양방향 LSTM 기반 한국어 음성 비속어 필터링

나보영

목차

연구 배경

- 인터넷 이용시간 증가 & 1인 미디어 사용시간 증가
- 음성 채팅 이용자 수 증가

→ **언어 폭력 증가**

현재 대안

1. 각 플랫폼 규정에 따른 사후 대응
2. 텍스트 중심 방지 대책

특히 음성 어플리케이션의 경우 대부분이 사후 대응

연구 목표 : 음성 어플리케이션에 대해 사전 대응 시스템 제시

음성 파일에서 욕설을 감지해서 사전에 필터링 하는 시스템

이를 통하여 콘텐츠 및 플랫폼 사용자를 더 효과적으로 보호

기존 연구

- SVM 기법 활용 필터링 모델
- CNN 사용 비속어 필터링 모델(이미지 활용)
- 품사 처리 방식 + RNN, LSTM, GRU 결합 모델
- 반전역(SEMI-GLOBAL) 문자 정렬 기법

모델 구조

1. STT 단계

- 발화(음성 데이터) -> 텍스트 데이터 변환

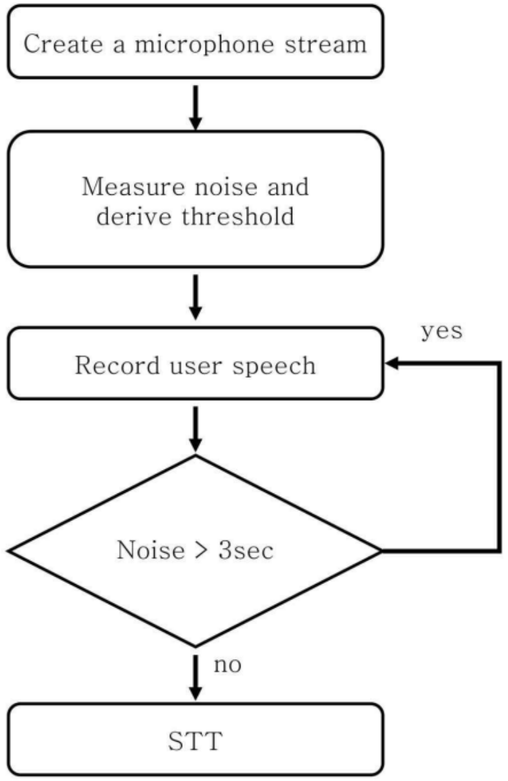
2. 필터링 단계

- 양방향 LSTM 기반 딥러닝 모델로 필터링 작업
- MFCC) 임베딩 기술을 이용하여 사용자의 발음 정확도가 모델에 영향을 주지 않도록 한다.

GOOGLE CLOUD SPEECH-TO-TEXT API

- 음성 데이터 -> 텍스트 데이터 변환
- 문장 내 단어 등장 시간 정보 저장

STT 단계


Create a microphone stream
Measure noise and derive threshold
Record user speech
yes
Noise > 3sec
no
STT

필터링 단계

1. 텍스트 데이터로 변환 된 문장을 단어 단위로 전달
2. FASTTEXT과 MFCC로 각각 FEATURE 벡터를 생성
3. 양방향 LSTM 모델에 전달
4. BAHDANAU ATTENTION 까지 수행 후 나온 ATTENTION SCORE 더하기
5. SOFTMAX 함수에 적용 -> 확률 변환
6. 앙상블 하기 즉 각 모델의 최종 확률들의 평균 구한뒤 비속어 여부 판단
7. 비속어로 판단시 앞서서 저장해 두었던 단어 등장 시각을 확인하여 음성 데이터에서 묵음 처리

FASTTEXT

- 하나의 단어를 여러 개의 SUBWORD로 분할하여 인베딩하는 방법
- 각 SUBWORD에 대한 인베딩 벡터를 계산 후 벡터의 평균을 구하여 단어의 벡터 표현 생성
- 초성, 중성, 종성으로 분리 -> FASTTEXT 임베딩 과정

