TEXT-TO-SPEECH WITH PROMINENCE CONTROL
A COMPARATIVE STUDY OF THREE TEXT-TO-SPEECH APPROACHES TO IMPROVE PROSODY

DT2112 SPEECH TECHNOLOGY

Santiago Roman, Oliver Möller

KTH Royal Institute of Technology

March 2022

ABSTRACT

This project compares three distinct text-to-speech (TTS) approaches, aiming to assess their effectiveness in producing natural and intelligible speech. The first method involves a baseline TTS system called SpeechT5, which functions as a complete encoder-decoder model. The second method extracts attention from an encoder model (Bert), utilizes the attention as normalized scores, modifies the SSML Prosody tag based on these scores, and synthesizes speech. The third method employs PyReaper to obtain pitch information from audio, Wave2Vec to acquire word timestamps, and matches high pitches with corresponding word timestamps to modify the SSML Prosody tag and synthesize speech. To evaluate the performance of each approach, a Mean Opinion Score (MOS) test will be conducted, with ten volunteers rating the quality of synthesized speech for five different phrases. The goal is to determine which of the three methods yields the highest score and user satisfaction. The paper will present the general processes, results, attention heatmaps, and sample audio files generated during the study, providing valuable insights into the strengths and weaknesses of each TTS approach.

1 Introduction

Text-to-speech (TTS) systems have been widely adopted in various applications, ranging from virtual assistants and audiobooks to language learning tools, providing users with an auditory representation of written text. Despite their prevalence, the speech output generated by these systems often lacks the natural prosody found in human speech, resulting in a less engaging and less intelligible user experience. In this project, we aim to enhance the prosody of speech output produced by TTS systems by adjusting the emphasis and prominence of individual words within a sentence.

To achieve this objective, we explore different methods to manipulate prosody, such as employing a transformer model to generate attention weights for each word in the input sentence and modifying the prosody based on these attention scores. Another approach involves extracting pitch information to group words in a phrase and separately adjusting the prosody for each group. In an effort to further improve the speech output's prosody, we will fine-tune the probabilistic weight using pitch and other acoustic features extracted from audio files. This adjustment aims to better match the emphasis and meaning of the words in the sentence. Subsequently, we will apply the optimized probabilistic weight to each word in the sentence to modify the overall prosody.

Lastly, we will synthesize speech output from the modified sentence and assess the quality of the prosody through a listening test involving multiple participants. This project seeks to demonstrate that our approach can substantially improve the naturalness and expressiveness of TTS systems, ultimately enhancing user experience across a diverse range of applications. By addressing the challenges in achieving natural prosody, our research contributes valuable insights to the ongoing development of more sophisticated and user-friendly TTS systems.

2 Background

2.1 Transformers

The transformer model was introduced in the paper "Attention is All You Need" by Vaswani et al. [11]. This architecture revolutionized natural language processing by utilizing self-attention mechanisms to process input sequences in parallel rather than sequentially, as traditional recurrent neural networks (RNNs) and long short-term memory (LSTM) networks do. Transformers are highly scalable, capable of handling long-range dependencies, and have become the foundation for numerous state-of-the-art models in various NLP tasks.

2.2 Encoder Models

BERT is one of the most notable encoder models based on transformers (Bidirectional Encoder Representations from Transformers). Developed by Devlin et al. [5], BERT is a pre-trained model designed for a wide range of NLP tasks, including text classification, named entity recognition, and sentiment analysis. BERT's bidirectional architecture allows it to capture context from both left and right sides of a token, resulting in a richer understanding of the input text.

2.3 Encoder-Decoder Models

T5 (Text-to-Text Transfer Transformer) is a prominent example of an encoder-decoder model based on transformers. Introduced by Raffel et al. [10], T5 is designed to handle various NLP tasks by converting them into a text-to-text format. The model consists of an encoder that processes the input text and a decoder that generates the output text, with both components using the transformer architecture.

