Self-supervised learning for speech recognition on low-resource languages. (Application to Tunisian or other).

Nour Essayegh

Research project
Supervised by : Anis Hoayek
École nationale supérieure des Mines de Saint-Étienne
2022

***Abstract*— This research paper aims at exploiting self-supervised methods to build a speech recognition model on Tunisian dialect.**

In this paper, the adaptation of automatic speech recognition model will be the main interest. Using different training methods after manipulating and preparing various audio data (Labeled and unlabeled) to improve the word error rate for such low resource language. This research not only aids as an speech recognition model for Tunisian dialect but also for several other dialect as soon as the amount of data gathered is acceptable.

***Index Terms*— Automatic speech recognition, self-supervised training, low resource language, Voice activity detection, XLS-R, Connectionist Temporal Classification.**

I. INTRODUCTION

Arabic is a Semitic language among the oldest in the world. The evolution of this language from antiquity to the present day has given birth to several linguistic registers; some forms have disappeared and others are still spoken. Four principle linguistic registers can be highlighted corresponding to the ancient Arabic language history.[1]

1) Old Arabic, which is not used currently and can be found in ancient literacy work
2) Classical Arabic, which is the language of Islams Holy Book
3) Modern Standard Arabic, the official language of all the Arab countries
4) Dialectal Arabic, which is the mother tongue of every Arabic speaker and used in daily life for spoken communication, changing from one Arab country to another.

A language will be called an under-resourced language if it has the following characteristics: lack of a unique writing system or stable orthography, limited presence on the web, lack of linguistic expertise, lack of electronic resources for speech and language processing,[2] vocabulary lists, etc. Dialects in general and Dialectal Arabic more specifically is an example of this category of languages since it is not standardized and does not usually have official status. Even though Modern standard Arabic is taught in schools and written conversations, most Arabic speakers are unable to produce sustained spontaneous speech in modern Standard Arabic in unscripted situations, it resorts to a mix between their dialect and standard Arabic.

In Natural Language Processing (NLP) context, Standard Arabic has had its share of attention. Several sophisticated tools had been developed regarding the language but it has been proved not to be able to effectively model Arabic dialects. From this problem emerges the necessity to construct dedicated tools to process dialectal Arabic. Working with dialects can have a huge positive impact on the standardization and preservation of their structure and identity. Voice activation algorithms using more complex language can also be a long-run perspective since using voice assistants has become ubiquitous with the popularity of Google Home, Amazon Echo, Siri, Cortana, and others.

The objective of this project is to produce a fairly effective speech recognition model on Tunisian dialect. The development of such a system faces the challenges of the lack of annotated resources and tools, apart from the lack of standardization at all linguistic levels (phonological, morphological, syntactic, and lexical). This project is therefore part of a research perspective to succeed in transferring low-density audio files to an adaptable language model. For that we are going to train our baseline on labeled and unlabeled data to get the best Word error score possible.

II. LITERATURE SURVEY

In recent decades, researchers have been increasingly interested in automatic speech recognition (ASR) [6]. ASR began with simple systems that responded to a limited number of sounds and has evolved into sophisticated systems that respond fluently to natural language. Because of the need and desire to automate simple task or be able to be understood by machines there has been increasing interest in ASR technology [7]. ASR can generally be defined as the process of deriving the transcription of speech, known as a word sequence, in which the focus is on the shape of the speech wave [6]. Currently, ASR is widely applied in many functions, such as weather reports,

automatic call handling, stock quotes, and inquiry systems [7]. Recently, human-machine communication has improved widely by the introduction of neural network. This introduced the capacity to tack into account speech from a large amount of users with a large spectrum of vocabulary and languages. There also excites more and more techniques to speech processing were the user's voice is more easily recognized in an isolated recognition system by having the speech uttered word for word with pauses in between them.This was first developed using an energy based approve of the speech signal but with the neural network.[4] Working on low resource languages makes the task harder. Some studies using different training techniques as semi supervised training (SST)[8] based only on unlabeled data to improve the word error rate in future test with out-of-domain data meaning improve the labeling and speech recognition of written languages. Another study[1] that was more focused on low resource languages and especially Tunisian dialect.They presented a historical overview of the Tunisian dialect and its linguistic characteristics. A basic 3-gram language model[9] was used on a labeled data set with a Word Error Rate of 22.6%. We will be using the same labeled data set for a part of the training of our model.

