다.

작은 데이터셋에서도 잘 동작 가능.

단점:

데이터셋이 크면 예측 시간이 오래 걸림.(모든 훈련 데이터와의 거리를 계산해야 하기 때문)

데이터의 차원이 높으면 ("차원의 저주" 문제로 인해) 성능이 떨어질 수 있음. 적절한 k 값을 선택하는 것이 중요함.

FFT와 k-NN,은 keytap과 같은 acoustic keylogging 도구에서 키 입력 소리의 주파수 특성을 분석하고 예측하는 데 중요한 역할을 한다.

여기서 Acoustic keylogging은 키보드 입력의 소리를 기록하고 분석하여 사용자가 어떤 키를 눌렀는지를 추측하려는 기법을 의미한다. 기본적인 아이디어는 키보드의 각 키가 눌렸을 때 특유의 소리나 진동 패턴을 만든다는 것이다. 이러한 패턴을 분석하면 사용자가 어떤 키를 눌렀는지 예측할 수 있다.

Acoustic keylogging의 주요 특징:

비침투성:

이 공격은 키보드나 컴퓨터 시스템에 직접 접근하지 않아도 수행될 수 있다. 대신, 공격자는 키 입력 소리를 기록할 수 있는 거리 내에서 마이크를 배치해야 한다.

소리 분석:

키 입력의 소리는 FFT (Fast Fourier Transform)와 같은 시그널 처리 기법을 사용하여 분석될 수 있다. 이렇게 얻은 데이터는 분류 알고리즘에 의해 특정 키와 연결될 수 있다.

학습 필요:

Acoustic keylogging은 특정 키보드의 소리 패턴을 학습하는 과정이 필요하다. 여기에는 키보드의 모델, 타입, 사용자의 입력 습관 등 다양한 변수가 포함될 수 있다.

keytap과 Acoustic Keylogging의 관계:

keytap은 acoustic keylogging의 실용성과 가능성을 연구하고 실증하기 위한 도구로 개발되었다. 프로젝트는 키보드 입력 소리를 기반으로 어떤 키가 눌렸는지 예측하는 기능을 제공하며, 이를 위해 k-최근접 이웃 (k-NN)과 같은 분류 알고리즘을 사용한다.

keytap의 개발은 acoustic keylogging의 위험성을 인식시키는 데 의미를 둔다. 이러한 연구를 통해 사용자와 조직은 소리 기반의 공격에 대비할 수 있게 된다.

acoustic keylogging은 키보드의 입력을 비밀리에 기록하고 분석하는 공격 기법을 나타내며, keytap은 이러한 공격의 가능성과 실용성을 탐구하기 위한 도구로 볼 수 있다.

keytap은 acoustic keylogging을 구현하기 위한 도구로서, 키보드 입력 소리를 기반으로 키를 예측한다: 학습(Training)과 예측(Prediction)을 통해 수행

1. 학습 (Training):

1. 이 단계에서는 키보드의 각 키를 여러 번 누르면서 발생하는 소리를 녹음
2. 이렇게 수집된 데이터는 각 키의 소리의 "지문" 또는 고유한 특성을 만드는 데 사용
3. 주요 목적은 키보드의 각 키에 대한 소리 프로파일 또는 특징 벡터를 생성
4. 이 데이터는 나중에 실시간 예측을 위한 참조로 사용

2. 예측 (Prediction):

1. 사용자가 키를 입력할 때, keytap은 마이크를 통해 소리를 실시간으로 캡처하고 분석
2. 캡처된 소리 데이터는 학습 단계에서 생성된 키의 프로파일과 비교되어 가장 유사한 프로파일을 찾음.
3. keytap은 k-최근접 이웃 (k-NN) 분류 알고리즘을 사용하여 캡처된 소리가 어떤 키와 가장 유사한지 판단. 여기서 k는 사용자가 정의할 수 있으며, 보통 소수의 값을 가진다.
4. 알고리즘이 가장 유사하다고 판단하는 키의 프로파일을 기반으로 해당 키를 출력

이런 방식으로 keytap은 마이크를 통해 캡처된 소리 데이터를 기반으로 실시간으로 키 입력을 예측한다.

실행 방법:

1. 먼저 keytap 프로젝트를 가져와서 빌드한다.
2. 필요한 파일을 설치한다. 이는 프로젝트 문서나 README에서 안내되어 있다.
3. 학습 모드로 프로그램을 실행하여 각 키의 프로파일을 생성
4. 예측 모드로 프로그램을 실행하여 실시간 키 입력 예측을 시작

추론

- 한국어 자모 분리/결합
    - Jamo/Hangul_utils
- 한글 맞춤법 교정
    - SymSpellpy-ko

사용 방안:

1. 음성 데이터와 키보드 값을 매핑
2. 이 과정을 거친다면 한글 자모의 나열이 완성 (ex. ㄱㅣㄹㅓㄱㅣ)
3. 나열된 값을 Hangul_utils를 이용해 결합 수행 (ex. 기러기)
4. 이후 SymSpellpy-ko를 이용해 오/탈자 교정 수행
5. 이후 원래의 자모, 결합 결과, 교정 결과를 모두 출력