2.4 TTS

Text-to-Speech (TTS) systems are designed to convert written text into spoken language, mimicking human speech's natural patterns and expressiveness. Over the years, TTS technology has evolved significantly, from rule-based concatenative synthesis to more advanced neural approaches. Modern TTS systems, like SpeechT5, leverage state-of-the-art machine learning architectures, such as transformer-based encoder-decoder models, to generate high-quality, natural-sounding speech from text input. This TTS system is a key component of our research, serving as the baseline for comparison with other TTS approaches.

The input text is first tokenized and preprocessed in the TTS pipeline to handle punctuation, abbreviations, and other linguistic elements. Next, an acoustic model converts the processed text into spectrograms or other intermediate representations, capturing the nuances of human speech. Tacotron [13] is a prominent example of a text-to-spectrogram

system that employs a sequence-to-sequence model with attention. Finally, a vocoder or waveform model, such as WaveNet [8], transforms the intermediate representation into a raw audio waveform, producing the final speech output.

2.5 SSML

The Speech Synthesis Markup Language (SSML) is an XML-based markup language designed to provide greater control over the synthesized speech's characteristics, including pitch, rate, and volume. By using SSML tags, developers can manipulate the prosody of speech output generated by TTS systems, enabling a more natural and expressive result. [3]

2.6 Prosody

Prosody refers to the stress, rhythm, and intonation patterns in speech. It plays a critical role in conveying meaning and emotion and improving the naturalness and intelligibility of synthesized speech. This research explores various methods to adjust the prosody of speech output generated by TTS systems, aiming to enhance the overall user experience. [7]

3 Method

3.1 Data

The chosen text to be synthesized comes from the Common-Voice dataset [2]. The Common Voice dataset consists of more than 9000 hours of audio and its corresponding text, of which around 7000 hours are validated for 60 languages. The dataset also includes information on demographics such as accent, age, and sex. For the chosen task, we chose five phrases from the English dataset, and the five phrases belonged to phrases that were all more than 20 words long to ensure we would work with relatively long phrases.

The five phrases that will be synthesized by the different approaches to explore are:

- I'm never more aware of a room's acoustics than when I'm trying to enjoy a snack i have no intention of sharing.
- You don't have to be rich skinny popular or even own a cape to perform a random act of kindness.
- It was only when i got this close to it that the strangeness of it was at all evident to me.
- We miss you and miss having a friend like you and i am so happy that you two got to catch up.
- A part of me is also trying to come to terms with the fact that i won't be able to travel as much anymore.

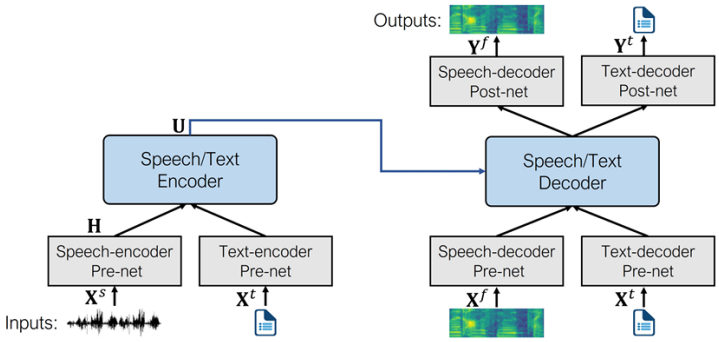
3.2 Experimental design / Planned Measurements

As part of the research methodology, we will generate speech output using various approaches for each of the five phrases. Ultimately, we aim to obtain five distinct results for each phrase, originating from the following sources:

- Original: The original audio from the Common Voice dataset.
- T5: An audio sample synthesized using the T5 encoder-decoder model.
- SSML: A plain audio sample produced by Amazon Polly without any modifications to the Prosody Tags.
- SSML + Attention: Audio generated by Amazon Polly with Prosody Tags modified based on attention scores.
- SSML + Pitch + Timestamps: Audio produced by first extracting pitch with PyReaper, matching words and timestamps using WhisperX, and subsequently modifying the prosody based on the identified words and pitch groupings.

