TAS:TTS with Auto-Spellcheck

머신러닝 TTS 오픈소스 개선 프로젝트

Team	2조	**Major**	Software Engineering
Team member	이다운 박준혁 김동현	**ID**	201420983 201620945 201420981

Table of Contents

1.개요

본 프로젝트는 텐서플로우를 이용한 딥러닝 TTS 오픈소스를 활용하여, 이론강의에서 습득한 딥러닝 지식을 직접 체험하는 동시에 머신러닝 관련 오픈소스 생태계에 기여하는 것이 목적이다. 딥러닝이 응용되는 다양한 분야 중에서도, NLP(Natural Language Processing, 자연어 처리)와 텍스트, 음성 처리 영역을 복합적으로 다루는 TTS 오픈소스 코드를 분석한다. 코드를 분석하며 머신러닝 프레임워크중 제일 메이저한 텐서플로우의 사용법을 습득하고, 분석대상인 오픈소스의 개선점을 찾아내어 기여하는 것이 기본 목표이다. 프로젝트의 기간이 매우 촉박하고 머신러닝 및 텐서플로우에 대한 학습이 거의 동반되지 않은 본 프로젝트 특성상, 딥러닝 지식, 그중에서도 딥러닝 성능 개선법에 대한 실질적 적용은 무리가 있다고 판단되어, 부차적인 목표로 설정하였다.

프로젝트 목표를 수립하면 다음과 같다.

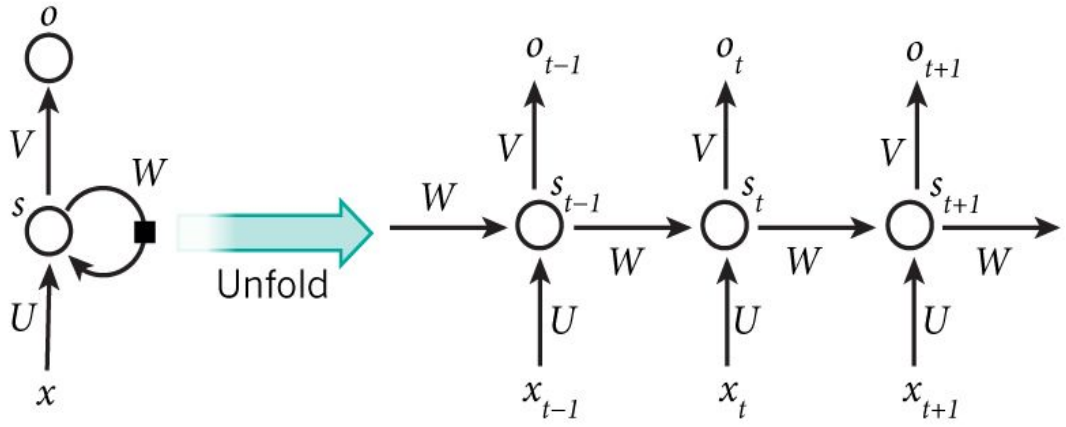
1) 기본 목표: 기존 오픈소스에서 개선점을 찾아서 기여한다.
2) 추가 목표: 이론강의에서 배운 딥러닝 성능 향상 기법을 제한적으로나마 적용하고, 기존 코드와의 산출물들을 비교한다.

2.머신러닝이란?

- **머신 러닝**
  머신러닝이란 하나부터 열까지 직접 가르치는 기계를 의미하는 것이 아니라, 학습할 거리를 일단 던져놓으면 이걸 가지고 스스로 학습하는 기계를 의미한다.

- **딥 러닝**
  딥러닝이란 머신 러닝의 한 갈래이다. 여러 비선형 변환기법의 조합을 통해 높은 수준의 추상화를 시도하는 기계학습 알고리즘의 집합 으로 정의되며, 큰 틀에서 사람의 사고방식을 컴퓨터에게 가르치는 기계학습의 한 분야라고 이야기할 수 있다. 딥러닝의 가장 큰 특징은 모델의 부피를 키우고, 데이터를 쏟아부으면 무조건적으로 그만큼의 성능향상을 보인다는 점이다.