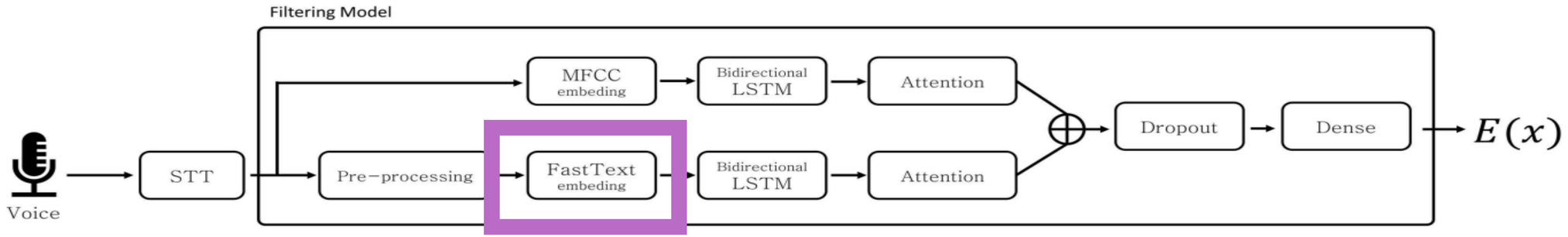
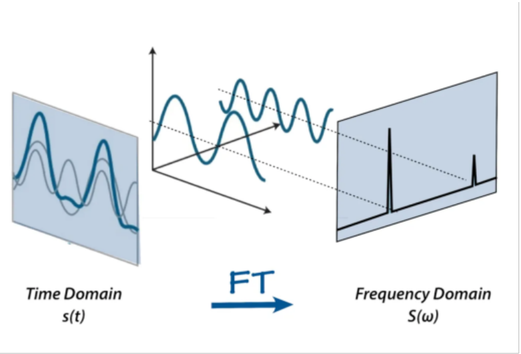


Fig. 4. Flowchart of the filtering steps centered around a profanity detection model.

필터링 단계

MFCC

1. 입력된 음성 신호를 구간으로 자른다. 즉 WINDOWING 을 거침
2. FASTFOURIER TRANSFORM(FFT)을 통해 시간 영역 -> 주파수 영역
3. SHORT TIME FOURIER TRANSFORM(STFT)을 통해 시간적 특성 고려한 스펙트럼 생성
4. MEL-FILTER BANK를 사용해 MEL-SPECTROGRAM을 생성
5. DISCRETE COSINE TRANSFORM(DCT) 처리
6. 39개의 MFCC FEATURE 생성



이러한 과정을 통하여 만들어진 FEATURE 사용

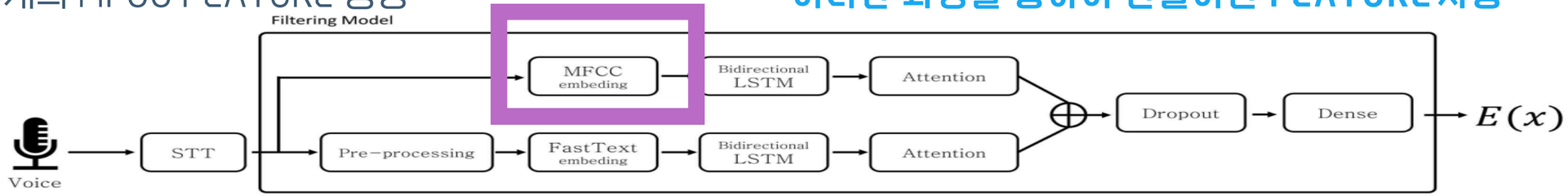


Fig. 4. Flowchart of the filtering steps centered around a profanity detection model.

