University of the Philippines Manila
College of Arts and Sciences
Department of Physical Sciences and Mathematics

PASABI: Pagmensahe ng Salitang Binigkas
A Filipino Speech-to-Text Messaging Application Using Recurrent Neural Networks

A special problem in partial fulfillment
of the requirements for the degree of
Bachelor of Science in Computer Science
Submitted by:

Damian Custer M. Fadri

June 2017

Permission is given for the following people to have access to this SP:

Available to the general public	Yes
Available only after consultation with author/SP adviser	No
Available only to those bound by confidentiality agreement	No

ACCEPTANCE SHEET

The Special Problem entitled "PASABI: Pagmensahe ng Salitang Binigkas A Filipino Speech-to-Text Messaging Application Using Recurrent Neural Networks" prepared and submitted by Damian Custer M. Fadri in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

Marvin John C. Ignacio, M.Sc. (*cand.*)
Adviser

EXAMINERS:

	Approved	**Disapproved**
1. Gregorio B. Baes, Ph.D. (*cand.*)	_____	_____
2. Avegail D. Carpio, M.Sc.	_____	_____
3. Richard Bryann L. Chua, Ph.D. (*cand.*)	_____	_____
4. Perlita E. Gasmen, M.Sc. (*cand.*)	_____	_____
5. Ma. Sheila A. Magboo, M.Sc.	_____	_____
6. Vincent Peter C. Magboo, M.D., M.Sc.	_____	_____

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

Ma. Sheila A. Magboo, M.Sc.
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Sciences and Mathematics

Marcelina B. Lirazan, Ph.D.
Chair
Department of Physical Sciences and Mathematics

Leonardo R. Estacio Jr., Ph.D.
Dean
College of Arts and Sciences

Abstract

PASABI is a Filipino text messaging mobile application with a speech-to-text functionality. The speech-to-text functionality makes use of Keras models produced with the separate PASABI desktop trainer. The trainer makes use of Recurrent Neural Networks for this task. Connectionist Temporal Classification is also utilized by creating a speech-to-text model that is trained by mapping characters in the transcription to the audio. By training the model directly to the characters, the need for speech datasets with phonetic transcriptions, or the development of algorithms to generate these phonetic transcriptions, is removed. The provided trainer can be used to develop models with new data, and be able to deploy it to the mobile application.
Keywords: speech-to-text, text messaging, speech recognition, artificial intelligence, neural networks

Contents

List of Figures

List of Tables

I. Introduction

A. Background of the Study

Speech recognition is the task of extracting a sequence of words from a speech utterance. While recognizing spoken words may seem intuitive for humans, the same does not necessarily go for machines[5].

Speech-based applications allow users to communicate with computers without the use of traditional input devices[6]. For mobile applications, there are several that takes a voice prompt as an input. Apple's Siri, a digital assistant that responds to a user's needs; EverNote for voice memo; and WhatsApp for instant messaging. In a sense, this promotes a more natural way for users to interact with their devices. Applications that support speech input also provide an alternative for input methods which require motor skills to work. In texting, the use of on-screen keyboards require pressing the right keys with one's fingers. Users may need several attempts to key in the right letter, or that function keys may be accidentally pressed[7]. For the elderly which are not as nimble-fingered as young people, the use of speech input can ease the difficulty. Persons with hand impairments can also benefit from such functionality.

In the local setting, several studies have been done for the development of a speech recognition system. A study by Ang et al.[8] presented a captioning system for TV News programs. The idea is to be able to display subtitles in real-time as the news gets delivered live. This provides a means for people with hearing disabilities to be updated on the currrent events in the country. A study by Bayona et al.[9] presents a speech recognition system on data which makes use of English and FIlipino in a single conversational context. In the study, they have presented their use of a new Filipino speech corpora. Also, a voice-input application for text-messaging called FiliText[10] was developed. FiliText makes use of a speech corpora called Filipino Speech Corpus (FSC), developed by the Digital Signal Processing Lab in University

of the Philippines Diliman[11]. The application provides a way for Filipino drivers to compose text messages while driving by just speaking to the phone. Also, people with physical impairments would also benefit from such an application. Given that they will have a hard time using an on-screen keyboard to write their messages, they can instead use their voice as an alternative.