- **RNN (Recurrent Neural Network)**
  RNN에 대한 기본적인 아이디어는 순차적인 정보를 처리한다는 데 있다. 기존의 신경망 구조에서는 모든 입력(과 출력)이 각각 독립적이라고 가정했지만, 많은 경우에 이는 옳지 않은 방법이다. 한 예로, 문장에서 다음에 나올 단어를 추측하고 싶다면 이전에 나온 단어들을 아는 것이 큰 도움이 될 것이다. RNN이 recurrent 하다고 불리는 이유는 동일한 태스크를 한 시퀀스의 모든 요소마다 적용하고, 출력 결과는 이전의 계산 결과에 영향을 받기 때문이다. 다른 방식으로 생각해 보자면, RNN은 현재지 계산된 결과에 대한 "메모리" 정보를 갖고 있다고 볼 수도 있다. 이론적으로 RNN은 임의의 길이의 시퀀스 정보를 처리할 수 있지만, 실제로는 비교적 짧은 시퀀스만 효과적으로 처리할 수 있다. 일반적인 RNN 구조는 그림1과 같이 생겼다.



<그림1>

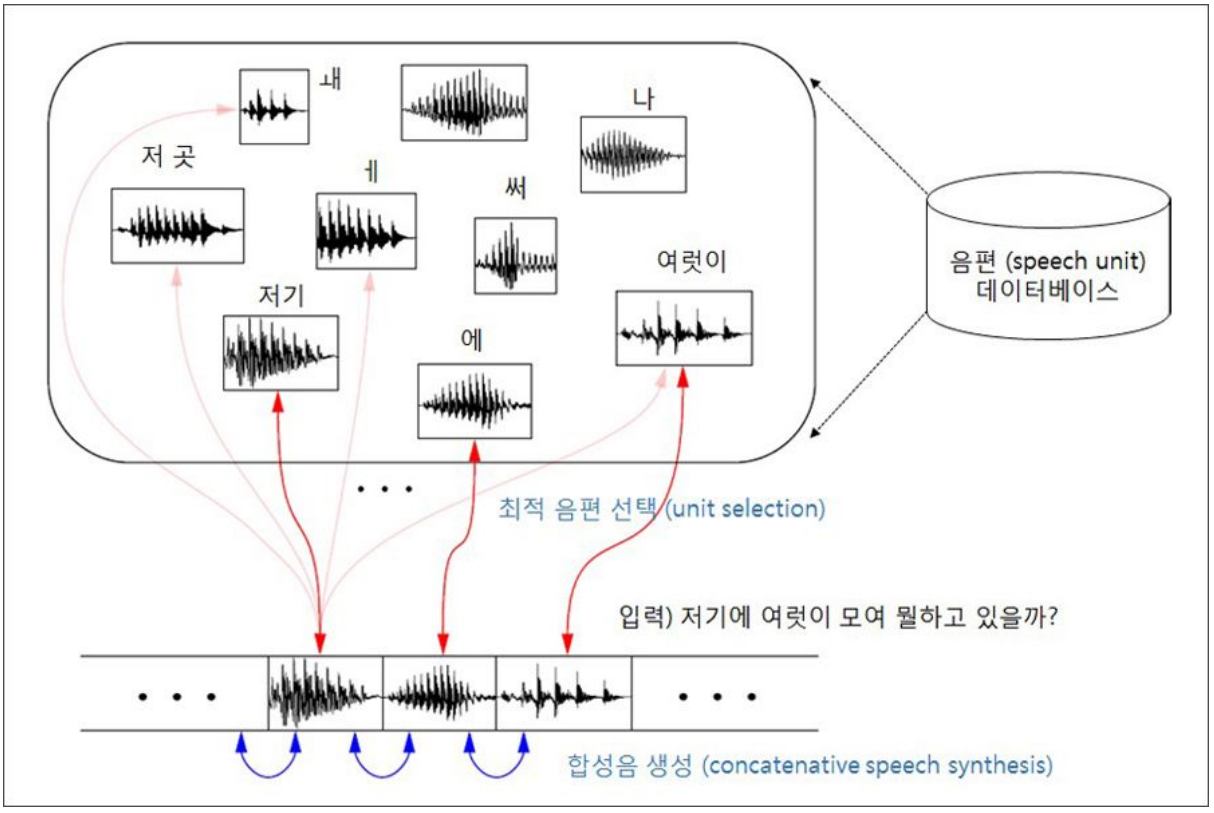
그림1 에서 RNN의 recurrent한 연결이 펼쳐진 것을 볼 수 있다. RNN 네트워크를 "펼친다"는 말은 간단히 말해서 네트워크를 전체 시퀀스에 대해 그려놓았다고 보면 된다. 즉, 우리가 관심있는 시퀀스 정보가 5개의 단어로 이루어진 문장이라면, RNN 네트워크는 한 단어당 하나의 layer씩 (recurrent 연결이 없는, 또는 사이클이 없는) 5-layer 신경망 구조로 펼쳐질 것이다.

