Yapay Zeka ile Sesten Yazıya Dönüştürme Vize Raporu

Samet Duran

24.04.2024

1 Giriş

Bu projede insanların konuşmalarının metne dönüştürülerek bilgisayar ortamına aktarılması amaçlandı bu yüzden yapay zeka ile sesten yazıya çevirme konusu seçildi. Bu konuda ki örnekler araştırıldı. Python dili kullanılmasına karar verildi. Openai şirketinin açık kaynak kodlu Whisper adlı modeli incelendi ve araştırıldı. Yapay zeka alanında ki dikkate alma konusu üzerinde araştırma yapıldı, Whisper modelinin gerçek zamanlı nasıl çalıştırılıcağı araştırıldı ve örnek kod bulundu, çalıştırıldı, incelendi. DeepSpeech modeli için türkçe veri seti bulunarak eğitilmeye çalışıldı ama hata alındığı için başarısız olundu. RealtimeStt kuruldu ve çalıştırıldı. Proje sonunda konuşmayı yazıya çeviren bir uygulama ortaya çıkacak.

2 Metodoloji

OpenAI Whisper modeli, sesli konuşmayı metne dönüştürmek için tasarlanmış bir yapay zeka modelidir. Bu model, ses sinyallerini analiz ederek ve her bir sesin hangi kelimelere karşılık geldiğini tahmin ederek çalışır. Whisper modeli, Şekil 1'te görüldüğü gibi bir kodlayıcı ve bir kod çözücü olmak üzere iki ana bileşenden oluşur.

2.1 Kodlayıcı:

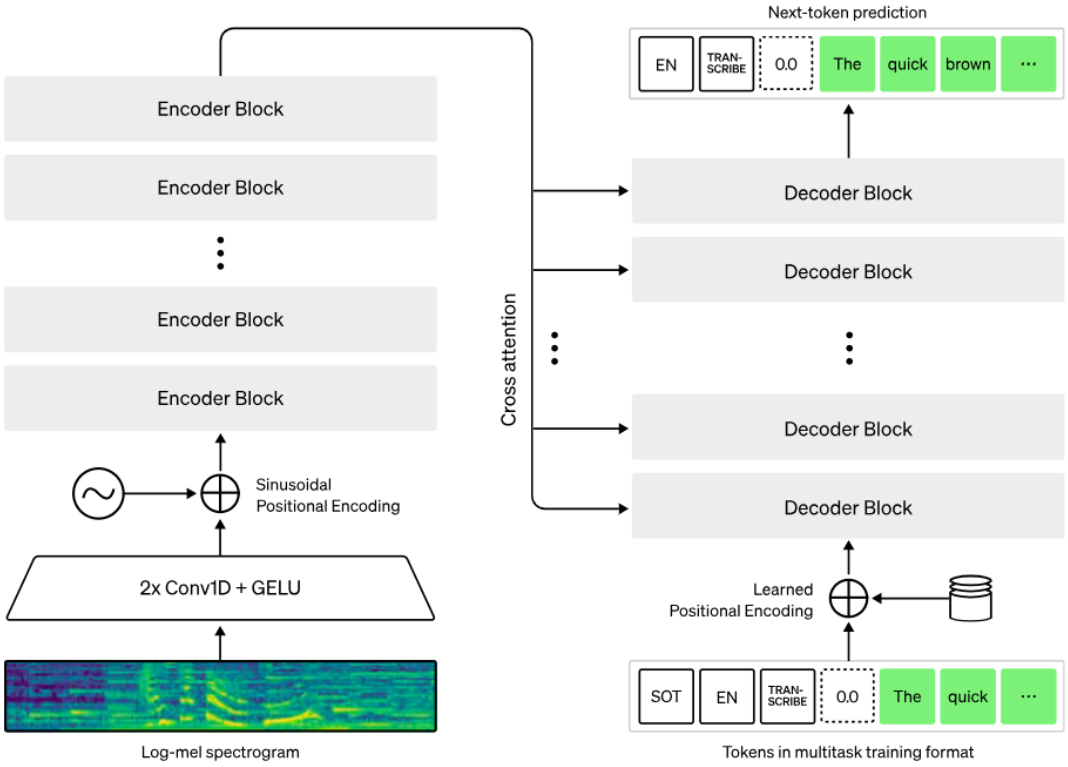
Kodlayıcı, ses sinyallerini alır ve bunları bir dizi sayıya dönüştürür. Bu sayılar, sinyallerin frekans ve genlik gibi özelliklerini temsil eder. Kodlayıcı, bir dizi sinir ağı katmanından oluşur ve her katman, sinyallerdeki daha karmaşık örüntüleri öğrenir.

