Evaluating w2v-BERT-2.0 Speech Tokenizer for Linguistic Performance in Speech Models

Abubakar Aliyu Badawi*, Celil Yilmaz*, Tayyab Tahir*, Ishfaq-Bhat*
*Department of Seatech, University dé Toulon, La Garde 83130, France
{abubakar-aliyu-badawi, celil-yilmaz, tayyab-tahir, eshfaz-bhat}@etud.univ-tln.com

***Abstract*—Speech-language models (SLMs) are crucial for natural language processing (NLP) and speech recognition, enabling applications that require the comprehension and generation of human language. This report integrates the W2v-BERT-2.0 tokenizer into existing speech-language models (SLMs) to improve speech-to-speech translation by processing speech directly, without converting it to text first, thereby eliminating the traditional intermediary text conversion step. The research involves replacing the HuBERT model's tokenizer with the open-source W2v-BERT-2.0 from Meta, leveraging its Conformer-based architecture and extensive pre-training on a multilingual corpus. Our methodology includes processing audio data, studying the W2v-BERT-2.0 architecture, extracting features, and training the model on extensive datasets. The enhanced model's syntactic performance is evaluated using the sBLIMP benchmark, the results shows some promising improvements in syntactic understanding and generalization, though further training and optimization, and some modifications are recommended. The findings suggest that adding the W2v-BERT-2.0 tokenizer to speech-language models (SLMs) can significantly improve their performance in certain language tasks, making speech-to-speech translation more efficient and accurate in various NLP applications.**

***Index Terms*—speech language models, w2v-BERT-2.0, tokenizers, natural language processing, deep learning, speech recognition**

I. Introduction

A. Speech-Language Models

Speech-language models (SLMs) are important in natural language processing (NLP) and speech recognition, serving as the backbone for applications that require understanding and generating human language. These models facilitate a range of functionalities, from voice-activated assistants and automatic transcription services to language translation. The evolution of SLMs, particularly through deep learning techniques, has significantly enhanced the capabilities of speech recognition systems, enabling them to grasp the context, nuances, and emotions in human speech more effectively.

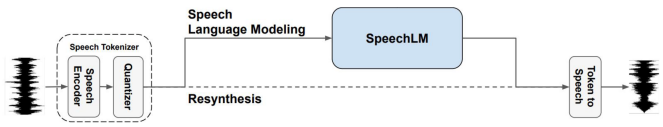

Fig. 1: Block Diagram of SLMs
Taken from: https://pages.cs.huji.ac.il/adiyoss-lab/twist/

The Speech Language Model (SLM) comprises several key components which is shown in Fig 1, each playing a vital role in processing and generating speech-based text. These components include:

1) **Speech Tokenizer** - This component is responsible for breaking down speech input into manageable tokens or units that can be further processed. These tokens often represent phonetic or linguistic elements that are foundational for understanding the speech.
2) **Speech Encoder** - It extracts feature representations from the speech tokens. This encoding process translates raw speech signals into a form that captures the essential characteristics of the speech, making it suitable for machine interpretation and further analysis.
3) **Quantizer** - Utilizes a K-means model to transform self-representations into clusters. This step is crucial for reducing the complexity of data while retaining the significant patterns, facilitating efficient processing in subsequent stages.
4) **SpeechLM (Generative Pre-trained Transformer, GPT)** - A generative model that leverages the Transformer architecture to generate speech tokens. It learns to predict the next item in a sequence, allowing for the generation of coherent and contextually relevant speech text based on the features and clusters identified by previous components.

B. Why Speech-Language Models Matter

Speech Language Models (SLMs) signify a groundbreaking shift in natural language processing and speech recognition, diverging from traditional methods that convert speech to text and vice versa. SLMs enable direct speech-to-speech processing, eliminating the need for text intermediary steps. This advancement not only streamlines human-machine interactions but also opens new avenues in language understanding and generation. By focusing on speech and its transformations, SLMs greatly enhance digital system accessibility and efficiency, marking them as a pivotal innovation in technology.

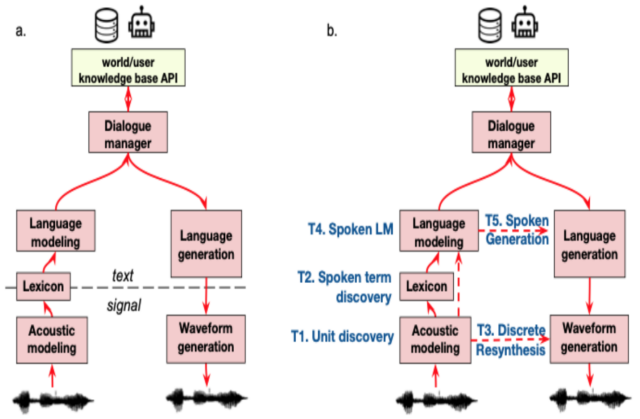

Fig. 2: Comparative Overview of Traditional and Advanced Speech Translation Methods

Taken from: https://www.zerospeech.com.