3.오픈소스 TTS

참고한 TTS 오픈소스는 “Kyubyoung/tacotron”이다[1]. Github에 등재된 오픈소스로, 음성 합성을 Tacotron 방식을 따르는 텐서플로우 기반 딥러닝 TTS이다. 영문 음성/텍스트 오픈소스 데이터셋인 LJSpeach를 학습하여, 입력 텍스트로부터 사람이 읽은 것 같은 음성 데이터를 만들어낸다.
Tacotron이란, 딥러닝이 대두되면서 제시된 텍스트-음성 합성 학습기법이다. 딥러닝을 위시한 기계학습이 활성화되기 이전에는 연결 합성 방식, 통계 모델 기반 TTS등의 기술로 음성합성을 구현했다. Tacotron, 나아가 머신러닝 기반 음성합성 이전 기술들의 특징을 정리하면 다음과 같다.[2]

- **연결 합성 방식**: 음성언어를 구성하는 음소, 혹은 단어, 혹은 문장 단위로 저장되있는 음성 데이터(=음편, Speech unit)를 입력 텍스트에 맞게 가져와서 합성하는 방식. 예를 들어, ‘나’, ‘는’, ‘저기’, ‘에’, ‘여럿이’, ‘모’, ‘여’, ‘있다’ 라는 음편이 저장되있고, ‘저기에 여럿이 모여있다’ 라는 문장이 들어오면, 음편 데이터 중 ‘저기’ ~ ‘있다’를 가져와 하나의 음성으로 합성한다. 미리 수집해놓은 음편DB와 입력된 문장이 유사할수록 원음에 가까운 좋은 음질이 만들어지는 장점이 있지만, DB구축을 위해 튜닝, 녹음비용이 많이 들고, 음편 접합부에서 왜곡이 발생할 경우 귀에 거슬릴 정도로 소리가 튀기도 한다.

[1] https://github.com/Kyubyong/tacotron
[2] 기계학습 분야에서 말하는 일반적인 ‘학습’과는 같은 용어를 사용하나, 내용은 상이하다.



