Cross Modal Knowledge Transfer on Sentiment Analysis using Multi Modal Models

Krishna Chaitanya Chinnam

Master of Science

Cognitive Science

School of Informatics

University of Edinburgh

2023

Abstract

In a world where agents based on large language models (LLMs) play increasingly crucial roles in our daily lives in the form of chatbots, virtual assistants, and content generation platforms, low-resource languages (LRLs) and, in turn, their speakers are at an ever-increasing disadvantage. While many solutions have been proposed to this problem, ranging from dataset collection to automatic data augmentation to cross-lingual knowledge transfer, we firmly believe that cross-modal and cross-lingual knowledge transfer represents another valuable and relatively underexplored avenue in addressing this issue. To this end, we explore two multi-modal models, SpeechT5 and SpeechLM for signs of cross-modal knowledge transfer for sentiment analysis. Despite our efforts, our results are inconclusive about the viability of cross-modal transfer for sentiment analysis in SpeechT5. This report encapsulates the project's methodology, experiments, their outcomes and our subsequent analysis along with the challenges encountered along the way. Furthermore, we outline potential directions for extending our work and also possible future research in the domain.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Krishna Chaitanya Chinnam*)

Acknowledgements

I would like to thank my supervisor Prof. Edoardo Ponti for his constant guidance both on and off the project. I would also like to thank Ramon Sanabria for setting up the project, helping me understand what I was supposed to be doing and always being willing to meet and talk about any stupid questions I might have. The hyperparameter tuning part of this report is based on work done by my project partner Max Leftley.

In addition, I would also like to thank my friends near and far who keep me from getting lonely in a new place. And finally, I would like to thank my parents, brothers and family who've always believed in me and my potential more than they actually should.

Table of Contents

ãñã́ kɨ̃ɨ̃́re baka-yígɨ
‘A snake bit him’. (Part of a legend.)

Chapter 1

Introduction

The above phrase comes from the Tuyuca language. It is spoken in Brazil and Colombia by the indigenous Tuyuca people. An interesting feature of the Tuyuca language is that verbs in the language often contain evidentials (Barnes 1984). In linguistics, “evidentials” refer to grammatical markers or structures found in some languages and are used to identify the source, manner or strength of evidence related to information in a sentence. This means the sentence doesn’t just relay information but talks about its source too. In this case, it is marked as second-hand information because it comes from a legend. This makes translating from Tuyuca to languages like English very difficult.

In addition to this Tuyuca is only spoken by about 1440 people and is classified as *Moribund* [noun: in terminal decline] by the Ethnologue (Eberhard, Simons, and Fennig 2022). And with very few experts who can translate to and from the language, it is extremely difficult to generate enough text, labelled or unlabelled, to train decent NLP models in the language. This is a story of not just Tuyuca, but of the more than 6000 low-resource languages (LRLs) spoken around the world (Magueresse, Carles, and Heetderks 2020). While there is no one agreed-upon definition of what classifies as a low-resource language, they are often characterized by scarcity of labelled data, limited online presence and lack of expertise. In the field of speech and language processing, these are languages that lack sufficient data to train statistical models on.

Many data augmentation methods have been proposed to overcome the scarcity of data in LRLs that range from automatically aligning text in LRLs to high resource languages using word-level (Saralegi, Manterola, and San Vicente 2011; Nasution, Murakami, and Ishida 2016), sentence level (Shchukin, Khristich, and Galinskaya 2016; Huang et al. 2016), and document level (El-Kishky and Guzmán 2020) alignment, to manually collecting and labelling more data (Strassel and Tracey 2016; Simpson et al.

2008). In addition, techniques like word substitution, back translation and dictionary induction are also used for data augmentation (Xia et al. 2019).

Other methods have focused on developing models that can make use of what little labelled data is available via transfer learning from high-resource languages to low-resource languages (Zoph et al. 2016) or developing multilingual models pre-trained on unlabelled data from multiple languages and then fine-tuned on languages where labelled data is available (Pires, Schlinger, and Garrette 2019).

In the field of speech, high-quality annotated recordings are used to train acoustic models, which are often absent for LRLs. Making this problem worse is the current prevalence of sequence-to-sequence deep neural networks like FastSpeech 2 (Y. Ren, Hu, et al. 2020) and Tacotron 2 (Shen et al. 2018) that require substantial amounts of training data compared to conventional statistical parametric speech synthesis. Models that are trained on multi-lingual data like XLSR (Conneau et al. 2020), which is pre-trained on speech from 53 languages, have shown that cross-lingual knowledge transfer is possible from high-resource languages like English to low-resource languages like Swedish and Turkish.

Multimodal models that handle both text and speech as both input and output, like SpeechT5 (Ao et al. 2021) and MAESTRO (Zhehuai Chen et al. 2022) have shown that joint training on both speech and text during pre-training can help in building models that outperform their uni-modal counterparts.

Given these advancements, it is not impossible to imagine the possibility of multi-modal models trained on both text and speech that are capable of both cross-modal and cross-lingual transfer. To this end, we try to probe the possibility of cross-modal and cross-lingual transfer for sentiment analysis and utilize SpeechT5 as the primary model for our experiments. This project focuses on cross-modal transfer, reserving the exploration of cross-lingual and cross-modal transfer for future endeavours.

The rest of this report is broken down into four chapters. Chapter 2 covers previous work related to knowledge transfer and also gives an overview of the models we use in our work. Chapter 3 describes our setup, datasets and frameworks and also describes the changes we made to these existing models. Chapter 4 focuses on our experiments, results and their analysis. Chapter 5 concludes the report and presents avenues for future research. Appendix A contains information about the code for the project.

Chapter 2

Background and Related Work

2.1 Knowledge Transfer

Knowledge Transfer or Transfer Learning is the process of transferring knowledge from a source task (or domain) to a target task. Depending on the source/target task it can be classified into different types like Transductive Transfer Learning and Inductive Transfer Learning.

Transductive transfer learning is when we have enough labelled data for the source task but very little or none for the target task. This has applications in domain adaptation where knowledge learned in one domain (say model trained on recognizing digits on letter posts) is used in another domain (like reading signposts). It is also helpful in cross-lingual transfer when transferring knowledge from a language with a lot of resources (like English) to a low-resource language (like Inuktitut).

Inductive transfer Learning on the other hand is when there is labelled data available for the target task but not for the source task. Word embeddings like word2vec and the pre-training followed by fine-tuning approach that is used in models like BERT (Devlin et al. 2018), RoBERTa (Y. Liu et al. 2019) and XLNet (Zhilin Yang et al. 2019) are both examples of this.

What we are trying to achieve in this project is an example of both these types of transfer learning. We want to show transfer learning in a cross-modal and cross-lingual setting which is an example of both domain adaptation (text to speech or speech to text) and cross-lingual transfer (both instances of transductive transfer learning). But before we check for knowledge transfer across languages and domains, we need to fine-tune the pre-trained model for specific tasks for which we want to probe transfer, which is an instance of inductive transfer learning.