DROPOUT

- 모델을 합친 후 과적합 문제를 방지 하기 위해서

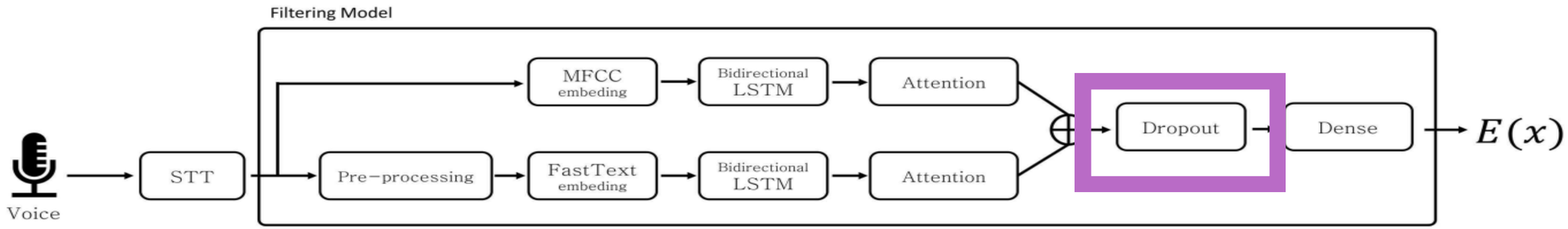


Fig. 4. Flowchart of the filtering steps centered around a profanity detection model.

필터링 단계

입력된 단어의 비속어 여부에 대한 확률값 도출 E(X)

비속어 판단 변수 α(MASKING PARAMETER) 보다 높다면 비속어로 판단

-> 묵음처리처리

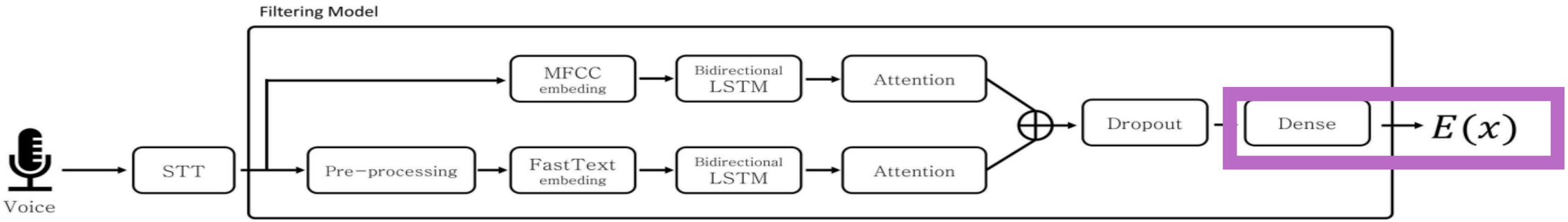


Fig. 4. Flowchart of the filtering steps centered around a profanity detection model.

실험 진행 : API + 필터링 단계의 비속어 탐지

$$N = \frac{n_1}{n_0} \times 100$$

Table 1. Comparison of STT systems.

	Google Cloud Speech-to-Text [21]	Naver Clova [22]	ETRI STT[23]
API response time	2.95s	5.91s	7.9s
False positive rate	22.19%	19.04%	23.7%

기존 연구와의 비교

0.87의 F1-SCORE -> 0.90의 F1-SCORE

87.2% 정확도 -> 91%의 정확도

Table 2. The parameter differences between the existing model [27] and our model.

	Cho [27]	Ours
Dataset	5,825	41,158
α	0.45	0.5
Shuffle	False	True

실험

문장의 길이를 설정

짧은 길이 : 1~8 글자

중간 길이 : 9~18 글자

긴 문장 : 19~28 글자

각각 문장 길이에 대해 10회 측정 후 평균

Table 3. Comparison of system processing time based on sentence length.

	Short Sentence	Middle Sentence	Long Sentence
STT	2.84s	3.93s	4.08s
Filtering	1.42s	1.46s	1.52s
Total	4.26s	5.39s	5.6s

데이터 셋

문장 기반 욕설 감지 데이터셋

온라인 커뮤니티에서 크롤링

개선점

하나의 음성 입력에 대해 5초 내외 -> 실시간 애플리케이션 적용 어려움

추후 연구를 통하여 멀티 스레딩을 통해 여러 문장을 동시에 처리

-> 실시간 비속어 필터링 가능 개선 예정