3.3 SpeechT5

SpeechT5 [1] is a full encoder-decoder model based on T5 model made by Microsoft. SpeechT5 is a pre-trained encoder-decoder model, that contains specific pre and post nets, to allow it to be multi-modal (speech or text), in its input and output. SpeechT5 can receieve text or speech as input, converts speech/text to a unified space of hidden representaitons, which is then fed to the encoder-decoder model, and in the same way it is output as either speech or text. With the appropriate modalities, SpeechT5 is able to perform Automatic Speech Recognition, Speech Translation, Speech Identification, Text-to-speech, Voice Conversion, and Speech Enhancement.



(a) The model architecture of SpeechT5

Figure 1: SpeechT5 Architecture

We used SpeechT5 to produce audios for all 5 of the phrases. This will serve as another point of comparison to the audios produced by the Vanilla SSMLs, and the audios of the Prosody enhanced SSMLs that will be modified by our approaches with Attention, and Pitch Extraction.

3.4 SSML

By embedding SSML tags within text content, developers can enhance the way TTS systems process and vocalize text, allowing for adjustments to pronunciation, pitch, volume, speed, and more. To translate the SSML text to voice, we used Amazon Polly.

An SSML markdown example is the next:

```
<speak>
    I'm never more aware of a room's acoustics than when I'm trying to enjoy
    a <prosody pitch="+10\%">snack</prosody> I have no intention of sharing.
</speak>
```

where we are grouping the contents of the phrase, and applying a separate prosody tag to each group. We will produce audios for the 5 phrases using Vanilla SSML, so no enhanced prosody tag, to have a point of reference. For the prosody tags that will be modified in the next approaches, the "pitch" attribute will be modified.

3.5 SSML + Attention

A prevalent approach for adjusting prosody involves emphasizing particularly important words in a sentence while decreasing the pitch for less significant ones. One technique to obtain importance scores for each word in a sentence is by examining the attention layers of transformer models. Transformers employ attention mechanisms through the computation of Queries, Keys, and Values. Each word within the input sequence is represented as a Query, Key, and Value vector. The relevance score between a Query and all Keys is determined using dot-product attention, a similarity metric. Subsequently, these scores are normalized via a softmax function, generating attention weights (figure 2). The corresponding Values are then multiplied by these weights to produce a weighted sum, effectively accentuating important words (high weight) and diminishing the impact of less relevant ones (low weight). This process enhances the model's ability to capture context and relationships between words. For our study, we extracted weights from a pretrained BERT model for convenience. To visualize the attention layers, we utilized BertViz, a library developed by Jesse Vig [12].

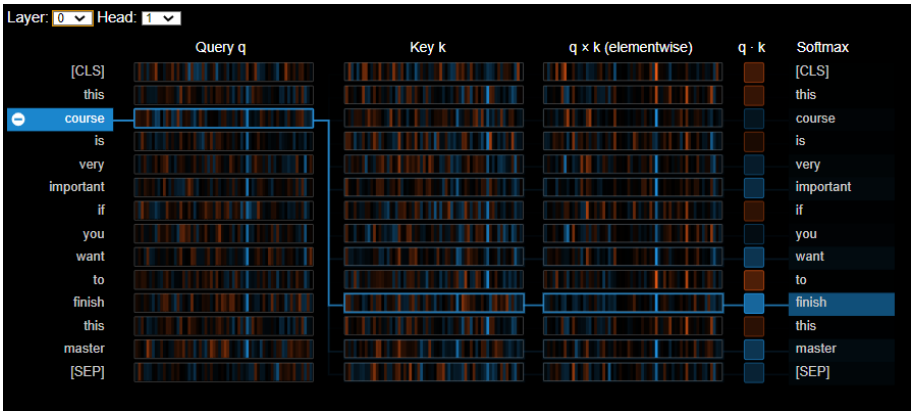


Figure 2: Visualizing how BERT computes attention weights from query and key vectors. Here it detects that the word "course" and "finish" have a strong relationship. The figure was created using the BertViz library [12].