2.2 Kod Çözücü:

Kod çözücü, kodlayıcı tarafından üretilen sayıları alır ve bunları metne dönüştürür. Kod çözücü, bir dizi sinir ağı katmanından oluşur ve her katman, sayı dizisinin hangi kelimelere karşılık geldiğini öğrenir.

2.3 Bloklar:

- Log-mel Spektrogram: Bu blok, ses sinyallerinin frekans ve genlik bilgilerini içeren bir görüntü oluşturur [14].
- 2x Conv1D + GELU(Gaussian Error Linear Unit): Bu blok, sinyallerdeki daha karmaşık örüntüleri öğrenir.
- Sinüsoidal Pozisyon Kodlama: Bu blok, ses sinyallerinin zamanla nasıl değiştiğini kodlar.
- Dikkate Alma: Bu blok, kodlayıcı tarafından üretilen sayıları ve kod çözücü tarafından üretilen kelimeleri ilişkilendirir.



Şekil 1: Whisper Çalışma Mantığı [3]

Dikkat Ağlarının Temeli: Dikkat ağları, bir girdinin belirli bölümlerine odaklanmak ve bu bölümlere dayalı bir çıktı üretmek için tasarlanmıştır. Bunu, "dikkat ağırlıkları" adı verilen bir dizi değer kullanarak gerçekleştirirler. Dikkat ağırlıkları, her bir girdi öğesinin önemini gösterir.

Çapraz Dikkat: Çapraz dikkat, birden fazla girdi dizisini işlemek için kullanılan bir dikkat mekanizmasıdır. Bu mekanizma, her bir girdi dizisindeki öğelerin diğer girdi dizileriyle nasıl ilişkili olduğunu öğrenir.

Çapraz Dikkatin Çalışma Şekli: Çapraz dikkat, her bir girdi dizisi için bir dizi dikkat ağırlığı hesaplayarak çalışır. Bu ağırlıklar, her bir girdi öğesinin diğer girdi dizilerindeki öğelerle ne kadar ilişkili olduğunu gösterir. Daha sonra, bu ağırlıklar, her bir girdi dizisinden gelen bilgileri birleştirmek için kullanılır.

Çapraz Dikkatin Uygulamaları: Çapraz dikkat, makine çevirisi, metin özetleme ve soru cevaplama gibi birçok NLP görevde kullanılır.

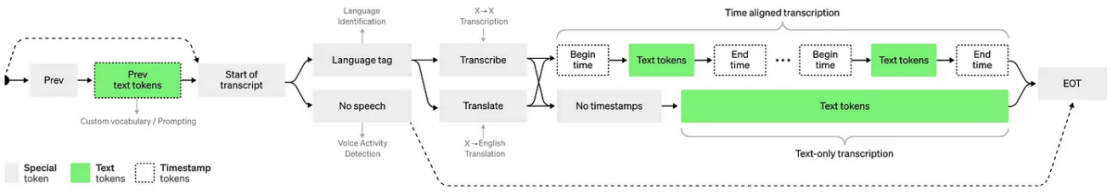
- Öğrenilmiş Pozisyon Kodlama: Bu blok, kodlayıcı tarafından üretilen sayı dizisinin zamanla nasıl değiştiğini kodlar.
- Sonraki Belirteç Tahmini: Bu blok, kodlayıcı tarafından üretilen sayılar ve kod çözücü tarafından üretilen kelimeler temelinde bir sonraki kelimenin ne olacağını tahmin eder.

2.4 Modelin Çalışma Aşamaları

- Ses sinyalleri kodlayıcıya girilir.
- Kodlayıcı, sinyalleri bir dizi sayıya dönüştürür.
- Kod çözücü, sayıları metne dönüştürür.

2.5 Whisper İş Akış Diyagramı

Aşağıda Şekil 2'de görüldüğü üzere Whisper modelinin iş akış diyagramı görülmektedir.



Şekil 2: Whisper İş Akış Diyagramı [3]

- Ses Girişi yapılır. Ses dalgaları dijital sinyallere dönüştürülür.
- Ses dalgalarında ses olup olmadığı belirlenir. Ses yoksa, işlem durur ve sonraki ses girişine kadar bekler.
- Dil Tanımlaması Yapılır.
- Aynı dilde çıktı oluşturulur veya isteğe bağlı çeviri yapılarak çıktı oluşturulur.
- Duruma göre zaman hizalı transkripsiyon veya sadece metin transkripsiyonu yapılır.