III. METHOD

A. Model Architecture

1) ***The self-supervised model : XLS-R****:* Our model was based on XLS-R, a large-scale model for cross-lingual speech representation learning based on wav2vec 2.0[5]. Because of the limits of the large amount of annotated data needed by the traditional speech recognition models, self-supervised learning is more promising since it can leverage unlabeled data to build better systems.

Firstly, we need to understand **wav2vec 2.0**. The audio signal in any language will be put into a multi-layer convolutional feature encoder to get latent audio representation($Z_1, \ldots, Z_T$ for T time-steps).After fed into the transformer g, we get the representation capturing the information of whole sequence($C_1, \ldots, C_T$). Besides, with the help of a quantization model, we can discretize the output of the feature encoder to a finite set of speech representations via product quantization, which is beneficial in learning contextualized representations[10].

- **Feature encoder :**
  The encoder consists of several blocks containing a temporal convolution followed by layer normalization and a GELU activation function. The raw waveform input to the encoder is normalized to zero mean and unit variance. The total stride of the encoder determines the number of time-steps T which are input to the Transformer.
- **Contextualized representations with Transformers :**
  The output of the feature encoder is fed to a context network that follows the Transformer architecture. Instead of fixed positional embeddings which encode absolute positional information, there is a convolutional layer similar to which acts as relative positional embedding. Also, the output of the convolution is followed by a GELU to the inputs and then apply layer normalization.

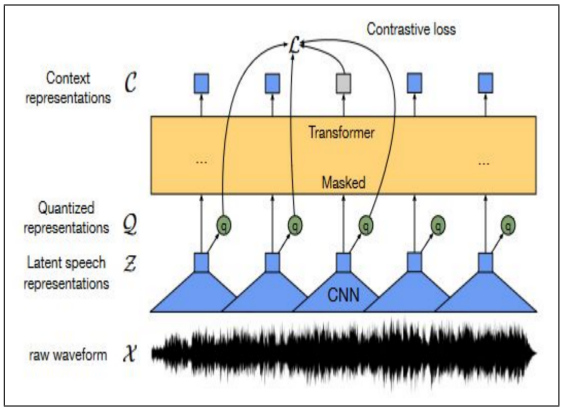

Fig. 1. Architecture of the model

The model learns contextualized speech representations by randomly masking feature vectors before passing them to a transformer network during self-supervised pretraining.

In order to deal with limited unlabeled data in some languages,cross-language training is necessary. Pretraining a single model on multiple languages at the same time, outperforms the single model trained on the single language.

Trained on more than 436,000 hours of publicly available speech recordings, XLSR outperforms the previous best model on the aspect of automatic speech recognition by multiple languages. In the picture below, XLSR is evaluated on four major multilingual speech recognition benchmarks(five languages of BABEL, 10 languages of CommonVoice, eight languages of MLS, and the 14 languages of VoxPopuli).

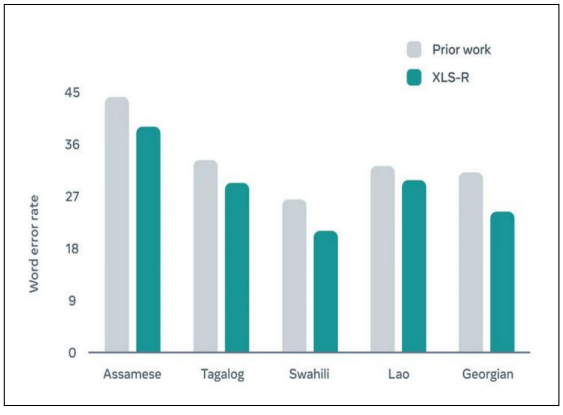

Fig. 2. Evaluation of XLSR on multiple languages

Fine-tuning To get more precise language representations of the language and making the link between the phonemes and the written phrases, we are using the

labeled data set to fine-tune the XLSR model. For the fine-tuning, a single linear layer is added on top of the pre-trained network to train the model on labeled data. ASR models transcribe speech to text, which means that