2.2 SpeechT5

Most of the experiments in this project build on the SpeechT5 model developed by Ao et al. (2021). We explain the basic structure of the model here so that the modifications we made and subsequent experiments can be understood more fully.

Inspired by the success of pre-training and fine-tuning methods in models like T5 (Raffel et al. 2020) the authors of SpeechT5 develop a unified speech and text model that can take either text or speech as input and output text or speech (or classes for classification). In order to achieve this the model uses what it calls pre-nets and post-nets (Fig 2.1).

2.2.1 Pre-nets

Pre-nets convert speech and text into a common representation. This common representation is learned during the pre-training phase using cross-modal vector quantization. The pre-nets are modal specific and there are 4 in total in the model: two speech pre-nets and two text pre-nets (one for the encoder and one for the decoder).

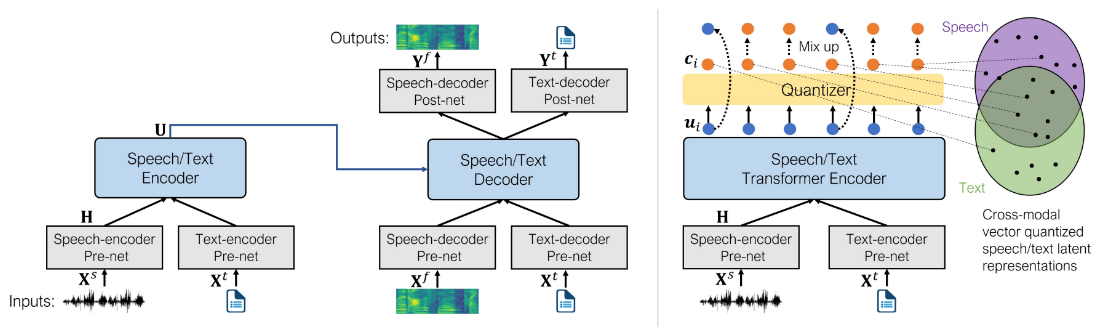
Speech pre-net for the encoder takes the raw waveform which is then passed through a CNN to extract speech representations. This is based on the convolutional feature extractor found in wav2vec 2.0 (Baevski et al. 2020). While the input speech for the encoder is in the form of raw waveforms the output speech is generated using a vocoder from the log Mel-filterbank output. Since the input to the decoder is the output of the decoder from the previous step the speech decoder pre-net is configured to take log Mel-filterbanks as input.

Shared embeddings are used for text pre-nets for both the encoder and the decoder. The pre-nets transform token indices into vector embeddings.

2.2.2 Post-nets

Speech decoder post-net takes the output of the decoder and outputs log Mel-filterbanks. These are then passed through 5 convolutional layers to generate residuals that are used to improve the predicted log Mel-filterbanks. The speech decoder post-net also has a module that uses the decoder output to predict the stop token.

Text decoder post-net takes the decoder output and predicts a probability distribution over all the possible tokens which is then normalized using a softmax function.



(a) The model architecture of SpeechT5 (b) The joint pre-training approach

Figure 2.1: Architecture and pre-training for SpeechT5 (Ao et al. 2021)

2.2.3 Training

Pre-training is done on text and speech separately. In the case of speech, it is trained on two types of tasks: bidirectional masked prediction and sequence-to-sequence generation. Bidirectional masked prediction is inspired by HuBERT (Hsu et al. 2021) and involves masking the input to the transformer encoder (output of speech encoder pre-net) and calculating loss over hidden representations output by the transformer. For sequence-to-sequence generation, the model takes masked input similar to the previous task and tries to generate the original speech which is output by the speech decoder post-net.

For text, we use a similar approach where the model is trained to reconstruct the original text. Adopting the text-infilling strategy of BART (Lewis et al. 2019) 30% of the text spans are masked and replaced by a mask token and the loss is calculated on the reconstructed text.

The model does not make use of labelled data to align text and speech. Instead, it uses a joint pre-training approach. Aligning speech and text representations is necessary for tasks like ASR (Automatic Speech Recognition) and TTS (Text-to-Speech). The model uses vector quantized embeddings to align speech and text representations using a shared codebook. Based on VQ-VAE (Van Den Oord, Vinyals, et al. 2017) and SemFace (S. Ren et al. 2021) the model maps continuous representations to discrete ones using nearest neighbour search. Some of the continuous representations are then replaced by the discrete ones when calculating cross-attention. The final loss is the sum of all the losses calculated for the mentioned tasks. This encourages the model to make use of cross-modal information.

2.2.4 Performance

One major advantage that a text-to-text encoder-decoder model like T5 has over an encoder-only model like BERT is that the text-to-text setting lends itself to easy fine-tuning without having to change the architecture based on the task. It also helps with transfer learning across tasks. In a similar vein authors of SpeechT5 fine-tune and evaluate the pre-trained model on a variety of tasks like ASR (Automatic Speech Recognition), TTS (Text-to-Speech), ST (Speech Translation), VC (Voice Conversion), SE (Speech Enhancement) and SID (Speaker Identification) and report state-of-the-art and close to state-of-the-art results in many of these tasks.

2.3 SpeechLM

Another model that we consider briefly is SpeechLM (Z. Zhang et al. 2022). It was developed by the researchers behind SpeechT5 and is designed to overcome some of the problems associated with SpeechT5. The main difference is that instead of using pre-nets to convert speech and text to a common representation, it makes use of a shared transformer paired with tokenizers that convert speech and text into the same discrete space.

2.3.1 Tokenizers

The authors make use of two alternative tokenizers: Phoneme-Unit tokenizer and Hidden-Unit tokenizer (Fig 2.2). The phoneme-unit tokenizer, inspired by PBERT (C. Wang et al. 2022), is used for discretizing both speech signal and text sequences. The speech tokenizer is comprised of an acoustic model that converts acoustic features into phoneme units. For text data, provided lexicons can be used to directly convert words to phonemes. These phoneme sequences are then upsampled by randomly repeating the phonemes so that the sequence length resembles that of speech.

Hidden-unit tokenizer tokenizes speech using k-means clustering. This is based on HuBERT and uses intermediate hidden states of 2nd round HuBERT model for clustering. Text data is also tokenized into the same hidden-unit space using a non-autoregressive hidden-unit model based on Fast-Speech (Y. Ren, Ruan, et al. 2019) which comprises of a text encoder, a duration model and a unit decoder.