These systems made use of CMU Sphinx[12], a speech recognition toolkit, which follows a standard pipeline. The pipeline involves three things: an acoustic model, a phonetic dictionary, and a language model. A speech utterance is first subdivided into speech frames. The primary assumption is that each frame holds a unique speech building block. Phonemes, defined to be the smallest unit of speech[1], are used in most cases. An acoustic model provides a means to map these speech frames into its correponding phoneme. Common implementations make use of probability distributions over all possible phonemes, and retrieving the most likely phoneme. This will give a sequence of phonemes predicted for each speech frame. The phonemes are then converted into words through a phonetic dictionary. The phonetic dictionary is a list of all words that the system can recognize. It provides a mapping for each word and the phonetic sequence corresponding to it. As such, search algorithms are implemented to be able to retrieve a word given the predicted phonemes. Finally, a language model provides predictions on what the next word could be, given the previous words.

The main idea is that the toolkit makes use of phonemes. While it provides a robust procedure on the speech recognition task, this poses the need for phonetic transcriptions. In the training of an acoustic model, phonemes are aligned to the appropriate speech frames. The phonetic dictionary also requires that each word would have a corresponding phonetic transcription. As such, development of speech recognition systems require that the speech corpus used provides the phonetic transcription of the speech utterances. However, such speech corpora are not easily created and

behind a pay wall. Speech corpora such as FSC provide only a text transcription for each utterance. In a sense, to make use of the CMU Sphinx library requires a developer to extract phonemes from a given speech corpora.

Alternatives include manually transcribing the phonemes or the development of algorithms for the extraction. The proponents of FiliText made use of the spelling of a word as a substitute for its phonemes. For instance, the word "saging" would have its phonemes as "/s/ /a/ /g/ /i/ /ng/". However, such an approach doesn't hold for some common words. "Mga", pronounced as "/m/ /a/ /ng/ /a/", would have a phonetic transcription of "/m/ /g/ /a/." Such algorithms can become more complex as more rules and linguistic asusmptions are added[13]. Also, in the case when a new speech dataset becomes available, a developer would need to develop a new phoneme extraction algorithm tailor-fit for the new dataset.

As a general field, Artificial Intelligence involves the development of intelligent computers. Intelligent, in the sense that it can take into account feeded information; and react based on that information. Human tasks are emulated by computers through implemention of rules and algorithms.

As an subset of this, Machine Learning aims to develop such behavior by "learning" it from observed data. This can be done with something as simple as a linear regression equation, to SVMS and Neural Networks. In particular, Neural Networks is an interesting design since its design is inspired by how the human brain works. Given simple units that are interconnected, one can produce complex and precise models in the real world.

A particular kind of neural network, called Recurrent Neural Network (RNN), has already been applied in several speech recognition tasks. The motivation for using RNNs is its capability to model data that are sequential or time-dependent. This is done by taking into consideration information from previous time frames. For data such as speech, sound from a point in time can be determined based on what

came before and after it. A variation called Bidirectional RNNs (BRNNs) are able to take into consideration future time frames as well[2]. A proposed methodology called Connectionist Temporal Classifcation[14], incorporated with BRNNs, removes the need to align speech utterances to its phonetic transcription. Rather, the method attempts to map the speech directly to its text transcription (i.e. characters as its sub-word components).

B. Statement of the Problem

For speech datasets which only provide text transcriptions, there is a need to extract phonemes from the text transcription. This leads to the following problems:

1. Manually transcribing phonemes from the text transcription is tedious and prone to human errors.

2. Development of algorithms to extract the phonemes requires the definition of rules and exceptions, which can be complex and specialized.

C. Objectives of the Study

This study aims to develop a mobile application that is capable of transcribing speech input into text. Its underlying structure will be utilizing a Speech-To-Text (STT) model that is trained without the use of phonetic transcriptions. Also, a desktop application capable of producing such models will also be developed.

Specifically, the system will have the following functionalities:

1. Allows the smartphone user to

   (a) Provide a voice input using the mobile's microphone

   (b) Obtain the corresponding text message based on the voice input

   (c) Edit the text message through an on-screen keyboard

(d) Select the contact where the text message will be sent to

(e) Send the text message

(f) View all the available STT models

(g) Select the STT model to be used by the application

2. Allows the AI Expert to

(a) Prepare the dataset

i. Select the dataset directory containing audio files and transcription file

ii. Build the JSON file containing references on all audio files in the dataset directory

iii. Split the JSON file into three sub-datasets with the inputted ratios

A. Training dataset JSON

B. Validation dataset JSON

C. Testing dataset JSON

iv. Display the resulting number of entries for each sub-dataset

(b) Create a STT model

i. Select the training and the validation dataset to be used for training

ii. Select the directory where the resulting model and its other configuration files will be saved

iii. Specify values for the hyperparameters

A. Training epochs

B. Learning rate

C. Minibatch size

iv. Start the training process of the STT model

v. Stop the training process of the STT model

vi. Continue a previously stopped training process.