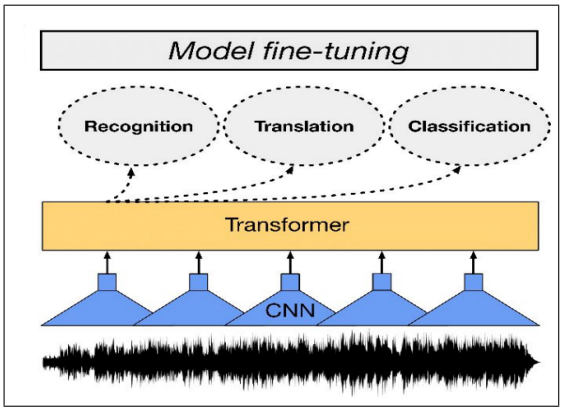

Fig. 3. Fine-tuning XLSR

we need :

- Tokenizer that processes the model's output format to text
- Feature extractor that processes the speech signal to the model's input format, e.g. a feature vector

The tokenizer For speech recognition, the model has to map this sequence of context representations to its corresponding transcription which means that a linear layer has to be added on top of the transformer block This linear layer is used to classify each context representation to a token class analogous to how a linear layer is added on top of BERT's embeddings for further classification after pre-training

For the tokenizer, We started by building a vocabulary file that contains the letters used in the phrases of the data set, the tokens start and end of a sentence, and an "unknown" token so that the model can deal with characters not encountered in the data set.

In a final step, we use the file to load the vocabulary into an instance of the Wav2Vec2CTCTokenizer class

The feature extraction Because of the continuity of speech signal, we need to discretize the signal—-sampling. The sampling rate is very important since it defines how many data points of the speech signal are measured per second. As a result, the higher the sampling rate is, the better approximation of the real speech signal is.

The pipeline for feature extraction and the processor combining the tokenizer and the feature extraction.

The preprocessing of the data is a fundamental step in the process, meaning we needed to extract the audio file as an array and get it sampling rate and other information before going through the training.

*2) **Connectionist Temporal Classification(CTC)**:* Since we are working on audio files, the supervised training part of the project needed an additional extension. Considering that labeled audio data can bring numerous issues due to the alignment of the audio file and it transcription.

First of all,we don't know how the characters in the transcript align with the audio. Maybe you can consider making a rule that "one character corresponds to twenty inputs". But that also does not work. Here is an example: as you can see in the Figure 4, people's rates of speech vary. Some people speak fast while some speak slowly even for the same sentence. So it is difficult to align the transcript to the audio. Here where comes the impotence of the Connectionist Temporal Classification (CTC) algorithm[11].

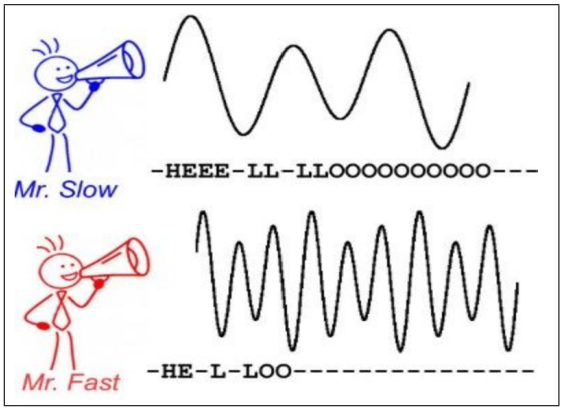

Fig. 4. People's rates of speech vary.

We use the CTC algorithm principally to calculate the loss function, for training model. CTC refers to the outputs and scoring, and is independent of the underlying neural network structure. So we can define our own model structure and add CTC part at the end. CTC loss scores can then be used with the back-propagation algorithm to update the neural network weights. The CTC algorithm works mainly on three majors:

- Encoding the text
- Loss calculation
- Decoding the text

We consider an input sequence of an audio $X = [x_1, x_2, ..., x_T]$, the corresponding output transcripts sequences $Y = [y_1, y_2, ..., y_u]$. The CTC algorithm introduces an output conditional distribution given an input X over all possible $Y's$ elements. So the CTC algorithm uses this conditional distribution given an input but does not assess the probability of a given output.

- Encoding the text :

When the characters are repeated more than one time-step in the image, The non-CTC methods would not work and give repeating characters in the result.

The CTC algorithm takes the way that firstly aligns input X and output Y by assigning an output character to each input time step and then it merges all the repeating characters into a single character, as shown in Figure 5.

But this way will cause one problem:

What about the words where there are duplicate characters?

For example, when we consider the word "hello". It has a repeating "l" and it will give a wrong result "helo" but not "hello".