Some of the highlights of some main importance of SLMs include:

- **Enhanced Speech Recognition:** SLMs significantly improve the accuracy of speech recognition technologies. They excel in challenging environments with background noise or unclear pronunciation, enhancing the robustness and user-friendliness of digital systems.
- **Addressing Language Diversity:** With approximately 3000 languages worldwide, many lack comprehensive text data, making transcription to text expensive and time-consuming. For the majority of these languages, no adequate text corpora exist, with only about 100 of the most common languages having substantial transcription resources. SLMs offer a pivotal advantage by focusing on direct speech processing, bypassing the need for text data and facilitating linguistic diversity and inclusivity in digital communication.
- **Natural Language Interaction:** SLMs lead in enabling conversations with computers to feel as natural as talking to a human friend. This advancement facilitates intuitive voice commands and interactions, broadening the accessibility and ease of use of digital devices for a wider audience.

II. Aim

To explore and assess the efficacy of integrating the W2v-Bert-2.0 tokenizer into an existing model that tends to reproduce the open-source version of Google's AudioLM, aiming for enhanced performance in speech-to-speech translation by direct speech processing, without the intermediary step of text conversion. This approach seeks to leverage the advanced capabilities of W2v-Bert-2.0 in handling speech embeddings for improved accuracy and efficiency in speech language modeling.

The current model uses a speech tokenizer called Hubert model for extracting the embeddings and the features from the speech data, we will try to replace this tokenizer with the open source model called w2v-bert-2.0 from Meta.

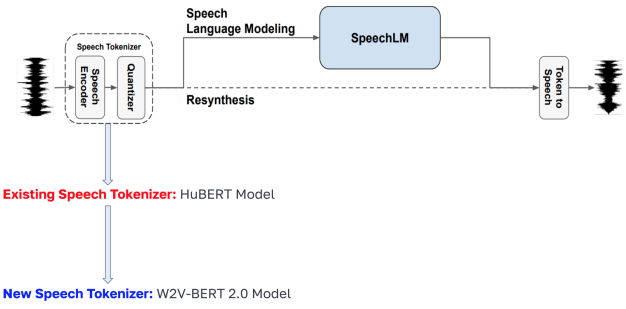

Fig. 3: Proposed Aim of the Project

A. The Role of Tokenizers in SLM

Tokenizers are integral to the preprocessing phase in both speech and text processing pipelines. They segment input data into manageable units called tokens, which can be phonemes, syllables, words, or subword elements. This segmentation is crucial as it influences how linguistic information is interpreted and processed by the models. Effective tokenization preserves semantic meaning, allowing for more efficient training and functionality of speech-language models.

III. Objectives

The project is structured around several key objectives that we aim to achieve in order to enhance the performance of the speech language model through the integration of the W2v-Bert-2.0 tokenizer. The key objectives include:

1) Process the audio data into a TSV file, containing the training and testing datasets.
2) Study the architecture and functionalities of the W2v-Bert-2.0 tokenizer.
3) Extract features from the audio files using the advanced capabilities of the W2v-Bert-2.0 tokenizer, ensuring high-quality feature representation.
4) Process the extracted features, preparing them for model training.
5) Perform a training run on a dataset comprising 100 hours of audio data, establishing a baseline for the model's performance and effectiveness.
6) Conduct an extensive training run with a larger dataset of 60,000 hours, aiming to significantly improve the model's accuracy and robustness through deep learning.
7) Evaluate the performance of the speech language model.

IV. Datasets

Our SLM, w2v-BERT2.0, was trained using two distinct audio datasets: "train-clean-100" and "Libri-light Large". The "train-clean-100" dataset, consisting of 100 hours of high-quality, clean audio, provides a controlled environment ideal

for initial model training, focusing on fundamental linguistic features and phonetic accuracy. In contrast, the "Libri-Light Large" dataset, with its extensive 60,000 hours of varied audio recordings, offers a rich landscape for enhancing the model's robustness and adaptability to real-world conditions. This comprehensive training approach ensures that the w2v-BERT2.0 model not only learns from clean, noise-free examples but also gains exposure to a wide array of speech patterns, accents, and dialects, thereby maximizing its effectiveness and generalization across different auditory environments.

A. Train-clean-100 Dataset

The "train-clean-100" dataset, comprising 100 hours of meticulously curated, clean audio recordings, serves as an essential resource for training and benchmarking automatic speech recognition (ASR) systems. This dataset is particularly valued for its high-quality, noise-free audio samples, which are pivotal for initial model training phases where the focus is on learning basic linguistic structures and phonetic details without the confounding influence of background noises or distortions.

Utilizing "train-clean-100" allows for the development of robust foundational models capable of accurately decoding spoken language into text. The clean nature of the recordings ensures that ASR systems can focus on understanding the nuances of language such as accents, intonations, and spoken syntax, which are critical for producing accurate transcriptions. This dataset is also frequently used as a benchmarking tool to assess the clarity and the precision of ASR systems under controlled conditions, thereby providing insights into the effectiveness of the model's acoustic and language processing capabilities.