vii. Plot the training and validation error against the number of iterations

(c) Evaluate a STT model

i. Select the directory of the model to be evaluated

ii. Select the training and testing dataset to be used for testing

iii. Start the testing process of the STT model

A. Display the STT model's predictions along with the true transcriptions

B. Display the character error rate (CER) of the model

iv. Stop the testing process of the STT model

(d) Save the logs of the trainer application

D. Significance of the Project

By training a model to map a speech utterance directly to its text transcription, there will be no need to acquire datasets with phonetic transcriptions. In the case when only a text transcription is available, there will be no need to manually transcribe the phonemes or develop an algorithm to derive the phonetic transcription from the text.

Development of a text messaging application would not be constrained on the use of phonemes. Given a new speech corpus, there will be no need to go over the traditional speech recognition pipeline again. This avoids redefining the acoustic model and the construction of a phonetic dictionary specific to the new speech corpus.

In addition, as an extension of FiliText, the application can be beneficial for drivers or people with hand impairments, given that the application is a way for hands-free text messaging.

E. Scope and Limitations

1. The mobile application will only recognize Filipino words.
2. Only the text message composition part will have the speech input functionality (i.e. no speech input for inputting contacts or other commands).
3. The development of models will not be a functionality of the mobile application, but on a separate desktop application.
4. The application will only be for offline use.
5. The transcription accuracy of the application will depend on the STT model it is currently using.
6. The transcription speed of the application will depend on the smartphone's processing power.

F. Assumptions

1. The audio files of the speech corpus is in .wav format
2. The user's smartphone that will be used runs on the Android OS.
3. The user's smartphone will have all the necessary dependencies installed.
4. THe user's smartphone will be running all necessary dependencies upon use.
5. There will be little to no noise when the mobile application is to be used.

II. Review of Related Literature

The use of RNNs has been prevalent in the field of speech recognition. Since speech can be represented as a data sequence (in this case, a sequence in time), models that are able to capture the temporal information of data is necessary. Several studies have shown the use of RNNs for speech recognition.

Graves et al.[14] introduced the concept of Connectionist Temporal Classification (CTC), wherein a network can be trained without the need for explicit alignment between the input and output data. In the case of speech, speech frames that form a speech utterance and its corresponding labels per frame. This is particularly useful since it is difficult to acquire pre-aligned data.

Different kinds of approaches exist for this task. A particular approach is the phoneme-level approach. The speech input is first translated to its phoneme transcription, and gets further translated to a word by looking it up on a pronunciation dictionary.

Chorowski et al.[15] presents the use of an RNN Encoder-Decoder model incorporated with an attention mechanism. Phonemes were used as sub-word components to be outputted by the decoder. The attention mechanism helps in constructing the context vector by deciding where to look at the input sequence to predict the phoneme at a particular time step. A window is used for such a task; a window is a subsequence of input time steps that maps to elements of the phoneme sequence. The idea behind is that a phoneme corresponds to a particular input frame and its adjacent frames. They have implemented the encoder with a Deep Maxout network[5] to transform the speech features. The encoder was connected to the decoder, implemented with a gated BRNN to account for long-term dependencies.

Graves et al.[16] demonstrates the use of BRNNs with Long Short-Term Memory (LSTM) cells for speech recognition. The use of LSTM provides a way so the back-propagated error do not blow up or exponentially decay over time[3]. The system was

developed by utilizing two neural networks. The first one is to model the acoustic data to its phonetic transcription with the use of BRNN with LSTM cells, incorporated with the concept of CTC. The second neural network is used to predict a phoneme given previous phonemes, acting as a language model for the system. Given the probability distributions from both networks, a beam search is used to find an effective transcription for the speech input. The proposed architecture has yielded 18.4% - 23.9% phoneme error rate (PER) with varying parameters.

Fernandez et al.[17] also used a BRNN-LSTM architecture with prefix search decoding. This consists of calculating the probabilities of successive extensions of possible transcriptions, and is used to find the most likely one.

Another approach is the character-level approach, wherein the speech input gets translated to its word transcription directly. This is done by training the network to directly map the speech input to its corresponding characters.

