ARABIC SPEECH-TO-TEXT

Authors : Ahmed Gamal Ahmed ,Motaz Saied Saleh ,Myar Hany Mohammed, Mona Badr Mostafa

Affiliation: Alexandria University, Alexandria, Egypt

Abstract :

This paper presents a comprehensive methodology for fine-tuning high-quality Arabic Text-to-Speech (TTS) models, with a focus on Egyptian Arabic while supporting multiple dialects. Leveraging data sourced from YouTube, we detail the workflow encompassing data preprocessing, transcription, diacritic restoration, and phoneme alignment. Advanced tools such as Spleeter, noisereduce, and OpenAI Whisper were employed to clean, segment, and transcribe audio data. The text normalization process included diacritic restoration using deep learning models and phoneme alignment via transliteration. The fine-tuning of Tacotron2 and FastPitch models on the curated dataset yielded significant improvements in naturalness and speech clarity, with FastPitch achieving the lowest Mel Cepstral Distortion (MCD) score of 28.39 after 40 epochs. These findings underscore the potential of leveraging web-based data and state-of-the-art TTS models to advance Arabic speech synthesis.

I. INTRODUCTION

Arabic Text-to-Speech (TTS) systems have long grappled with the linguistic complexities of the Arabic language. These challenges stem from its rich morphology, diacritic-dependent pronunciation, and the presence of numerous dialectal variations. While classical Arabic remains the standard for formal communication, regional dialects dominate everyday interactions, necessitating models that can handle diverse linguistic inputs.

This research aims to address these challenges by fine-tuning pretrained TTS models using a dataset curated from YouTube content. By leveraging openly available tools for data processing and transcription, alongside advanced models such as Tacotron2 and FastPitch, we explore methods to enhance the naturalness, clarity, and expressiveness of Arabic TTS outputs. Special emphasis is placed on Egyptian Arabic due to its widespread usage and rich phonetic characteristics, though the methodology supports adaptation to other dialects. This study also examines metrics such as Mel Cepstral Distortion (MCD) to evaluate model performance, setting a benchmark for Arabic TTS development.

II. DATA COLLECTION AND PREPROCESSING

Data Source:

- YouTube videos were scraped using yt_dlp, an advanced tool for downloading video and audio content efficiently.
- A set of relevant Arabic YouTube channels was identified, focusing on diverse content to capture multiple dialects and tones.
- Metadata, including video titles, descriptions, and durations, were collected and analyzed to filter suitable content.

- Keywords used to extract data included:
    - Group 1: سياسة مصرية, سياسة عربية, محادثات مصرية, تعليم اللغة العربية, تعليم اللغة المصرية, برامج ترفيه مصرية, برامج وثائقية, ثقافة مصرية, تاريخ مصر, قصص عربية قصيرة, قواعد اللغة العربية, طعام مصري, أدب مصري, معالم سياحية في مصر, أحداث سياسية مصرية, مبادرات تنمية مصرية, النحو والصرف, الفصحى والعامية, تاريخ اللغة العربية, معاني المفردات العربية, تقاليد مصرية, قصص مصرية قصيرة, النطق الصحيح في اللغة العربية, اللهجة العامية المصرية نوتة, إعلام مصري.
    - Group 2: نشرات أخبار مصرية, خطب سياسية عربية, تصريحات رسمية, مؤتمرات صحفية عربية, تحليل سياسي عربي, دروس تعليم اللغة العربية, شرح قواعد النحو, تعليم النطق الصحيح, قصص تعليمية للأطفال بالعربية, دروس علمية مصرية, برامج حوارية عربية, مقاطع مسرحيات مصرية, مقابلات تلفزيونية عربية, قصص عربية مسموعة, وثائقيات عن الحضارة المصرية, تجارب اجتماعية بالعربية, كلمات ملهمة وخطب تحفيزية, تسجيلات طلابية عربية, مؤتمرات عربية تعليمية, قراءة شعرية عربية, تسجيلات نادرة لخطب سياسية, برامج عن تاريخ العرب, مقابلات مع شخصيات تاريخية, تسجيلات إذاعية قديمة, برامج إذاعية قديمة.
    - Group 3: فن شعبي مصري, أغاني تراثية مصرية, رقصات مصرية تقليدية, أسواق مصرية قديمة, عمارة مصرية تاريخية, عادات وتقاليد الزواج في مصر, السينما المصرية الكلاسيكية, أمثال شعبية مصرية, الحرف اليدوية المصرية, الآلات الموسيقية التقليدية المصرية, الثقافة البدوية في مصر, العائلة في المجتمع المصري, الاحتفالات الوطنية المصرية, أزياء مصرية تقليدية, حكايات شعبية مصرية, ألعاب الأطفال التراثية في مصر, الديكور الداخلي المصري التقليدي, الزراعة في الريف المصري, الحياة اليومية في القاهرة القديمة, الطبيعة في جنوب مصر, الأساطير الفرعونية المصرية, المسلسلات الرمضانية المصرية, الدراما الاجتماعية المصرية, الأكلات الشعبية المصرية في المناسبات, الخط العربي في الفن المصري