The BERT model utilized in our study comprised 12 attention layers and 12 attention heads. For each head in every layer, there is an attention weight matrix of size $len\_sequence * len\_sequence$. Given a sequence of 20 tokens, we obtain a $12 * 12 * 20 * 20$ torch tensor. We investigated two approaches to compute a single importance score for each token.

In the first approach, we created a 20*20 tensor by aggregating over all attention heads in the last layer, which arguably contains the most information as it integrates outputs from previous layers. For each token, we computed the importance score by aggregating all output weights, exploring both mean and sum aggregation methods, which yielded similar results.

In the second approach, we aggregated information across all layers instead of solely focusing on the last one. This approach is advantageous as each layer captures different levels of abstraction and linguistic features. Initial layers typically concentrate on local syntactic structures and surface-level patterns, while deeper layers capture more complex semantic relationships and long-range dependencies. We found this approach to be more stable and exhibit less variance. A visualization of the computed scores can be seen in figure 3.

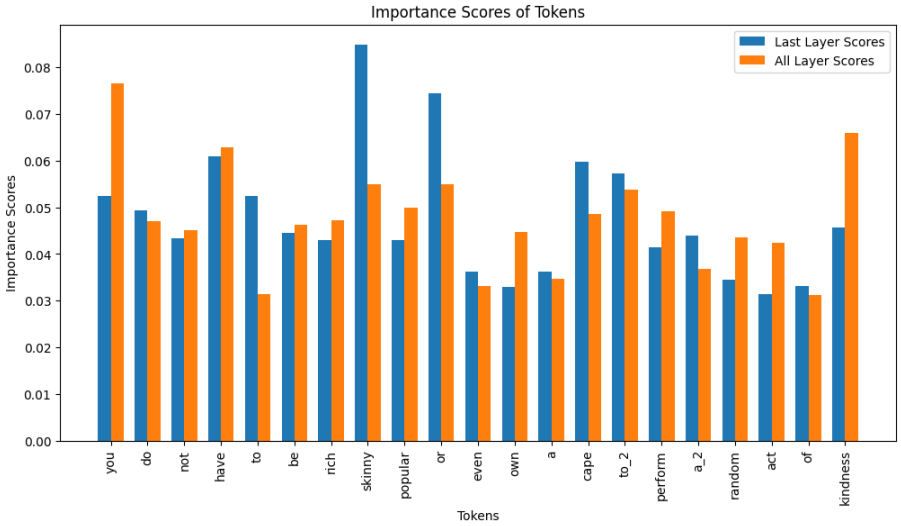


Figure 3: The computed word importance scores for the sentence *"You do not have to be rich skinny popular or even own a cape to perform a random act of kindness"* - In orange, using the attention weights from all layers, in blue, using only the weights from the last layer.

To generate SSML based on the computed importance scores, our method identifies words with the highest and lowest importance. For highly important words, we increase the pitch, while for less relevant words, we decrease it. To avoid abrupt prosodic shifts that can result from modifying the pitch of only one word using SSML, we implemented an approach that incorporates neighboring words within the SSML tags, ensuring smoother and more natural-sounding prosodic adjustments. Below is a sample SSML output, generated by our method:

```
<speak>
<prosody pitch="+15%">you do not</prosody>
have to be rich skinny popular
<prosody pitch="-15%">or even own a cape</prosody>
to perform a random act of kindness
</speak>
```

3.6 SSML + Pitch + Timestamp

In this approach, we will extract pitch with Pyreaper, match the pitch to the words using the timestamps of the words, obtained with WhisperX, and then use several heuristics to group words and apply the SSML prosody tag to those groups of words.

To explain the process done in this step, we will use one of the phrases as an example:

1. I'm never more aware of a room's acoustics than when I'm trying to enjoy a snack I have no intention of sharing.

REAPER (Robust Epoch And Pitch EstimatoR) [6] is a tool that extracts pitch as the location of voiced-speech, and fundamental frequency. Pyreaper acts as a wrapper for Reaper. The tool receives an audio file as input, and outputs a time series, with the moments of voicing state, and fundamental frequency. For this approach, we will use the voicing state (voiced or unvoiced) as pitch. Figure 4 is the output of REAPER, as a binary signal representing the voiced and unvoiced moments.