Graves et al.[18] demonstrates the use of CTC in a speech recognition system that models speech input directly to its text transcription. They used a network with five Bidirectional LSTM hidden layers, along with a CTC loss function. The set of labels used are the standard English alphabet, with punctuation marks, spaces, and blank characters. In addition, the outputs of the network are integrated with the output of a word language model. The language model predicts what word could follow preceding words. The combined model rates with a 26% word error rate.

Hannun et al.[4] has developed a speech recognition system that is also capable of mapping speech input to its text transcription. This eliminates the need for a dictionary to translate sub-word components to words since the system is essentially taught how to spell. Their architecture makes use of an BRNN with a clipped rectified linear function. The network is composed of 3 fully-connected layers with the 4th layer having bidirectional recurrent connections. A n-gram language model is also used to supplement the results, checking which word is most likely given the output sequence

produced by the network. Further variations include introducing a noisy channel to their clean speech data so the system can be robust to noisy inputs. Rather than removing the noise from a speech utterance, they trained the network to be able to predict from noisy speech. The system rated a 16% word error rate.

Maas et al.[13] also implemented a speech recognition system that maps acoustic input directly to its text transcription. The network utilized the CTC loss function, with three fully connected layers and the fourth layer having bidirectional recurrent connections. For their model, a character-level language model was integrated with the network. The character language model predicts what the next letter could be given previous letters. Their system evaluated a word error rate of 21.4%.

Ko et al.[19] presents a data augmentation technique for speech data. The idea is to directly process the audio by changing its speed by factors of 0.9, 1.0, and 1.1. This is helpful for languages with a limited dataset, including Filipino.

III. Theoretical Framework

A. Automatic Speech Recognition

Automatic speech recognition (ASR) systems convert speech from a recorded audio signal to text. Humans convert words to speech with their speech production mechanism. An ASR system aims to infer those original words given the observable signal[1]

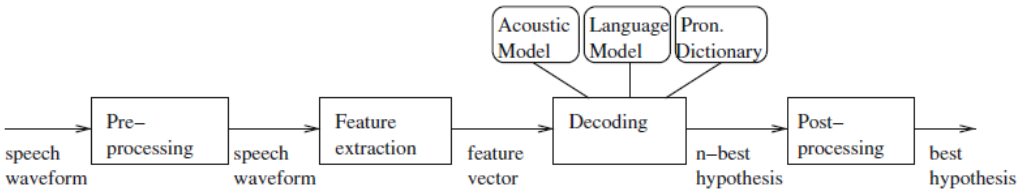


Figure 1: Schematic outline of a typical speech recognition system[1]

The process of a typical speech recognition system can be divided into the following consecutive steps[12]:

1. **Pre-processing** This is the initial stage wherein speech signals are worked on to prepare the dataset on being fed to the training algorithm.

2. **Feature extraction** The is the stage in which the relevant information about the speech signal is extracted.

3. **Decoding** This is the stage that does the actual recognition, employing an acoustic and language model as well as a dictionary.

   (a) **Acoustic model** This is the model that maps the sound data to a set of sub-word representations such as phonemes or characters.

   (b) **Language model** This is the model that deals with the probability distribution over sequence of words.

(c) **Pronunciation Dictionary** This is a list of words with their corresponding sub-word transcriptions.

B. Recurrent Neural Networks

Recurrent neural networks (RNNs) are a class of artificial neural network architecture that makes use of loops for information persistence. The general idea is that it is a series of standard neural networks connected through their hidden layers. As such, each step in the sequence (or time, in the case of speech) will correspond to a single neural network. For RNNs, there will be an input for each time step with its corresponding output.

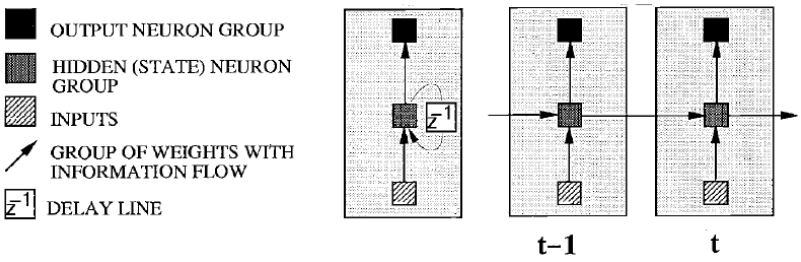


Figure 2: General structure of an RNN, unfolded in two time steps[2]

1. **Bidirectional recurrent neural networks (BRNNs)** are another class of RNNs that extend the steps in time covered by RNNs. With bidirectional, this involves training using all available input information in the past and future of a specific time frame[2]. In the case of a speech input, a prediction text must take into consideration the entire acoustic sequence.