One important way in which these tokenizers differ from the pre-nets in SpeechT5 is that they are offline, which means they are done training by the time we start pre-training

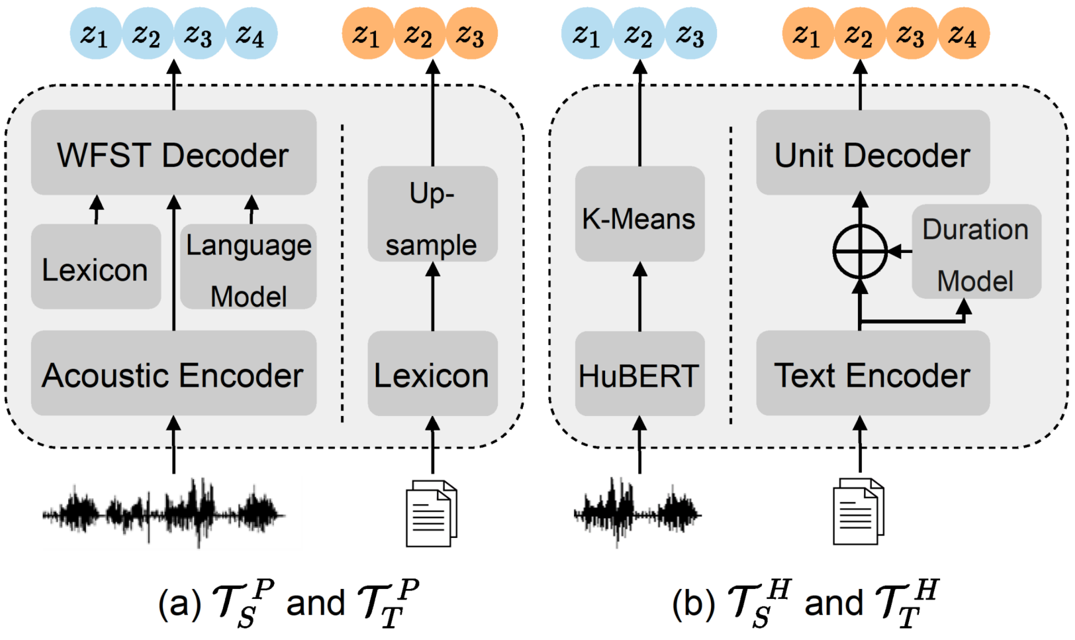


Figure 2.2: Architecture of (a) Phoneme-Unit Tokenizer and (b) Hidden-Unit Tokenizer (Z. Zhang et al. 2022)

and don't change during the pre-training or fine-tuning phases. Moreover, the tokenizers are trained on small amounts of labelled ASR data whereas SpeechT5 is completely unsupervised.

2.3.2 Shared Transformer

The model utilizes a shared transformer (Fig 2.3) that processes both speech and text inputs. Before passing through the shared transformer, speech input first goes through a speech transformer. During pre-training some of the positions in the speech sequence are randomly replaced by corresponding unit embeddings derived from the speech units generated using the tokenizer. This helps in aligning the speech and text representations during pre-training. For the text input, we don't use a separate transformer. Instead, the text data is tokenized using either phoneme-unit or hidden-unit tokenizer and then passed through a unit embedding layer after which it goes through the shared transformer.

2.3.3 Training

To train the shared transformer (and other parts of the model) the model utilizes two types of loss: Unit-based Masked Language Modelling (UMLM) loss and Unit-based Connectionist Temporal Classification (UCTC) loss. UMLM loss is used for speech

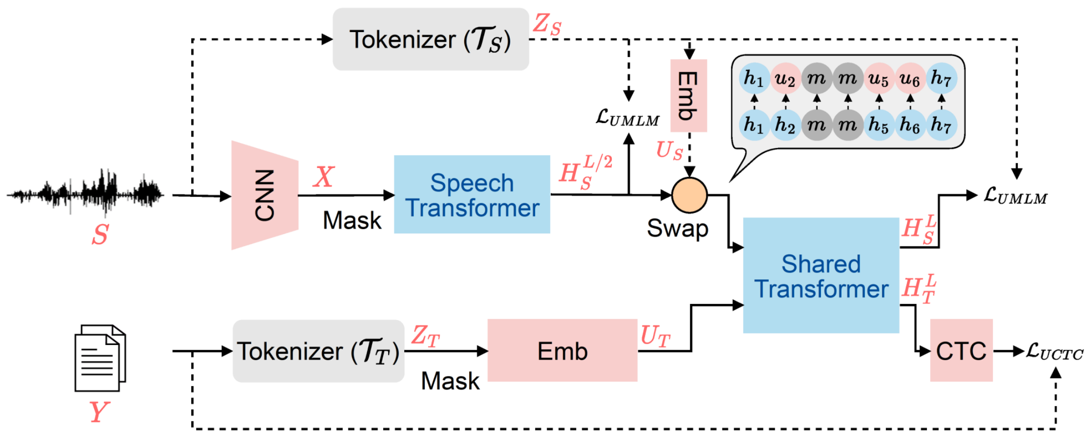


Figure 2.3: Architecture of the SpeechLM model (Z. Zhang et al. 2022)

Model	ASR (w/o LM)	ST (en-de)
SpeechT5	4.4	24.44
SpeechLM	3.4	24.3

Table 2.1: ASR score (% word error rate) without a language model and ST core (case-sensitive BLEU) for translation from English to German for SpeechT5 and SpeechLM. ST score for SpeechT5 is for the version without decoder initialization

and is calculated on the predicted hidden tokens at masked positions. It is calculated both on the output of the Speech Transformer and the output of the Shared Transformer. UCTC loss is used for text and is calculated on the output of the Shared Transformer by trying to reconstruct the masked input. The total loss is the sum of both UMLM loss and UCTC loss.

2.3.4 Performance

While the model is not evaluated on as many tasks as SpeechT5, it still performs really well across tasks like ASR, ST and Universal Representation Evaluation. In fact, SpeechLM performs better than SpeechT5 on ASR. Results comparing SpeechT5 and SpeechLM for ASR and ST are shown in Table 2.1. One reason for this could be that it utilizes some labelled data for training the tokenizers which helps in creating better shared representations for text and speech.

Chapter 3

Methodology

3.1 Setup

Owing to the time and resources required for pre-training large models from scratch, we use the weights of the pre-trained model, made available on the GitHub repository for SpeechT5[1], for fine-tuning our models. The pre-trained model was trained on 960 hrs of LibriSpeech and the LibriSpeech LM Dataset (Panayotov et al. 2015). The vocabulary and the sentencepiece model are also taken from this repository.

The fine-tuning was run on the `mlp` research cluster with an NVIDIA GeForce RTX 2080 Ti GPU with a memory of 11GB.

3.2 Datasets

To probe knowledge transfer we start with the Sentiment Analysis task. Sentiment analysis is a standard classification task in NLP, where you are given some text or speech and are asked to classify the sentiment of the whole text or utterance into one of the valid classes. These classes depend on the dataset and can range from just binary classification (Positive/Negative) to categorical labels like emotions (Happy, Sad and Angry) and even multi-label classification where each utterance can have multiple labels associated with it. For example, a sentence can be both sad and angry at the same time.