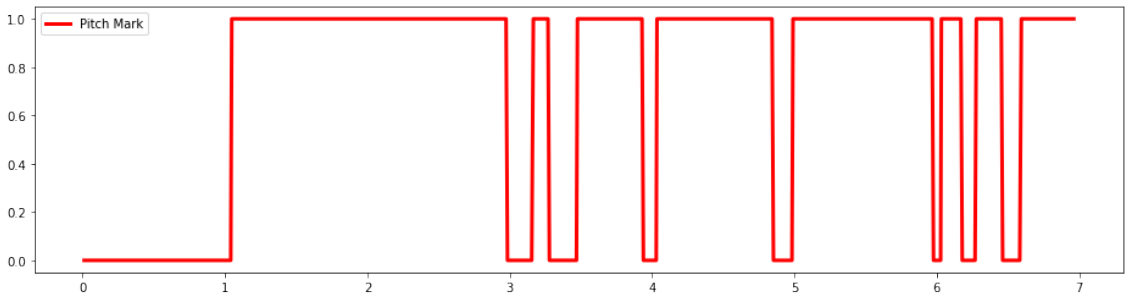


Figure 4: PyReaper Pitch

After obtaining the pitch, we need to match the words to the signal, so we need to obtain the timestamps of the words. For this we will use WhisperX [4]. WhisperX is a tool that builds upon OpenAIs Whisper [9]. Whisper is an ASR model that produces accurate transcriptions, but has a poor word-level timestamp production. WhisperX improves this by performing forced aligment with a Phoneme-based ASR model, like wav2vec2.0. In this way, we leverage both their capabilities to obtain precise word timestamps, and accurate transcriptions.

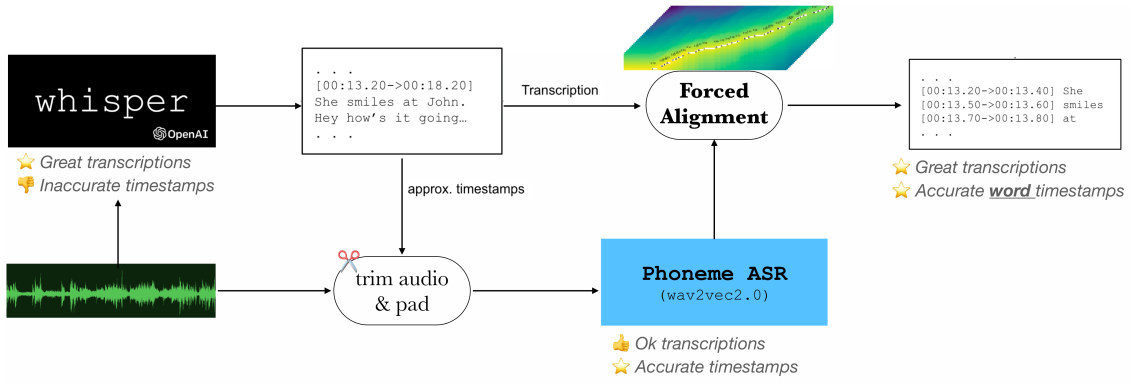


Figure 5: WhisperX

Figure 6 shows the previously extracted pitch signal with the words in their corresponding temporal spot, obtained by WhisperX.