2.6 Dikkate Alma

Dikkate Alma [1]: Bu blok, kodlayıcı tarafından üretilen sayıları ve kod çözücü tarafından üretilen kelimeleri ilişkilendirir.
Dikkat Ağlarının Temeli: Dikkat ağları, bir girdinin belirli bölümlerine odaklanmak ve bu bölümlere dayalı bir çıktı üretmek için tasarlanmıştır. Bunu, "dikkat ağırlıkları" adı verilen bir dizi değer kullanarak gerçekleştirirler. Dikkat ağırlıkları, her bir girdi öğesinin önemini gösterir.

Çapraz Dikkat: Çapraz dikkat, birden fazla girdi dizisini işlemek için kullanılan bir dikkat mekanizmasıdır. Bu mekanizma, her bir girdi dizisindeki öğelerin diğer girdi dizileriyle nasıl ilişkili olduğunu öğrenir.

Çapraz Dikkatin Çalışma Şekli: Çapraz dikkat, her bir girdi dizisi için bir dizi dikkat ağırlığı hesaplayarak çalışır. Bu ağırlıklar, her bir girdi öğesinin diğer girdi dizilerindeki öğelerle ne kadar ilişkili olduğunu gösterir. Daha sonra, bu ağırlıklar, her bir girdi dizisinden gelen bilgileri birleştirmek için kullanılır.

Çapraz Dikkatin Uygulamaları: Çapraz dikkat, makine çevirisi, metin özetleme ve soru cevaplama gibi birçok NLP görevde kullanılır.

2.7 Whisper Modelini Çalıştırma

OpenAI'nin Whisper Konuşmadan Yazıya Dönüştürme konusunda geliştirilmiş modeldir.

2.7.1 Genel Bilgi:

Konuşmadan yazıya (STT) dönüştürme, sesli konuşmayı metne dönüştüren bir teknolojidir.

2.7.2 Whisper:

- Açık kaynaklı bir ASR modelidir.
- Şuanlık gerçek zamanlı şekilde çalışmıyor, ses dosyası yüklenerek çalıştırılabiliyor.
- Çok dilli bir konuşma tanıma sistemi olarak öne çıkar [2].
- 680.000 saatten fazla çok dilli ve webden toplanan veriyle eğitilmiştir [3].
- Bu büyük veri havuzu, benzersiz vurguları, arka plan gürültüsünü ve teknik jargonu daha iyi tanımasına olanak sağlar
- Whisper, hızlı ve doğru sonuçlar elde etmek için tasarlanmıştır.

```
import whisper

model = whisper.load_model("base")
result = model.transcribe("audio.mp3")
print(result["text"])
```

Şekil 3: Çalıştırma Kodu

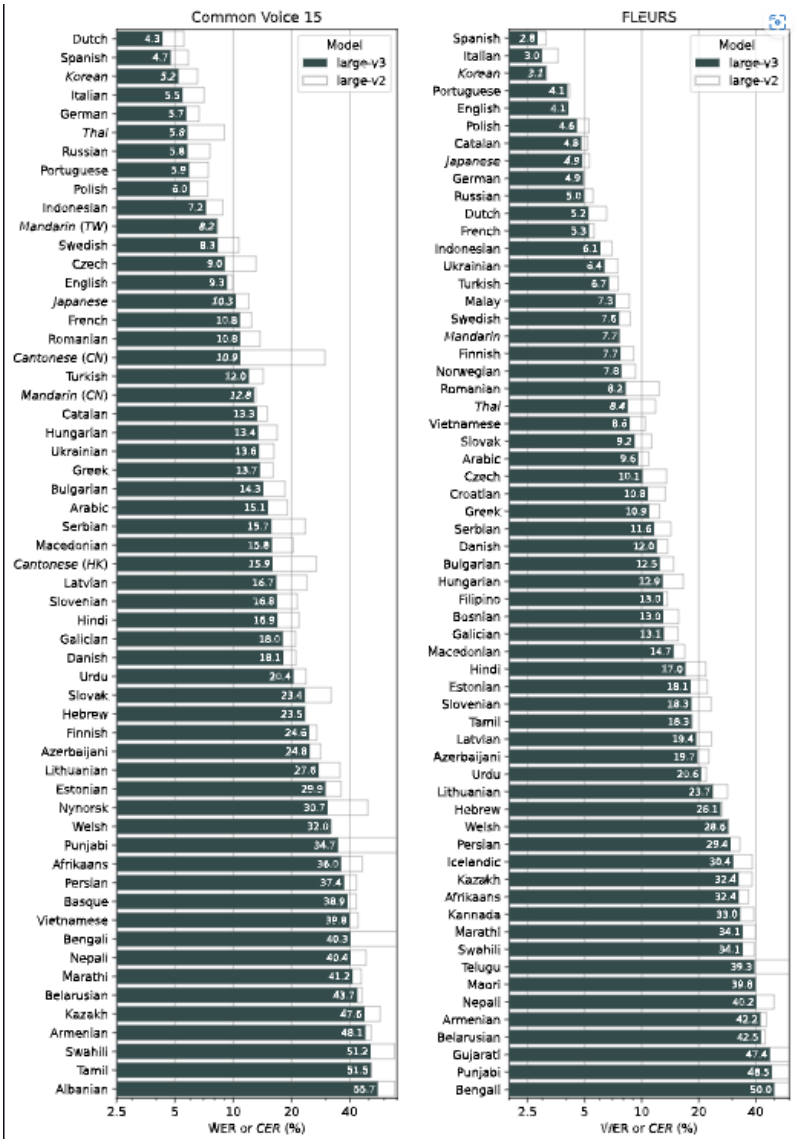
Yukarıda Şekil 3'de görüldüğü üzere whisper modelinin çalıştırmak için gerekli kod parçası örneği bulunmaktadır.