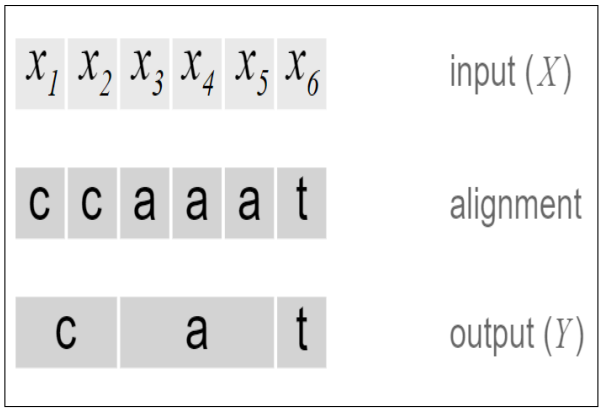

Fig. 5. The alignment for CTC methods

So the CTC introduces a specific character “epsilon” ϵ as a token. As shown in Figure 6, this token ϵ does not correspond to anything and is placed between the characters in the output text. Finally we just simply remove them from the output and get the target transcript.

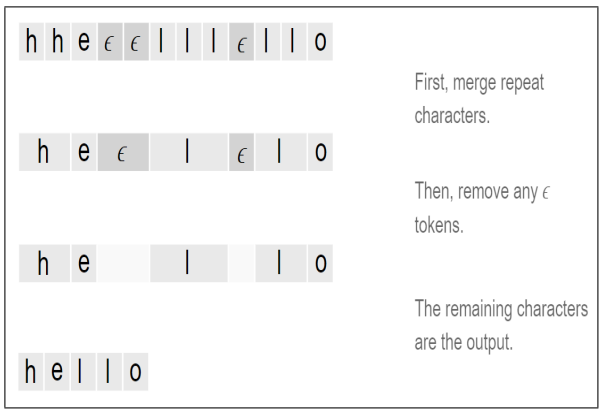

Fig. 6. The alignment with token ϵ for CTC methods

So CTC handles successfully the problems in the alignment by introducing a blank token ϵ.

- Loss calculation : Firstly, to calculate the loss value for the CTC algorithm, we must know the conditional distribution at each time-step. The CTC algorithm uses the conditional distribution given an input, noted as $p_t(a_t|X)$, X is input and a_t is output characters at each time-step. After having defined the conditional probability over the output character at each time step, we can calculate the CTC conditional probability. Here is the equation:

$$P(Y|X) = \sum_{A \in A_{X,Y}} \prod_{t=1}^{T} p_t(a_t|X) \quad (1)$$

The sum refers to all the possible aligment referred as $A_{X,Y}$ and the multiplication refers to the probability of each character in one alignment. So through this method, the CTC algorithm gives us a way to go from probabilities at each time-step to the probability of an output sequence. We can efficiently compute the conditional probability $P(Y|X)$. $P(Y|X)$ can also be differentiable, so we can use gradient descent to train the model.

- Decoding the text :

After having trained the model, the objective is to use it to obtain the most likely transcript given an output conditional probability matrix. As you can see the equation below:

$$Y^* = \underset{Y}{argmax} \quad P(Y|X) \quad (2)$$

Decoding the text by examining every potential output is very time consuming.

One solution that can be considered is “the best path” or also called "the greedy algorithm". It means that at every time step, it calculates the best alignment by considering the character with max conditional probability. The equation is indicated below[11]:

$$A^* = \underset{A}{argmax} \prod_{t=1}^{T} p_t(a_t|X) \quad (3)$$

We can also consider more advanced decoders like the "beam search decoding", "prefix-search decoding". They also make use of language structure information to get better results.

IV. DATA PREPROCESSING

The main difficulty while working with under-resourced languages is the lack of annotated resources and standardization at all linguistic levels. That is why choosing carefully our data sets is a fundamental step in the process.

For most under-resourced languages, there are no existing corpora that can be used for the development of ASR systems. Therefore, data collection is generally an integral part of ASR development in these languages. In order to train such models very large amounts of speech and textual data are generally required.

A. Labeled data

The data set used in the project is from the Tunisian Arabic Railway Interaction Corpus (TARIC). The general architecture of a standard ASR system integrates three main components: acoustical (acoustic-phonetic) modeling, lexical modeling (pronunciation lexicon/vocabulary), and language modeling. Any state-of-the-art ASR system works in two modes: model training and speech decoding.