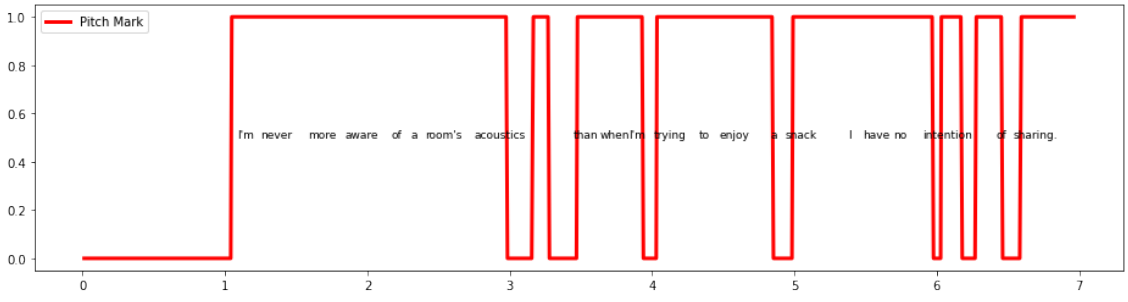


Figure 6: Pitch + Timestamps of Words from WhisperX

At this point we have the words matched to the pitch obtained with PyReaper. By observing Figure 6, we see that there are a lot of groups of words according to the pitch signal. If we apply a prosody tag to every group (that would include a separate tag for the sub-text "" and also the word "of"). So to avoid using an unnecessary amount of Prosody SSML tags, we will look and "smooth" the pitch signal to build the best combination of possible groups to apply the SSML Prosody Tags to those groups.

For this we developed a smoothing algorithm for the signal. The algorithm takes all the lengths of the HIGH and LOW segments, and builds an array of the HIGH lengths, and an array of LOW lengths. The smallest element from each array is obtained and compared. If the smallest length is the HIGH length, then the HIGH length becomes LOW, and if the smallest length is the LOW length, then the LOW length becomes HIGH. Once this is done, the lengths are recalculated and the process starts again. The process iterates until the stopping criteria, which is to get to a predetermined number of groups of tags.

The pseudo-code of the algorithm looks as follows:

```
fn(pitch_arr, pitch_times_arr, number_of_desired_tags = 3):
Begin

    modified_pitch  = pitch_arr.copy()

    while True:
        upward_indices <- get indices when the signal switches to 1
        downward_indices <- get indices when the signal switches to 0

        pairs_up_lengths <- get array of tuples with start and end of HIGH segment
        pairs_down_lengths <- get array of tuples with start and end of LOW segment
```

```
        up_lengths <- get array of length in time of HIGH segments
        down_lengths <- get array of length in time of LOW segments

        shorter_up <- get shorter HIGH segment
        shorter_down <- get shorter LOW segment

        if shorter_down<= shorter_up
            modified_pitch <- modify pitch signal to make shorter_down HIGH
        else:
            modified_pitch <- modify pitch signal to make shoerter_up LOW

    if number_of_separated_groups == number_of_desired_tags:
        break

    return modified
```

End

The example pitch signal is shown in Figure 7, after the smoothing. In this way we have 3 marked groups in which we can encapsulate the prosody tags, with the modified pitch value.

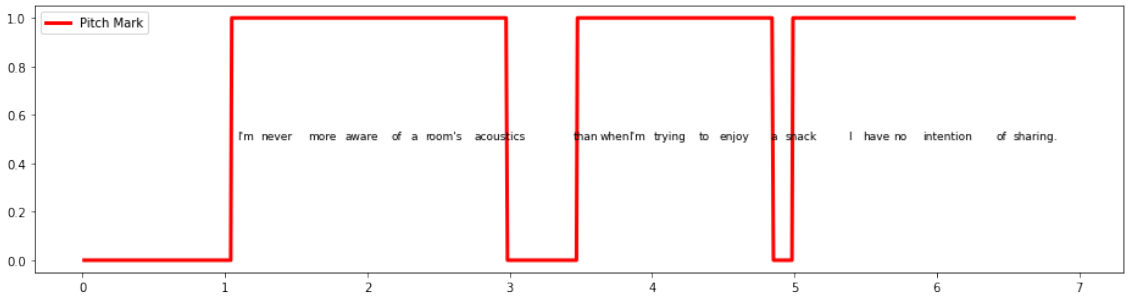


Figure 7: Softened Pitch, with phrases grouped

With this we output the following SSML markdown snippet:

```
<speak>
    <prosody pitch="10%">I'm never more aware of a room's acoustics than</prosody>
    <prosody pitch="10%">when I'm trying to enjoy a snack</prosody>
    <prosody pitch="10%">I have no intention of sharing.</prosody>
</speak>
```

The grouping makes the best arrangement, but it has the limitation that since the level of the signals is the same for all of them (0 or 1), we can not make an assumption of value of pitch, so all of the values were put to a constant 10%.

