Meeting Perception Planning for Conversational AI

Harish Vadlamani
M. Ahsan Ghani
Jiayin Lei
Yuesheng Luo

Table of Contents

1. Introduction

Seasalt AI presents a variety of solutions such as SeaMeet and SeaSuite which specialize in conference room solutions that listen, transcribe, summarize, follow up, and perceive. SeaSuite is a Full Stack Cloud Communication AI which consists of three integrated platforms such as SeaCode, SeaChat, and SeaVoice. Whereby SeaCode is an integrated development environment for conversational AI, SeaChat is a state of the art conversational AI for omni-channel communication, and SeaVoice is a deep speech recognition and neural speech synthesis platform.

This project aims to to extrapolate the two existing functions of the current meeting perception planning system. We are emphasizing modifications for improvements to Summarization as well as Action Extraction through multi-label classification methodologies. Furthermore, we aim to deploy proof of concept implementations for sentiment analysis and speech emotion recognition on text and audio data respectively accessible by Fast API endpoint on a friendly user interface.

The current implementations for both of the previously mentioned Action Extraction and Summarization techniques were BERT and BART models respectively, with a Rasa classifier for the primary multi-class classification problem. We attempted LUKE, a deep contextualized entity representation with entity-aware self-attention and discovered that despite the enhanced results for named entity recognition, the model was struggling to perform imperative action extraction.

We implemented ULMFiT for Action Extraction for multi-class classification preliminarily and refined the model to a binary classification head for increased accuracy thereby circumventing unnecessary ambiguity and granularity between the positive classes. Analogously, the ULMFiT model demonstrated spectacular performance on sentiment analysis as well since both of the previously mentioned implementations are binary classification procedures as discussed below.

Furthermore, the speech emotion recognition implementation incorporated a wav2vec 2.0 for analyzing acoustic features from audio files such as an irritated client's intonations towards a customer service representative. The motivating friendly user interface designed internally provides a holistic distribution of the client's emotions throughout the conversation for call center based application and supplements interactions with the customer service representatives.

Finally, we attempted short and long summarization with BART as well as T5 models with varying results attributed predominantly to not being able to truncate concise utterances any further. In addition to long passages of utterances lacking contextual representations from the previous conversational dialogues to adequately generate a summary without redundant phrases.

2. Literature Review

Information extraction (IE) is the process of automatically extracting structured information from unstructured and/or semi-structured machine-readable documents, while information retrieval (IR) is the process of utilizing queries to locate material such as documents of an unstructured nature (typically text) that fulfills an information requirement from huge collections.

The task of recognising named entities such as person names, place names in a document is known as named entity recognition (NER).This is especially important for languages with limited resources, such as many remote dialects of Eastern languages.

The task of recognising entity mentions such as noun, pronoun as well as mentions of people, places and deriving meaningful connections between these mentions is known as relationship extraction. The emphasis is on employing phrase level joint inference in Markov logic networks to jointly extract entity mentions and relations (MLNs).

The semantic relationship between the components of a noun compound is determined by noun compound interpretation such as a sequence of two or more nouns. For example, “student protest” implies “protest by students” whereas “student housing” means “housing for students.”

Coreference resolution connects reference entities with co-referential mentions. The job of detecting a series of words (such as 'zebra crossing') whose semantic, syntactic, or lexical features cannot be completely anticipated from their component words is known as multiword expression extraction. This paper aims to highlight several of the previously mentioned theoretical fundamentals in hands-on implementation exercises for tasks such as action extraction, short/long summarization, sentiment analysis as well as speech emotion recognition.

2.1 Action Extraction

Singh [1] highlights numerous parsing techniques such as named entity resolution, named entity linking as well as coreference resolution. Amongst these, the most pertinent ones are relation extraction and event extraction. Whereby identifying two human entities in a dialogue, referred to as the speaker and receiver, supplemented by the relationship between them or the imperative being passed from one to another serves as the basis for our conversation scenario.

Anand et al. [2] evaluated the feasibility of implementing a Universal Language Model Fine-tuning for Text Classification by applying various pre-processing techniques before training the language and the classification model. Resulting in an F1-scores improvement for training and testing across Naive Bayes, Logistic Regression as well as support vector machines.

Inspired by their promising results, we selected ULMFiT in preference over LUKE, a deep contextualized entity representations with entity-aware self-attention by Yamada et al. [3] which focused exclusively on pretrained contextualized representations of words and entities based on the bidirectional transformer. Yamada et al. [3] find that the model achieves impressive empirical performance on a wide range of entity-related tasks such as the following as demonstrated in 3.2.

- Open Entity (entity typing)
- TACRED (relation classification)
- CoNLL-2003 (named entity recognition)

- ReCoRD (cloze-style question answering)
- SQuAD 1.1 (extractive question answering)

2.2 Sentiment Analysis

Pan et al. [9] highlights the versatility of creating classification models based on DNN, LSTM, Bi-LSTM, and CNN to determine the trend of the user's implicit sentiment text. The categorization model of word-level attention mechanism is investigated using the Bi-LSTM model. This paper's experimental findings on the public dataset reveal that both the existing LSTM series classification model and the CNN classification model can produce strong sentiment classification effects. Whereby the CNN model performs better than the DNN model.

In comparison however, AlBadani et al [10] demonstrates an innovative machine learning approach for sentiment analysis on Twitter incorporating the Universal Language Model Fine-Tuning and Support Vector Machines. This paper illustrates that the classification accuracy and detection performance of Twitter sentiment detectors are highly dependent on the classification methods' performance as well as the quality of input characteristics.

That being said however, conventional machine learning approaches pose a hurdle in transitioning to automated processes due to the significant time and latency requirement. The author aims to introduce a revolutionary effective sentiment analysis approach based on deep learning architectures that combines "universal language model fine-tuning" (ULMFiT) and support vector machine (SVM) to improve detection efficiency and accuracy.

The technique provides a novel deep learning strategy for Twitter sentiment analysis that incorporates user comments to determine people's opinions about certain items. The model shows improvements on accuracy performance to 99.78% on the Twitter US Airlines dataset. Due to the colloquial conversational nature of Twitter, IMDB, as well as client and customer service representative correspondence, we anticipate analogous correlations between the training and validation datasets in terms of sentiment analysis for transcribed dialogues as shown in 4.2.

2.3 Speech Emotion Recognition

Schneider et al. [11] experimentally demonstrate unsupervised pre-training for speech recognition by learning representations of raw audio. Large volumes of unprocessed audio data are utilized to train wav2vec. The generated representations are subsequently used to enhance acoustic model training. A noisy contrastive binary classification challenge is used to train CNN.

The closest evaluation metric for this experiment's performance is a logarithmic scale of the Mel-filterbank. The log-mel filterbank is a scale of pitches judged by listeners to be equal in distance from one to another. The reference point between this scale and normal frequency measurement is defined by equating a 1000 Hz tone with a pitch of 1000 mels.

When just a few hours of transcribed data is available, the trials on WSJ lower WER of a strong character-based log-mel filterbank baseline by up to 36%. On the Nov92 test set, the method scores 2.43 percent WER. While utilizing two orders of magnitude less labeled training data, this surpasses DeepSpeech2 by Amodei et al [12] the best documented character-based system.

Baevski et al. [13] experimentally demonstrate that learning strong representations from speech audio alone and fine-tuning on transcribed speech outperforms the best semi-supervised approaches while being conceptually simpler as demonstrated in Section 5.3.

Wav2vec 2.0 masks voice input in the latent space and solves a morphosyntactic task specified on a combination of jointly acquired underlying interpretations. On the clean/other test sets, experiments using all labeled data from Librispeech get 1.8/3.3 WER.

Wav2vec 2.0 exceeds prior state of the art on a 100 hour subset while requiring 100 times less labeled data when the quantity of labeled data is reduced to one hour. Even with just 10 minutes of labeled data and 53k hours of unlabeled data, the WER is still 4.8/8.2.

2.4 Short Summarization

Lewis et al. [4] presents a denoising autoencoder for pretraining sequence-to-sequence models. Summarization is inherently a sequence-to-sequence task consisting of a neural network that takes a sequence from a specific domain such as the text vocabulary as the input and outputs new sequences in another domain such as a summary vocabulary. Furthermore, an inference model yields an encoder-decoder structure where individual models are used for training and predicting.

Bidirectional Auto-Regressive Transformers (BART) is learned by using an arbitrary noising function to distort text and then building a model to recover the original text. It employs a typical Transformer-based neural machine translation architecture that generalizes Bidirectional Encoder Representations from Transformers (BERT), as well as generative pre-trained transformer (GPT, left-to-right decoder), as well as more modern pre-training approaches as shown in Section 6.3.

Devlin et al [5] present the previously mentioned BERT model, which stands for Bidirectional Encoder Representations from Transformers, which is aimed to pre-train deep bidirectional representations from unlabeled text by conditioning on both left and right context in all layers.

Consequently, the pre-trained BERT model may be fine-tuned with only one extra output layer to provide state-of-the-art models for a variety of tasks, such as question answering and language inference, without requiring significant task-specific architectural modifications.

BERT is both theoretically and experimentally simple. It achieves new state-of-the-art results on eleven natural language processing tasks, including increasing the following:

- GLUE score = 80.5 %
- MultiNLI accuracy = 86.7 %
- SQuAD v1.1 Test F1 = 93.2
- SQuAD v2.0 Test F1 = 83.1

2.5 Long Summarization

Raffel et al. [6] evaluate the Limits of Transfer Learning with a Unified Text-to-Text Transformer where a model is first pre-trained on a data-rich summarization task before being fine-tuned on a downstream task has the potential to outperform on numerous applications. Transfer learning's effectiveness has extrapolated numerous techniques and methodologies.

Raffel et al. [6] provide a single framework which translates all text-based language issues into a text-to-text format and explores the landscape of transfer learning approaches for natural language processing. On hundreds of language comprehension tasks, the research examines pre-training goals, architectures, unlabeled data sets, transfer methodologies, as well as various other parameters. They produce state-of-the-art results on several benchmarks, including summarization, question answering, and text categorization as demonstrated in Section 7.3.

The Text-To-Text Transfer Transformer (T5) is an exceptionally powerful resource as it can be implemented across multilingual applications as demonstrated by Nagoudi et al. [7] with a strong emphasis on the performance of mT5 (multilingual T5) as well as AraT5 (Arabic T5) trained on language specific datasets resulting in exceptional performance on evaluation benchmarks such as ARLUE by Mageed [8]. The previously mentioned multilingual approaches are tremendously constructive to our application due to the current and updated model's potential and tentative implementations in Chinese for the Seasalt Voice Meeting Perception user interface.

3. Action Extraction

Model API endpoint.

3.1 Problem Statement

Information extraction (IE) serves as a foundation for the action extraction process whereby the intention is to highlight a directive or imperative gesture from an assigner (speaker) to an assignee (receiver), both parties being distinguished with the help of named entity recognition.

The Meeting Perception Planning implementation by Seasalt aims to gather actionable items such as deliverables from internal meetings which can be used as follow up items shown in Table 1 for subordinates like the following utterance where they are requested to deliver on a timeline.

Utterance	Classification
Can you show this presentation to senior management by tomorrow afternoon	actionable

Table 1: Sample actionable utterance containing a timeline and deliverable.

3.2 Current Implementation

A multi-class classification implementation where the input meeting transcription utterances and was previously deployed into the production environment with Rasa's Dual Intent and Entity Transformer (DIET) classifier. This binary implementation was further improvised to a multi-class classification program categorizing actionable utterances into positive and negative classes as per the following examples demonstrated in Table 2.

Utterance	Class
M2: Can you think of any other solutions of initiative that will help achieve and make it more inclusive for everyone?	question
Speaker: If you have to look up words in the dictionary or something just like	imperative

Utterance	Class
take very good notes of your process so we can compare later an kindof compile all the different techniques we were using for that.	
Speaker: OK, one thing I'd like to see more of on the document before we present it tomorrow is like specific challenges for whatever languages you're assigned to.	desire
Speaker: As she insisted that we should have some kind of issue there, and then you might have, like you know, a good manual curve to keep track of everything, right?	suggestion
Cody: Also had some formatting peer comments to finish and I'll submit this today and I've been running some tests on topics audio data because right now the way the training sampling works is it picks a random sample from the clip	statement
speaker04: I'm going to say a word, and you just tell me whatever comes to your mind, right or wrong, answers ready.	non-actionable

Table 2: Sample utterances for sentiment analysis classifications.

Taking into consideration the classification pipeline followed by a summarization task on the actionable utterances, we observed that segregation of more granular positive classes was ambiguous and yielded diminishing returns since the information was discarded downstream.

3.3 Proposed Modifications

We propose pretrained versions of Universal Language Model Fine-tuning (ULMFiT) for Text Classification which improves upon the previous implementation. Furthermore, we implemented a reiterative exercise for cross-verifications on the model's classification predictions and updated the training set to learn imperatives on directives which were very specifically pertinent to dialogues for meeting minutes in accordance with corporate requirements and business feedback.

As per business requirements, a greater preference was placed on very selective classifications whereby "actionable" was defined as a task involving a tangible deliverable, preferably with respect to a "person", "place" or "thing" supplemented with a timeline for completion such as "today", "tomorrow" or "by the end of the week". In academic and professional circumstances the previously mentioned deliverables are critical to monitor and evaluate a colleague or subordinate's performance. Henceforth, the aforementioned characteristics became a critical focus during annotations to outline fundamental expectations for the model's behavior.