[1]https://github.com/microsoft/SpeechT5

Dataset	Positive	Negative	Neutral	Mixed	Disagreement	Total
Train	1279	227	4223	48	-	5777
Valid	262	51	1124	3	14	1454

Table 3.1: Class distribution of SLUE SA dataset before cleaning

Dataset	Positive	Negative	Neutral	Mixed	Disagreement	Total
Train	1279	227	4223	-	-	5729
Valid	262	51	1124	-	-	1437

Table 3.2: Class distribution of SLUE SA dataset after cleaning

3.2.1 SLUE

SLUE (Spoken Language Understanding Evaluation) (Shon et al. 2022) is a suite of benchmark tasks developed for testing models on spoken language understanding tasks. We use the dataset for the sentiment analysis (SA) task which consists of utterances taken from the VoxCeleb dataset (Nagrani, Chung, and Zisserman 2017) along with their corresponding text and labels. Each utterance has been hand-labelled as `Positive`, `Negative` or `Neutral` by multiple human evaluators. Utterances which clearly contain both positive and negative emotions are labelled as `Mixed`.

For the dev and test set, each utterance was labelled by multiple evaluators. Whenever the majority of the evaluators disagreed on an utterance, it is labelled as **Disagreement**.

The authors of the paper leave it up to the user whether to include labels like `Mixed` and `Disagreement` when training and testing. In line with the evaluation of the SA dataset on baselines in the SLUE paper, we have chosen to remove utterances that are labelled as `Mixed` or `Disagreement` from both fine-tuning and dev sets. The dataset details before and after cleaning are shown in Tables 3.1 and 3.2.

3.3 Adapting SpeechT5

While the authors of speecht5 fine-tune and test the model on a variety of tasks like Automatic Speech Recognition (ASR), Text To Speech (TTS), Speech to Text (ST), Voice Conversion (VC) and Speaker Identification (SID), Sentiment Analysis is not one of the tasks they explored. To get the model running on sentiment analysis we had to make some changes to the existing code base of SpeechT5.

3.3.1 Fairseq

We use the Fairseq framework (Ott et al. 2019) developed by Facebook AI Research to implement our changes in SpeechT5. The decision to go with Fairseq instead of another framework like HuggingFace largely came down to the fact that the original implementation of the model was in Fairseq and the code for this is readily available on the official GitHub repository for SpeechT5. In addition, Fairseq comes with many advantages like easy changes to the architecture of the model via configuration changes, the ability to create custom data collaters, data pre-processing, easy saving and retrieval of model parameters, convenient out-of-the-box logging, hyperparameter tuning via Hydra and the option to speed up training using mixed-precision and distributed training.

3.3.2 Speech-To-Class (S2C)

We first replicated the results of the Wav2Vec 2.0 baseline presented in the SLUE paper (Shon et al. 2022) for sentiment analysis on speech so we could compare it with the SpeechT5 model. We then repurposed the SID task by changing the output classes from Speaker IDs to sentiment labels in the output dictionary.

This required increasing the length of the utterances used to classify speech from 8000 to 250000 (speech is sampled at 16KHz). This is likely due to the fact that while identifying the speaker can be done with very little speech, finding out the emotion in it requires both semantic understanding and prosody, which requires context and as such needs longer utterances to properly classify.

Another change we had to make was a limitation of the GPUs we had available. Because of the large number of speech positions required to classify speech, we needed GPUs that could handle very long vectors. This limited both the type of GPUs we could use and also the batch size we could have. Using any batch size greater than one resulted in out-of-memory exceptions in our GPUs. Because having very small batch sizes (in this case, 1) can lead to unstable training we set the `update_frequency` variable to 16, which controls the number of batches for which gradients are accumulated before the weights are updated, to stabilize the training. This number was chosen after a few experiments with different batch sizes and update frequencies.

3.3.3 Text-To-Class (T2C)

Since the primary focus of SpeechT5 was speech, text classification is not one of the tasks available by default in it. So we had to create a new ‘t2c’ task for speecht5 which was modelled after the speech-to-class (s2c) and text-to-speech (t2s) tasks that are already available in the model.

This involved having two different dictionaries (where the model usually had only one), one for input text and the other for output classes, writing functions to load text and classes and collate them into batches for the model to process and to generate classes from input text by using the appropriate pre-nets and post-nets.

Because the input vectors for text input are much smaller compared to speech it was possible to have batch sizes larger than 1 during training.

3.4 Knowledge Transfer between S2C and T2C

The first thing we tried was to transfer the weights from the model trained on speech to text and the weight from the model trained on text to speech. The first problem was the difference in dictionaries. In the case of speech, the model was reusing the dictionary built for text to create encoders for the output classes (since it didn’t need the text dictionary anyway while classifying speech). So we created a new dictionary for classes when creating the t2c task. This class dictionary was different from the text dictionary in s2c in that it didn’t have `<ctc_blank>` and `<mask>` tokens as part of it. The text dictionary had those as part of pre-training. So we first added these to the class dictionary as well to align the output dimensions. While this allowed the sharing of weights, there was still a mismatch in the architectures.

The reason for the mismatch in architectures turned out to be the text dictionary which was being repurposed for classification tasks in s2c. The architecture of the text part of the model (like text pre-nets and post-net) was dependent upon the size of the dictionary and when the text dictionary was being reused it changed the architecture of the text processing parts of the model. This was not an issue because the text-processing machinery was redundant when classifying speech. To align the architectures a new class dictionary was introduced in s2c as well. This meant that weights could finally be transferred between the models without any problems.

3.5 Adapting SpeechLM

3.5.1 Classifying Speech

The researchers behind SpeechLM provide trained models along with scripts to pre-train, fine-tune and extract features. The `extract_features` function tokenizes the input speech and then outputs a 768-dimensional vector for each token. To do sentiment analysis on speech we train different classifiers on top of the extracted speech features. The architectures of the classifiers are explained in detail in the next chapter.

3.5.2 Classifying Text

Unlike with speech, SpeechLM does not provide any readily available functions to extract features from text. When we started working with SpeechLM our initial plan was to create an `extract_features` function for text fashioned after the one for speech. However, this had to be paused due to time constraints and also having to test different classifiers for speech input.

3.6 Hyperparameter Tuning

3.6.1 Hydra

We make use of the Hydra (Yadan 2019) framework for hyperparameter tuning of our model. Hydra is an open-source Python framework that makes creating and managing multiple configurations much simpler. This makes tedious manual processes like hyperparameter tuning easy and automatic. This is done through yaml configuration files where the user can specify the parameters that need to be altered along with the values that need to be considered. Hydra can do this hyperparameter optimization parallelly by running multiple instances of the same job with different configurations. Moreover, Hydra integrates easily with Fairseq, making it the ideal choice for tuning our models.

3.6.2 Tuning our models

SpeechT5 and its variations were the only models for which Hydra config files were created. This is because all changes made to these models were within the fairseq framework, allowing for the use of the fairseq-hydra-train tool that does the parameter sweeps. This was not the case for SpeechLM-based models where the final classifier layers were built using just PyTorch modules and as such would have required custom scripts to perform hyperparameter tuning.