2. **Long Short-Term Memory (LSTM)** networks are a class of RNNs that use memory blocks instead ofregular nodes. These memory blocks consist of gates and a memory cell designed to be able to control the flow of propagated information. It is designed to overcome the error back-flow problems that RNNs face, also known as the vanishing gradient problem[3].

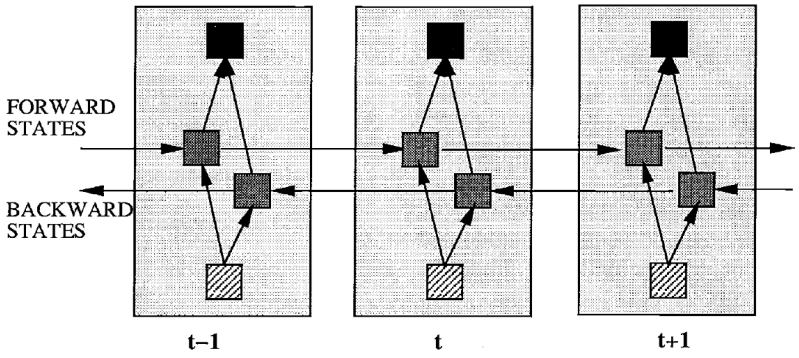


Figure 3: Structure of a Bidirectional Recurrent Neural Network[2]

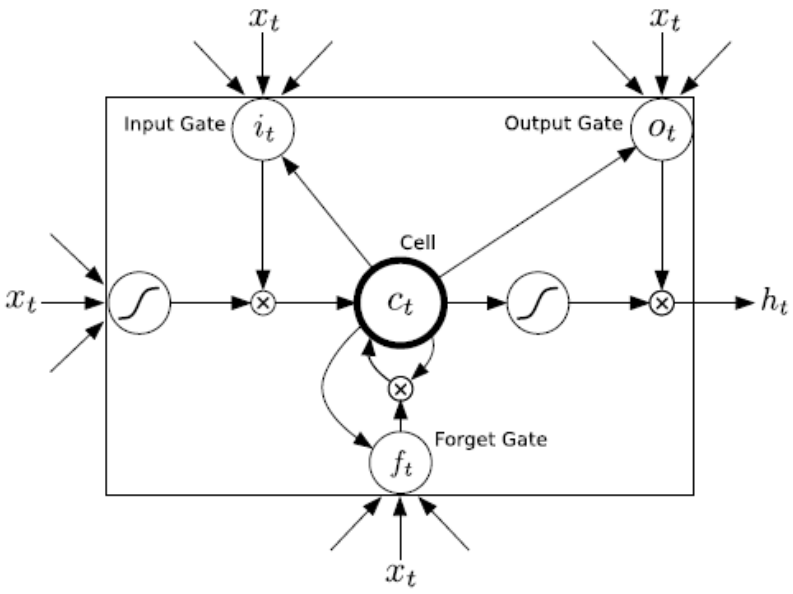


Figure 4: Structure of an LSTM Cell[3]

C. Connectionist Temporal Classification

Connectionist Temporal Classification (CTC) is a procedure useful for performing supervised learning on sequence data[14]. The general process involves making the RNN output a probability distribution out of all possible transcriptions. By getting the most likely path from all possible paths, the most likely output sequence can be derived from it.

The procedure was proposed in accordance with the need to map input sequences to its corresponding output sequence without specifying its alignment per sequence

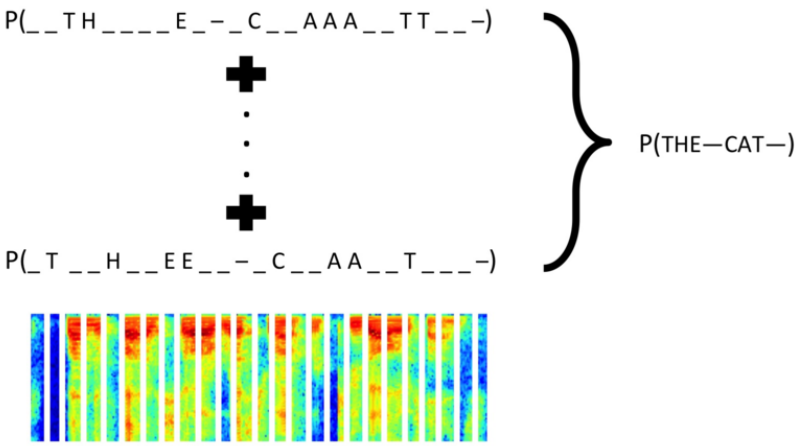


Figure 5: Workflow of CTC[4]

step. This is a common problem for real-world sequence learning as the act of aligning an input to an output is considered to be tedious and error-prone[14].