Furthermore, supplemental data had to be manually generated in the form of data augmentation as per corporate requirements, following a very specific format of actionable deliverables as previously mentioned. The absence of grammatical errors and redundant terminology in addition to addressing the class imbalance further facilitated the model's understanding. Despite catering to actionable deliverables discussed in meeting minutes, this approach generalizes well to the domain of information extraction through conversational dialogues and can be further fine tuned for more specific assignments through further feature engineering.

Finally, we carried out extensive error analysis whereby we analyzed misclassified utterances. Numerous such utterances followed patterns from the training data which were incorrectly annotated and by iterating over thousands of utterances in the training set to exclude improper annotations such as first person imperatives, we improved on the model's misunderstandings as per the following misclassifications highlighted in the results section below.

3.4 ULMFiT Model

The Universal Language Model Fine-tuning (ULMFiT) is an inductive transfer learning approach developed by Jeremy Howard and Sebastian Ruder to all the tasks in the domain of natural language processing which sparked the usage of transfer learning in NLP tasks.

The ULMFiT approach to training NLP models is heralded as the ImageNet moment in the domain of Natural Language Processing. The model architecture used in the entire process of the ULMFiT approach is ubiquitous and is the well-known AWD-LSTM architecture. The ULMFiT approach can be broadly explained as per the following three steps in Figure 1:

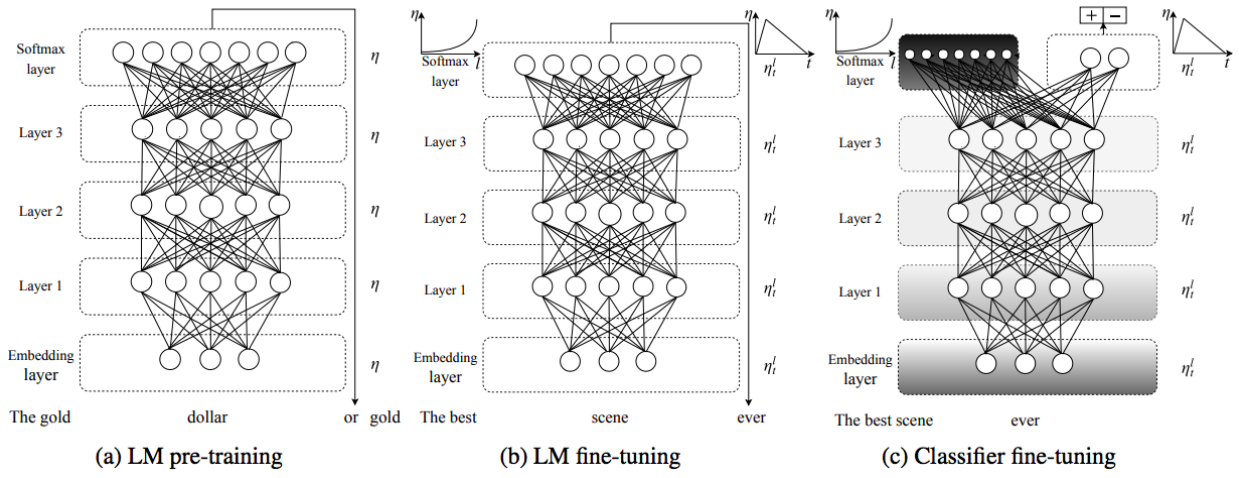

Figure 1: 3 steps of the ULMFiT model implementation

Step 1: Training a general corpus language model

A language model is first trained on a corpus of Wikipedia articles known as Wikitext-103 using a self-supervised approach, i.e. using the training labels in itself to train models, in this case training a language model to learn to predict the next word in a sequence. This resulting language model learns the semantics of the English language and captures general features in the different layers.

This pretrained language model is trained on 28,595 Wikipedia articles and the training process is very expensive and time consuming and is luckily open-sourced in the Fastai library to use.

Step 2: Fine-tuning pretrained language model to downstream dataset

Despite having a vast language model pre-trained, it's always likely that the specific downstream task we would like to build our NLP model is a part of a slightly different distribution and thus need to fine-tune this Wikitext 103 language model.

This step is much faster and it converges much faster as there will be an overlap to the general domain dataset. It only needs to adapt to the idiosyncrasies of the language used and not learn the language per say.

Since NLP models are more shallow in comparison to a computer vision model, the fine-tuning approaches need to be different and thus the paper provides novel fine-tuning techniques like discriminative fine-tuning, slanted triangular learning rates and gradual unfreezing.

Gradual Unfreezing

The idea behind gradual unfreezing is that fine-tuning a classifier on all layers can result in catastrophic forgetting and thus each layer starting from the last layer is trained one after the other by freezing all the lower layers and only training the specific layer in question in Figure 2.

The paper empirically found that after training the last layer of the model with a learning rate of lr, the subsequent layers can be trained one after another by reducing lr by a factor of 2.6.

```
In [114…
learn.freeze_to(-2)
learn.lr_find()
```

```
Out[114…
SuggestedLRs(valley=0.007585775572806597)
```

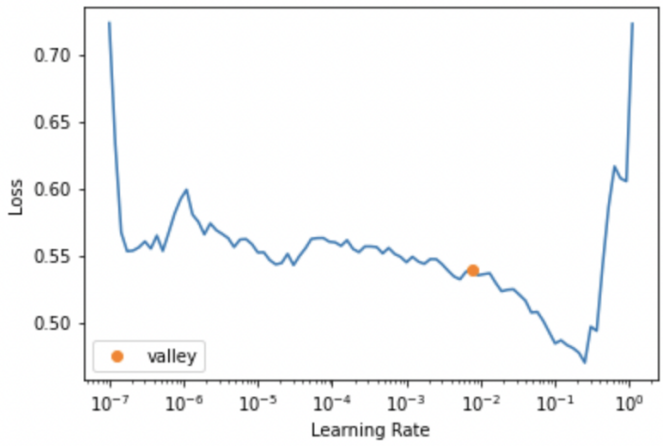

Figure 2: Implementation of gradual unfreezing of layer 2 while training using Fastai

Discriminative Fine-tuning

Since different layers of the model capture different types of information and thus they should be fine-tuned to different extents and the idea is similar to using discriminative learning rates in transfer learning tasks. In this case we would like to fine-tune the pre-trained language model to learn intricacies of the downstream dataset and not corrupt the learnings i.e weights of the

pre-trained language model trained on Wikipedia data where it learns general structures of the English language as demonstrated in Table 3.

```
learn.fit_one_cycle(1, slice(1e-3/(2.6**4), 3e-2))
```

epoch	train_loss	valid_loss	accuracy	precision_score	recall_score	fbeta_score	time
0	0.446834	0.229334	0.971731	0.974359	0.996255	0.985185	00:01

Table 3: Implementation of discriminative fine-tuning using a slice operator where earlier models are trained at lower learning rate of 1e-3 and final layers at a higher learning rate of 3e-2

Slanted Triangular Learning Rates

The idea behind slanted learning rates is that for a pre-trained language model to adapt/fine-tune itself to the downstream dataset, the fine-tuning process should ideally converge faster to a suitable region in the parameter space and then refine its parameters there as shown in Figure 3.

So the slanted learning rates approach first linearly increases the learning rates for a short period and then linearly decays the learning rate slowly which is a modification of Leslie Smith's triangular learning rate approach where the increase and decrease is almost the same.

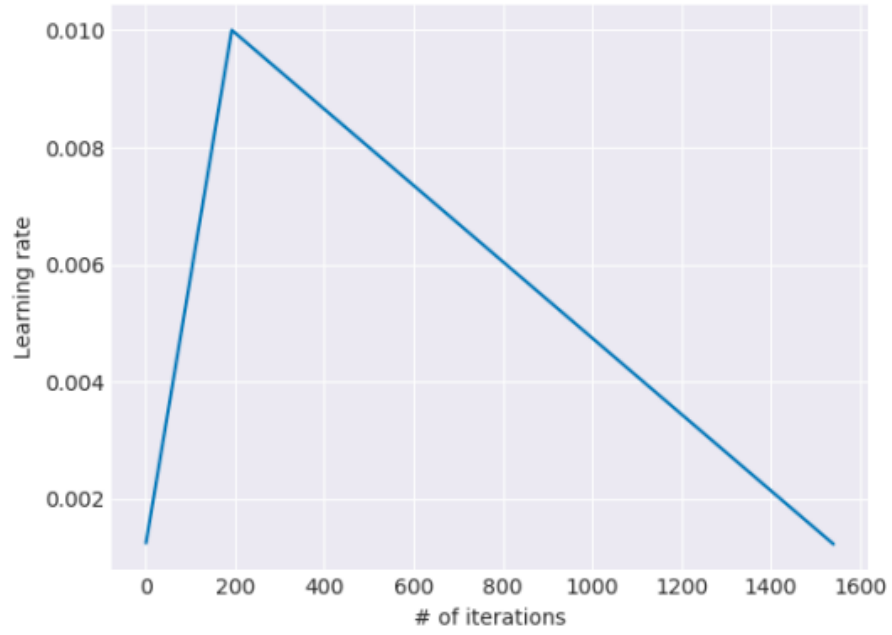

Figure 3: hyperparameter tuning of learning rate with respect to number of iterations

Step 3: Training a classifier on the downstream NLP task

Now that we have a language model fine-tuned to our downstream NLP dataset we can use the encoder portion of the fine-tuned language model which is the part that learns the features of the

language used in the downstream dataset as the base to build a text classifier for tasks such as sentiment analysis, spam detection, fraud detection, document classification etc.

The encoder saved is then appended by a simple classifier consisting of two additional linear blocks consisting of the standard batch normalization and dropout, with ReLU activations for the intermediate layer and a softmax activation at the last layer for the classification purpose. Fine-tuning a classifier is a very critical task in a transfer learning method and is the main reason why transfer learning approaches failed until ULMFiT came along.

Overly aggressive fine-tuning can result in catastrophic forgetting and too cautious fine-tuning can lead to extremely slow convergence. To tackle this problem, ULMFiT introduces a novel fine-tuning technique in gradual unfreezing besides also using slanted triangular learning rates and discriminative fine-tuning to successfully train classifiers using pre-trained language models.

3.5 Results

Multi-class classification model:

The initial model was trained to classify 6 intents desire, imperative, question, statement, suggestion and non-actionable. By looking at the results from our trained model we realized that the intents were too fine-grained and there was a class overlap in predictions as illustrated in the following Figures 4-5 and Tables 4-5 respectively:

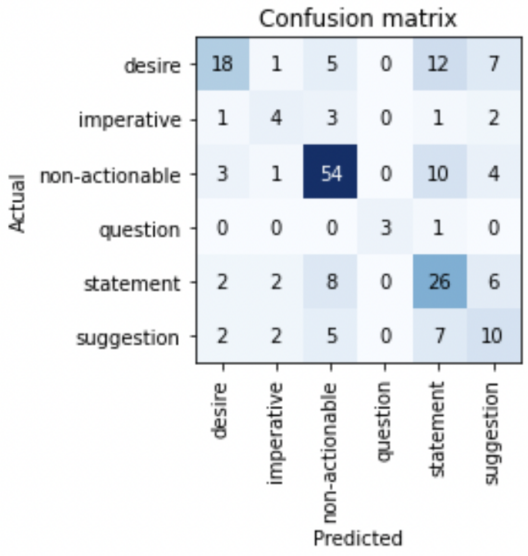

Figure 4: classification model with 6 intents

```
[('desire', 'statement', 12),
 ('non-actionable', 'statement', 10),
 ('statement', 'non-actionable', 8),
 ('desire', 'suggestion', 7),
 ('suggestion', 'statement', 7),
 ('statement', 'suggestion', 6),
 ('desire', 'non-actionable', 5),
 ('suggestion', 'non-actionable', 5),
 ('non-actionable', 'suggestion', 4),
 ('imperative', 'non-actionable', 3),
 ('non-actionable', 'desire', 3)]
```

Figure 5: Intent overlap in predictions

	text	sentiment	predicted_sentiment
609	And so I'll treat it as zero right now, because I don't want to double count my nodes, right otherwise would be inaccurate here.	desire	statement
69	I hope that they'll have a lift or at least a ramp for the overhead bridge.	desire	statement
25	I would need some more clarity on this.	desire	statement
78	I just have to fix a bug where when you say about the intense, it puts aspace between the like brackets and parentheses.	desire	statement
753	I'm gonna pass in, of course, the graph, as well as that node.	desire	statement
375	So I wish I could get like true back in this scenario, but I'm not going to get a true.	desire	statement
754	So we're going to assume I have a function here, I'll call it explore.	desire	statement
661	I'm going to list out my nodes to represent how I'm going to do the iterations to be In a traversal at every node as my starting point, so I'm going to start at node zero.	desire	statement
628	So I'll give me the nodes like 015, and so on.	desire	statement
614	So now that I have my visited set, I can pass it along, as I start the traversals.	desire	statement
759	And so what I'll do here is let me start with the iterative code.	desire	statement
659	And I'm going to mark my nodes as visited as I go, because like usual for our undirected graphs, you want to watch out and prevent any cycles that you may get trapped in.	desire	statement
750	So I'll choose to do this explore method recursively.	desire	statement
656	So I'm going to treat each of these nodes as just being a single note, of course.	desire	statement

Table 4: model struggles to identify intents which are quite similar like ‘desire’ and ‘statement’

Binary classification model:

The final model was thus trained with a focus on getting high-precision in predicting the ‘actionable’ statements from a meeting transcript with a focus to only get the top highlights from the meeting. In order to train this model, we re-annotated the above task to a binary classification problem and considering there was a huge class imbalance we curated a training set for the positive ‘actionable’ class.