4 Evaluation

To evaluate the performance of our proposed approaches, we employed the Mean Opinion Score (MOS) by asking ten participants, primarily KTH students, to rate the naturalness of each generated audio sequence on a scale of 1 to 5. The unadjusted Amazon Polly output served as the baseline, assigned a rating of 3, with all other ratings relative to this baseline. This experiment was conducted for five distinct word sequences (Figure 8).

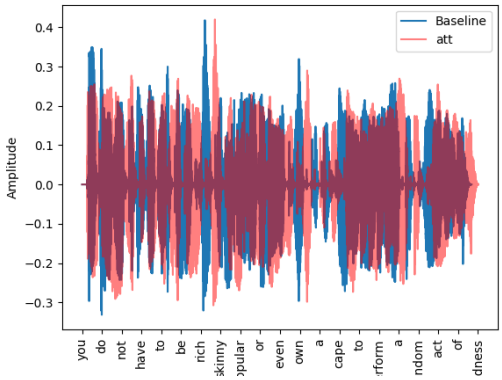
Figure 8: Methodology for User Study: Participants were provided with unrestricted access to each voice sample, enabling repeated auditions and evaluations. Ratings were obtained for each audio sample on a 5-point scale, where higher scores corresponded to superior quality. The red audio signal is the adjusted one, the blue is the Amazon Polly baseline.

Our findings indicate that the end-to-end T5 model outperformed other methods, achieving a MOS of 3.7. The pitch-extraction approach also demonstrated an improvement in the naturalness of the TTS system. However, utilizing information from the attention layers led to a decrease in performance (table 4).

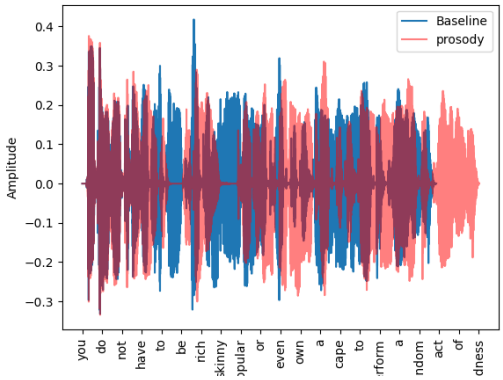
Approach	MOS
SSML	3.00
SSML + Attention	2.80
SSML + Pitch	3.24
T5	3.70

Table 1: User Study Result: MOS for the three approaches

Figure 9b presents an encouraging outcome when adjusting prosody based on the extracted pitch. The adjusted pitches introduce pauses at appropriate timestamps, contributing to a more natural-sounding voice. However, Figure 9a also highlights a significant limitation of our approachês. In numerous instances (for both approaches), our heuristics for adjusting prosody using SSML tags fail to substantially alter the signal, resulting in a high overlap with the original audio. In several cases, participants could not discern a difference between the baseline and the adjusted audio. To circumvent this issue, a more refined heuristic should be developed, which entails adjusting more than just a single SSML attribute (in our case, only the pitch was adjusted).



(a) Red: Resulting audio signal after adjusting the prosody using word importance scores, based on attention. Blue: Amazon Polly Baseline



(b) Red: Resulting audio signal after adjusting the prosody based on extracted pitch. Blue: Amazon Polly Baseline

Figure 9: Resulting audio signals after adjusting the prosody.