D. Filipino Speech Corpus

The speech corpus presented in a study on Filipino ASR[9] entitled Interdisciplinary Signal Processing for Pinoys - Project 06 - Tagalog Corpus (ISIP06 - TGL) was provided by the Digital Signal Processing Laboratory, Electrical and Electronics Engineering Institute, College of Engineering, University of the Philippines Diliman. The corpus contains speech utterances from multiple speakers along with its corresponding text transcriptions. The speakers are from different demographic locations and of varying age.

The dataset consists of directories for each speaker. Inside a speaker directory are the speech recordings for that particular speaker, along with a text file that contains all the transcriptions of every recordings in that directory. The dataset is primarily in Tagalog, but also contains English utterances and other local dialects as well. All audio files are in .wav format.

The corpus contains various types of speech utterances. This includes:

1. utterances of words and phrases, such as names of persons and places, nouns, and enumeration of digits, and

2. utterances of sentences, such as reading from a provided script to spontaneous Q&A exchange

E. Baidu DeepSpeech

Baidu provides a base implementation of their character-based speech transcription approach in their Github page[20]. The available source code consists of the scripts required to train and test a model. While the code is built as a sample for the LibriSpeech dataset, it can be modified for other speech datasets as well.

F. GNURoot Debian

GNURoot Debian is an application that emulates a Linux terminal in Android. It serves as a mini-system inside the phone, wherein a user can install libraries and packages or run scripts in. The mobile application is available in the Google Play Store and does not require any root access to be installed.

IV. Design and Implementation

A. System Overview

The PASABI system comprises of two applications: a mobile application and a desktop application. The mobile application is used for the actual text messaging services. The desktop application is for the development of models to be used by the mobile application.

1. **PASABI Mobile Application**

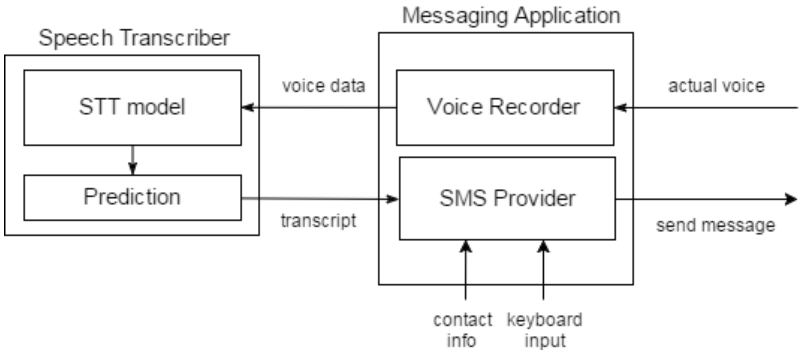


Figure 6: System Overview for the PASABI Mobile Application

2. **PASABI Desktop Application**

B. Use Case Diagram

PASABI considers two types of users, namely the smartphone user and the AI expert. The smartphone users are the primary users of the application; they are able to compose a text message through an on-screen keyboard, with an additional feature of composing a message through speech input. The AI Expert is tasked in facilitating the development of the models that will be used on the speech-to-text functionality of the mobile application.

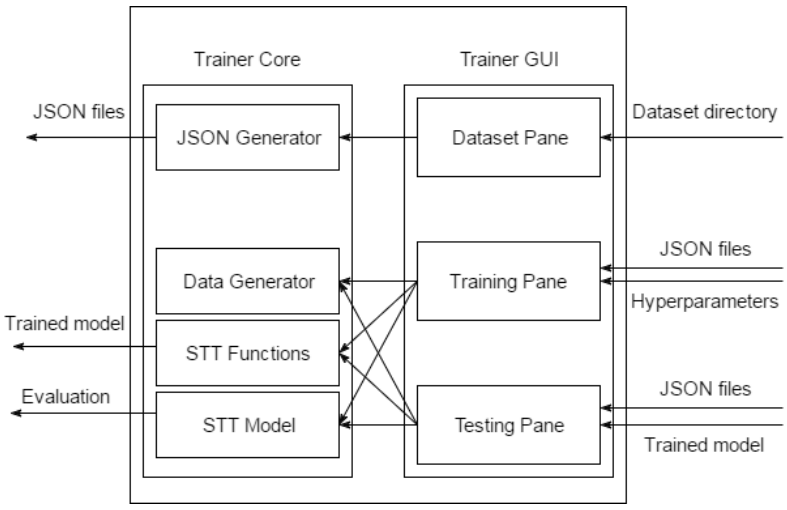


Figure 7: System Overview for the PASABI Desktop Application

1. **Smartphone User**

   The smartphone user is able to provide a voice input to the mobile application. The voice input received by the mobile application is the speech data that will be converted to text. In turn, the smartphone user is able to get the text transcription given the users voice input.