Below we see an example of model running on Seasalt’s internal meeting consisting of 385 utterances sorted by prediction confidence which produces around 9 instances of ‘actionable’ utterances with 6 of them correctly classified and the last one that was later ignored after adding a threshold criteria based on the prediction confidence:

	text	predicted_sentiment	predicted_confidence
194	Are we got better results from our 20 LM adding the the finance data set to it	actionable	0.995027
170	An after working on this video certification and hopefully I can finish that in by Friday.	actionable	0.968201
208	So we were thinking to automate the issue creation from our server directly where the Cron Jobs going to run, but the idea was to attach the synthesized audio file directly to that issue, which could work, but apparently can only work manually	actionable	0.959383
29	This damn meeting in Seattle in the past year	actionable	0.906901
235	Tomorrow we're going to have this engine meeting demo from the Capstone undergrads and.	actionable	0.874238
315	Like should we be tokenizing by?	actionable	0.814643
163	He took some time on Cody's PR, and I think you know it's a little bit complicated for now and	actionable	0.788257
363	That's probably something we can talk about next week or tomorrow.	actionable	0.650301
98	Did you use Google Translator?	actionable	0.500793

Table 5: model predicts actionable utterances with a high confidence level, thresholds can be applied to trim this list

The final action extraction model developed using ULMFiT was deployed on Seasalt’s SeaMeet interface which one could use to record a meeting which would use the action extraction model to summarize the transcript with the most valuable actionable items as demonstrated in Figure 6.

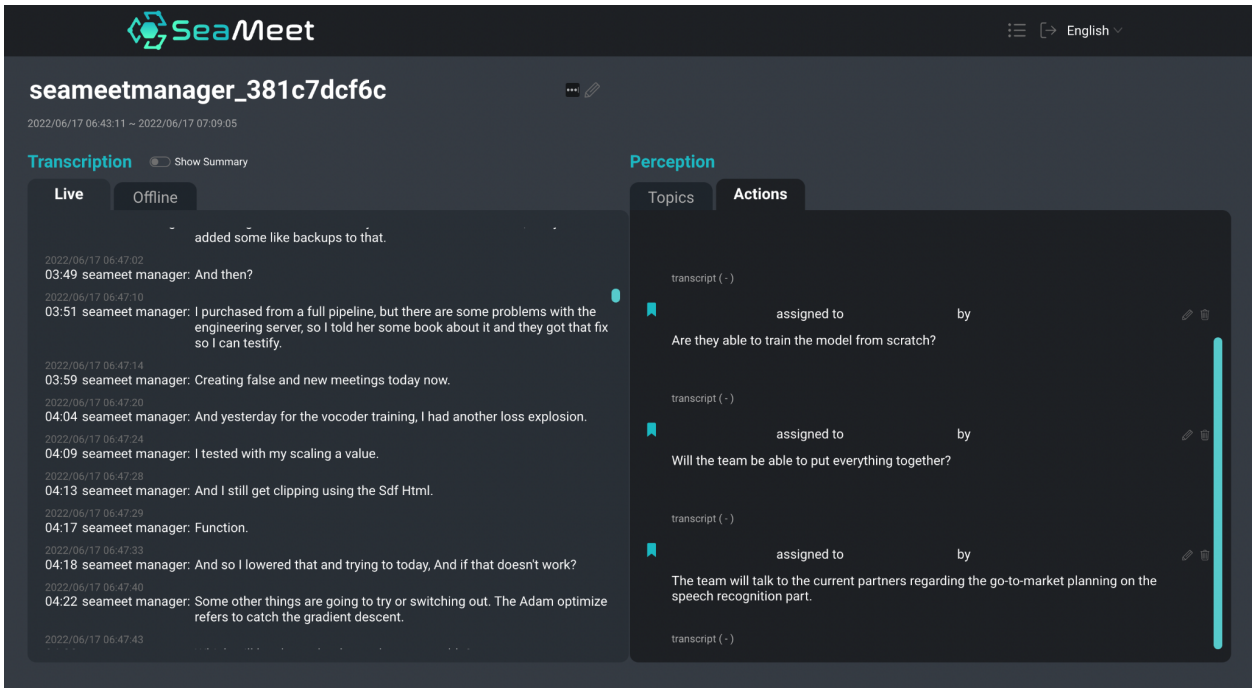

Figure 6: Final action extraction model deployed on Sesalt's SeaMeet interface

3.6 Discussions

Initially, we proposed the LUKE model specializing in deep contextualized entity representations with entity-aware self-attention with the intention of leveraging its spectacular ability to enhance named entity recognition tasks such as identifying the speaker and receiver which was not previously implemented. Through fine-tuning and experimentation, unfortunately we discovered that while LUKE demonstrated a high accuracy for named entity recognition, it failed to extract the imperative clauses in utterances from a speaker to a receiver for the following reasons.

Primarily, receiving human entities are often implied and seldom specifically mentioned. Secondly, a typical speaking human being does not mention their name in third person. Additionally, the assignee is not recognized since previous utterances are not always considered.

We experimented with building the action extraction model using various transformer architecture models from the HuggingFace libraries but found that the performance wasn't better than our final model in ULMFiT. The hypothesis is that considering we had lesser data for our downstream domain task we couldn't converge to an optimal model using the more complex transformer architectures and considering that the ULMFiT model has simpler LSTM based architecture(AWD-LSTM), it was able to converge faster and thus perform better.

In conclusion, we found much more promising results with Universal Language Model Fine-tuning for Text Classification which improves the previous implementation by the following. Primarily, ULMFiT incorporates a single architecture and training process. Second, the model works across tasks varying in document size, number, and label type. Also, ULMFiT does not require custom feature engineering or pre-processing. Finally, the previously mentioned model does not require additional domain specific labels as it is a semi-supervised approach.

4. Sentiment Analysis

Live Model.

4.1 Problem Statement

As a proof of concept, we evaluated several machine learning models for sentiment analysis. Pre-training on Twitter and IMDB datasets seems to have the most effective output since both of the previously mentioned datasets are speaking a colloquial language in comparison to written text such as Wikipedia which is very formal. A typical use case for sentiment analysis is the multi-class classification of utterances as per the following representation in Table 6.

index	text	class
0	match 1 : tag team table match bubba ray and spike dudley vs eddie guerrero and chris benoit bubba ray and spike dudley started things off with a tag team table match against eddie guerrero and chris benoit . according to the rules of the match , both opponents have to go through tables in order to get the win . benoit and guerrero heated up early on by taking turns hammering first spike and then bubba ray. a german by benoit to bubba took the wind out of the dudley brother . spike tried to help his brother , but the referee restrained him while benoit and guerrero	pos
1	some have praised lost as a disney adventure for adults . i don't think so -- at least not for thinking adults . \n\n this script suggests a beginning as a live - action movie , that struck someone as the type of crap you can not sell to adults anymore . the " crack staff " of many older adventure movies has been done well before , (think the dirty dozen) but represents one of the worse films in that motif . the characters are weak . even the background that each member trots out seems stock and awkward at best .a tomboy mechanic whose father always wanted sons , if we have not at least seen these before	neg
2	warning does contain spoilers . open your eyes if you have not seen this film and plan on doing so , just stop reading here and take my word for it . you have to see this film . i have seen it four times so far and i still have n't made up my mind as to what exactly happened in the film . that is all i am going to say because if you have not seen this film , then stop reading right now . if you are still reading then i am going to pose some questions to you and maybe if anyone has any answers you can email me and let me know what you think . i remember my grade 11 english teacher quite well	pos

Table 6: sample utterances receiving a binary classification for sentiment analysis

The motivation behind this exercise was to draw statistics for customer sentiment throughout the conversation from transcribed speech for which ULMFiT model yields the best results, as per the following example embedded in a friendly user interface as demonstrated in Figure 7.

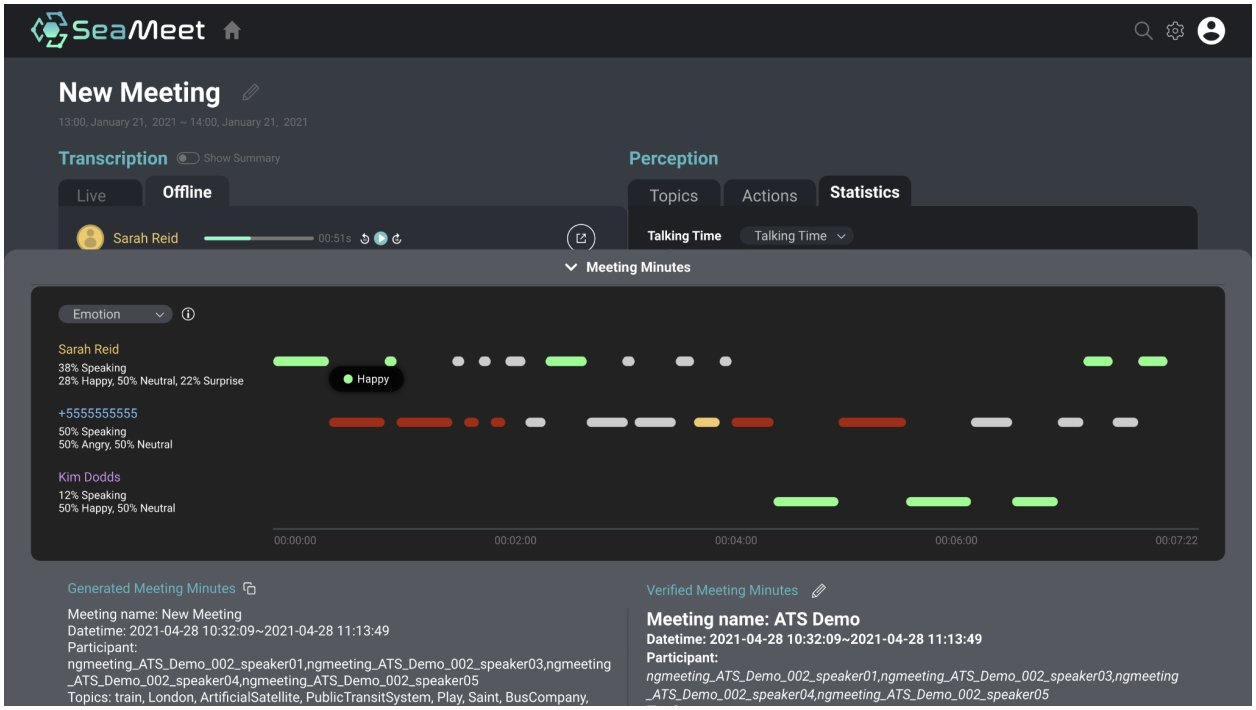

Figure 7: Client's emotion distribution for the customer service representative.

4.2 Proposed Implementation

Inspired by AlBadani et al. [10] where they demonstrated an innovative machine learning approach for sentiment analysis on Twitter incorporating the Universal Language Model Fine-Tuning and Support Vector Machines, we implemented analogous methodologies on conversational dialogues transcribed from internal manually annotated datasets of Youtube videos simulating irritated customers speaking to customer service representatives.

Howard et al. [15] support this approach by introducing techniques that are very important for fine-tuning the previously mentioned language model, such as the following which have been incorporated into our proposed modifications. Primarily, training the language model on a domain specific corpus facilitates in capturing the language's sentiment features in different layers, such as the interaction between an irritated client and a customer service representative.

Second, and more importantly the language model is trained on the target dataset using discriminative fine-tuning and slanted triangular learning rates to learn task-specific features such as irritable conversations over financial transactions, for example. Last but not least, the classifier is adjusted for the target task utilizing discriminative fine-tuning, slanted triangular learning rates, as well as progressive unfreezing in order to maintain low-level representations.

4.3 ULMFiT Model

The Universal Language Model Fine-tuning (ULMFiT) is an inductive transfer learning approach developed by Jeremy Howard and Sebastian Ruder to all the tasks in the domain of natural language processing which sparked the usage of transfer learning in NLP tasks.

The ULMFiT approach to training NLP models is heralded as the ImageNet moment in the domain of Natural Language Processing. The model architecture used in the entire process of the ULMFiT approach is ubiquitous and is the well-known AWD-LSTM architecture. The ULMFiT approach can be broadly explained as per the following three steps illustrated in Figure 8.

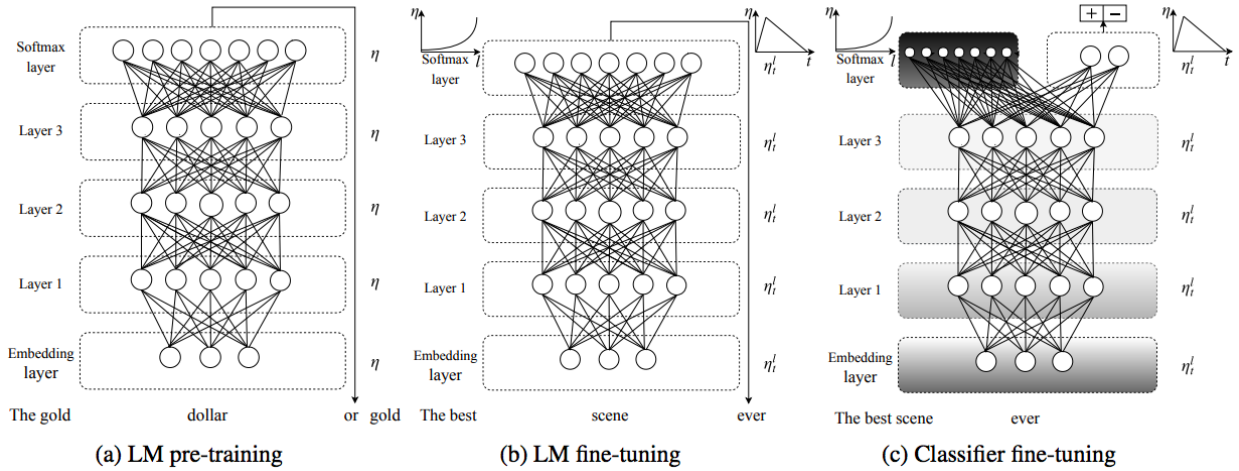

Figure 8: 3 steps of the ULMFiT model implementation

Step 1: Training a general corpus language model

A language model is first trained on a corpus of Wikipedia articles known as Wikitext-103 using a self-supervised approach, i.e. using the training labels in itself to train models, in this case training a language model to learn to predict the next word in a sequence. This resulting language model learns the semantics of the English language and captures general features in the different layers.