B. Libri-Light Large Dataset

The "Libri-Light Large" dataset, encompassing a staggering 60,000 hours of audio recordings, represents a substantial resource for training and refining automatic speech recognition (ASR) systems at scale. This vast dataset is derived primarily from audiobooks and is designed to provide a diverse array of spoken content, featuring multiple accents, dialects, and varying speech modalities. Its primary strength lies in its volume and variety, offering an unparalleled breadth of linguistic data.

This extensive collection of audio is instrumental in training ASR systems to cope with a wide range of speech patterns and unpredictable real-world audio environments. The sheer scale of "Libri-Light Large" allows for deep learning models to be exposed to and learn from a myriad of phonetic contexts and dialectical variations, significantly enhancing the model's ability to generalize across different speakers and conditions. This is crucial for developing ASR systems that maintain high levels of accuracy and reliability in practical, everyday use scenarios.

V. Overview of the Model [w2vBERT-2.0]

The W2v-BERT 2.0 model represents a significant advancement in speech language processing, combining the strengths of Conformer-based architecture with extensive pre-training over a vast multilingual corpus. With 600 million parameters and training on 4.5 million hours of audio from more than 143 languages, it is designed for superior performance in tasks such as Automatic Speech Recognition (ASR) and Audio Classification. This model offers unprecedented linguistic diversity and depth, setting a new standard for speech-based applications [1].

VI. Methodology

A. Identifying the Optimal Layer for Audio Feature Extraction

We evaluate the performance of each layer within the W2v-Bert-2.0 model for feature extraction by training a linear classifier to classify the embeddings derived from each layer with its associated labels. The label file is crucial for training our model, mapping audio files to their phonetic representations at a sampling rate of 50Hz. This detailed mapping allows for precise training, aligning audio segments with specific phonetic labels.

By measuring classification accuracy for each layer, the process identifies the layer that yields the highest accuracy, which is the 16th layer with an accuracy of 88.6%, indicating the most effective layer for feature extraction in the context of the project's objectives. the layer-wise accuracy is shown in Figure 4.

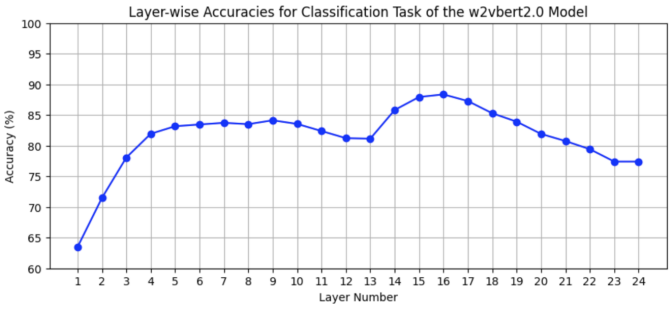

Fig. 4: Layer-wise accuracy for the Classification task of the W2VBert-2.0 model

B. Data Pre-processing

The data preprocessing stage is an essential component of our project, aimed at enhancing speech language models using the W2v-BERT 2.0. Audio data, specifically in `.wav` format from the LibriSpeech dataset located on the Mundus partition, forms the basis of our analysis. This process begins by dividing the datasets into training and testing sets with an 80:20 ratio (train-clean 100) and a 99.9:0.01 ratio (Librilight). Subsequently, these files are copied into directories in the cluster. The final phase involves feature extraction using the W2v-BERT 2.0

model, where the features are generated and saved for further model training and performance evaluation.

C. Training the Kmeans Model

We trained a K-means model on pre-processed audio features to categorize speech patterns efficiently. We achieve this by adapting a script from fairseq GitHub repository, specifically the `"train_kmeans.py"` and name it `"train_kmeans_w2vbert.py"`.

This process, leveraging the MiniBatchKMeans algorithm, was designed to ensure optimal clustering of audio data, significantly enhancing our understanding of diverse audio characteristics. Through careful selection of algorithm parameters and feature sampling, we achieved a robust model capable of distinguishing the audio clusters.

D. Feature Extraction

The process utilizes a pre-trained W2v-Bert-2.0 model to analyze audio files, converting them into a compact representation known as speech tokens. We achieve the feature extraction process by adapting a script from fairseq GitHub repository, specifically the `"dump_hubert_codes.py"` and name it `"dump_w2vbert_codes.py"` so that it will utilize the structure of our new model.

The extraction results in three files: a **.bin** file containing the speech tokens, and a **.len** file which lists the length of each embedding sequence. After obtaining the speech tokens we remove adjacent repetitions, e.g. [0, 0, 1, 2, 3, 3] becomes [0, 1, 2, 3]. The **.dur** file contains the duration of each of the collapsed units. We will not use it, but it's useful in keeping track of the original duration in time of each speech token.

E. Speech Tokenizer and Processing Pipeline

Figure 5 illustrates the architecture and workflow of a speech language model, specifically focusing on the speech tokenizer component. The pipeline begins with an input WAV file, representing raw audio data, which is then fed into a CNN Feature Encoder. This encoder is responsible for extracting latent speech representations from the raw audio. These representations capture essential characteristics of the speech, making them suitable for further processing.