II. DATA COLLECTION AND PREPROCESSING

Audio Processing:

- pydub: This Python library was utilized for handling audio files, specifically for slicing, trimming, and format conversion:
    - Slicing and Trimming: Long audio clips were divided into smaller, manageable segments, removing unnecessary parts such as long silences or irrelevant speech.
    - Format Conversion: Audio files were converted into a standardized format (e.g., WAV) suitable for processing by subsequent tools.
- Spleeter: A powerful tool for source separation, Spleeter was used to isolate vocals from background music:
    - Vocal Isolation: By separating the vocal track from instrumental or noise components, Spleeter ensured the dataset primarily contained clean speech audio.

    - Signal Enhancement: The isolated vocal track was then processed to retain only the speech content while discarding residual background interference.
- noise reduce: This library was employed to apply noise reduction techniques:
    - Noise Profile Estimation: A noise profile was generated from silent or low-activity segments in the audio.
    - Noise Suppression: The identified noise was subtracted from the audio, resulting in cleaner and more comprehensible speech segments.
- audioop: A low-level audio processing module in Python, used for volume adjustments:
    - Volume Normalization: Audio clips were adjusted to achieve consistent loudness levels across the dataset.
    - Peak Limiting: Ensured no audio clip exceeded a specific loudness threshold, maintaining clarity and avoiding distortion.

A. Data Source

- **YouTube** was selected as the primary data source due to its vast repository of diverse Arabic speech content. Using the **yt_dlp** tool:
    - Videos from Arabic-speaking creators were scraped.
    - Metadata such as titles, descriptions, and durations were analyzed to ensure content relevance.
    - A final selection of channels spanned topics such as news, entertainment, education, and podcasts to capture varied tones and dialects.

B. Audio Processing

The audio processing pipeline was designed to ensure high-quality input for transcription and TTS model training:

- **Slicing and Trimming:**
    - Long audio clips were segmented into shorter, manageable durations, typically 5-10 seconds.
    - Silences, overlapping speech, and irrelevant segments were identified and removed.
- **Format Conversion:**
    - All audio files were standardized to **WAV format** for compatibility.

- **Vocal Isolation:**
    - **Spleeter** was utilized to separate vocal tracks from background music.
        - **Overview:**
            - Spleeter is an open-source deep learning tool developed by Deezer for source separation. It uses pre-trained models to isolate vocal components from music tracks efficiently.
            - By employing convolutional neural networks (CNNs), it separates audio into multiple stems (e.g., vocals, instruments).
        - **Process:**
            1. Input audio files were processed to generate vocal and instrumental stems.
            2. Only the isolated vocal stem was retained for further preprocessing.
            3. Residual noise in the vocal track was minimized to ensure clarity.
- **Noise Reduction:**
    - Using the **noise reduce** library:
        - **Overview:**
            - noise reduce is a Python library designed for adaptive noise reduction using spectral subtraction.
            - It applies machine learning techniques to identify noise profiles and subtract them from audio signals.
        - **Process:**
            1. Noise profiles were generated by analyzing silent or low-activity segments in audio files.
            2. Filters were iteratively applied to suppress noise while preserving the integrity of speech.
            3. Output files were reviewed to ensure minimal distortion and improved intelligibility.