To develop ASR systems an amount of annotated resources is needed even if it’s limited. The corpus used is gathered in collaboration with the Tunisian National Railway Company (SNCFT) and corresponds to recordings of information requests in railway stations. It was created to be used in similar projects and made open source. This corpus, called TARIC (Tunisian Arabic Railway Interaction Corpus), contains information requests in the Tunisian dialect about the railway services such as train type, train schedule, train destination, train path, ticket price, and ticket booking. TARIC corpus is created in three

stages: first, audio recording; second the orthographic transcription, and finally the transcription normalization.

It contains more than 21k statements with a vocabulary of 3207 words for over 10 hours of audio data. 108 different speakers were recorded where 60 are males and 37 are females. The data set came into separate audio files paired with a text document that contains the transcription and start and end time of each speech segment.

B. Unlabeled data

For the unlabeled data, it was gathered from different Tunisian TV shows replays on YouTube. It was the most efficient way to get a good amount of data that can be used. Only the audio tracks were extracted and each file corresponded to the audio of a single episode of a specific Tunisian TV show. Over 450 hours of audio content was gathered to build the unlabeled database. This corpus can be easily extended regarding the among of available clips on the internet.

The difficulty with this unlabeled data is that it requires pretreatment since it contains a lot of noisiness and some music segments that can be confused with speech and influence our evaluation score on the training part. The challenge was to find a good Voice activity detection model that can isolate the speech segment and ignore music or random noise in the audio files.

Voice activity detection :

Voice activity detection (VAD) is the detection of the presence or absence of human speech, used in speech processing. Since speech signals are not stationary, the signal energy can be used as an indicator for speech presence. Speech adds energy to the signal, such that high-energy regions of the signal are likely speech.

The VAD is a binary operation, either we have speech of not. From that we can define a threshold λ such that when the energy of the signal e(x) is above the threshold, the VAD indicates speech activity.

$$VAD(x) = \begin{cases} 0, \text{if} & e(x) < \lambda \\ 1, \text{if} & e(x) >= \lambda \end{cases} \tag{4}$$

The issue with this classical method is that it will be very sensitive to each change of tone or background noise and music. Since building our data set is a fundamental step in training our baselines we explored machine learning-based methods for voice activity detection. A library treating this problem is pyannote.audio [4].

VAD with Pyanote.audio :

Pyannote.audio is an open-source toolkit written in Python and based on PyTorch machine learning framework. This library provides a set of trainable end-to-end neural building blocks that can be combined and jointly optimized to build speaker diarization pipelines. It also comes with several pre-trained models but we will be interested in the voice activity detection pre-trained model.

The voice activity detection model is based on bidirectional-RNN which makes this method more efficient than the energy-based method.

In fact, a standard recurrent neural network (RNN) can also deal with sequential data. The particularity of this kind of network is that the information cycles through a loop. When it makes a decision, it considers the current input and also what it has learned from the inputs it received previously.

Given an input sequence $x = (x_1, ..., x_T)$, a standard recurrent neural network computes the hidden vector sequence $h = (h_1, ..., h_T)$ and output vector sequence $y = (y_1, ..., y_T)$ by iterating the following equations from t = 1 to T:

$$h_t = H(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \tag{5}$$

$$y_t = W_{hy}h_t + b_y \tag{6}$$

where the W terms denote weight matrices (e.g. W_{xh} is the input-hidden weight matrix), the b terms denote bias vectors (e.g. b_h is hidden bias vector), H is the hidden layer function and T being the sequence length .

So RNN has proven to be efficient for short sequential data it has been found that the Long Short-Term Memory (LSTM) architecture, which uses purpose-built memory cells to store information is better at finding and exploiting long-range context.

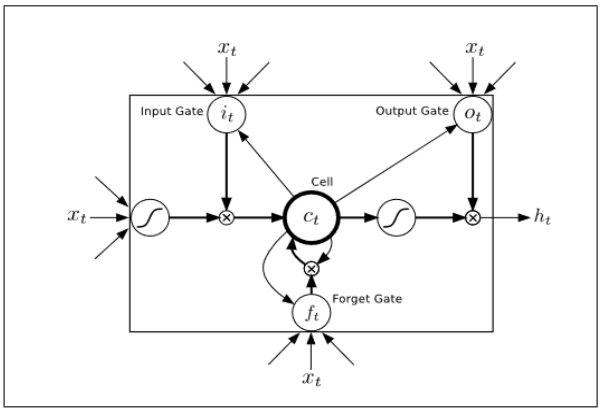

Fig. 7. Long Short-term Memory Cell

Since Conventional recurrent networks only make use of the previous context and consider that with audio data there is no reason not to exploit future context as well. Bidirectional RNNs (BRNNs) come in handy by processing the data in both directions with two separate hidden layers, which are then fed forwards to the same output layer[3].