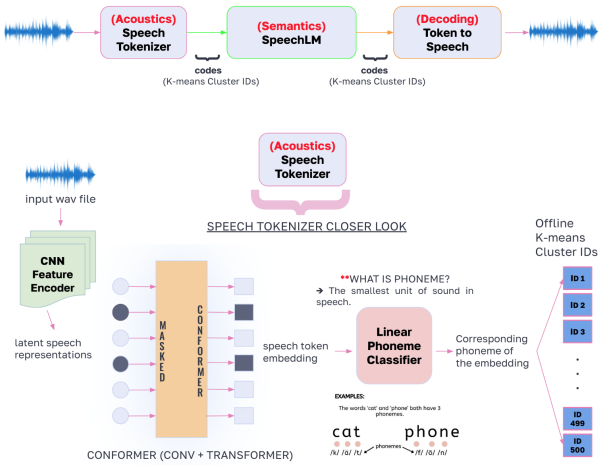

Fig. 5: Overview of the Feature Extraction

Following the feature encoding, the process enters a core component highlighted in the diagram—the Conformer module, which combines convolutional neural network (CNN) and Transformer technologies. This Conformer is crucial for adapting the feature representations to a form that balances local features, processed by CNN layers, with global dependencies handled by Transformer layers. It outputs speech token embeddings, which are subsequently classified by a Linear Phoneme Classifier.

The phoneme classifier plays a vital role in identifying and classifying the phonemes, the smallest units of sound in speech, from the speech token embeddings. Each embedding is mapped to a phoneme, as demonstrated by the examples 'cat' and 'phone' shown in the diagram, helping the model to understand and process the linguistic elements of the speech.

Moreover, the model utilizes offline K-means clustering, assigning cluster IDs to these embeddings, which are then used in both semantic processing by the SpeechLM (Speech Language Model) and decoding phases. The SpeechLM leverages these cluster IDs to understand and generate language-based predictions, while the decoding phase translates these tokens back into audio form, completing the speech-to-speech process.

F. Training and Evaluation Workflow

Figure 6 provide workflow diagram for the training and evaluation, our training process starts with a single batch of coded .FLAC files which are fed into our transformer-based language model via the script *train_lm.py*. After training the model on 60,000 hours of audio, the trained model's codes are input to an evaluation model through the *eval_lm.py* script. This script calculates log probabilities, which are then used by the ZRC Toolkit for benchmarking. The evaluation focuses on the model's syntactic performance, with the ZRC Evaluation Dataset serving as the target for these assessments. The outcome is a syntactic performance score that quantifies

the model's effectiveness in processing and understanding language structures.

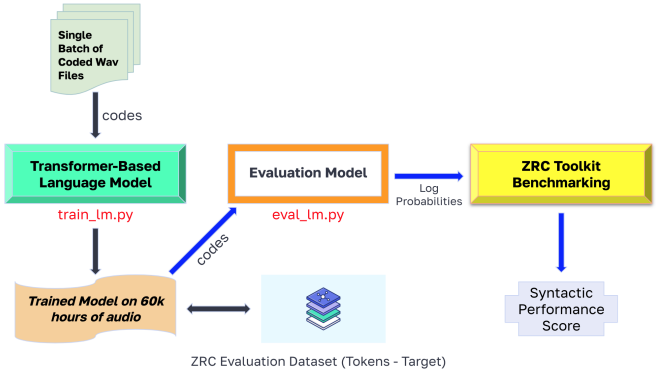

Fig. 6: Training and Evaluation Workflow

G. Model Training

The model training was done completely on the cluster. Santiago gave us a piece of the code which was used to train the HuBERT model, we then adapted the code to work with our model by changing some parameters such as the dimension of the tensors and so on. We transferred all the adapted scripts to a directory on the cluster, and created a new conda environment with all the dependencies we need some of which are:

- **numpy** (version 1.22.0) - A fundamental package for scientific computing with Python, providing support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.
- **pytorch** (version 2.2.1) - An open-source machine learning library for Python, used for applications such as computer vision and natural language processing, mainly developed by Facebook's AI Research lab.
- **scipy** (version 1.10.1) - A Python-based ecosystem of open-source software for mathematics, science, and engineering, particularly for scientific computing.
- **transformers** (version 4.39.0.dev0) - A library by Hugging Face offering state-of-the-art general-purpose architectures for Natural Language Processing (NLP) including BERT, GPT, RoBERTa, etc.
- **librosa** (version 0.10.0) - A Python package for music and audio analysis, providing the building blocks necessary to create music information retrieval systems.
- **scikit-learn** (version 1.0.2) - A machine learning library for Python, featuring various classification, regression, and clustering algorithms.
- **ffmpeg** (version 4.3) - A complete, cross-platform solution to record, convert and stream audio and video, which is a crucial tool for processing multimedia content.
- **wandb** - A tool for visualizing and tracking your machine learning experiments, making it easier to share results with collaborators and manage your models.