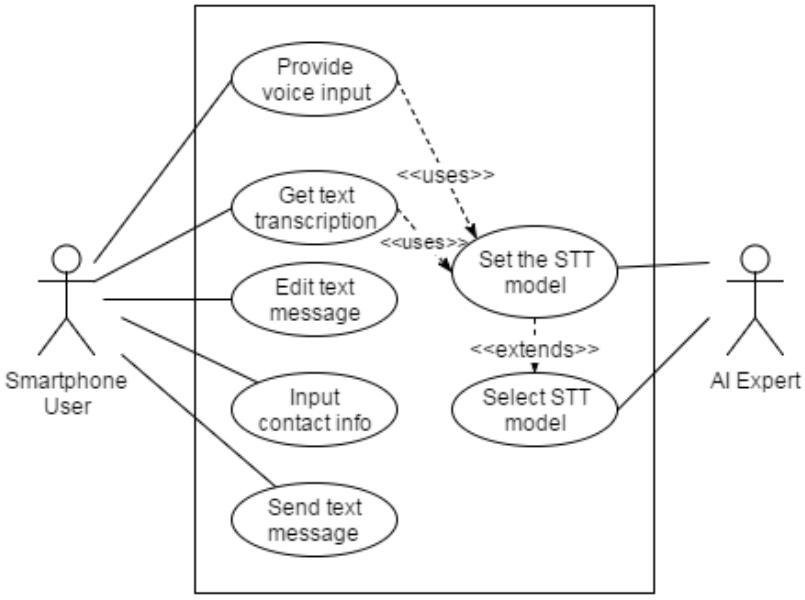


Figure 8: Use case diagram of the mobile application

The smartphone user is free to edit the text message through the use of the phones on-screen keyboard. This option is provided so as not to restrict the smartphone user into using the speech-to-text functionality of the mobile application. The smartphone user is also able to select which contact he/she wants to send the text message to. Finally, the smartphone user is able to send the message to the specified contact.

2. **AI Expert**

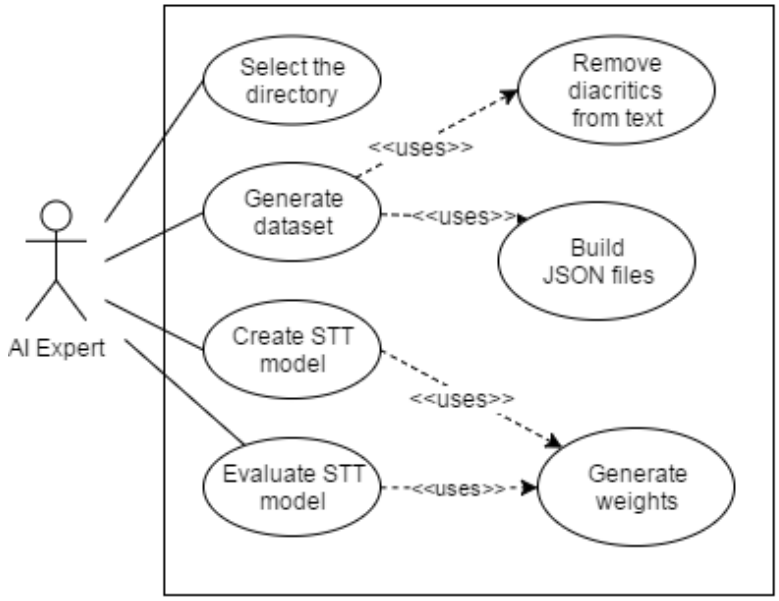


Figure 9: Use case diagram of the desktop application

The AI Expert is able to prepare the dataset to be used for training. This includes the selection of a training set and validation set from the whole dataset. The AI expert is also able to build the dataset into a format accepted by the neural network.

In terms of training a model, the AI expert is able to select which built dataset will be used. In addition, there are also options on what network architecture will be used along with the training hyperparameters. Hyperparameters include

the number of epochs, learning rate, and minibatch size. The AI expert is also able to see the training error and validation error for each epoch, to monitor the progress of the training. For each epoch, a model will be saved to benchmark the current progress of the training.