Chapter 4

Experiments, Results and Analysis

4.1 Baseline

The SLUE paper reports results for sentiment analysis using a variety of models, one of them being wav2vec 2.0. End-to-end sentiment classification using wav2vec 2.0 has an F1 score of 48.6 according to the SLUE paper. We start off by trying to replicate this result. Based on the scripts provided in the GitHub repository for SLUE we were able to achieve similar results on our machines using wav2vec 2.0 (Table 4.1).

4.2 Speech-To-Class

The first test we ran after repurposing the Speaker Identification task (called s2c in the code base) for sentiment analysis was by changing the output dictionary to contain sentiment classes (`Positive, Negative` and `Neutral`) instead of speaker identifiers. This resulted in a model that just predicted `Neutral` all the time. We suspected that this might be because of either the relatively small number of speech positions that were being used (8000, which corresponds to about half a second of speech) or a result of the batch size being too small (batch size of 1).

Metric	SLUE	Ours
Precision	-	51
Recall	-	47.67
F1	48.6	48.79

Table 4.1: Sentiment Analysis on wav2vec 2.0 baseline

Metric	S2C (8k speech positions)	S2C (250k speech positions)
Accuracy	78.22	80.31
Precision	92.74	53.76
Recall	33.33	42.69
F1	29.26	44.76

Table 4.2: Performance of Speech-To-Class (S2C) models with different maximum allowed speech positions

Next, we tried increasing the maximum speech positions in the model from 8000, which was what was being used for speaker identification by the authors, to 250000, which was what was used during pre-training of the SpeechT5 model. This model predicted all three labels during evaluation and performed better than the model that predicted neutral all the time (Table 4.2). This indicates that predicting the sentiment of a speech sample requires the model to look at longer instances of speech compared to just identifying the speaker.

4.3 Text-To-Class

We then created a new t2c task for sentiment analysis on text. The t2c model was trained on transcripts of speech in the sentiment analysis task of SLUE. This resulted in our best-ever model (compared to all the other models in this project) with an F1 score of 51.73, which is higher than our wav2vec 2.0 baseline. But this is not an equal comparison given that the wav2vec 2.0 baseline is an end-to-end speech model where as our t2c model deals with only text, which is an easier modality to deal with. For a more fair comparison, we can look at the results presented in the SLUE paper. The authors of SLUE train DeBERTa-L (He et al. 2020) on the ground truth text (just like we did for our t2c model) to compare to with other E2E and pipeline (like ASR with a speech model followed by SA with a text model) approaches. The results are presented in Table 4.3.

As we can see, SpeechT5 modified for classification on text performs worse than the NLP topline presented in SLUE. It is possible that pre-training together with speech using pre-nets to learn joint contextual representations negatively affects the model's ability to process text directly.

Metric	SpeechT5 T2C	DeBERTa-L
Accuracy	80.38	-
Precision	58	-
Recall	49.57	-
F1	51.73	66.8

Table 4.3: Performance of our SpeechT5 Text-to-Class (T2C) model compared with DeBERTA-L on Sentiment Analysis from SLUE

4.4 Knowledge Transfer between T2C and S2C

4.4.1 Aligning Architectures of S2C and T2C

As we started experimenting with the transfer of weights between T2C and S2C we noticed that the architectures of our models differed. First, we noticed that the dictionary used for T2C had two extra classes: `<ctc_blank` and `<mask>`. These are used during pre-training on text data and are not really needed for classification. But to align the output dimensions of both models we decided to add both of these classes to the dictionary. The resulting T2C model performed worse than our original T2C model which had fewer classes in the output. This result was surprising, so we looked into the output of both models to see if they differed in the number of classes they predicted. But this turned out not to be the case and both models only predicted either `Positive`, `Negative` or `Neutral` in all cases (at least during evaluation).

To try and improve the performance of the model after updating the output dictionary, we tried training the model with various batch size and update frequency combinations. The results are presented in Table 4.4. From this table, we can see that while the F1 scores do change with batch sizes and update frequencies, their effect is minimal. While larger batch sizes did decrease the training time considerably (not shown here) it doesn't seem to improve the model. But this can be chalked up to the fact that the number of samples used for an update remains the same (16) in all these cases. We chose 16 because this was the size used in the fine-tuning setup of SpeechT5.

The next thing we observed was that the reuse of the text dictionary in the S2C model was causing its architecture to differ from the architecture of the T2C model. So after creating a new dictionary to match the architecture of T2C, we fine-tuned the new S2C model again on SLUE. The results are shown in Table 4.5. While the accuracy and precision of the model did go down we also observe that the newer model has higher

Model	Batch Size	Update Frequency	Accuracy	Precision	Recall	F1
Original	1	16	80.38	58	49.57	51.73
A	1	16	78.78	56.62	47.99	48.42
B	4	4	79.96	46.92	46.56	46.63
C	8	2	79.19	50.98	45.44	46.9

Table 4.4: Comparison of T2C models trained with different batch sizes and update frequencies

Metric	S2C (New)	S2C (Old)
Accuracy	78.36	80.31
Precision	49.37	53.76
Recall	43.46	42.69
F1	45.11	44.76

Table 4.5: Comparing S2C models before and after dictionary changes

recall and F1 score. However, the improvement in the scores is really small and might as well be because of the influence of random initializations during the fine-tuning of the models.

4.4.2 Transferring Weights Between Models

Zero Shot Learning

The obvious thing to try once we had aligned the architectures of both S2C and T2C models was to use the weights trained on one model in another and check for knowledge transfer between the modalities. The first thing we did was to check for any traces of zero-shot learning across modalities. This involved directly using the best checkpoint of the fine-tuned S2C model in T2C model for inference on the validation set. But the T2C model just ended up predicting `Neutral` in all the cases. We then checked for zero-shot transfer from text to speech. Unfortunately, even this resulted in a model that just predicted `Neutral` all the time.

We suspected that the heavy skew in the class distribution towards `Neutral` in the dataset was making the model predict `Neutral` for most cases and was preventing the transfer of knowledge between modalities. To overcome this we tried to balance the class distribution so that each class had roughly equal representation in the dataset. This

Dataset	Positive	Negative	Neutral	Total
Original	1279	227	4223	5729
Pruned	227	227	227	681
Duplicated	3837	4086	4223	12146

Table 4.6: Class distribution of sentiment analysis datasets after pruning and duplication

Model	Accuracy	Precision	Recall	F1
Original	78.36	49.37	43.46	45.11
Pruned	48.99	33.4	28.15	27.18
Duplicated	78.57	48.02	44.06	44.99

Table 4.7: Performance of S2C model on different datasets

was done using two methods: Pruning Dataset and Duplicating Data. For pruning the dataset, we reduced the number of samples for `Positive` and `Neutral` cases to match that of `Negative`, which had the least representation in the dataset. For duplication, the samples that were labelled as `Positive` or `Neutral` were repeated till the number of samples was roughly equal across the classes. The sizes of the datasets before and after these operations are shown in Table 4.6.