Aşağıda Şekil 4'de gösrüldüğü gibi whisper modelinin büyüklük ve hız karlışatırılması verilmiştir. Büyüklük arttıkça doğruluk artarken işlem hızı da yavaşlıyor.

Size	Parameters	English-only model	Multilingual model	Required VRAM	Relative speed
tiny	39 M	`tiny.en`	`tiny`	~1 GB	~32x
base	74 M	`base.en`	`base`	~1 GB	~16x
small	244 M	`small.en`	`small`	~2 GB	~6x
medium	769 M	`medium.en`	`medium`	~5 GB	~2x
large	1550 M	N/A	`large`	~10 GB	1x

Şekil 4: Büyüklük ve hız karşılaştırması [2]

Aşağıda Şekil 5'de görüldüğü gibi whisper modelinin large-v2 ve large-v3 büyüklüklerinin Common Voice 15 ve Fleurs veri kümelerinin dillere göre sözcük hata oranları verilmiştir. Değer azaldıkça hata oranı azalmaktadır.



Şekil 5: Sözcük hata oranı [2]

2.8 Whisper Modelini Gerçek Zamanlı Olarak Çalıştırma

Aşağıda Şekil 6'de Whisper modelini gerçek zamanlı olarak çalıştırmak için kullanılan kodun bir kısmı verilmiştir.

```python
import argparse
import os
import numpy as np
import speech_recognition as sr
import whisper
import torch

from datetime import datetime, timedelta
from queue import Queue
from time import sleep
from sys import platform


def main():
    parser = argparse.ArgumentParser() # Bir ArgümanParser nesnesi oluşturur, bu, komut satırı argümanlarını işlemek için kullanılacak.
    parser.add_argument("--model", default="small", help="Model to use",
                        choices=["tiny", "base", "small", "medium", "large"])
    parser.add_argument("--non_english", action='store_true',
                        help="Don't use the english model.")
    parser.add_argument("--energy_threshold", default=1000,
                        help="Energy level for mic to detect.", type=int)
    parser.add_argument("--record_timeout", default=2,
                        help="How real time the recording is in seconds.", type=float)
    parser.add_argument("--phrase_timeout", default=3,
                        help="How much empty space between recordings before we "
                             "consider it a new line in the transcription.", type=float)
    if 'linux' in platform:
        parser.add_argument("--default_microphone", default='pulse',
                            help="Default microphone name for SpeechRecognition. "
                                 "Run this with 'list' to view available Microphones.", type=str)
    args = parser.parse_args() # Komut satırı argümanlarını analiz eder ve bunları args adlı bir nesneye dönüştürür.

    # Kuyruktan bir kayıt alındığı son zaman.
    phrase_time = None # Bu, son ses parçasının zamanını tutmak için bir değişken tanımlar.

    # İş parçacığı kaydı geri aramasından veri aktarmak için güvenli iş parçacığı kuyruğu.
    data_queue = Queue() # Bu, ses verilerini geçici olarak depolamak için bir kuyruk oluşturur.

    # Konuşma sona erdiğinde algılayabilen güzel bir özelliği olduğu için Ses Tanıyıcı'yı sesimizi kaydetmek için kullanıyoruz.
    recorder = sr.Recognizer() #Bu, SpeechRecognition kütüphanesinden bir Tanıyıcı nesnesi oluşturur. Bu, sesi metne dönüştürmek için kullanılacak.
    recorder.energy_threshold = args.energy_threshold # Bu, mikrofonun ne kadar gürültü algılayacağını belirler. Bu, ses kaydının ne zaman başlayıp duracağını etkiler.

    # Kesinlikle bunu yapın; dinamik enerji telafisi, enerji eşiğini, SpeechRecognizer'ın kaydı asla durdurmadığı bir noktaya kadar önemli ölçüde düşürür.
    recorder.dynamic_energy_threshold = False # Bu, dinamik enerji eşiğini kapatır. Bu, enerji eşiğinin otomatik olarak ayarlanmasını engeller.
```