As seen in figure 8 the outputs of the Bidirectional RNNs are fed forward into the Output Network. This network transforms what was learned by the recurrent network into a value between 0 and 1 that can be interpreted as the probability that the audio clip contains speech or not. The output network has only one hidden layer with tanh activation functions and a logistic activation function for the single output of the final layer.

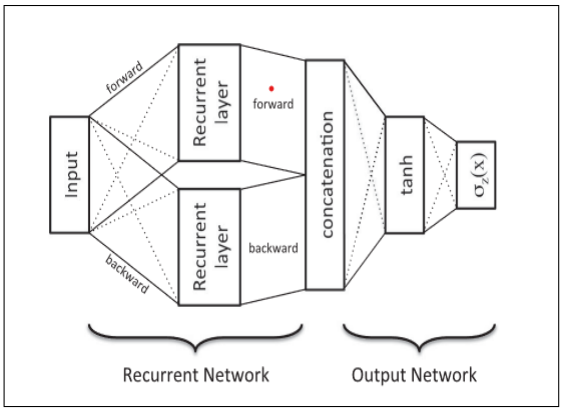

Fig. 8. Recurrent network and output Network

The Voice Activity Detection pipeline based on the Bidirectional RNNs has proven to be very efficient despite background music or change of tone in input audio speech files for example, and that is exactly what we need for our data preparation regarding the unlabeled data since we need to be able to identify and extract only the speech segments.

V. WORK DONE AND RESULTS ANALYSIS

Preparing the training :

The first step for us was to gather the unlabeled data. Youtube was our first source. Over 600 hours of audio data was gathered. We had to use the Voice activity detection model to extract The speech segment. The generating the filtered corpus took over 6 hours of training. We tested over 20 min of output files manually to evaluate the accuracy of our voice activation model. This data set was then used to train our baseline XLS-R and learn the phonetic representations of our language in addition to the previous representation learned before.

After this step we had to train our model on the Labeled data. For that we used speech brain,an open-source and all-in-one conversational AI toolkit based on PyTorch. This toolkit provides various useful tools to speed up and facilitate research on speech and language technologies.

Speech brain enabled us to combine our trained ASR model with CTC. To run the experiment we needed to prepare the input files needed.

- One csv file containing the series of information
- All separated audio clips containing only one sentence or oral expression

We had to separate our labeled data into small audio containing only one sentence and build a csv file with all the information needed. We were able to extract starting and ending time of each segment of speech and this was the why we obtained short separated audios. From the 'txt' we then built our CSV file that contains the sentences, duration path and ID and the phonetics of each small audio file.

The processing of this data to enable speechbrain to train on the labeled data is the following :

1) We had a "txt" format information file containing the words for the sentences, the starting and ending times. We used a library called "re" which is based on "regular expression" to extract directly the useful time information we mentioned above.
2) After having the starting and ending times, with the library "librosa" with a sampling rate at 22050, we used these time information to separate the long original audios into short separated audio clips.
3) And then, we used the "phonetis_arabic" library to obtain the phonetics for each separated audio clip containing one single sentence.
4) After giving each new audio clip a unique ID, All the information were combined into a CSV file.

After having the training data ready, We were able to run the experiments using a YAML format file to set hyper-parameters. The YAML file contained all the information to initialize the classes when loading them, like learning rate, batch size, and also structure of the Neural network model, etc.

Results and interpretations :

After several attempts and a lot of technical difficulties regarding the GPU memory of the virtual machine provided to us by OVHcloud we manage to fixe a small batch size to run our baseline. The best result for our ASR based on CTC is **29.7%** WER(Word error rate) tested on the test data set and **29.2% WER** on the dev data set. According to the results of ASR system using TARIC data set (the same data set we use in labeled data set) for the Tunisian dialect in the article [1], the result achieves at 21.5% WER in dev data set and at 22.6% WER in test data set. The comparison is legit since we are working on the same data set, still our result are slightly worse that those presented with a simple 3-gram algorithm used in article [1]. We can try to justify those result with some factors that might seem off : After spending a lots of time adapting the code and the libraries on Virtual Machine, the code was able to run with a small batch size since even with a virtual machine we faced limitations regarding the GPU memory. In addition to this technical limitation we can identify other hypothetical reason of our less performing model.