However, this did not result in any noticeable improvements in the model. In the case of the pruned dataset, we notice a large drop in the F1 score from 45.11 to 27.18 (Table 4.7). This can be attributed to the decrease in the size of data available for training. The total size of the dataset decreases from 5729 to 681 which is about an 88% decrease in the size of the dataset. On the other hand, repeating the underrepresented classes doesn't improve the performance of the model either. While the size of the data available for training increases from 5729 to 12146, a 112% increase, the F1 score of the model actually decreases (though this decrease is only nominal). There was no zero-shot learning from this model as well, with the model once again just predicting Neutral all the time even though the s2c model was trained on a dataset with equal class representation. Perhaps having more samples for Negative and Positive classes (and not just repeating the available samples) could improve the model, but collecting this data and testing it is beyond the scope of this project.

Model	Accuracy	Precision	Recall	F1
T2C (from SpeechT5)	79.96	46.92	46.56	46.63
T2C (from S2C)	78.57	77.90	43.53	43.76

Table 4.8: Performance of T2C model directly trained from pre-trained SpeechT5 compared with T2C model trained on a fine-tuned S2C model (which was again trained from SpeechT5)

Few Shot and Many Shot Learning

After zero-shot learning, we moved on to check for Few-shot learning. As we mentioned previously, the model starts off by predicting `Neutral` all the time when weights are transferred. But this is not the case when starting training directly from the pre-trained SpeechT5. So we wanted to check if the learning sped up for the T2C model when starting from a model trained on S2C when compared to starting directly from the pre-trained SpeechT5 model. The indicator we were using to verify this was to see when the model started doing better than just predicting `Neutral` in both cases. From the results, we didn't see any noticeable difference, with both models starting to predict classes other than Neutral from the 4th epoch or so. And moreover, there was no improvement in the final performance of the model in both cases. The final scores of both models are shown in Table 4.8.

To investigate this further, we plotted the progression of training loss and validation loss over epochs for both models, shown in Figure 4.1. As can be seen from the graph the validation loss is lowest pretty much at the start of training for both the models. This could be either due to a high learning rate resulting in issues like divergence, overshooting and instability or it might be an issue with the architecture of our model. High learning rate also provides an explanation for the large drop in validation loss for the model trained from s2c (and an increase in training loss). But to test this more thoroughly we would need to run a proper hyperparameter search and plot the loss across different learning rates. Owing to time constraints, we couldn't get Hydra completely working on our model to perform an effective parameter search.

4.5 Issues with Hyperparameter Tuning

While we initially planned to parallelly adapt, test and tune our models as the project progressed, we immediately ran into problems with getting Hydra to work with SpeechT5.

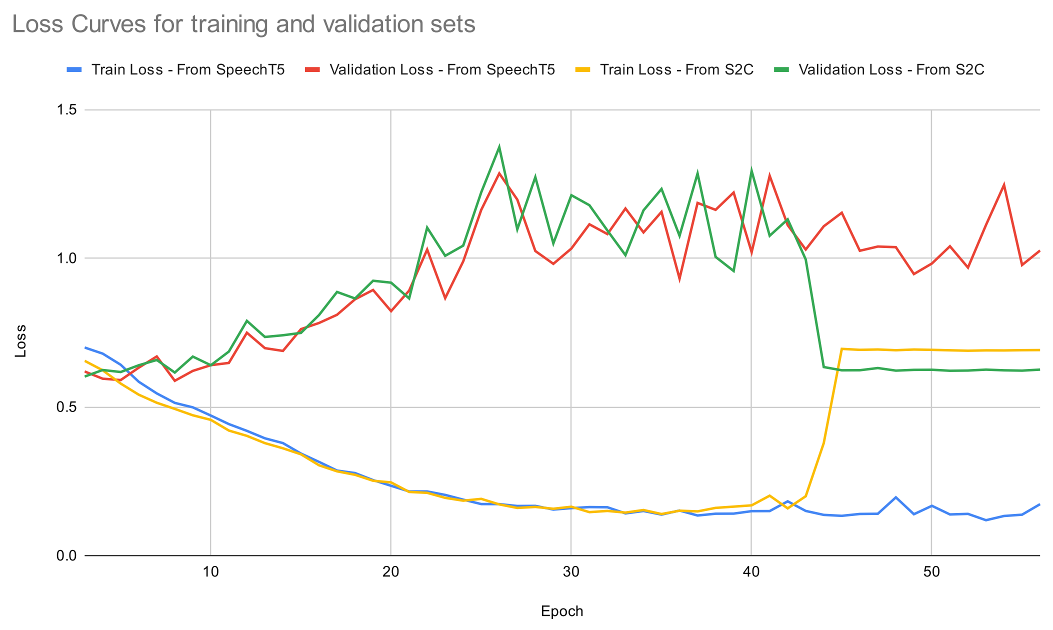


Figure 4.1: Progression of loss over epochs for training and validation sets for T2C model trained from SpeechT5 and T2C model trained from S2C

This was largely down to the use of legacy Fairseq modules in SpeechT5 (which are on their way to being deprecated). This meant newer versions of the Hydra framework were not compatible with our model and we had to revert to the older version of Hydra. Moreover, the newer version of Fairseq also integrates better with Hydra by providing DataClasses that define parameters and their default values and can be easily overridden with configuration files. This was not the case with legacy Fairseq modules where the parameters had to be manually altered through `ArgumentParse` objects. All these issues meant we couldn't get hyperparameter tuning with Hydra working properly on our models. This necessitated the use of manual interventions to tune hyperparameters which slowed down the tuning of our models and ultimately resulted in us having to decide which parameters we could prioritize for tuning (and which we had to hold off on) while simultaneously refining our models.

4.6 Switching to SpeechLM

Late into our project, we decided to pivot from SpeechT5 to SpeechLM to test for cross-modal transfer. Our reason for this was two-fold: we had already spent a large amount of time trying to probe for cross-modal transfer for sentiment analysis in

SpeechT5 without much success, and SpeechLM was a newer and simpler model that outperformed SpeechT5 in some cases.

Another major advantage that SpeechLM has over SpeechT5 is that it utilizes a small amount of labelled ASR data to train the tokenizers. We believe this can lead to better shared representations for speech and text during pre-training compared to SpeechT5 which as a result might allow for better cross-modal transfer.

With this in mind, we trained a classifier on top of SpeechLM for sentiment analysis. The architecture of the classifier is shown in Figure 4.2. The encoder from SpeechLM was used to extract features of dimension `[num_tokens, 768]` for speech which was used as input for the classifier. We ran two different experiments, one where the weights of the encoder were fixed during fine-tuning for sentiment analysis and another where they were updated. In both cases, we ended up with a model that just predicted `Neutral` all the time.

We then switched to using a GRU-based classifier instead. This was done so that all of the input sequence could be used to predict the sentiment of the input speech. This did slow down the training of the model considerably but we didn't notice any improvement in the final predictions of the model. Even the GRU-based classifier predicted `Neutral` all the time. We had to stop our experiments with SpeechLM owing to the time constraints inherent in an MSc Dissertation. We discuss possible ways to fix these issues and other ideas we initially planned on exploring in the next chapter.