This pretrained language model is trained on 28,595 Wikipedia articles and the training process is very expensive and time consuming and is luckily open-sourced in the Fastai library to use.

Step 2: Fine-tuning pretrained language model to downstream dataset

Despite having a vast language model pre-trained, it's always likely that the specific downstream task we would like to build our NLP model is a part of a slightly different distribution and thus need to fine-tune this Wikitext 103 language model.

This step is much faster and it converges much faster as there will be an overlap to the general domain dataset. It only needs to adapt to the idiosyncrasies of the language used and not learn the language per say.

Since NLP models are more shallow in comparison to a computer vision model, the fine-tuning approaches need to be different and thus the paper provides novel fine-tuning techniques like discriminative fine-tuning, slanted triangular learning rates and gradual unfreezing.

Gradual Unfreezing

The idea behind gradual unfreezing is that fine-tuning a classifier on all layers can result in catastrophic forgetting and thus each layer starting from the last layer is trained one after the other by freezing all the lower layers and only training the specific layer in question as in Fig. 9.

The paper empirically found that after training the last layer of the model with a learning rate of lr, the subsequent layers can be trained one after another by reducing lr by a factor of 2.6.

```
In [114…
learn.freeze_to(-2)
learn.lr_find()
```

```
Out[114…  SuggestedLRs(valley=0.007585775572806597)
```

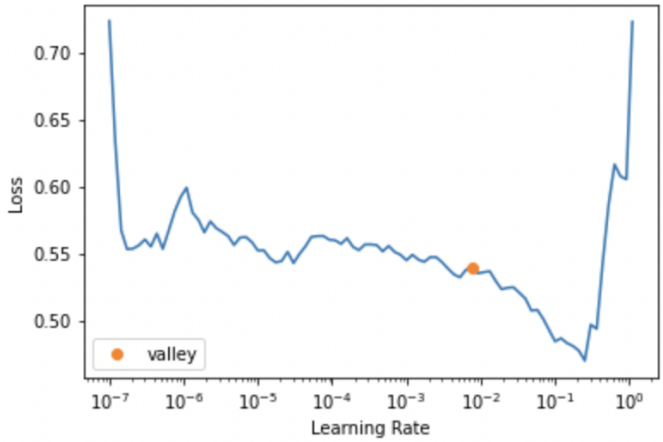

Figure 9: Implementation of gradual unfreezing of layer 2 while training using Fastai

Discriminative Fine-tuning

Since different layers of the model capture different types of information and thus they should be fine-tuned to different extents and the idea is similar to using discriminative learning rates in transfer learning tasks. In this case we would like to fine-tune the pre-trained language model to learn intricacies of the downstream dataset and not corrupt the learnings i.e weights of the pre-trained language model trained on Wikipedia data where it learns general structures of the English language as demonstrated in Table 7.

```
learn.fit_one_cycle(1, slice(1e-3/(2.6**4), 3e-2))
```

epoch	train_loss	valid_loss	accuracy	precision_score	recall_score	fbeta_score	time
0	0.446834	0.229334	0.971731	0.974359	0.996255	0.985185	00:01

Table 7: Implementation of discriminative fine-tuning using a slice operator where earlier models are trained at lower llr of 1e-3 and final layers at a higher lr of 3e-2

Slanted Triangular Learning Rates

The idea behind slanted learning rates is that for a pre-trained language model to adapt/fine-tune itself to the downstream dataset, the fine-tuning process should ideally converge faster to a suitable region in the parameter space and then refine its parameters there as shown in Figure 10.

So the slanted learning rates approach first linearly increases the learning rates for a short period and then linearly decays the learning rate slowly which is a modification of Leslie Smith's triangular learning rate approach where the increase and decrease is almost the same.

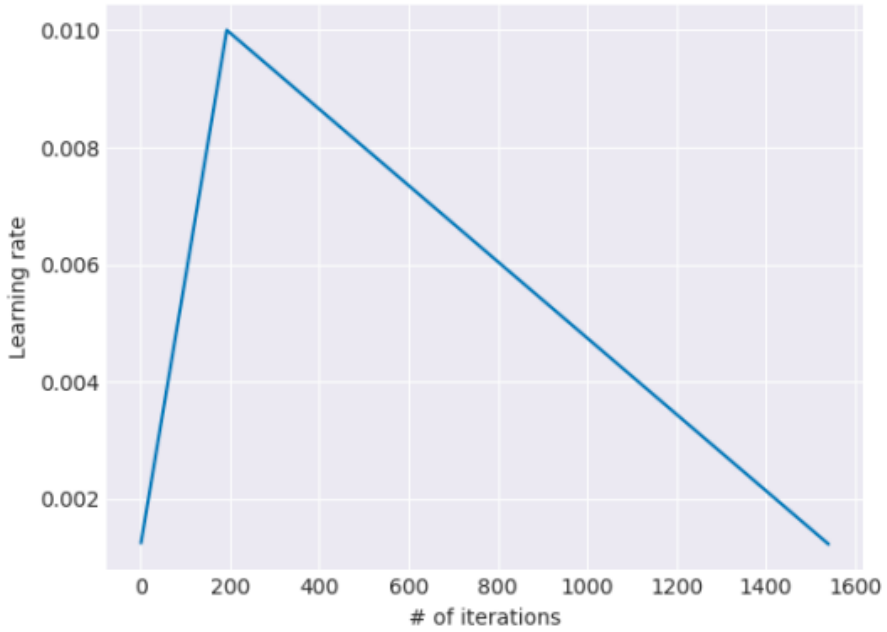

Figure 10: hyperparameter tuning of learning rate with respect to number of iterations

Step 3: Training a classifier on the downstream NLP task

Now that we have a language model fine-tuned to our downstream NLP dataset we can use the encoder portion of the fine-tuned language model which is the part that learns the features of the language used in the downstream dataset as the base to build a text classifier for tasks such as sentiment analysis, spam detection, fraud detection, document classification etc.

The encoder saved is then appended by a simple classifier consisting of two additional linear blocks consisting of the standard batch normalization and dropout, with ReLU activations for the intermediate layer and a softmax activation at the last layer for the classification purpose. Fine-tuning a classifier is a very critical task in a transfer learning method and is the main reason why transfer learning approaches failed until ULMFiT came along.

Overly aggressive fine-tuning can result in catastrophic forgetting and too cautious fine-tuning can lead to extremely slow convergence. To tackle this problem, ULMFiT introduces a novel fine-tuning technique in gradual unfreezing besides also using slanted triangular learning rates and discriminative fine-tuning to successfully train classifiers using pre-trained language models.

4.4 Results

Despite challenges encountered in the transcription process, our ULMFiT model exhibits decent performance in binary classification of positive and negative. During our deployment to the production server this was modified to include all three of positive, negative and neutral. A classification report of the previously mentioned binary classifications is represented as illustrated in Table 8.

	precision	recall	f1-score	support
0	0.89	0.88	0.89	6250
1	0.89	0.89	0.89	6250
accuracy			0.89	12500
macro avg	0.89	0.89	0.89	12500
weighted avg	0.89	0.89	0.89	12500

Table 8: classification matrix for IMDB sentiment classification proof of concept (positive and negative)

Furthermore, ULMFiT served as the underlying model driving this graphical user interface publicly hosted on HuggingFace spaces for further experimentation as shown in Figure 11.

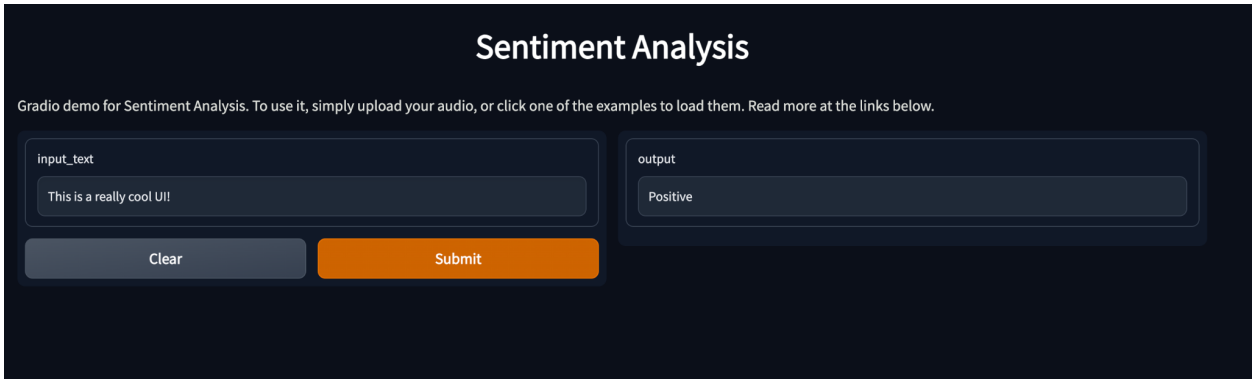

Figure 11: sentiment analysis model UI as deployed on huggingface spaces

4.5 Discussions

As per the previously mentioned sample utterances, we observed that the binary classifications for sentiment analysis demonstrated a higher accuracy than the multi-class classifications including a “neutral” class. This can be significantly attributed to the ambiguity of applying a neutral threshold such as the upper and lower bounds of neutral classification from zero.

Fine-tuning on meeting minutes and placing upper and lower bounds for a neutral threshold requires an empirical analysis and is highly recommended for further investigation before production deployment. In the meantime, the previously mentioned classification report demonstrates a high precision of true positive therefore concurrently reinforcing the F1-Score.

During the training process, we evaluated over many epochs and observed a “sweet spot” at epoch 10 where the training loss dropped significantly and the training accuracy jumped

significantly as well. Unfortunately, this performance improvement in the training set was not reflected in the testing loss or testing accuracy.

On the contrary however, beyond epoch 10 we observed the testing loss increase slightly and testing accuracy decrease slightly and therefore we would select 10 as the ideal epoch as per the following graphical demonstration in Figures 12 and 13 respectively.

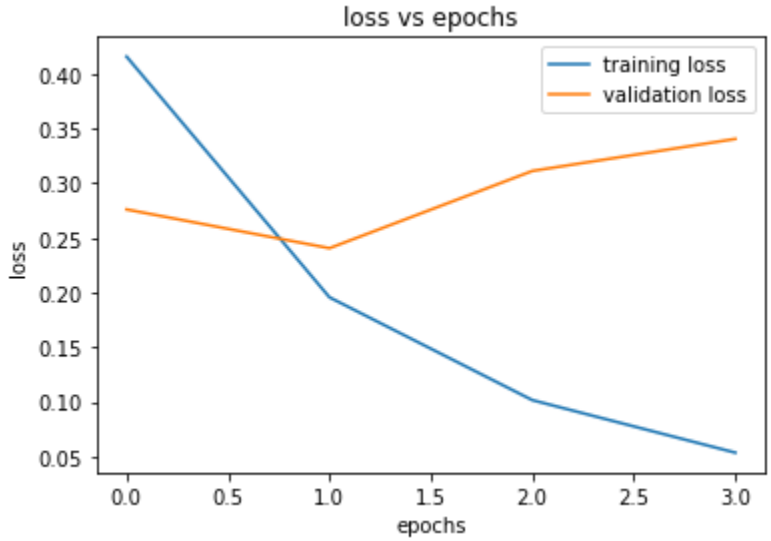

Figure 12: Training and Validation Loss comparison with respect to increasing epochs

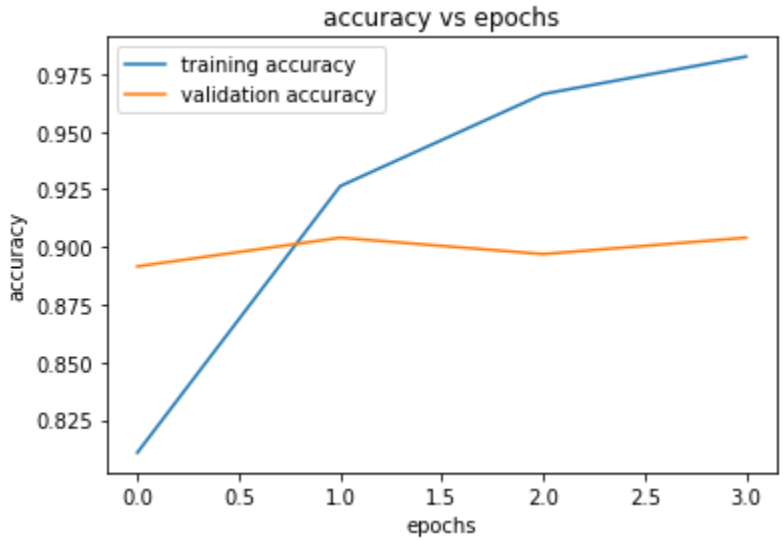

Figure 13: Training and Validation Accuracy comparison with respect to increasing epochs

5. Speech Emotion Recognition

Live Model.