After installing all the dependencies, we coded a small bash script that will run our training script. The script includes the name of the training job, the estimated allocated time for the cluster, and also requests a GPU, specifically the **NVDIA A100-20G**. As a caution, we ensured that there was a checkpoint mechanism saving the latest parameters for possible interruptions or scheduling problems, however, the mechanism works only when the training and validation loss decreases.

VII. Results

A. Training on 100 hours of Audio Dataset [train-clean-100]

The first training was done on the baseline dataset we have access to that contains 100 hours of audio data in various subdirectories. We split the dataset into train and valid with an 80 and 20 percent ratio with each containing the bin, len and dur file extracted using the w2v-BERT 2.0 model. The code we adopted was initially designed for the larger dataset, we then reduced the model size by adjusting some parameters like the number of heads and layers of the model and applying dropout, which led to some better results as compared to when the default parameters were set. We achieved the least training loss with the default parameters but with the highest evaluation loss, which means the model overfit. We tried several configurations of the model size because the initial configuration was meant for the large dataset, we were able to get a decent loss of training loss of 3.158 and a validation loss of 3.188, below is a snapshot of the two best runs we could get for the 100 hours of the dataset. Figure 7 shown the runs we performed some runs overfit due to the larger model size, but the best runs are depicted in Figure 8. The plots below show the training and evaluation losses as well as the learning rate and number of iterations associated with each run.

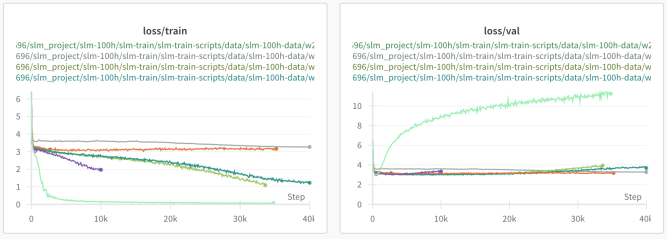

Fig. 7: Train and evaluation losses plot

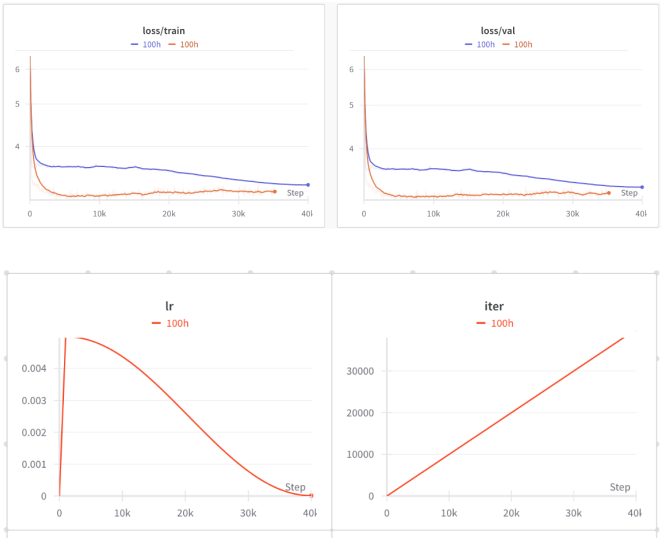

Fig. 8: Train and evaluation losses plot for the Best Model Configuration

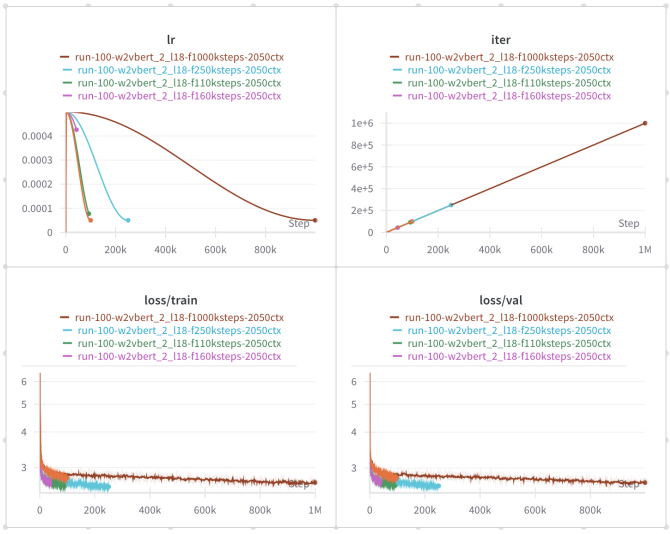

Fig. 9: Train and evaluation losses plot

B. Training on 60,000 hours of Audio Dataset [Libri-Light Large Dataset]

The second model training was on the Libri-Light Large Dataset which consists of 60,000 hours of audio data, we split the data into train and valid in a ratio of 99.99 to 0.001 percent with each containing the bin, len and dur file extracted using the w2v-BERT 2.0 model. We were able to perform just two complete runs (the previous attempts were executed on a different node where we didn't have high priority, and our tasks were halted to accommodate jobs with higher priority) for one and two epochs respectively due to GPU constraints, the dataset was too large and took around 7 days to process **3,280,000,000 tokens** which is equivalent to 1 epoch using the **NVIDIA a100-20** which was the only option we could have at that time, we tried using the a100-80 several times to speed up the process but unfortunately, we usually get kicked out due to higher priority jobs.