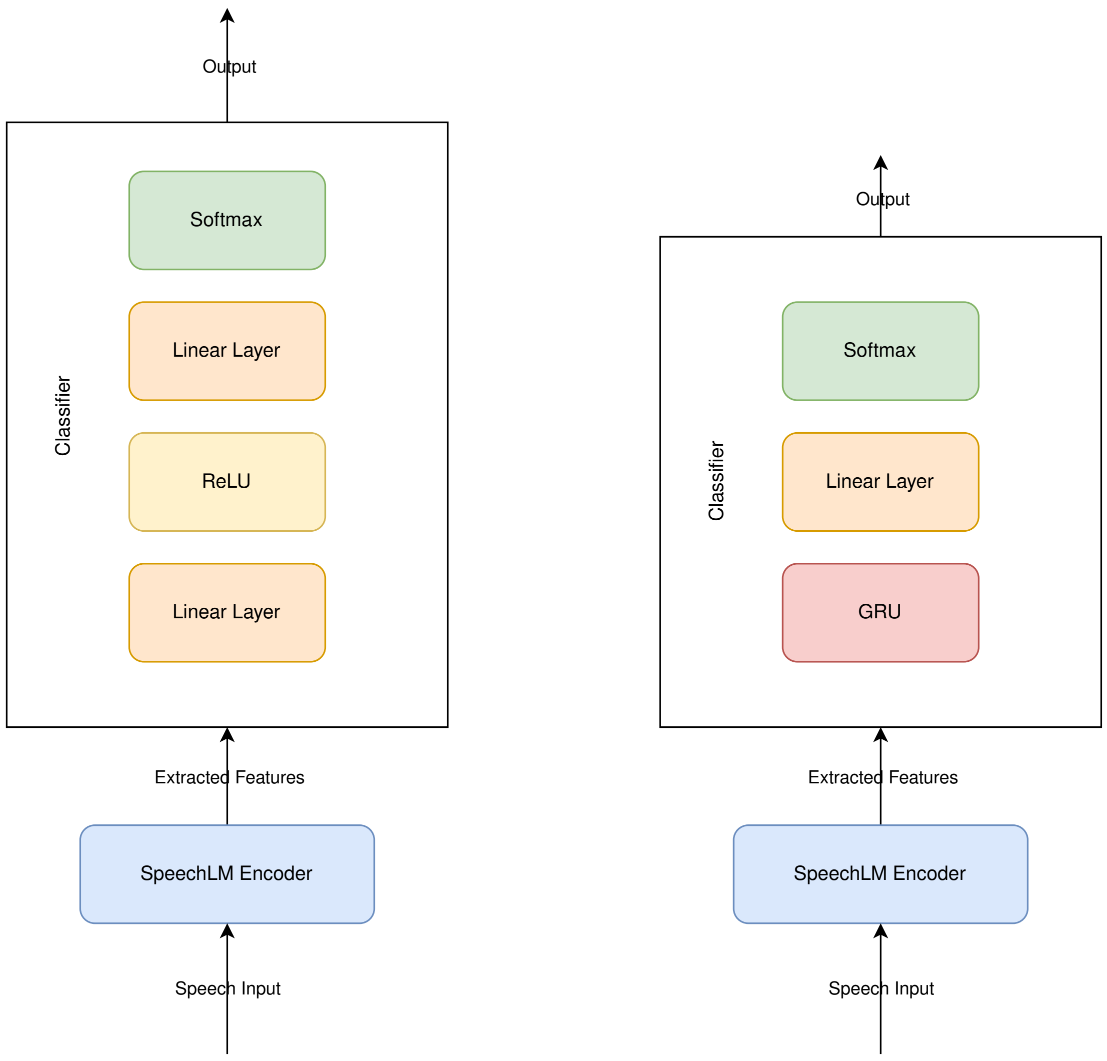


Figure 4.2: Architecture of different classifiers used with SpeechLM

Chapter 5

Conclusions

5.1 Summary

In this project, we conduct a series of experiments to test for knowledge transfer for the sentiment analysis task in cross-modal models. We make use of state-of-the-art cross-modal models like SpeechT5 and SpeechLM to this end. Since these models were not originally designed for or tested with sentiment analysis previously, we made our own changes to these models to make them usable for sentiment analysis task. We make use of the SA task from the SLUE toolkit to train and validate our models. The data from the task is first pre-processed in line with existing practices to suit our needs.

Over the course of this project, we probe for cross-modal knowledge transfer in various forms, like zero-shot learning, few-shot learning and many-shot learning. To facilitate proper and efficient transfer of weights between models trained on text and speech, we experiment with several changes to the original models like making changes to the output dictionaries, max input sizes and classifier architectures. In addition to this, we also experiment with different class percentages in the dataset to account for class imbalances in the dataset. We also test for knowledge transfer in both directions: from text to speech and from speech to text. We document and present our findings in the project in this report. In the end, while we don't find any compelling evidence to suggest SpeechT5 allows for cross-modal knowledge transfer for sentiment analysis, we believe a lot more analysis needs to be done before we can prove or even suggest that this is definitively the case.

5.2 Limitations and Future Work

While our experiments have been thorough and planned out to test various aspects of cross-modal transfer in the models we chose, they are far from complete. There were a lot of tests we couldn't do and a lot of research directions we couldn't take. The biggest of these is testing for cross-lingual transfer along with cross-modal transfer. Initial negative results with cross-modal transfer necessitated the need for further experiments with various models, architectures and datasets, including the switch to SpeechLM towards the end of the project. While we were able to build a classifier for sentiment analysis on speech with SpeechLM, the performance of the resulting model ended up being worse than our modified SpeechT5 models, despite SpeechLM being a better model than SpeechT5.

SpeechT5

In the case of SpeechT5, any future work aimed at continuing what we started should start with hyperparameter tuning for SpeechT5 to rule out the possibility that parameters like learning rate, batch_size, length of input speech and text or update frequency played any role in preventing knowledge transfer across modalities. Upcoming experiments can also focus on tasks other than sentiment analysis. Working with just the encoder of SpeechT5 and using the features extracted from the last layer of the encoder to classify speech and text, similar to what we did with SpeechLM, is also a direction worth exploring.

SpeechLM

When it comes to SpeechLM we believe that given its reported performance on Speech Translation and SUPERB (S.-w. Yang et al. 2021), which provides a comprehensive test suite for pre-trained models on speech tasks like Keyword spotting and intent classification, the model should be capable of sentiment analysis. Future work should focus on testing out different classifiers and also using features from different layers (rather than just using the features from the last layer) or a combination of layers to test for sentiment analysis. Similar work can be done with the text representation in SpeechLM. Tasks other than sentiment analysis also provide opportunities for further exploration of cross-modal transfer in SpeechLM.