- **Volume Normalization:**
    - **audioop** adjusted all clips to a uniform loudness level of approximately -23 LUFS, suitable for speech synthesis.
        - **Overview:**
            - audioop is a low-level Python module for audio processing tasks, including volume normalization and peak limiting.
            - It operates directly on raw audio data, making it efficient for large datasets.
        - **Process:**
            1. Volume levels across all clips were analyzed.
            2. Adjustments ensured consistent loudness while maintaining audio quality.
- **Resampling:**
    - All audio clips were resampled to a consistent **22050 Hz** to match the requirements of TTS models.

C. Transcription

- OpenAI Whisper: Whisper is an advanced, multilingual transcription model developed by OpenAI. It leverages state-of-the-art deep learning techniques to convert spoken language into text.
    - Process:
        1. Audio Segmentation: Each audio clip was segmented into manageable durations to enhance transcription accuracy.
        2. Model Processing: Whisper processed each segment to generate preliminary transcriptions.
        3. Language Detection: Whisper's automatic language detection confirmed the language of the spoken content, ensuring accurate Arabic transcription.
    - Accuracy Review:

- The transcription output was reviewed manually to identify and correct errors, including misinterpretation of words, incorrect punctuation, or missed segments.
- Low-quality transcriptions—such as those with significant background noise or unclear speech—were discarded to maintain dataset quality.

D. Data Cleaning:

- Duplicate transcriptions and non-Arabic text entries were removed.
- Special characters and emojis were filtered out.
- A final dataset of 244 high-quality WAV files paired with accurate transcriptions was created.

E. Mel Spectrogram Extraction:

- Mel spectrograms were extracted as required for training:
  - **Overview:** Mel spectrograms represent audio signals as a time-frequency distribution, emphasizing perceptually relevant frequencies.
  - **Process:**
    1. A Python script, extract_f0.py, was executed to compute mel spectrograms for all audio files.
    2. These spectrograms were used as input features for TTS models, ensuring accurate alignment between text and speec

F. Text Normalization:

- Shakkelha Model: Employed for diacritic restoration to ensure proper pronunciation.
  - Methods: This model uses deep learning techniques to analyze the grammatical structure and restore missing diacritics. The steps include:
    1. Morphological analysis: Determines the grammatical structure of words.
    2. Contextual learning: Applies context-based rules to predict accurate diacritics.
    3. Validation: Cross-checks predicted diacritics against known rules to ensure correctness.
- Diacritics were added systematically across all transcriptions.

G. Transliteration:

- Buckwalter Transliteration: Used to convert Arabic script into phoneme-aligned text.
    - Methods: Buckwalter transliteration replaces Arabic characters with Latin script representations:
        1. Character mapping: Converts each Arabic letter to its predefined Latin equivalent.
        2. Contextual alignment: Aligns transliteration with phoneme representations required by TTS models.
        3. Validation: Ensures the phoneme alignment matches pronunciation patterns for Arabic.
    - This step ensured better alignment with TTS model requirements.

III. MODELS AND ARCHITECTURES

Tacotron2:

- Architecture: Tacotron2 is a sequence-to-sequence model with an attention mechanism. It maps input text to a mel-spectrogram representation. The architecture consists of:
    - Encoder: Processes the input text sequence into a fixed-length vector.
    - Attention: Aligns the input and output sequences.
    - Decoder: Converts the aligned representation into mel-spectrogram frames.
    - Post-Processing: A WaveNet vocoder generates audio from the mel-spectrogram.
- How to Use:
    1. Prepare training data as aligned text-audio pairs.
    2. Use a GPU-enabled training environment for efficiency.
    3. Train the model for a specified number of epochs, monitoring loss metrics.
    4. Use the trained model to synthesize speech by providing input text.
- Performance:
    - Training Loss: 2.9477
    - Validation Loss: 2.5164

- Observations: Poor naturalness and clarity.