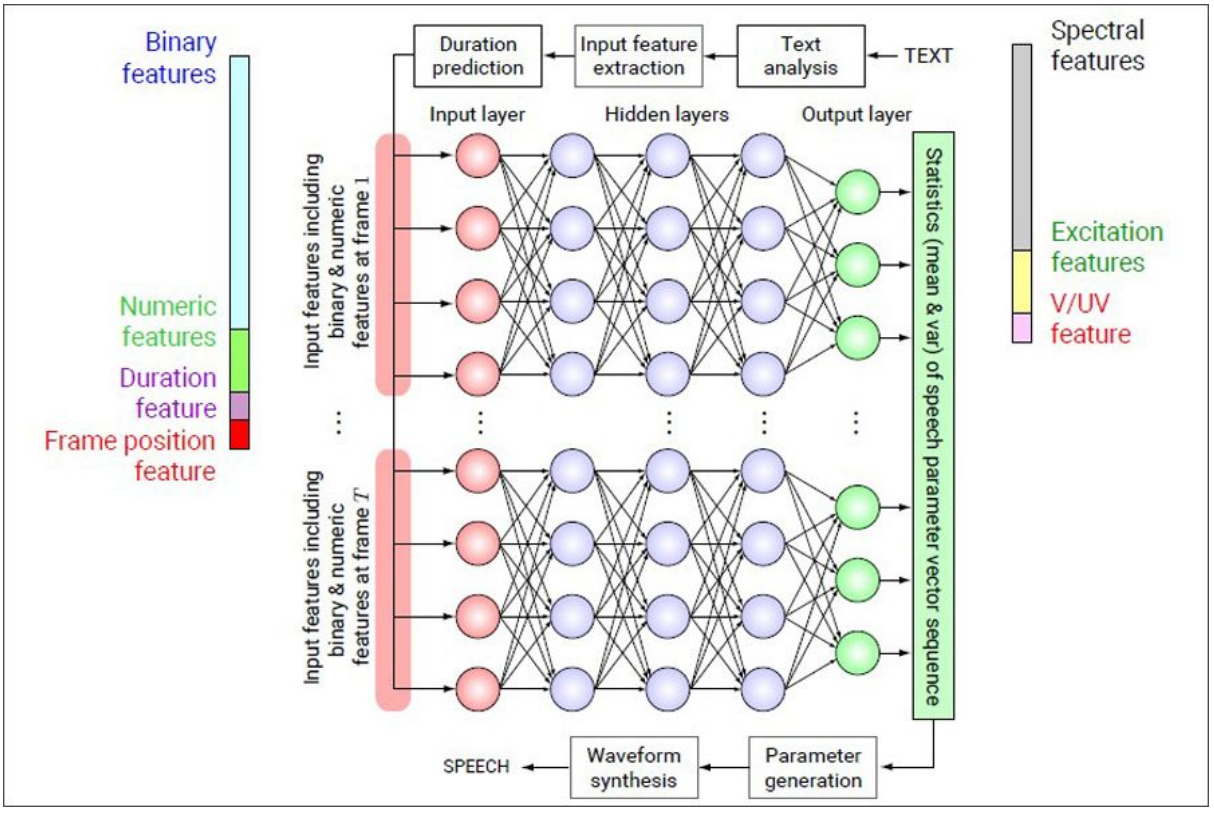
<그림2, 연결합성방식의 TTS개요>

- **통계모델 기반 TTS**: 2000년대 이후, 음성합성 영역에서 통계모델을 활용하는 움직임이 있었다. 은닉 마르코프 모델(Hidden Markov Model, HMM)이 그 대표적인 예이다. HMM은 음성신호의 주요 파라미터를 각각 음소, 삼음소별로 학습[3]한다. 그 후 음성 신호의 주요 파라미터를 생성하고, 생성된 파라미터를 이용해 최종 음성 신호를 생성한다. 음성 그 자체가 아닌, 음성신호의 산술통계학적 파라미터만을 저장하기 때문에 연결 합성 방식보다 규모의 측면에서 효율적이다. 또한, 화자 적응, 감정제어, 음색 변환 등에서도 더 우월한 성능을 보인다. 하지만, 음성 합성 과정에서의 스무딩 현상으로 인해 소리의 명료도(Intelligibility)가 떨어지는 단점이 있다.

하드웨어 기술의 발전과 더불어 급속도로 발전한 기계학습의 수혜를 입어, 음성 합성에서도 딥러닝을 사용한 기술이 개발되었다. 딥러닝을 사용한 음성 합성은 기본적으로 다음의 성질을 공유한다.

- 입력과 출력 구조로 되어있음.
- 음성신호에서 텍스트 전처리를 통해 언어적 특징벡터를 추출 → 입력
- 음성신호의 보코더(Vocoder)파라미터 → 출력
- 위의 입력-출력 셋을 가지고 학습.