5 Discussion

Adjusting prosody presents a considerable challenge. While both approaches offer promising insights, they fall short of fully addressing the issue. Further research is required to determine the optimal utilization of importance scores derived from attention layers. In the current implementation, a simplistic heuristic adjusts the pitch only for highly important or unimportant words. However, prosody encompasses more than just pitch, and a fixed adjustment (e.g., 15%) fails to produce natural-sounding voice outputs. Participants in our user study emphasized the significance of well-timed pauses in generating natural-sounding TTS, a feature that our word-importance score does not appear to encode. Furthermore, the SSML tags encompass a large variety of options that were not explored, and if all used adequately, could create better speech. It was not under the scope of this project to explore all of the SSML tags, that could've yield better results.

In one instance, employing the original audio's pitch (our second approach) resulted in a highly-rated output, surpassing the audio generated by the end-to-end T5 model. This finding indicates that further exploration of this approach could yield more promising results. This gives intuition as to how getting the correct groups, can make an SSML system perform better in creating speech.

We would also like to mention that our results were computed on 10 volunteers, which may lack the expertise and experience to evaluate speech, as none of them were Native speakers, and also 10 datapoints is not enough to build a conclusion, but it is what fit under the scope of the project.

It is important to mention that commercially there exist a range of options, such as "Synthesys" (https://synthesys.io/), that provide better TTS approaches, that are unfortunately not open sourced and require membership to use.

6 Future Work

For Future Work, we could explore and develop an actual NLP task that given a text, can be input into a language model to produce SSML tags with improved prosody. There would be challenges in obtaining the data, mainly the SSML tagged material, but it would be an interesting language generation task. It would lie under the task of "generating code".

For the pitch extraction + word timestamps approach, since the limitation was using the voiced moments plot, it was evident after the development of the project, that the fundamental frequency information plot that is also output by REAPER would've served better for the intended results. So for future work instead of using the voiced moments plots, to use the fundamental frequency graph.

As mentioned in Discussion, also exploring the use of other SSML tags is left for further work, and also exploring the effect of using different values for the tags attributes, such as volume, rate and pitch for the prosody tag.

As proposed in the presentation, to really evaluate change in prosody, shorter length of phrases is also something that could be explored, and would be easier to evaluate if the generated language has better prosody or not.

References

[1] Junyi Ao et al. *SpeechT5: Unified-Modal Encoder-Decoder Pre-Training for Spoken Language Processing*. 2021. DOI: 10.48550/ARXIV.2110.07205. URL: https://arxiv.org/abs/2110.07205.

[2] Rosana Ardila et al. "Common Voice: A Massively-Multilingual Speech Corpus". In: *CoRR* abs/1912.06670 (2019). arXiv: 1912.06670. URL: http://arxiv.org/abs/1912.06670.

[3] AWS. *Supported SSML Tags*. Available at https://docs.aws.amazon.com/polly/latest/dg/supportedtags.html (2023).

[4] Max Bain et al. *WhisperX: Time-Accurate Speech Transcription of Long-Form Audio*. 2023. DOI: 10.48550/ARXIV.2303.00747. URL: https://arxiv.org/abs/2303.00747.

[5] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2018. DOI: 10.48550/ARXIV.1810.04805. URL: https://arxiv.org/abs/1810.04805.

[6] Google. *REAPER*. 2014. URL: https://github.com/google/REAPER.

[7] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 1st. USA: Prentice Hall PTR, 2000. ISBN: 0130950696.

[8] Aaron van den Oord et al. *WaveNet: A Generative Model for Raw Audio*. cite arxiv:1609.03499. 2016. URL: http://arxiv.org/abs/1609.03499.

[9] Alec Radford et al. *Robust Speech Recognition via Large-Scale Weak Supervision*. 2022. DOI: 10.48550/ARXIV.2212.04356. URL: https://arxiv.org/abs/2212.04356.

[10] Colin Raffel et al. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. 2019. DOI: 10.48550/ARXIV.1910.10683. URL: https://arxiv.org/abs/1910.10683.

[11] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[12] Jesse Vig. *BertViz*. 2017. URL: https://github.com/jessevig/bertviz.

[13] Yuxuan Wang et al. *Tacotron: Towards End-to-End Speech Synthesis*. 2017. DOI: 10.48550/ARXIV.1703.10135. URL: https://arxiv.org/abs/1703.10135.