- We used directly a library called "phonetis_arabic" to get the phonetics (the input is the separated audio clips; the output is the phonetics corresponding to the sentence in audio clips). But we weren't able to evaluate the accuracy of the phonetics on Tunisian dialect.
- There might still be some noises in the audio clips. For the labeled data, the recording of the audios happened in a train station. Some noises in the background can have also a bad influence on the training result.
  For the unlabeled data, we might need in addition to the voice activity detection model used a denoising neural network model able to only filter the speech

and not the background music for example.

VI. CONCLUSIONS AND PERSPECTIVES

Working with Tunisian dialect was a challenge by itself but being able to explore some existing baseline and adapting it to a new language with limited data was a very interesting way to approach Natural Language Processing(NLP) for low resource languages. We were able to obtain an acceptable score of 29.7% WER on the test data. This opens up the door to exploring different dialects and other low resource languages in general. We still had some technical problems on different level throughout the working process that's why some parameter tuning and optimization is still necessary.

REFERENCES

[1] A. Masmoudi, F. Bougares, M. Ellouze, Y. Esteve, L. Belguith,(2018) "Automatic Speech Recognition System for TunisianDialect", Language Resources and Evaluation, Springer Verlag

[2] L. Besacier, V-B. Le, C. Boitet, V. Berment (2014). ASR AND TRANSLATION FOR UNDER-RESOURCED LANGUAGES

[3] A.Graves, N. Jaitly and A. Mohamed(2013),HYBRID SPEECH RECOGNITION WITH DEEP BIDIRECTIONAL LSTM,[Online]. Available: https://www.cs.toronto.edu/~graves/asru_2013.pdf

[4] Tom Bächstöm (2019) Voice activity detection (VAD) - Introduction to Speech Processing, Aalto University Wiki.[Online]. Available: https://wiki.aalto.fi/pages/viewpage.action?pageId=151500905

[5] Meta AI (2018) XLS-R: Self-supervised speech processing for 128 languages. [Online]. Available: https://ai.facebook.com/blog/xls-r-self-supervised-speech-processing-for-128-languages/

[6] D. Yu and L. Deng, Automatic Speech Recognition. London, U.K.: Springer, 2016.

[7] B.-H. Juang and L. R. Rabiner, "Automatic speech recognition—A brief history of the technology development," Georgia Inst. Technol., Atlanta Rutgers Univ. Calif. St. Barbara, Tech. Rep., 2005, p. 67, vol. 1.

[8] J. Billa,(2021) "Improving low-resource ASR performance with untranscribed out-of-domain data" [Online]. Available: https://arxiv.org/pdf/2106.01227.pdf

[9] N-gram Language Models, Stanford university [Online]. Available: https://web.stanford.edu/ jurafsky/slp3/3.pdf

[10] A. Baevski, H. Zhou, A. Mohamed, M. Auli,(2020) "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations" [Online]. Available: https://arxiv.org/pdf/2006.11477.pdf

[11] Harald Scheidl, "An Intuitive Explanation of Connectionist Temporal Classification", TDS, (2018)

[12] H. Bredin, R. Yin, J. Manuel Coria, G. Gelly, P. Korshunov,M. Lavechin, D. Fustes, H. Titeux, W Bouaziz, M. Gill, Pyannote.audio: Neural building blocks for speaker diarization, 2019.

[13] G. Gelly and J. Gauvain,Optimization of RNN-Based Speech Activity Detection, 2018.

[14] Hannun, "Sequence Modeling with CTC", Distill, 2017.

[15] X. Kong, J. Choi, S.Shattuck Hufnagel, "Evaluating Automatic Speech Recognition Systems in Comparison With Human Perception Results Using Distinctive Feature Measures", IEEE, 2017. [Online]. Available:https://arxiv.org/abs/1612.03990

[16] P. von Platen, Fine-Tune XLSR-Wav2Vec2 for low-resource ASR with Transformers, 2021. [Online]. Available:https://huggingface.co/blog/fine-tune-xlsr-wav2vec2