5.1 Problem Statement

Speech Emotion Recognition (SER) is a performance parameter for conversational analysis that can be used to evaluate customer satisfaction.The corporate requirement designated for this model to be used in call centers to classify calls according to emotions and can be used as a performance parameter for identifying customer satisfaction, our proof of concept addresses the previously mentioned utterance by utterance for the purpose of simplicity and size of audio files. A sample output of the corresponding multi-class classifications for audio files can be illustrated as per the following Table 9.

name	path	emotion
d05 (2)	/content/data/aesdd/disgust/d05 (2).wav	disgust
s16 (6)	/content/data/aesdd/sadness/s16 (6).wav	sadness
a18 (5)	/content/data/aesdd/anger/a18 (5).wav	anger
h17 (6)	/content/data/aesdd/happiness/h17 (6).wav	happiness
f18 (3)	/content/data/aesdd/fear/f18 (3).wav	fear

Table 9: sample speech emotion recognition classifications for audio files

5.2 Proposed Implementation

Inspired by Baevski et al. [13] we experimentally demonstrated that learning representations from speech audio alone and fine-tuning on audio files of irritated customers in Wav2vec 2.0 offers a more refined approach whereby the second phase of training is supervised fine-tuning, during which labeled data is used to teach the model to predict particular words or phonemes.

This approach incorporates masking voice input and solves a problem involving both morphology and syntax such as classifying emotion towards customer service representatives. Aspiring for exceptional accuracy, our proof of concept training and validation sets were extracted from the following datasets and then tested on in house conversational dialogues and finally deployed through a continuous integration pipeline to a friendly graphical user interface.

- Crowd-sourced Emotional Multimodal Actors Dataset (Crema-D)
- Ryerson Audio-Visual Database of Emotional Speech and Song (Ravdess)
- Surrey Audio-Visual Expressed Emotion (Savee)
- Toronto emotional speech set (Tess)

5.3 Model Diagram

Wav2Vec 2.0 employs a self-supervised training method based on contrastive learning for automatic speech recognition. It requires less labeled data to get satisfactory results while learning speech representation on a large, unlabeled sample.

Additionally, it applies the transformer to improve the voice representation with context after using convolutional layers to preprocess the raw waveform. Finally, the objective is a weighted sum of contrastive and diversity loss functions and quantization is used to create targets in self-supervised learning which is represented as per the following illustration in Figure 14.

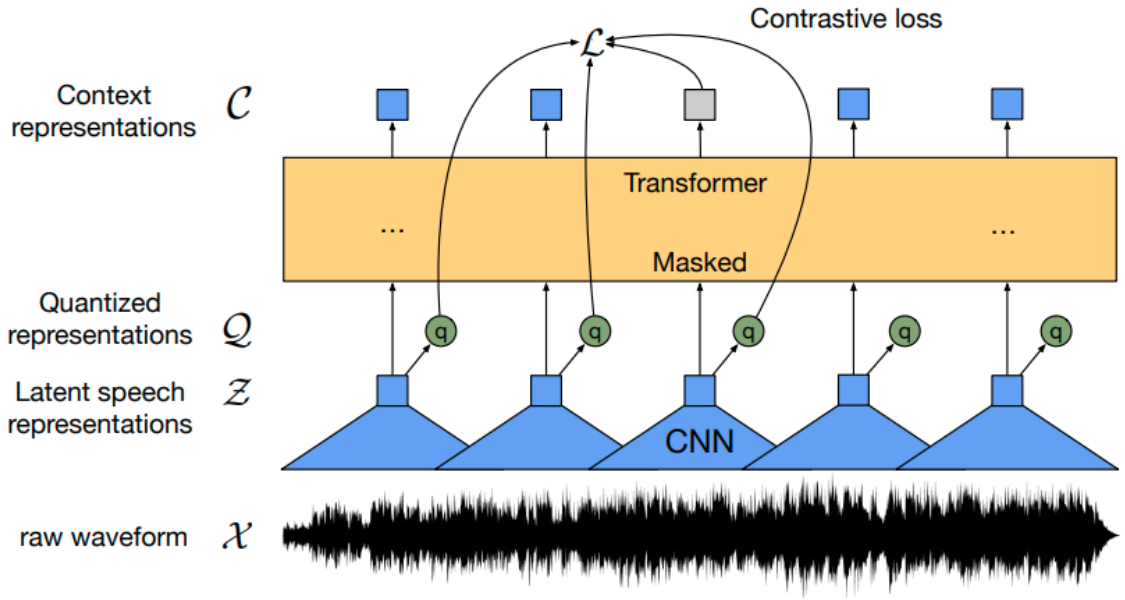

Figure 14: graphical representation of wav2vec 2.0 architecture consisting of a waveform, latent speech representations, quantized representations, context representation and contrastive loss

The previously mentioned model can learn about data even in the absence of labels due to a self-supervised, task-independent deep learning approach known as contrastive learning. By identifying which sorts of phrases are similar and which ones are distinct, wav2Vec 2.0 learns generic properties about the dataset.

In the event that labels are sparse, the pre-trained model with a generic grasp of the data may be tailored for a particular objective such as speech emotion recognition to significantly enhance label efficiency and potentially outperform various other supervised approaches in comparison.

5.4 Results

During our proof of concept, we observed a sharp rise in confidence by eliminating two of the least frequently occurring classes such as “calm” and “surprise”. The following classification report demonstrates relatively high precision across all classes representing strong true positives over false positives.

Furthermore, the classification report also demonstrates relatively high recall scores representing stronger true positives over false negatives. Therefore, we see reasonable accuracy despite class imbalance in the training set as demonstrated in the following Table 10.

	precision	recall	f1-score	support
angry	0.79	0.70	0.74	1473
calm	0.64	0.71	0.68	140
disgust	0.52	0.51	0.52	1445
fear	0.60	0.53	0.57	1420
happy	0.56	0.59	0.57	1445
neutral	0.59	0.62	0.60	1296
sad	0.59	0.68	0.64	1415
surprise	0.84	0.82	0.83	488
accuracy			0.62	9122
macro avg	0.64	0.65	0.64	9122
weighted avg	0.62	0.62	0.62	9122

Table 10: classification report for speech emotion recognition including imbalance classes “calm” and “surprise”

5.5 Discussions

Analyzing audio files is particularly challenging due to lack of human understanding regarding acoustic features such as intonation, complex tones, vowels and voiceless fricatives [16]. Visualizations facilitate this understanding such as the waveplot and spectrogram for happiness demonstrated in Figures 15 and 16 respectively, in comparison to the waveplot and spectrogram for anger demonstrated in Figures 17 and 18 respectively.

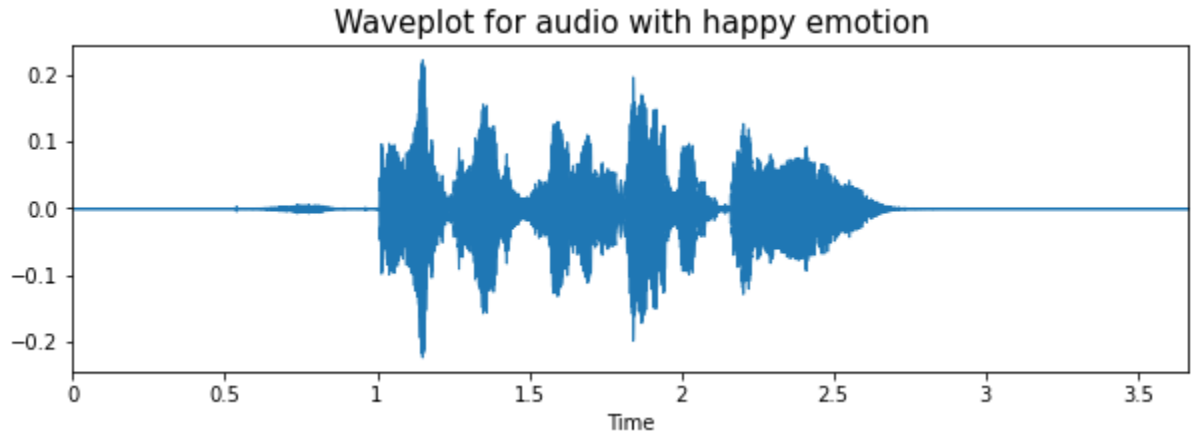

Figure 15: Waveplot for audio with happy emotion

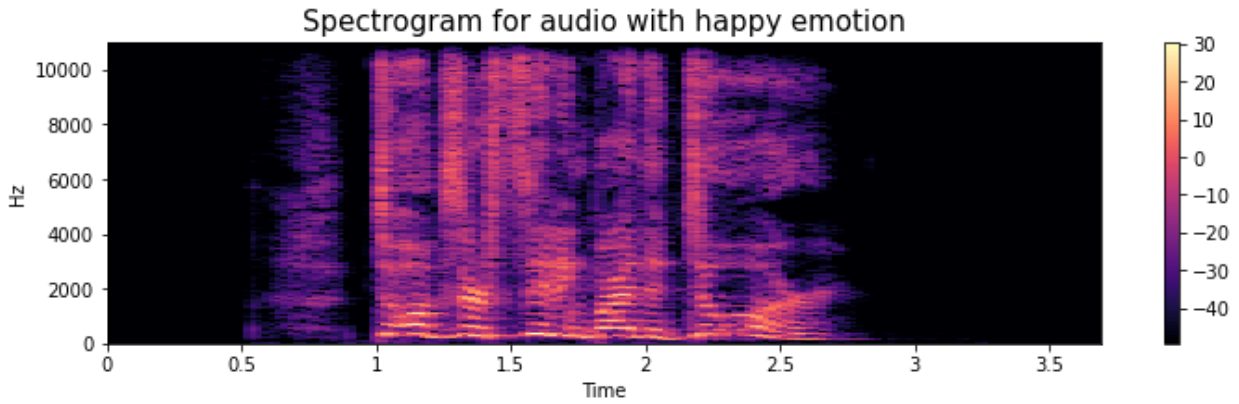

Figure 16: Spectrogram for audio with happy emotion

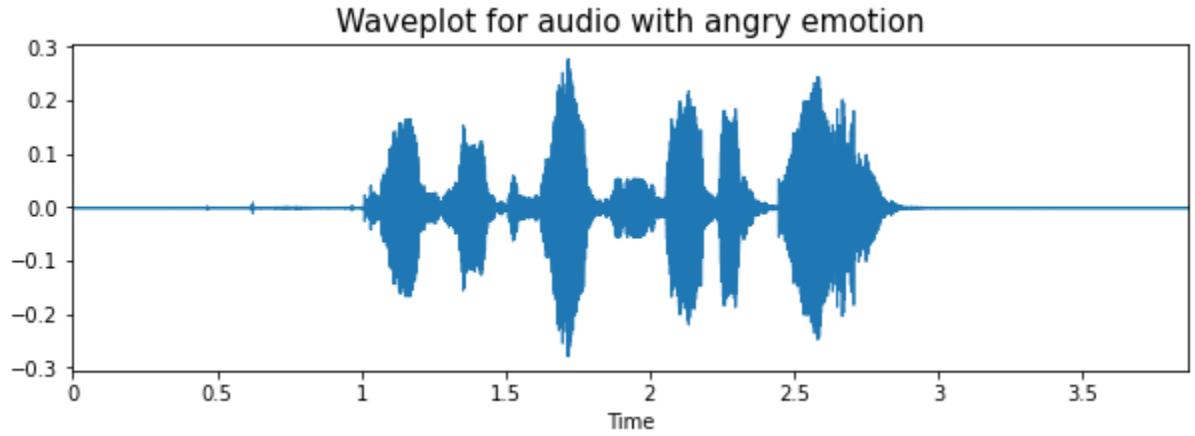

Figure 17: Waveplot for audio with angry emotion

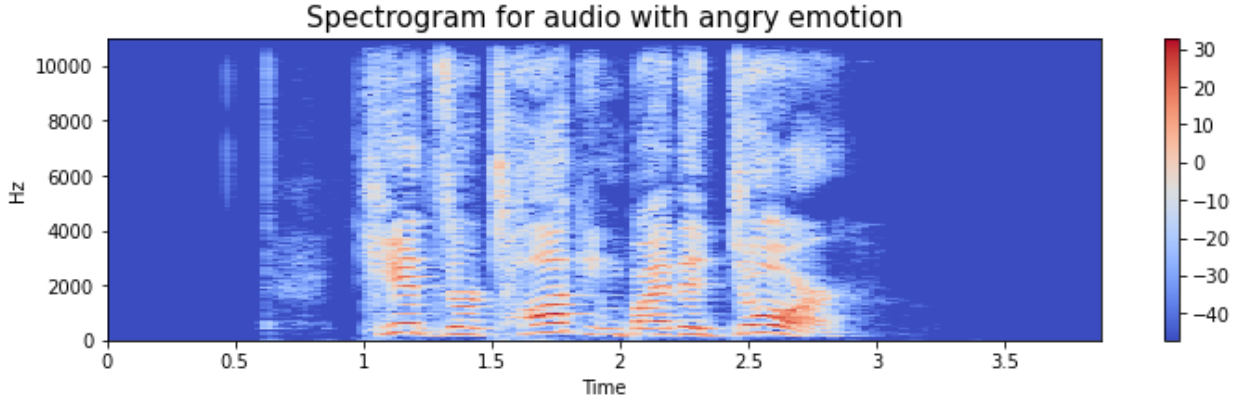

Figure 18: Spectrogram for audio with angry emotion

During the training process, we evaluated over many epochs and observed a “sweet spot” at epoch 40 where the training loss dropped significantly and the training accuracy jumped significantly as well. Unfortunately, this performance improvement in the training set was not reflected in the testing loss or testing accuracy. On the contrary however, beyond epoch 40 we observed the testing loss increase slightly and testing accuracy decrease slightly and therefore we would select 40 as the ideal epoch as per the following graphical demonstration in Figure 19.