For the first run in "dark red" in figure 9, we set the batch size to 16 and the gradient accumulation to 1 (due to OOM error when set to 16) and set the maximum iteration count to 990,000 which is equivalent to around 1 epoch, after the training the train loss and validation loss were around 2.675 and 2.671 respectively. For the second run in "blue", we set the batch size to 16 and gradient accumulation to 16 as well and set the maximum iteration count to 250,000 which is equivalent to 2 epochs, after the training the train loss and validation loss were around 2.558 and 2.50 respectively.

C. Evaluating the Performance of the Model

To evaluate the performance of our trained model, we use the sBLIMP (syntax Benchmark of Linguistic Minimal Pairs) and additional syntactic acceptability metrics. The sBLIMP benchmark is specifically designed to assess the syntactic processing capabilities of speech language models by testing their ability to differentiate between grammatical and ungrammatical sentences. This benchmark, along with the sBLIMP Acceptability Metrics and the Syntactic Level Acceptability assessment, provides a comprehensive framework for evaluating how well the model understands and processes different syntactic features.

1) The sBLIMP (syntax Benchmark of Linguistic Minimal Pairs): The sBLIMP (syntax Benchmark of Linguistic Minimal Pairs) is an essential tool for evaluating the syntactic processing capabilities of speech language models. Originally adapted from the BLIMP benchmark developed by Warstadt, Parrish, and colleagues in 2019, sBLIMP focuses on assessing the ability of models to differentiate between grammatical and ungrammatical sentences. The benchmark consists of pairs of sentences where one is syntactically correct and the other contains a deliberate grammatical error. These minimal pairs are specifically designed to test one syntactic feature at a time, thereby isolating the model's ability to understand and process different grammatical constructs without the interference of lexical complexity.

Syntax: the sBLIMP Acceptability Metrics: This component of the benchmark, adapted from the original BLIMP, involves a set of linguistic minimal sentence pairs comprising matched grammatical and ungrammatical sentences. Like sWUGGY, the task is to decide which of the two sentences is grammatical based on the probability of the sentence. The test and development sets contain 63,000 and 6,300 sentence pairs respectively, with no sentence pair overlap. Stimuli were filtered to contain

only LibriSpeech vocabulary and were synthesized to maintain natural prosodic contours.

Syntactic Level (sBLIMP Syntactic Acceptability): This metric assesses whether models possess knowledge of the syntactic structure of the language. Models are presented with pairs consisting of a grammatical sentence (e.g., 'the dogs sleep') and an ungrammatical correspondent (e.g., 'the dog sleep'). The syntactic acceptability accuracy measures how often the model correctly assigns a higher probability to the grammatical sentence than to the ungrammatical one, across a variety of syntactic phenomena. Character-based language models trained on LibriSpeech annotations achieve an accuracy of around 68%, while humans and larger language models score in the 80-90% range. For reference, random chance would result in an accuracy of 50%.

D. Training for 1 Epoch

Results of the sBLIMP Benchmark

Type	N	Score	Standard Deviation
anaphor_agreement	200	0.4755	0.2378
argument_structure	900	0.5006	0.2468
binding	500	0.4950	0.2457
control_raising	500	0.4855	0.2444
determiner_noun_agreement	800	0.4709	0.2579
ellipsis	100	0.5175	0.2468
filler_gap_dependency	800	0.5099	0.2436
irregular_forms	200	0.4975	0.2374
island_effects	700	0.5046	0.2517
npi_licensing	700	0.5088	0.2539
quantifiers	300	0.5100	0.2409
subject_verb_agreement	600	0.5025	0.2535
Average Weighted Score		**0.4977**	-

Fig. 10: Leader board Comparison Table

E. Training for 2 Epochs

Results of the sBLIMP Benchmark

Type	N	Score	Standard Deviation
anaphor_agreement	200	0.4725	0.2321
argument_structure	900	0.5036	0.2494
binding	500	0.4915	0.2519
control_raising	500	0.4850	0.2432
determiner_noun_agreement	800	0.4878	0.2601
ellipsis	100	0.5200	0.2505
filler_gap_dependency	800	0.5081	0.2446
irregular_forms	200	0.4913	0.2343
island_effects	700	0.5146	0.2449
npi_licensing	700	0.5088	0.2585
quantifiers	300	0.5142	0.2393
subject_verb_agreement	600	0.5050	0.2518
Average Weighted Score		**0.5010**	

Fig. 11: Results of the sBLIMP Benchmark

Aggregate Scores by Syntactic Feature

F. Comparing the Results

Comparatively, when examining the results from one epoch against two epochs, it is apparent that while certain categories show improved scores, the overall average weighted score has seen a marginal increase from 0.4977 to 0.5010. This indicates that while the model is progressively learning to differentiate between grammatical correctness in more challenging categories, the gains are incremental and highlight the need for further training or perhaps adjustments in model architecture or training parameters. Additionally, the performance on sentence pairs involving reflexive pronouns, such as "THAT NIECE COULDN'T DISTURB HERSELF" vs. "THAT NIECE COULDN'T DISTURB HIMSELF", remains low, underscoring persistent difficulties in the model's ability to generalize across syntactically diverse constructions. This comparison underscores the importance of extended training and refinement to overcome these limitations and achieve significant improvements in syntactic comprehension.