Şekil 6: Whisper Gerçek Zamanlı Örnek Kod Parçası [4]

2.9 RealtimeSTT

RealtimeSTT, gerçek zamanlı uygulamalar için kullanımı kolay, düşük gecikmeli konuşmadan metne kütüphanesidir [8]. Kullanılan bileşenler:

2.9.1 Ses Etkinliği Algılama

İlk ses etkinliği tespiti için WebRTCVAD [9]. Daha doğru doğrulama için SileroVAD [10].

2.9.2 Konuşmadan Metne

Anında (GPU hızlandırmalı) transkripsiyon için Faster-Whisper [11].

2.9.3 Uyanık Kelime Algılama

Uyandırma kelimesi algılama için Porcupine [12].

RealtimeStt kuruldu,çalıştırıldı ve kodları incelendi. Şekil 7 de ornek kod parçası bulunmaktadır.

```
recorder_config = {
    'spinner': False,
    'model': 'small',
    'language': 'tr',
    'silero_sensitivity': 0.4,
    'webrtc_sensitivity': 2,
    'post_speech_silence_duration': 0.4,
    'min_length_of_recording': 0,
    'min_gap_between_recordings': 0,
    'enable_realtime_transcription': True,
    'realtime_processing_pause': 0.2,
    'realtime_model_type': 'tiny',
    'on_realtime_transcription_update': text_detected,
    #'on_realtime_transcription_stabilized': text_detected,
}
```

Şekil 7: RealtimeSTT örnek kod parçası [13]

2.10 Sesten Yazıya Çevirmenin Metodolojisi

3 VeriTabanı ve Veriler

DeepSpeech modelini değiştmek için kendi veri tabanı sitesinden alınan Türkçe veri seti kullanılacak.

Whisper dışında Mevcut diğer yaklaşımlar sıklıkla daha küçük, daha yakın eşleştirilmiş ses-metin eğitim veri kümeleri kullanmakta, [15], [16], [17] veya geniş ancak denetimsiz ses ön eğitimi kullanmaktadır. Whisper büyük ve çeşitli bir veri kümesi üzerinde eğitildiği ve herhangi bir veri kümesine göre ince ayar yapılmadığı için, konuşma tanıma alanında rekabetçi bir ölçüt olarak bilinen LibriSpeech performansında uzmanlaşmış modelleri geçememektedir [18], [19]. Bununla birlikte, Whisper'ın sıfır atış performansını birçok farklı veri kümesinde ölçtüğümüzde, çok daha sağlam olduğunu ve bu modellerden %50 daha az hata yaptığını görüyoruz. Whisper'ın ses veri kümesinin yaklaşık üçte biri İngilizce değildir ve dönüşümlü olarak orijinal dilde yazıya dökme veya İngilizceye çevirme görevi verilir. Bu yaklaşımın özellikle konuşmadan metne çeviriyi öğrenmede etkili olduğunu ve CoVoST2'deki denetimli SOTA'dan İngilizce çeviriye sıfır atışta daha iyi performans gösterdiğini görüyoruz.