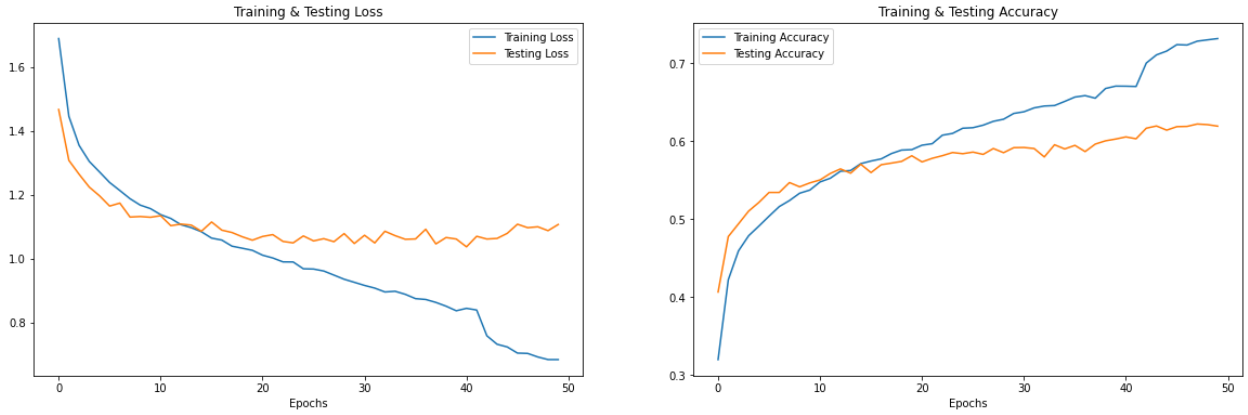

Figure 19: Training and Testing Loss and Accuracy comparison with respect to increasing epochs

As per the following confusion matrix we can see that there is a significant class imbalance for two categories such as “calm” and “surprise”. We observed a sharp rise in confidence by eliminating two of the least frequently occurring classes such as “calm” and “surprise” during our proof of concept. Strictly adhering to corporate requirements however, we had to maintain all of the previously mentioned classes which are graphically represented in the next Figure 20.

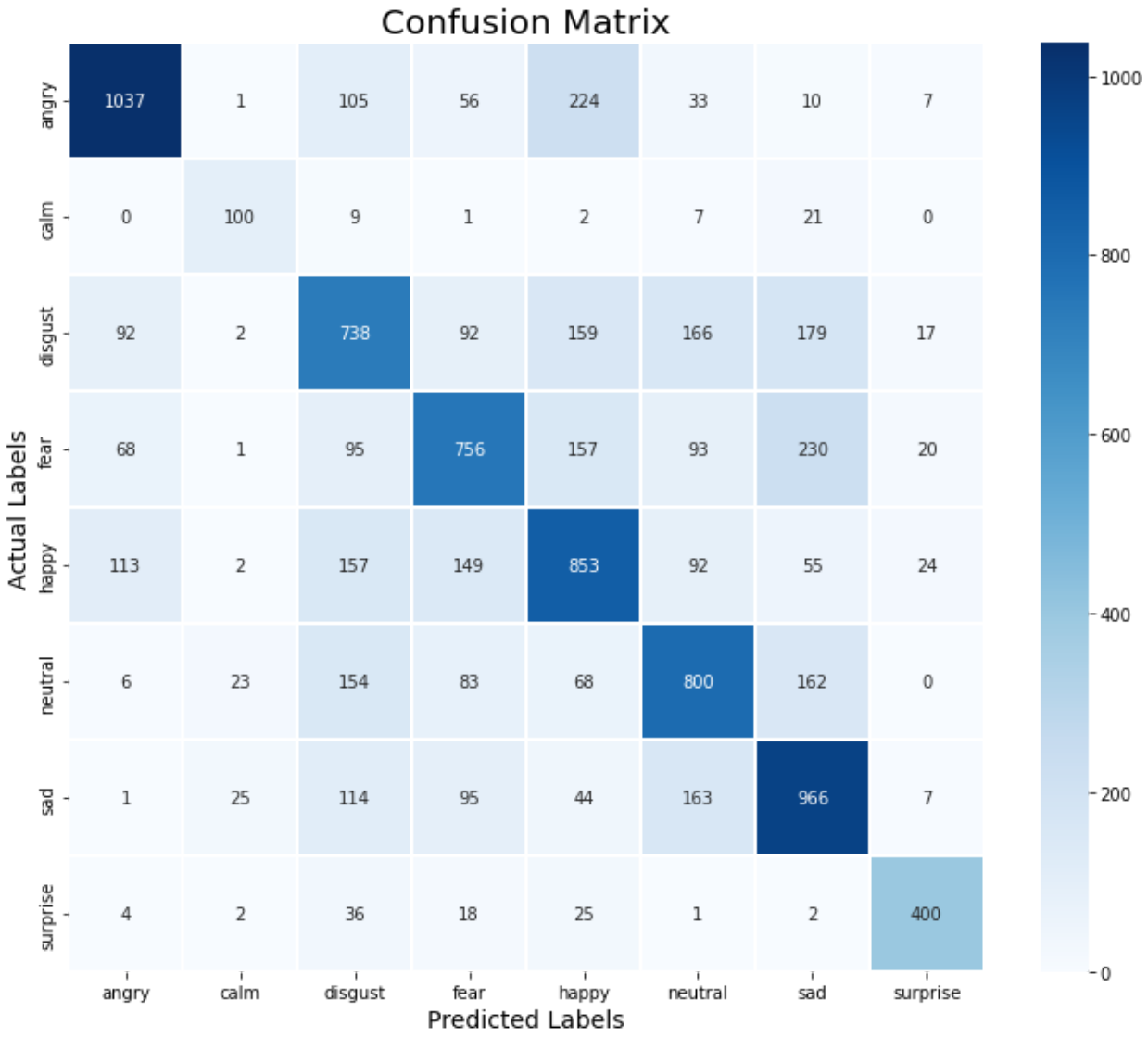

Figure 20: confusion matrix for speech emotion recognition including imbalance classes “calm” and “surprise”

Despite the challenges faced in analyzing audio files, this wav2vec 2.0 implementation serves as a foundation for the friendly user interface illustrated in Figure 7 which offers a holistic distribution of a client’s emotions to the corresponding customer service representative.

For future reference, and motivated by corporate strategy for image classifications we would highly recommend for the technical department to take a deeper dive into a new cutting edge methodology called data2vec whereby it extrapolates upon the concepts of wave2vec 2.0 by being applicable to vision, speech as well as text inputs [17].

Furthermore, ULMFiT served as the underlying model driving this graphical user interface publicly hosted on HuggingFace spaces for further experimentation as shown in Figure 21.

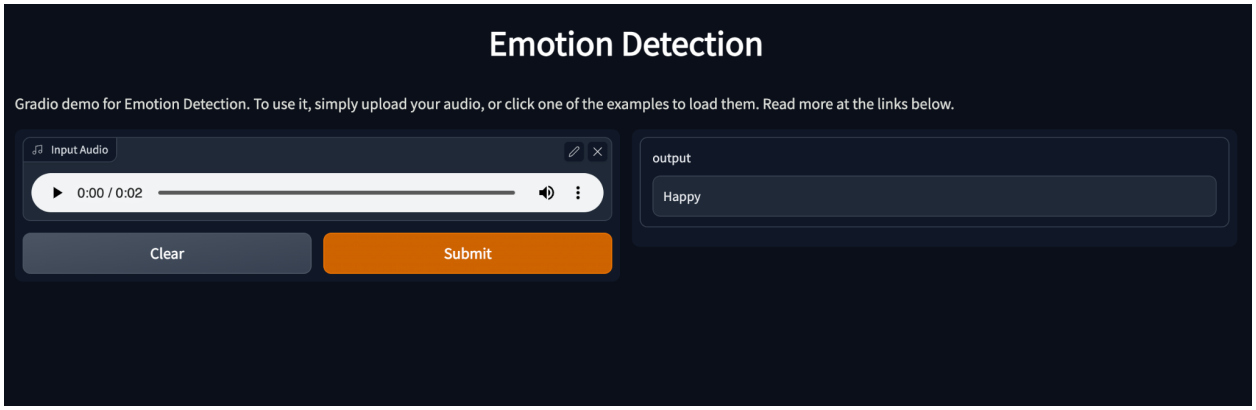

Figure 21: [speech emotion recognition model] UI as deployed on huggingface spaces

6. Short Summary

6.1 Problem Statement

The short summary is focused on the turn level, which means that given a single line from a speaker in a recorded meeting, the model would generate a summarization on this single line. In terms of use case, it is incorporated with the rest of SeaWord. The product accepts a recording of a meeting and parses the audio into speech text, later feeding the generated speech text to derive summarization. Since the existing models occasionally return obscure summaries, due to the noise from text to speech system, there is a high priority requirement to address the coherence and accuracy of the previously mentioned summaries.

6.2 Current Implementation

A transformers library has been implemented alongside a pre-trained BART model which has been fine-tuned on the SAMSum corpus. A BERT score has been implemented to gauge the performance of summarization. BERTScore uses the pre-trained contextual embeddings from BERT and matches words in candidate and reference sentences by cosine similarity. It has been used to associate with human judgment on sentence-level and system-level evaluation. BERT Score computes precision, recall, and F1-score, helping evaluate different language generation.

The data used for summarization is the SAMSum dataset. For multilingual summarization, the data is a machine translated version of the SAMSum Corpus in the target language such as Chinese. The SAMSum data corpus is made of 16369 conversations created and written down by linguists fluent in English with manually annotated summaries in json format, and has been partitioned based on an 80/10/10 split for train, dev, test respectively.

- SAMSum Data Example
    - Dialogue: Amanda: I baked cookies. Do you want some?
      Jerry: Sure!
      Amanda: I'll bring you tomorrow :-)
    - Summary: Amanda baked cookies and will bring Jerry some tomorrow.

6.3 Proposed Modifications

The team proposed a procedure which implements a preliminary step of evaluating the performance of the model by manually annotating a gold standard test set followed by evaluating on metrics like Precision, Recall and F1-Score for debugging and troubleshooting.

When we annotated the gold standard test data, we observed that the data of the current speech-to-text system are not so clean. For example, many complete sentences are cut into more than one line, and for short summarization, each line is a single data. Therefore, these incomplete sentences could lead to missing information. The current speech-to-text system results contain noise, while the SAMSum data are much cleaner. Hence, we decided to clean and rebuild test data as well as try some data augmentation methods in the training process. A few examples of SAMSum data and the test data from the speech-to-text system are demonstrated in Figure 22.

Dialogue
Blair: Remember we are seeing the wedding planner after work Chuck: Sure, where are we meeting her? Blair: At Nonna Rita's Chuck: Can I order their seafood tagliatelle or are we just having coffee with her? I've been dreaming about it since we went there last month Blair: Haha sure why not Chuck: Well we both remmber the spaghetti pomodoro disaster from our last meeting with Diane Blair: Omg hahaha it was all over her white blouse Chuck: :D Blair: :P
Summary
Blair and Chuck are going to meet the wedding planner after work at Nonna Rita's. The tagliatelle served at Nonna Rita's are very good.

```
 'Speaker_8: So yesterday I got a bunch of small things done, resolve that dependency conflict and got the refinance b
ot deployed and working.\n',
 'Speaker_8: Test link on GBM.\n',
 'Speaker_8: I fixed those excessive warnings on elastic selector and updated the Andrew Chat\n',
 'Speaker_8: and it got files\n',
 'Speaker_8: terasa two format\n',
 'Speaker_8: and then I started translating the Korean menu\n',
 "Speaker_8: so I'll continue working on that.\n",
 "Speaker_8: I still have like all of the training data to translate, but I can use Sam's.\n",
 'Speaker_8: Google Translate script to help with that.\n',
```

Figure 22: SAMSum data example [14] and test data from speech-to-text system.

We concatenated the incomplete sentences and manually built a gold standard test dataset on the cleaned data. We tried three mini models with each pretrained on 100 SAMSum data to find the appropriate data augmentation method. The three mini models are such as the following:

- The original model (BART)
- The model trained with data augmented by Python packages (Textattack)
- The model trained with data generated by SeaMeet STT

Besides these three mini models, we tried generating a short sentence training set for each utterance extracted from SAMsum corpus instead of a summary from the whole dialogue.

We separated the summarization for the whole dialogue into sentences. Then for each utterance in the dialogue and for each sentence in the summarization, we compare the BERT score between them. If the BERT score exceeds the certain threshold, we would add these utterance, summary sentence pairs into our training set. We tried different values of the threshold and for now 0.5 seems to be a suitable value, considering the number of the data we need for training. Here is an example of the short sentence training set.

The origin data:

- Dialogue:
  A: Who wants to go hiking this weekend?
  B: me!
  C: I'm in as well.
  D: count me in! I'll come with E of course.
  E: I can drive.
  B: I'll bring some snacks.
  C: Sounds good.
- Summary: A, B, C, D and E are planning to hike this weekend. E will drive. B will bring snacks.

The short summarization version:

- Dialogue: E: I can drive.
- Summary: E will drive.

These short utterance summaries are used to train a separate model specified for short sentence summarization to compare with the original one. We assumed that a specified short summary model with training data closely resembling the testing data would yield improved performance.

We tried models with these short utterance summaries and compared the BERT score with the original model and our findings are demonstrated via tabular representations in Tables 11-13.