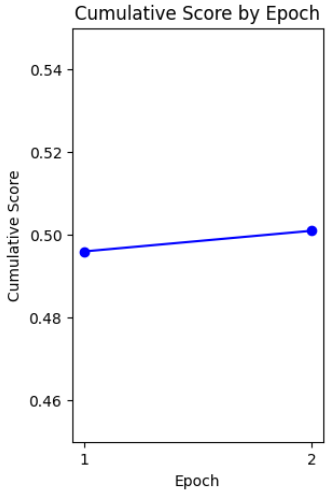

Fig. 12: Score Comparison for 1 and 2 epochs

G. Current Leader-board Chat

The results obtained for our model's syntactic processing capabilities, as shown in the "Syntactic dev" column, demonstrate a notable improvement compared to the random baseline and some of the other entries in the leaderboard. For instance, while the random baseline achieves a syntactic dev accuracy of approximately 49.35 percent, our model, particularly in the experiments led by Dr. Sriram Ganapathy and colleagues, significantly outperforms this baseline, reaching up to 51.96 percent in some configurations despite the low compute budget we had. This improvement underscores the effectiveness of the w2v-BERT2.0 integration in enhancing syntactic understanding. However, it is important to note that despite these gains, the results still indicate room for further optimization, as there are entries with higher performance, such as those

Sentence (Grammatical)	Sentence (Ungrammatical)	Score
ELLA LEADS HERSELF	ELLA LEADS ITSELF	0.7500
WALTER WAS HARMING HIMSELF	WALTER WAS HARMING ITSELF	0.5000
THIS LEGISLATURE EMBARRASSED ITSELF	THIS LEGISLATURE EMBARRASSED HERSELF	0.5000
CAROLINE DISGUSTS HERSELF	CAROLINE DISGUSTS ITSELF	0.7500
SPAIN ANNOYS ITSELF	SPAIN ANNOYS HERSELF	0.2500
THAT NIECE COULDN'T DISTURB HERSELF	THAT NIECE COULDN'T DISTURB HIMSELF	0.0000
APRIL CONFUSED HERSELF	APRIL CONFUSED HIMSELF	0.5000
MARIA WASN'T SCARING HERSELF	MARIA WASN'T SCARING ITSELF	0.7500
THOMAS REVEALED HIMSELF	THOMAS REVEALED ITSELF	0.2500

TABLE I: Aggregate Scores by Syntactic Feature for 1 epoch

Sentence (Grammatical)	Sentence (Ungrammatical)	Score
ELLA LEADS HERSELF	ELLA LEADS ITSELF	0.7500
WALTER WAS HARMING HIMSELF	WALTER WAS HARMING ITSELF	0.5000
THIS LEGISLATURE EMBARRASSED ITSELF	THIS LEGISLATURE EMBARRASSED HERSELF	0.5000
CAROLINE DISGUSTS HERSELF	CAROLINE DISGUSTS ITSELF	0.7500
SPAIN ANNOYS ITSELF	SPAIN ANNOYS HERSELF	0.2500
THAT NIECE COULDN'T DISTURB HERSELF	THAT NIECE COULDN'T DISTURB HIMSELF	0.2500
APRIL CONFUSED HERSELF	APRIL CONFUSED HIMSELF	0.5000
MARIA WASN'T SCARING HERSELF	MARIA WASN'T SCARING ITSELF	0.7500
THOMAS REVEALED HIMSELF	THOMAS REVEALED ITSELF	0.5000

TABLE II: Aggregate Scores by Syntactic Feature for 2 epochs

from Kim et al., who achieved syntactic dev scores of around 51.96 percent. This comparison highlights the competitive performance of our model while also indicating the potential for additional refinement and tuning to achieve even greater syntactic processing accuracy.