FastPitch:

- Architecture: FastPitch is a pitch-aware sequence-to-sequence model. It introduces explicit pitch prediction to improve speech expressiveness and naturalness.
  - Encoder: Encodes text input.
  - Pitch Predictor: Models pitch contours to capture prosodic features.
  - Decoder: Synthesizes mel-spectrograms using both text and pitch representations.
- How to Use:
  1. Prepare text-audio pairs and ensure pitch annotations are included.
  2. Train the model with hyperparameter tuning for optimal performance.
  3. Validate using pitch-accurate metrics.
  4. Use the trained model to synthesize text with controlled pitch variation.
- Performance:
  - 12 Epochs:
    - Training Loss: 11.0213
    - Gradient Norm: 17.0301
    - Observations: Moderate naturalness and clarity.
  - 40 Epochs:
    - Training Loss: 8.0657
    - Gradient Norm: 12.8098
    - Observations: Best naturalness and clarity.

Mel Cepstral Distortion (MCD):

MCD measures the difference between two waveforms, quantifying the similarity between generated and original speech. A lower MCD value indicates higher quality synthesis.

- Tacotron2 Model MCD: 43.208201637997256
- FastPitch Model (12 Epochs) MCD: 34.190841844522296

- FastPitch Model (40 Epochs) MCD: 28.387458742018264

These results highlight that FastPitch with 40 epochs achieves the lowest MCD, indicating the closest match to the original audio.

IV. TRAINING AND FINE-TUNING

Fine-tuning the pretrained models required systematic adjustments to optimize their performance for Arabic speech synthesis. Key aspects included:

- Dataset Preparation: The curated dataset underwent additional preprocessing to ensure optimal alignment of audio-text pairs. Silence trimming and audio normalization techniques were applied to remove discrepancies. Specifically:
- Silence trimming: Ensured that extraneous silence before or after speech was eliminated using tools like sox.
    - Audio normalization: Standardized volume levels across all samples to avoid discrepancies during model training.
    - Hyperparameter Optimization:
    - Learning Rate: Adjusted dynamically using a learning rate scheduler to prevent overfitting or underfitting.
    - Batch Size: Experimented with different batch sizes to balance memory constraints and model performance.
    - Epochs: Extended training for FastPitch to 40 epochs based on observed performance improvements.

Model-Specific Fine-Tuning:

Tacotron2: Attention parameters were carefully tuned to improve alignment stability during synthesis.

FastPitch: Added pitch contour adjustments and enabled fine-grained control over prosody during training.

Regularization Techniques:

- Dropout layers: Applied to prevent overfitting.
- Weight decay: Introduced to regularize model weights and improve generalization.
- Evaluation: Intermediate checkpoints were saved, and their performance was evaluated using MCD and subjective listening tests to select the best-performing model.

IV. RESULTS AND CONCLUSION:

Results:

The study evaluated the performance of Tacotron2 and FastPitch models based on Mel Cepstral Distortion (MCD), training loss, and subjective assessments of speech naturalness and clarity. Key findings include:

- Tacotron2:
    - Higher MCD values, indicating less accurate synthesis.
    - Moderate naturalness but lacked expressiveness and clarity.
- FastPitch:
    - At 12 epochs: Improved clarity with MCD of 34.19, though some pitch inconsistencies were noted.
    - At 40 epochs: Achieved the best naturalness and clarity, with an MCD of 28.39, closely matching original speech.

Conclusion:

This work demonstrates the feasibility of fine-tuning Arabic TTS models using publicly available YouTube data. A total of **360 videos** were scraped, leading to **346 unique entries** after duplicate removal. The curated dataset of 244 transcribed WAV files provided a robust foundation for training. The results highlight the importance of a robust preprocessing pipeline and the advantages of advanced architectures like FastPitch in achieving high-quality speech synthesis. While this study focused on Egyptian Arabic, the methodology can be adapted to other dialects and languages. Future work will explore larger datasets, additional dialects, and enhancements in prosody modeling to further improve TTS quality and applicability.

REFERENCES:

1. [1] Google Speech Commands Dataset
2. [2] OpenAI Whisper Documentation
3. [3] Tacotron2 and FastPitch Research Papers
4. https://github.com/AhmedGamal7207/tts-arabic-pytorch/tree/master
5. https://www.kaggle.com/code/ahmedgamal7207/speech-recognition-final-project-phase-1#Getting-Transcriptions

Keywords: Arabic TTS, Tacotron2, FastPitch, Fine-Tuning, Speech Synthesis, NLP, Deep Learning