In addition to the data cleaning and augmentation methods proposed above, we also experimented with numerous off the shelf models on HuggingFace. First, we selected two distilled BART models pre-trained on the SAMSum dataset to see if it would give any substantial improvement. In particular we tried two of the most downloaded models:

- *distilbart-cnn-12-6-samsum* from Shilschmid.
  - This model was trained using Amazon SageMaker and the Hugging Face Deep Learning container. We chose this model to find out if a distilled smaller student model could outperform the current model and provide easy deployment for practical use.
- *bart-large-cnn-samsum* from Shilschmid.
  - This model was trained using Amazon SageMaker and the Hugging Face Deep Learning container. It uses the same BART base model as the current model and uses the same SAMSum dataset for fine tuning. We selected this model to verify that the current model is trained correctly and converged.

  With the same training set and similar methods from our original model, these models result in close but deficient scores compared to the original model.

Second, we explored models pretrained on other dataset, for example:

- *pegasus-large* from Google
  - The Pegasus model is trained with sampled gap sentence ratios on both C4 and HugeNews, and stochastically samples important sentences. The Pegasus model came out more recently than the BART model and performed better on multiple

summarization tasks. It was not tested in SeaSalt's early experiment for the current model. For such reasons, we decided to compare it with the current model.

- *pegasus-xsum* from Google
    - This is a similar model to the above one. It is trained on the Xsum data set. The Extreme Summarization (XSum) dataset is a dataset for evaluation of abstractive single-document summarization systems. We tested it as another alternative candidate model.

As expected, the results were even worse, since the Pegasus dataset contains mostly articles and patents, which does not align with our dialogue format test data. Finally, we reached out to the T5 model, a generative unsupervised model, hoping it would make a breakthrough:

- *T5-large-samsum-deepspeed* by Henryu-lin
    - This model was trained using Microsoft's AzureML and DeepSpeed's ZeRO 2 optimization. It was fine-tuned on the SAMSum corpus from t5-large checkpoint. We selected it since it was the only T5 model we can find pre-trained on the SAMSum dataset.

To our dismay, the T5 model generates repetitive and long summary. We suspected, since the model was pre-trained in an unsupervised manner and then fine-tuned on SAMSum dataset which has much longer dialogues, the generative nature of the model makes T5 tend to repeat itself when the input sentence provides insufficient information, thus giving a longer than expected summary. We reserve this method for later experiments with the long summary.

6.4 Model Diagram

An autoregressive decoder is used to determine the probability of the original document (right) after the damaged document (left) has been encoded using a bidirectional model. We employ representations from the final hidden state of the decoder and input an uncorrupted document to the encoder and decoder for fine-tuning.

The autoregressive decoder can be directly fine-tuned for sequence generation tasks such as abstractive question answering and summarization. BART is remarkably useful when fine-tuned for text generation but also works well for comprehension tasks as demonstrated in Figure 23 [4].

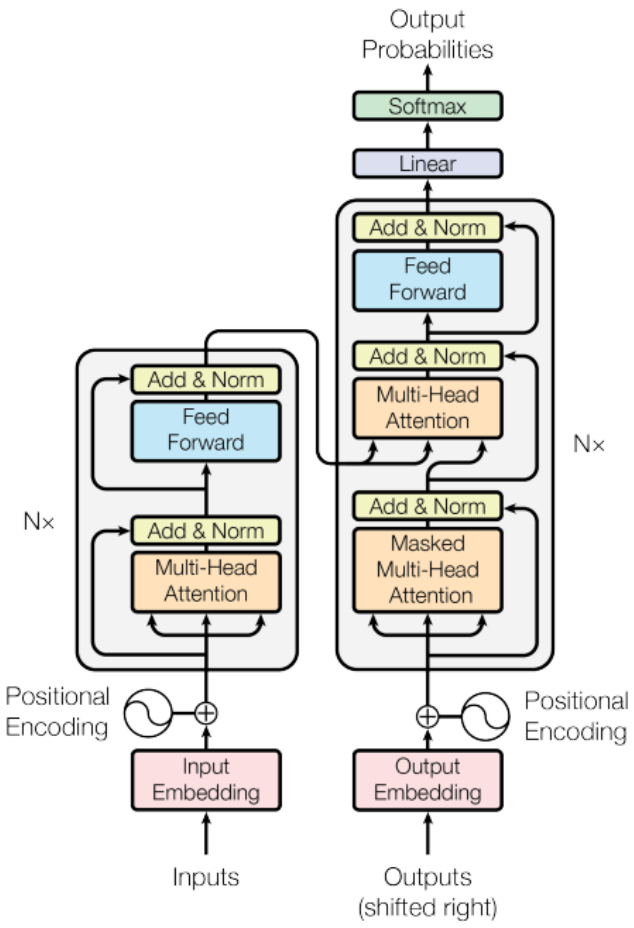

Figure 23: base BART model diagram [4].

6.5 Results

The BERT scores are produced by cosine similarities between contextual embeddings, and the numerical range of BERT Score is between -1 and 1. In practice, BERTscore is usually found to be in a small range. However, BERT Score would be easier to interpret and work with if it has a natural range. Then we would rescale BERT Score to have a range between 0 and 1. We can see that after cleaning, the precision, recall and F1-Score all increase as per the following Table 11.

		Precision	Recall	F1-Score
After cleaning	No rescale	0.939	0.935	0.937
	Rescaled	0.638	0.611	0.624
Before cleaning	No rescale	0.916	0.922	0.919
	Rescaled	0.501	0.537	0.518

Table 11: The BERT score of test data before cleaning and after cleaning.

Here we compared the results of four mini models. They are model trained with 100 lines of SAMSum data, model trained with 100 lines of SAMSum data with text noise added, model trained with 100 lines of SAMSum data created with SeaVoice and model trained with 98 lines of SAMSum data created with short summary setting the threshold at 0.5 as shown in Table 12.

	Number of empty result	Rescaled Precision with empty result	Rescaled Precision non-empty result
100 lines of SAMSum	22	0.387	0.625
100 lines of SAMSum with text noise added	50	0.089	0.632
100 lines of SAMSum created with SeaVoice	32	0.293	0.641
98 lines of SAMSum with short summary (threshold=0.5)	0	0.583	0.583

Table 12: The BERTprecision score comparison between four mini models.

Since these are mini models and only use 100 lines to train the BART model, there would be some empty results. The BERT score of the 100 lines of SAMSum with text noise added are low compared to others and it generates more empty results. However, we notice that with 98 lines of SAMSum with short utterance summary, this did not generate empty results.

At last, we compared three models. The first model is the original model which is fine-tuned on the whole SAMSum data corpus. The second is the model fine-tuned with short summary data extracted from SAMSum corpus, which only contains 699 training data as shown in Table 13.

	Precision	Recall	F1-Score
Original model SAMsum data (16k)	0.638	0.611	0.624
Model with short summary data (669)	0.579	0.656	0.617
Fine-tuning short summary data on original model	0.567	0.613	0.590

Table 13: The BERT score comparison between three models.

The last model is fine-tuning the short summary data extracted from SAMSum data on the original model. We observed that the recall of the model with short summary data is higher than the two models. However, the precision of the original model is still the highest.

6.6 Discussions

As per the previously mentioned minimodels, the SeaVoice generated training set gives the best non-empty precision followed by Textattack added noise. This reinforces our hypothesis that adding noise to the training set can boost model performance. Currently, this requires a manual transcription process and is therefore an impractical method of adding noise, since it requires excessive manual power generating augmented data for the whole SAMSum dataset. While adding noise with Textattack also possesses difficulty, namely the large proportion of empty results. Hence, we refute augmenting the dataset via this method.

The separate short summary model also does not meet our expectations. Although, compared to the original model, the short summary model performs better at eliminating empty predictions in the mini model phase, but as the number of training data increases to the whole SAMsum dataset, the original model produces close to zero empty predictions. Whereas the number of short sentences we can extract from SAMSum is limited to 669, making this a suitable threshold to filter the short sentences. Consequently, we face a significant challenge for evaluating this model's full potential In the absence of sufficient training data for the short summary model.

7. Long Summary

7.1 Problem Statement

The long summarization is at the document level. For example, the system generates one summary for an entire meeting transcript. The current model performs fairly well on the long summary task, since it is trained on long dialogues from the SAMSum dataset. Despite the relatively better performance compared to short summary, we assumed that improvement can be achieved by experimenting with state-of-the-art NLP models such as T5.

7.2 Current Implementation

Identical to the short summary model, the long summary model implements a pre-trained BART model that has been fine-tuned on the SAMSum corpus. For assessing summarizing performance, a BERT score has been introduced.

The SAMSum dataset was used to summarize the data. The data is a machine translated version of the SAMSum Corpus in the target language, such as Chinese, for multilingual summary. The SAMSum data corpus consists of 16369 conversations recorded and written down by linguists fluent in English, with manually annotated summaries in json format. It has been partitioned based on an 80/10/10 split for train, dev, and test, and can be accessed via the following database access points:

- SAMSum Corpus

7.3 Proposed Modifications

Due to the limited time schedule and failed attempts with data augmentation from a short summary, we decided to evaluate the possibility of replacing the current model with T5. In contrast to BERT-style model's restriction to output class label or span of input text, T5 reframes the NLP tasks to a text-to-text-format, specialized for tasks when both the input and output are text strings. We suspected this change of method could generate different results from the current model and potentially make improvement.

Particularly the model *T5-large-samsum-deepspeed* provided by Henryu-lin on Hugging Face. It is trained using Microsoft's AzureML and DeepSpeed's ZeRO optimization. ZeRO is a powerful memory optimization technique that enables effective training of large models with trillions of parameters. It was fine-tuned on the SAMSum corpus from t5-large checkpoint.

The result did not show a promising upgrade over the existing BART model. Although it demonstrated better performance from recall, the focus of our project is to enhance precision, as SeaSalt desires it to be more concise and accurate. Both the precision and F1 score are lower than the existing model, calculated with both BERT-score and Rouge score. Rouge-N measures the number of matching 'n-grams' between our model-generated text and a 'reference'.

For Rouge-1 we would be measuring the match-rate of unigrams between our model output and reference, and Rouge-2 would use bigrams. Rouge-l measures the longest common subsequence (LCS) between our model output and reference. It is a good metric for assessing both machine translation and automatic summarization tasks. After analyzing the generated sentence, we discovered that T5 generated are much longer compared to the more concise current model's result. The T5 model also produced repetitive sentences for certain parts of its outputs, which contributes to a significantly high recall serving counterintuitively to the summarization task.

7.4 Model Diagram

Tasks such as translation, question answering, and classification, feed the model text as input and train it to generate some target text. This allows for using the same model, loss function, hyperparameters, etc., across our diverse set of tasks.

The changes compared to BERT include adding a *causal* decoder to the bidirectional architecture and replacing the fill-in-the-blank cloze task with a mix of alternative pre-training tasks [6] as demonstrated in the following Figure 24.

Whereby a cloze task is an activity or evaluation consisting of a language passage with some words or signs deleted where the participant is required to replace the missing language item.

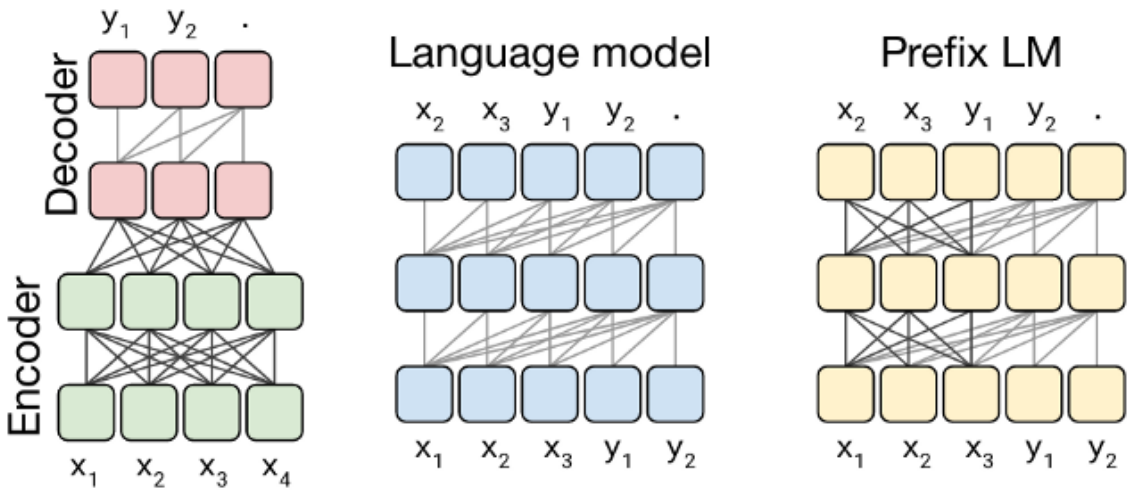

Figure 24: T5 model structure [6].

7.5 Results

Above is the BERT score of BART model and T5 model. These scores are rescaled, and we can observe that the precision of T5 is lower than the precision of BART, while the recall of the T5 model is higher than the BART's. This is because the T5 model would go into repetitive generations and render longer results than the result of theBART model as demonstrated in the following Table 14.

	Precision	Recall	F1-Score
BART	0.442	0.411	0.425
T5	0.248	0.454	0.348

Table 14: The BERT scores of BART model and T5 model.

Table above shows the precision, recall and F1 score of the Rouge score. Rouge-1 uses unigrams and Rouge-2 uses bigrams. Rouge-l measures the longest common subsequence between our model output and reference. We observed that the BART model has higher precision than the T5 model, while the T5 model has higher recall than the BART model. Consequently, the results are consistent with the BERT score as demonstrated in the following Table 15.