				Phonetic (Within)		Phonetic (Across)		Lexical		Syntactic	Semantic		Semantic (Weighted)	
#	Author	Budget	Set	clean	other	clean	other	all	in vocab.		synth.	libri.	synth.	libri.
4	**Random baseline**	0	dev	49.39%	49.97%	49.94%	50.17%	49.60%	49.73%	49.35%	1.48	6.79	1.48	6.79
			test	49.79%	49.37%	50.23%	50.11%	50.08%	50.06%	50.49%	0.17	6.44	0.83	0.55
35	Dr.Sriram Ganapathy et al.	60	dev	2.92%	4.50%	3.52%	6.99%	49.60%	49.73%	49.35%	6.80	7.16	6.80	7.16
			test	2.92%	4.36%	3.67%	7.17%	50.08%	50.06%	50.49%	0.24	0.07	0.76	2.08
37	Dr.Sriram Ganapathy et al.	60	dev	10.64%	13.16%	11.99%	17.57%	49.60%	49.73%	49.35%	6.80	7.16	6.80	7.16
			test	10.01%	13.39%	12.07%	18.68%	50.08%	50.06%	50.49%	0.24	0.07	0.76	2.08
38	Dr.Sriram Ganapathy et al.	60	dev	9.46%	12.35%	11.68%	17.63%	49.60%	49.73%	49.35%	6.80	7.16	6.80	7.16
			test	9.02%	12.34%	11.33%	18.44%	50.08%	50.06%	50.49%	0.24	0.07	0.76	2.08
39	Dr.Sriram Ganapathy et al.	60	dev	8.48%	10.34%	9.76%	14.22%	49.60%	49.73%	49.35%	6.80	7.16	6.80	7.16
			test	7.75%	10.78%	9.56%	15.04%	50.08%	50.06%	50.49%	0.24	0.07	0.76	2.08
17	Kim et al.	7	dev	7.17%	9.56%	9.30%	14.89%	51.68%	52.87%	51.96%	4.30	16.03	4.30	16.03
			test	6.50%	9.95%	9.04%	15.44%	51.36%	51.91%	50.43%	8.23	16.76	8.10	12.21
19	Kim et al.	7	dev	7.17%	9.57%	9.32%	14.94%	51.68%	52.87%	51.96%	1.05	20.29	1.05	20.29
			test	6.50%	9.95%	9.17%	15.46%	51.36%	51.91%	50.43%	2.40	14.92	4.51	5.54

Fig. 13: Leader board Comparison Table
Taken from: https://zerospeech.com/tasks/task$_4$/$results$/

VIII. Conclusion

In conclusion, the integration of the w2v-BERT2.0 tokenizer into our speech language model has demonstrated some promising improvements in various aspects of syntactic processing, as evidenced by the sBLIMP benchmark results. Despite being trained for a maximum of 2 epochs, the model performed better than some of the models as shown in the leaderboard in Figure 9.

IX. Recommendations

To further enhance the performance of our speech language model, we recommend the following strategies:

- **Code Verification and Testing:** Due to the complexity of the pipeline, particularly the crucial step of merging 10 split binary files into a single file for model training, rigorous verification and testing of the code are essential. Any errors in these processes can invalidate all results. Implementing automated unit tests, continuous integration, and code reviews will help ensure the integrity of the code throughout the development lifecycle, minimizing the risk of bugs and inconsistencies in the final output.
- **Extended Training Duration:** Increasing the number of training epochs to a range of 50 to 100 epochs could significantly improve the model's accuracy. Prolonged training allows the model to better learn and generalize from the vast dataset, leading to enhanced syntactic and phonetic processing capabilities.
- **Hyperparameter Optimization:** Systematically modifying and tuning the model's hyperparameters is crucial. We suggest conducting experiments to identify the optimal settings for learning rate, batch size, gradient accumulation steps, and other relevant hyperparameters. This process involves running multiple configurations and selecting the ones that yield the best performance on validation metrics.
- **Enhanced Computational Resources:** Given the substantial size of the binary files (13GB) and the intensive computational demands, it is essential to utilize high-performance GPUs. We recommend employing the NVIDIA A100-80G GPU or other comparable high-memory, high-compute GPUs. Relying solely on the

A100-20 GPU significantly prolongs the training process, making it impractical to run for the required number of epochs. Investing in more powerful hardware will expedite training and facilitate achieving the desired accuracy levels.

By implementing these recommendations, we anticipate substantial improvements in the model's performance, enabling it to meet and exceed current benchmarks in speech language processing tasks.

REFERENCES

[1] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli, Wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations, *ArXiv*, 2020. https://arxiv.org/abs/2006.11477.

[2] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin, A neural probabilistic language model, *Journal of Machine Learning Research*, 2003. http://www.jmlr.org/papers/volume3/bengio03a/bengio03a.pdf.

[3] Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and others, Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups, *IEEE Signal Processing Magazine*, 2012. https://ieeexplore.ieee.org/document/6296526.

[4] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le, Sequence to sequence learning with neural networks, *Advances in Neural Information Processing Systems*, 2014. https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019. https://aclanthology.org/N19-1423.

[6] Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman, BLIMP: A Benchmark of Linguistic Minimal Pairs for English, *arXiv preprint arXiv:1912.00582*, 2019. https://arxiv.org/abs/1912.00582.

[7] Sandra Cuervo and Ricard Marxer, Scaling Properties of Speech Language Models, *ArXiv preprint arXiv:2404.00685*, 2024. https://arxiv.org/abs/2404.00685.

[8] Zalán Borsos, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin, Mohammadreza Sharifi, Dominik Roblek, Olivier Teboul, David Grangier, Marco Tagliasacchi, and Neil Zeghidour, AudioLM: A Language Modeling Approach to Audio Generation, *ArXiv*, 2022. https://arxiv.org/abs/2209.03143.