4 Bulgu ve Tartışma

4.1 DeepSpeech

DeepSpeech modelini eğitmek için Türkçe veri seti bulundu ve indirildi. İlk önce PyCharm kullanarak [5] ilgili makalede ki kodlar kullanılarak eğitilmeye çalışıdı fakat başarısız olundu. Daha sonra DeepSpeech dökümantasyonu [6] ve örnek DeepSpeech eğitim colab dosyasına [7] bakılarak colab üzerinden eğitilmeye çalışıldı fakat Şekil 8 de bulunan hata alındı ve başarısız olundu.

```
%cd /content/DeepSpeech/
! bin/import_cv2.py ../tr/cv-corpus-17.0-2024-03-15/tr
```

```
/content/DeepSpeech
Traceback (most recent call last):
  File "/content/DeepSpeech/bin/import_cv2.py", line 18, in <module>
    from deepspeech_training.util.downloader import SIMPLE_BAR
ModuleNotFoundError: No module named 'deepspeech_training'
```

Şekil 8: DeepSpeech modeli eğitilmeye çalışırken alınan hata

Kaynakça

[1] G. Kalra, "Attention networks: A simple way to understand cross-attention." https://medium.com/@geetkal67/attention-networks-a-simple-way-to-understand-cross-attention-3b396266d82e, Jul 18, 2022.

[2] openai, "Whisper," Nov 17, 2023.

[3] openai, "Research introducing whisper," September 21, 2022.

[4] davabase, "Whisper real time." https://github.com/davabase/whisper_real_time, Mar 14, 2024.

[5] W. Mousa, "Building a speech to text with ai correction system: A step by step tutorial using deepspeech in python." https://medium.com/@waleedmousa975/building-a-speech-to-text-with-ai-correction-system-a-step-by-step-tutorial-using-deep Mar 20, 2023.

[6] Mozilla, "Training your own model." https://deepspeech.readthedocs.io/en/r0.9/TRAINING.html, 2020.

[7] abbycabs, "Training a dutch speech-to-text model." https://colab.research.google.com/github/acabunoc/Tutorial-train-dutch-model/blob/master/DeepSpeech_train_a_model%2C_CV_Dutch.ipynb, Jul 12, 2020.

[8] K. Beigel, "Realtimestt." https://github.com/KoljaB/RealtimeSTT, April 14, 2024.

[9] wiseman, "py-webrtcvad." https://github.com/wiseman/py-webrtcvad, Feb 15, 2021.

[10] snakers4, "Silero vad." https://github.com/snakers4/silero-vad, Apr 2, 2024.

[11] otakutyrant, "Faster whisper transcription with ctranslate2." https://github.com/SYSTRAN/faster-whisper, Apr 2, 2024.

[12] ErisMik, "Porcupine." https://github.com/Picovoice/porcupine, Apr 15, 2024.

[13] K. Beigel, "realtimestt-test." https://github.com/KoljaB/RealtimeSTT/blob/master/tests/realtimestt_test.py, April 14, 2024.

[14] L. Roberts, "Understanding the mel spectrogramr," Mar 6, 2020.

[15] W. Chan, D. S. Park, C. A. Lee, Y. Zhang, Q. V. Le, and M. Norouzi, "Speechstew: Simply mix all available speech recognition data to train one large neural network," *CoRR*, vol. abs/2104.02133, 2021.

[16] D. Galvez, G. Diamos, J. Ciro, J. F. Cerón, K. Achorn, A. Gopi, D. Kanter, M. Lam, M. Mazumder, and V. J. Reddi, "The people's speech: A large-scale diverse english speech recognition dataset for commercial usage," *CoRR*, vol. abs/2111.09344, 2021.

[17] G. Chen, S. Chai, G. Wang, J. Du, W. Zhang, C. Weng, D. Su, D. Povey, J. Trmal, J. Zhang, M. Jin, S. Khudanpur, S. Watanabe, S. Zhao, W. Zou, X. Li, X. Yao, Y. Wang, Y. Wang, Z. You, and Z. Yan, "Gigaspeech: An evolving, multi-domain ASR corpus with 10, 000 hours of transcribed audio," *CoRR*, vol. abs/2106.06909, 2021.

[18] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *CoRR*, vol. abs/2006.11477, 2020.

[19] Y. Zhang, D. S. Park, W. Han, J. Qin, A. Gulati, J. Shor, A. Jansen, Y. Xu, Y. Huang, S. Wang, Z. Zhou, B. Li, M. Ma, W. Chan, J. Yu, Y. Wang, L. Cao, K. C. Sim, B. Ramabhadran, T. N. Sainath, F. Beaufays, Z. Chen, Q. V. Le, C.-C. Chiu, R. Pang, and Y. Wu, "Bigssl: Exploring the frontier of large-scale semi-supervised learning for automatic speech recognition," *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, p. 1519–1532, Oct. 2022.