	Precision			Recall			F1-Score		
	P_1	P_2	P_1	R_1	R_2	R_1	F1_1	F1_2	F1_1
BART	0.446	0.293	0.428	0.426	0.284	0.411	0.407	0.263	0.391
T5	0.383	0.227	0.378	0.649	0.488	0.641	0.454	0.286	0.448

Table 15: The Rouge scores of BART model and T5 model.

7.6 Discussions

As per the previously mentioned experimental findings, we can conclude that the T5 model presents a higher recall score in comparison to the BART model which is representative of higher true positive values with respect to false negative values. On the contrary however, the T5 model presents a lower precision score in comparison to the BART model which is representative of higher true positive values with respect to false positive values. Henceforth, despite a rise in F1-Score driven by a higher Recall score, we are unable to implement this modification in preference to the previously implemented pre-trained BART model,

Theoretically speaking, we found that if certain utterances were sufficiently large, the model would not have visibility into the preceding conversation's utterances. As a T5 model is driven by contextual representations and learns from self-attention this running history of conversation becomes tremendously important. Ideally, we would like to test this new model on an entire meeting but it is more susceptible to handling monologues and less receptive to a dialogue.

For future considerations, we would like to highly recommend further research and development with mT5 which is a multilingual pre-trained T5 model presenting the ability to code switch between English and Chinese in the Seasalt use case. Furthermore, we highly recommend longT5 where we anticipate performance improvements of Transformer-based neural models from the previous findings due to increasing the input length or increasing the model size.

8. Data

The data used for summarization is the SAMSum dataset. For multilingual summarization, the data is a machine translated version of the SAMSum Corpus in the target language such as Chinese. The SAMsum data corpus is made of 16369 conversations created and written down by linguists fluent in English with manually annotated summaries in json format, and has been partitioned based on an 80/10/10 split for train, dev, test respectively. The SAMSum dataset was prepared by Samsung R&D Institute Poland and is distributed for research purposes. We also performed some data preprocessing on this SAMSum data, including data cleaning, sentence segmentation, data augmentation and manual annotations.

The data for action extraction is transcribed from SEASALT internal company meetings such as daily morning scrum sessions in addition to the ICSI meetings dataset consisting of over 70 hours of meeting recordings in mp3 format. This dataset supplements the BART model as well as the Rasa classifier, the transcription of which have been uploaded to the SeaWord repository. These datasets have been partitioned based on an 80/10/10 split for train, dev, test respectively.

We are using the IMDB dataset having 50K movie reviews for natural language processing or Text analytics. This is a dataset for binary sentiment classification containing substantially more data than previous benchmark datasets. We provide a set of 25,000 highly polar movie reviews for training and 25,000 for testing. So we can predict the number of positive and negative reviews using either classification or deep learning algorithms.

For emotion recognition, we incorporated the following datasets:

- CREMA-D
- RAVDESS

- SAVEE
- TESS

CREMA-D (Crowd-sourced Emotional Mutimodal Actors Dataset) is a data set of 7,442 original clips from 91 actors. These clips were from 48 male and 43 female actors between the ages of 20 and 74 coming from a variety of races and ethnicities (African America, Asian, Caucasian, Hispanic, and Unspecified). This dataset includes six different emotions (Anger, Disgust, Fear, Happy, Neutral and Sad) and four different emotion levels (Low, Medium, High and Unspecified). This CREMA-D is made available under the Open Database License.

The RAVDESS is a validated multimodal database of emotional speech and song. The database is gender balanced consisting of 24 professional actors, vocalizing lexically-matched statements in a neutral North American accent. Speech includes calm, happy, sad, angry, fearful, surprise, and disgust expressions, and song contains calm, happy, sad, angry, and fearful emotions. Each expression is produced at two levels of emotional intensity, with an additional neutral expression. This dataset is an open access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Surrey Audio-Visual Expressed Emotion (SAVEE) database has been recorded as a pre-requisite for the development of an automatic emotion recognition system. The database consists of recordings from 4 male actors in 7 different emotions, 480 British English utterances in total. The sentences were chosen from the standard TIMIT corpus and phonetically-balanced for each emotion. This dataset includes six basic emotions and neutral (Common, Anger, Disgust, Fear, Happiness, Sadness, Surprise, Neutral). The database is available free of charge for research purposes.

TESS (Toronto emotional speech set) is modeled on the Northwestern University Auditory Test No. 6. A set of 200 target words were spoken in the carrier phrase "Say the word _____' by two actresses (aged 26 and 64 years) and recordings were made of the set portraying each of seven emotions (anger, disgust, fear, happiness, pleasant surprise, sadness, and neutral). There are 2800 stimuli in total. This collection is published under Creative Commons license Attribution-NonCommercial-NoDerivatives 4.0 International.

For sentiment analysis and speech emotion recognition tasks, we scraped publicly available conversations to build our own test data. We collected Seasalt company internal recordings in addition to Youtube videos of irritated clients interacting with customer service representatives.

We segmented the videos using the SeaMeet platform which also parsed the corresponding transcriptions for these videos. Subsequently, we manually annotated the sentiment from these transcripts as well as the speech emotion from the respective audio recordings.

9. Future Work

Primarily, for action extraction and sentiment analysis the model works with a high precision as previously anticipated. We would highly recommend focusing on the key aspects of the data which help the model learn as per corporate requirements. For example, modifications to the internal transcription engine would significantly facilitate in reducing noise and repetitions throughout the training set, cleaner inputs would yield more tangible classification results.

Furthermore, fine tuning on colloquial dialogues in comparison to encyclopedias such as Wikipedia would help the model anticipate certain conversational lingo which is absent from formal literature. Additionally, datasets catering to information technology would help reduce the out of vocabulary terminology the model might encounter as software development engineers are discussing daily activities and referencing weekly or monthly actionable deliverables.

In the likely scenario of a corporate decision to develop a summarization tool that aims to preserve as much key information as possible, we strongly recommend continuing further investigations with unique variations of the T5 model such as multilingual T5 or longT5. These variations of pre-trained models can help cater to a multidisciplinary audience across geographical boundaries.

For future reference, and motivated by corporate strategy for image classifications we would highly recommend for the technical department to take a deeper dive into a new cutting edge methodology called data2vec which extrapolates upon the concepts of wave2vec 2.0 by being applicable to speech recognition, image classification, and natural language understanding [17].

The main concept is to forecast latent representations of the whole input data using a typical Transformer architecture and a masked view of the input. Data2vec predicts contextualized latent representations, which include information from the whole input, as opposed to modality specific targets, such as words, visual tokens, or units of human speech, which are local in nature.

Studies on the key benchmarks of voice recognition, picture classification, and natural language comprehension show a new state of the art or competitive performance to traditional methods which would further supplement corporate aspirations in migrating to visual representations.

10. Conclusion

In conclusion, various modules of the capstone project such as action extraction, summarization, sentiment analysis and speech emotion recognition implemented unique and individual pre-trained fine-tuned models such as ULMFiT, wav2vec 2.0, BART and T5 respectively.

For action extraction, the final model was trained to exclusively capture a meeting's tangible deliverables such as a person, place or thing accomplishing a goal within a timeline such as today, tomorrow, or by the end of the week. With a significant emphasis on obtaining "high-precision" predictions of the "actionable" statements from a meeting transcript.

After modifying the implementation into a binary classification problem, the final action extraction model created with ULMFiT was implemented on Seasalt's SeaMeet interface development server, which allows users to record meetings and utilize the action extraction model to highlight the most important deliverables from the meeting’s transcription.

Furthermore, for both the short and long summaries we attempted to improve the model. Although our efforts were futile due to the repetitiveness and redundancy of certain generative models, this exercise served as a fundamental learning curve through an error analysis process.

We experimented with data cleaning and augmentation to align the training and testing dataset more closely with corporate requirements. Additionally, we evaluated various models such as distilled BART and variations of T5 such as mT5 and longT5. Therefore, despite our attempt to update the current model, we believe there is valuable feedback from our experiments.

First, we provided sufficient evidence that the current model is well optimized both in terms of performance and reliability. We also found that a BART or T5 model is likely not the best approach to the short summarization task, since the utterances are already very concise and it seems as though the intention from Seasalt was more in the direction of sentence rewriting.

Second, although we refute further operations with data augmentation due to practical reasons, we offer the previous improvement methods that can be used later given enough training time and resources. Finally, T5 model modifications which specialize in high recall performance.

Throughout our proof of concept as well as production deployment, we observed that the binary classifications for sentiment analysis demonstrated a higher accuracy than the multi-class classifications including a “neutral” class. This can be significantly attributed to the ambiguity of applying a threshold such as the upper and lower bounds of neutral classification from zero.

The fine tuning of placing upper and lower bounds for a neutral threshold requires an empirical analysis and is highly recommended for further investigation before production deployment. In the meantime, the previously mentioned classification report demonstrates a high precision of true positive therefore concurrently reinforcing the F1-Score as shown in Table 8.

Despite the challenges faced in analyzing audio files, this wav2vec 2.0 implementation serves as a foundation for the friendly user interface illustrated in Figure 7 which offers a holistic distribution of a client’s emotions to the corresponding customer service representative which is a significant corporate undertaking in establishing the SeaMeet as well as SeaSuite interface.

Appendix

i. Models

Github repository - Master branch

Action Extraction:
- GitHub repo
- API Documentation
- FastAPI endpoint

Sentiment Analysis:
- GitHub repo
- Model UI

Speech Emotion Recognition:
- GitHub repo
- Model UI

Proof of concepts:
- Short summarization
- Long summarization
- Emotion recognition
- Emotion detection
- Sentiment analysis
- Youtube captions
- Youtube audio
- Annotations
- Processing transcripts
- Baseline model and ULMFiT POC branch
- HuggingFace transformers POC for action extraction

iii. References

[1] H. Singh, “Big Data and Natural Language Processing came together for better information extraction: Text Analytics,” International Journal of scientific research and management, Nov. 2016, doi: 10.18535/ijsrm/v4i11.15.

[2] S. Anand et al., “Suggestion Mining from Online Reviews using ULMFiT,” Apr. 2019. [Online]. Available: https://arxiv.org/pdf/1904.09076.pdf

[3] I. Yamada et al., “LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention,” Oct. 2020. Accessed: Jun. 20, 2022. [Online]. Available: https://arxiv.org/pdf/2010.01057.pdf

[4] M. Lewis et al., “BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension,” Oct. 2019. [Online]. Available: https://arxiv.org/pdf/1910.13461.pdf

[5] J. Devlin, M.-W. Chang, K. Lee, K. Google, and A. Language, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” May 2019. [Online]. Available: https://arxiv.org/pdf/1810.04805.pdf

[6] C. Raffel et al., “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer,” Jul. 2020. [Online]. Available: https://arxiv.org/pdf/1910.10683.pdf

[7] E. Moatez, B. Nagoudi, A. Elmadany, and M. Abdul-Mageed, “AraT5: Text-to-Text Transformers for Arabic Language Generation,” Mar. 2022. Accessed: Jun. 20, 2022. [Online]. Available: https://arxiv.org/pdf/2109.12068.pdf

[8] M. Abdul-Mageed, A. Elmadany, E. Moatez, and B. Nagoudi, “ARBERT & MARBERT: Deep Bidirectional Transformers for Arabic,” Jun. 2021. Accessed: Jun. 20, 2022. [Online]. Available: https://aclanthology.org/2021.acl-long.551.pdf

[9] D. Pan, L. Li, J. Yuan, and D. Sheng, “Deep neural network-based classification model for Sentiment Analysis,” Jul. 2019. Accessed: Jun. 20, 2022. [Online]. Available: https://arxiv.org/pdf/1907.02046.pdf

[10] B. AlBadani, R. Shi, and J. Dong, “A Novel Machine Learning Approach for Sentiment Analysis on Twitter Incorporating the Universal Language Model Fine-Tuning and SVM,” Applied System Innovation, vol. 5, no. 1, p. 13, Jan. 2022, doi: 10.3390/asi5010013.

[11] S. Schneider, A. Baevski, R. Collobert, and M. Auli, “wav2vec: Unsupervised Pre-training for Speech Recognition,” Sep. 2019. [Online]. Available: https://arxiv.org/pdf/1904.05862.pdf

[12] D. Amodei et al., “Deep Speech 2: End-to-End Speech Recognition in English and Mandarin Baidu Research -Silicon Valley AI Lab *,” Dec. 2015. Accessed: Jun. 19, 2022. [Online]. Available: https://arxiv.org/pdf/1512.02595.pdf

[13] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, “wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations,” Nov. 2020. [Online]. Available: https://arxiv.org/pdf/2006.11477.pdf

[14] B. Gliwa, I. Mochol, M. Biesek, and A. Wawer, “SAMSum Corpus: A Human-annotated Dialogue Dataset for Abstractive Summarization,” Proceedings of the 2nd Workshop on New Frontiers in Summarization, Nov. 2019, doi: 10.18653/v1/d19-5409.

[15] J. Howard and S. Ruder, “Universal Language Model Fine-tuning for Text Classification,” May 2018. Accessed: Jun. 22, 2022. [Online]. Available: https://arxiv.org/pdf/1801.06146.pdf

[16] C. Wang, “Speech Emotion Recognition Based on Multi-feature and Multi-lingual Fusion,” arXiv:2001.05908 [cs, eess], Jan. 2020, Accessed: Jun. 25, 2022. [Online]. Available: https://arxiv.org/abs/2001.05908

[17] A. Baevski, W.-N. Hsu, Q. Xu, A. Babu, J. Gu, and M. Auli, “data2vec: A General Framework for Self-supervised Learning in Speech, Vision and Language,” *arXiv:2202.03555 [cs]*, Feb. 2022, Accessed: Jun. 25, 2022. [Online]. Available: https://arxiv.org/abs/2202.03555