Jamo

https://github.com/JDongian/python-jamo

한글을 자모 단위로 나누어주는 NLP 도구

한국어 음절을 조합형 자모로 변환 후, 한글 호환 자모로 변환

Hangul_utils

https://github.com/kaniblu/hangul-utils

분리되어 있는 자모를 하나로 결합해주는 NLP 도구

SymSpellpy-ko

https://github.com/HeegyuKim/symspellpy-ko

SymSpellPy를 한글 특성에 맞춰 수정한 라이브러리

음소분해를 이용해 더 정확한 오타 교정 지원

사용법은 종전의 SymSpellPy와 동일

SymSpell

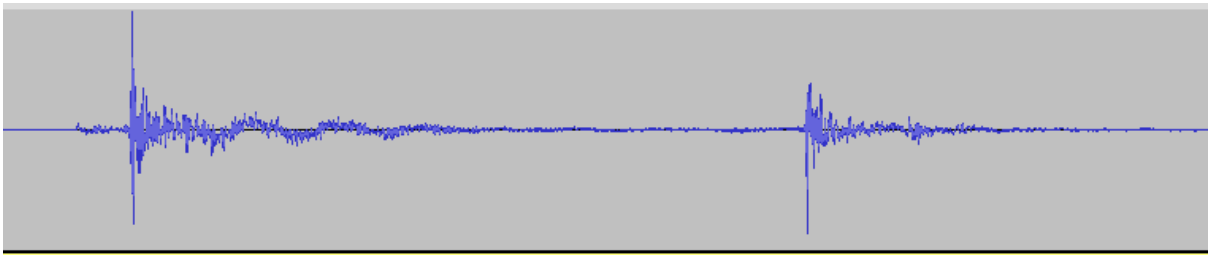
오타교정, 유사검색어 제안 기능에서 활용되는 알고리즘

Peter Norving 알고리즘의 단점 보완 (속도 향상, 다양한 자연어 지원)

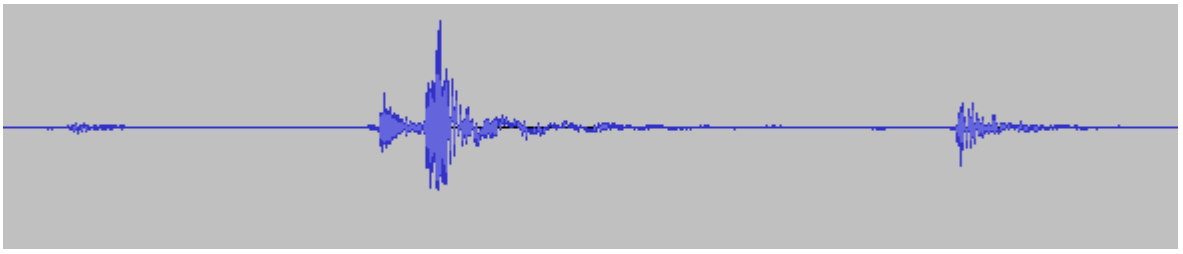
두 가지 접근 방법 사용: 오타 후보군 사전 계산, Delete 방식으로만 후보군 계산

키보드 시연

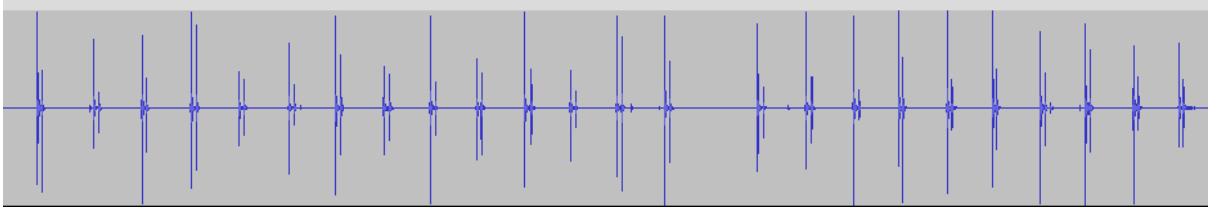
- 프로젝트는 펜타그래프 방식의 키보드(랩탑에 범용적으로 사용되는 방식)으로 진행되나, 프로젝트의 본격적인 구현에 앞서 차이 유무를 살피고자 다른 키보드로 키의 위치에 따른 소리 파형 양상 탐색
- 시연 자체는 기계식 적축 키보드를 이용 (기존 연구들은 대부분 기계식 청축과 같은 고소음 방식 사용)
- 키에 따라서 소리 파형의 차이가 존재함을 확인
- 이후 펜타그래프 키보드를 이용해 동일한 과정을 수행 (예정)
- ㄱ의 입력 결과



- ㄴ의 입력 결과



- 한글 자모 순서대로 입력한 결과



결과물 및 평가 방안

- 결과물은 가장 일치하는 것 1개 + 후보군 4가지 제시
- Key Stroke 분리 후 학습 데이터와 대조해 키 확인
- 단어 추론이 목표
- 이용자가 입력한 실제 키와 소리를 통해 예측된 키의 일치율 확인
- 교정 전/후 각각의 정확도 파악 (모든 기준은 한글 자모 단위)
    - 교정 전:
        - Key Stroke별 비교
        - 자모의 결합 이후 완성된 단어 비교
    - 교정 후:
        - 맞춤법 교정 후 교정된 단어 비교

추후 논의 사항

- Key Stroke 단위로 자동 Split 방안
- 실제 코드 작성 시작
- 구체적인 평가 방안 확정

추가 참고 자료

- 머신러닝을 이용한 오디오 비교
  https://www.section.io/engineering-education/machine-learning-for-audio-classification/
- 딥러닝 CNN 음향분류
  https://velog.io/@seaworld0125/%EB%94%A5%EB%9F%AC%EB%8B%9D-CNN-%EC%9D%8C%ED%96%A5%EB%B6%84%EB%A5%98
- 음성인식 기초 및 음악분류 & 추천 알고리즘
  https://jonhyuk0922.tistory.com/114