Cross-modal, Cross-lingual Transfer

Our holy grail when we started this project was to prove that cross-modal and cross-lingual transfer was possible. And that this would help NLP research in low-resource languages where labelled data is seldom available for training for many tasks. While we couldn't show that this was possible we nevertheless believe that future work should focus on not just showing this, but actively developing models that can take advantage of inherent similarities that exist between spoken language and written language along with similarities that exist across languages. To this end, we look forward to not just the emergence and widespread use of multi-modal and multi-lingual models, but also to taking part in the story of their creation.

Bibliography

Ao, Junyi et al. (2021). “Speecht5: Unified-modal encoder-decoder pre-training for spoken language processing”. In: *arXiv preprint arXiv:2110.07205*.

Baevski, Alexei et al. (2020). “wav2vec 2.0: A framework for self-supervised learning of speech representations”. In: *Advances in neural information processing systems* 33, pp. 12449–12460.

Barnes, Janet (1984). “Evidentials in the Tuyuca verb”. In: *International journal of American linguistics* 50.3, pp. 255–271.

Chen, Zhehuai et al. (2022). “Maestro: Matched speech text representations through modality matching”. In: *arXiv preprint arXiv:2204.03409*.

Conneau, Alexis et al. (2020). “Unsupervised cross-lingual representation learning for speech recognition”. In: *arXiv preprint arXiv:2006.13979*.

Devlin, Jacob et al. (2018). “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805*.

Eberhard, Gary F Simons, and Charles D Fennig (2022). *Ethnologue: Languages of the World*. URL: `https://www.ethnologue.com/`.

He, Pengcheng et al. (2020). “Deberta: Decoding-enhanced bert with disentangled attention”. In: *arXiv preprint arXiv:2006.03654*.

Hsu, Wei-Ning et al. (2021). “Hubert: Self-supervised speech representation learning by masked prediction of hidden units”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 29, pp. 3451–3460.

Huang, Gao et al. (2016). “Supervised word mover’s distance”. In: *Advances in neural information processing systems* 29.

El-Kishky, Ahmed and Francisco Guzmán (2020). “Massively Multilingual Document Alignment with Cross-lingual Sentence-Mover’s Distance”. In: *arXiv preprint arXiv:2002.00761*.

Lewis, Mike et al. (2019). “Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension”. In: *arXiv preprint arXiv:1910.13461*.

Liu, Yinhan et al. (2019). “Roberta: A robustly optimized bert pretraining approach”. In: *arXiv preprint arXiv:1907.11692*.

Magueresse, Alexandre, Vincent Carles, and Evan Heetderks (2020). “Low-resource languages: A review of past work and future challenges”. In: *arXiv preprint arXiv:2006.07264*.

Nagrani, Arsha, Joon Son Chung, and Andrew Zisserman (2017). “Voxceleb: a large-scale speaker identification dataset”. In: *arXiv preprint arXiv:1706.08612*.

Nasution, Arbi Haza, Yohei Murakami, and Toru Ishida (2016). “Constraint-based bilingual lexicon induction for closely related languages”. In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC’16)*, pp. 3291–3298.

Ott, Myle et al. (2019). “fairseq: A fast, extensible toolkit for sequence modeling”. In: *arXiv preprint arXiv:1904.01038*.

Panayotov, Vassil et al. (2015). “Librispeech: an asr corpus based on public domain audio books”. In: *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, pp. 5206–5210.

Pires, Telmo, Eva Schlinger, and Dan Garrette (2019). “How multilingual is multilingual BERT?” In: *arXiv preprint arXiv:1906.01502*.

Raffel, Colin et al. (2020). “Exploring the limits of transfer learning with a unified text-to-text transformer”. In: *The Journal of Machine Learning Research* 21.1, pp. 5485–5551.

Ren, Shuo et al. (2021). “Semface: Pre-training encoder and decoder with a semantic interface for neural machine translation”. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4518–4527.

Ren, Yi, Chenxu Hu, et al. (2020). “Fastspeech 2: Fast and high-quality end-to-end text to speech”. In: *arXiv preprint arXiv:2006.04558*.

Ren, Yi, Yangjun Ruan, et al. (2019). “Fastspeech: Fast, robust and controllable text to speech”. In: *Advances in neural information processing systems* 32.

Saralegi, Xabier, Iker Manterola, and Inaki San Vicente (2011). “Analyzing methods for improving precision of pivot based bilingual dictionaries”. In: *Proceedings* *of the 2011 Conference on Empirical Methods in Natural Language Processing*, pp. 846–856.

Shchukin, Vadim, Dmitry Khristich, and Irina Galinskaya (2016). “Word clustering approach to bilingual document alignment (wmt 2016 shared task)”. In: *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pp. 740–744.

Shen, Jonathan et al. (2018). “Natural tts synthesis by conditioning wavenet on mel spectrogram predictions”. In: *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, pp. 4779–4783.

Shon, Suwon et al. (2022). “Slue: New benchmark tasks for spoken language understanding evaluation on natural speech”. In: *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 7927–7931.

Simpson, Heather et al. (2008). “Human language technology resources for less commonly taught languages: Lessons learned toward creation of basic language resources”. In: *Proceedings of the LREC 2008 Workshop on Collaboration: interoperability between people in the creation of language resources for less-resourced langauges*.

Strassel, Stephanie and Jennifer Tracey (2016). “LORELEI language packs: Data, tools, and resources for technology development in low resource languages”. In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC’16)*, pp. 3273–3280.

Van Den Oord, Aaron, Oriol Vinyals, et al. (2017). “Neural discrete representation learning”. In: *Advances in neural information processing systems* 30.

Wang, Chengyi et al. (2022). “Supervision-guided codebooks for masked prediction in speech pre-training”. In: *arXiv preprint arXiv:2206.10125*.

Xia, Mengzhou et al. (2019). “Generalized data augmentation for low-resource translation”. In: *arXiv preprint arXiv:1906.03785*.

Yadan, Omry (2019). *Hydra - A framework for elegantly configuring complex applications*. Github. URL: `https://github.com/facebookresearch/hydra`.

Yang, Shu-wen et al. (2021). “Superb: Speech processing universal performance benchmark”. In: *arXiv preprint arXiv:2105.01051*.

Yang, Zhilin et al. (2019). “Xlnet: Generalized autoregressive pretraining for language understanding”. In: *Advances in neural information processing systems* 32.

Zhang, Ziqiang et al. (2022). “Speechlm: Enhanced speech pre-training with unpaired textual data”. In: *arXiv preprint arXiv:2209.15329*.

Zoph, Barret et al. (2016). “Transfer learning for low-resource neural machine translation”. In: *arXiv preprint arXiv:1604.02201*.

Appendix A

Code

Code for SpeechT5 and SpeechLM models comes from the official GitHub repository for SpeechT5 (https://github.com/microsoft/SpeechT5). All the code that we have written which includes scripts for cleaning up and pre-processing the data sets, scripts for training and evaluating our models, changes to SpeechT5, Hydra configuration files and classifiers built on top of SpeechLM are all available in the project's GitHub page (https://github.com/EdinburghNLP/t2s-xling). Note that the repository is private at the time of writing and you might need to request access to view it. The repository is also attached as a zip file for evaluation.