3



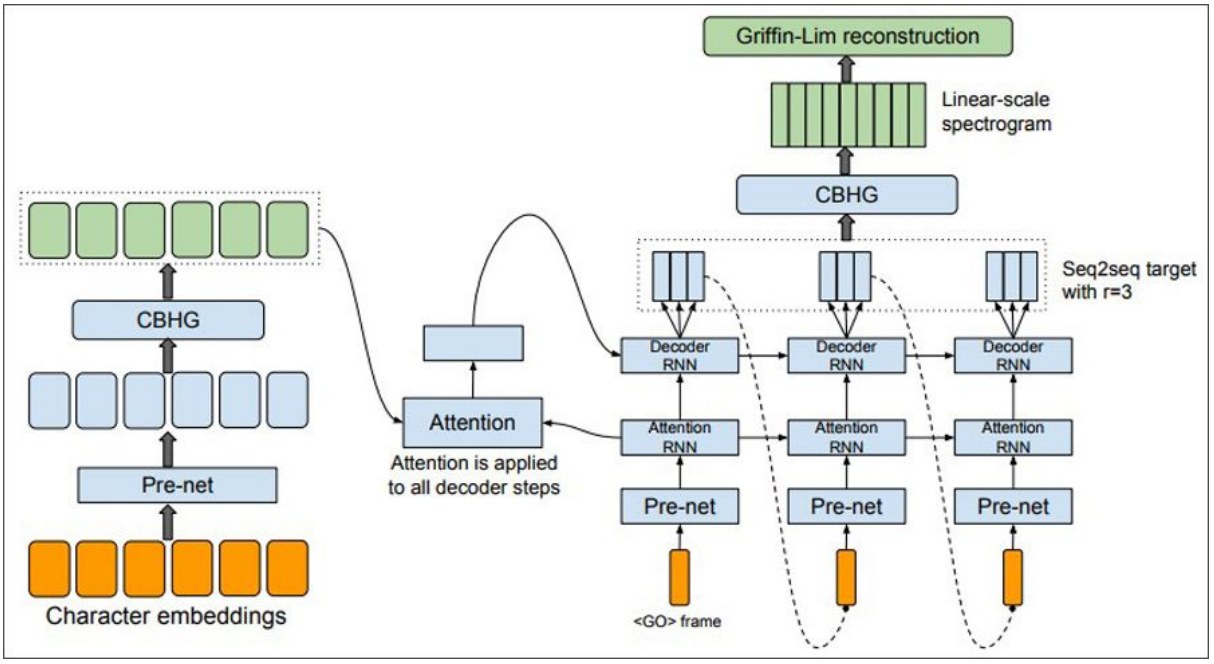
<그림3, 딥러닝 기반의 음성합성기술>

입력인 음성, 텍스트 속 해당 언어의 언어적 특징과, 그러한 언어적 특징으로부터 오는 음성신호의 보코더 파라미터들의 변화에 대한 비선형적 관계를 뉴런모델들이 자동으로 발견하고 학습하여 인공신경망을 구축한다. 이를 통해 고품질의 음성 합성을 구현할 수 있게 되었다.
기존 두 방식은 음성합성장치의 개발자가 프로그래밍 지식 뿐만 아니라 발성기관 구조 및 원리의 이해, 문자에서 발음표기호 변환하는 언어적 지식, 음성스펙트럼 분석 능력 등의 음성, 음운학적 지식이 필요했다. 이는 개발 진입장벽을 높이는 요소이기도 했다. 딥러닝 기반 방식은 이러한 배경지식 없이도 입출력 데이터만 충분히 많이 보유중이라면 음성합성장치의 개발이 가능하다.

일반적인 딥러닝 기반 음성합성기술에서 조금 더 발전한 방식이 Tacotron이다. 구글이 주력으로 사용하는 음성합성 학습 방법이다. 특징을 정리하면 다음과 같다.

- 문자열을 입력으로, 음성의 스펙트럼 특징벡터 열을 출력으로 지정한다.
- 문자열이 스펙트럼으로 변환되는 중간과정을 학습.
- 입력과 출력의 길이차가 발생하는데, 이를 Attention이라는 신경망 층(Layer)을 도입하여 입력과 출력 사이의 매핑(Mapping)관계를 학습한다.

텍스트 → 음성스펙트럼으로 변환되는 과정을 자동으로 습득하여, 텍스트 데이터를 말 그대로 음성으로 '변환'하는 방법을 학습하는 방식이다.



<그림4, 구글타코트론(Tacotro) 음성합성기술>

4.오픈소스 개선사항

기존의 오픈소스는 사용자의 텍스트 입력에 오탈자가 있을 경우, 이를 그대로 사용해 음성 데이터를 합성하기 때문에 출력 음성이 이상해지거나, 사용자의 의도와는 다른 결과물이 나올 수 있다. 이를 보완하기 위해, 사용자의 텍스트 입력으로 음성을 합성하기 전, 오탈자를 검사하고 자동으로 수정하도록 한다. 오탈자 검수는 python 라이브러리 'autocorrect'를 통해 검수된다. 이 과정을 통해 오탈자를 수정한 후 Tacotron 방식으로 학습을 하는 프로그램에 입력하면 그에 맞는 음성 파일이 출력된다.

4. 고찰

이번 프로젝트를 통해 머신 러닝에 대해 접해볼 수 있었다. 머신 러닝에 대한 기초 지식을 배울 수 있었으며, 이를 기반한 오픈소스로 제공된 프로젝트를 통해 명확한 이해와 오픈소스에 기여를 하는 방식으로 프로젝트에 참여했다.

프로젝트 처음에는 TTS 성능 향상에 초점을 맞춰 진행했지만, 충분한 학습과 Training을 위한 GPU 성능과 시간이 부족한 관계로 충분한 TTS Training을 통한 모듈 비교를 하지 못했고, 그로인해 잘못된 입력에 대한 오류 핸들링을 통해 사용성을 높여주는 방향으로 진행하게 되었다.

이후에 충분한 Training을 통해 여러 모듈을 비교하여 성능 향상을 직접 경험해보고 싶고, 지금 적용한 'autocorrect'에 RNN을 이용하여 문맥을 파악하여 문맥에 알맞는 단어를 선택 할 수 있도록 성능 개선하는 방향으로 진행하고 싶다.