In terms of testing a trained model, the AI expert is able to select which trained model will be tested. The AI expert is able to perform predictions using the trained model. Given the predictions, the error rate of the model is calculated. A log file listing the results of the testing is also generated. Finally, the AI expert is able to update the underlying STT model being used by the mobile application.

C. Flowcharts

1. **Sending a Text Message**

   Figure 10 shows the flowchart for the smartphone user on sending a text message. The user is given the option to input the message either from the on-screen keyboard or from the speech-to-text functionality of the application. In either case, the text message draft is displayed on the text field and is available for further editing through the phones keyboard. Once finalized, the message can be sent to the selected contact.

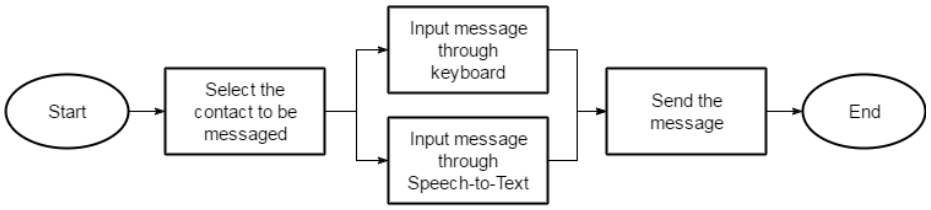


Figure 10: Sending a Text Message Flowchart

2. **Inputting a Message through STT**

   Figure 11 shows the flowchart on the subprocess Inputting a Message through

the STT model from Figure 10. The smartphone user needs to press the recording button to start the speech input and press the button again to stop the recording. After recording, the voice data is fed to the model to obtain the predicted transcription. The transcription is then sent back to the application as a text message draft.

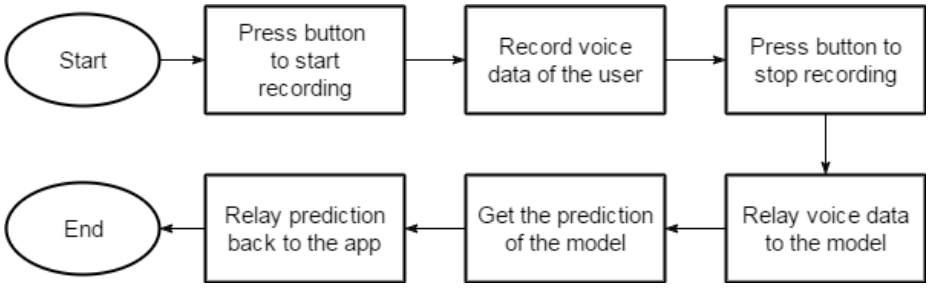


Figure 11: Inputting a Message through STT Flowchart

3. **Selecting the STT model to be used**

Figure 12 shows the flowchart on selecting the STT model to be used by the application. The list of available models is displayed, each directory containing all the necessary files to initialize the model. The user is able to select which model is to be used. After selecting a model, the STT model of the application is re-initialized into the selected model.

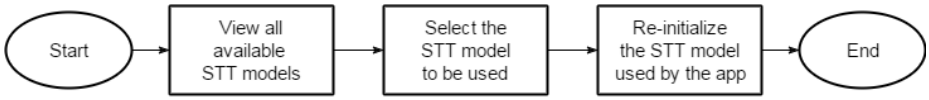


Figure 12: Selecting the STT Model to be Used Flowchart

4. **Re-initializing the STT model**

Figure 13 shows the flowchart of the subprocess Re-initialize the STT model in Figure 12. The directory of the STT model selected by the user is retrieved; the files inside are used to initialize the model. The model is reconstructed based

on the configuration file, and the weights are loaded onto the model with the weights file.

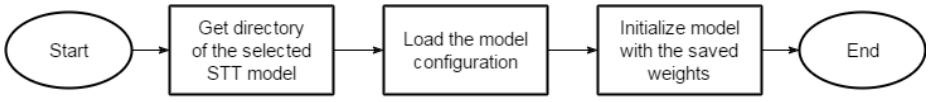


Figure 13: Re-initializing the STT model used by the Application Flowchart

5. **Preparing a dataset**

Figure 14 shows the flowchart on preparing a dataset. An AI expert is able to select the directory containing the speech files and the transcription. The AI expert is able to specify the ratios for the partitioning. Three ratios are to be provided: namely for the training dataset, for the validation dataset, and the testing dataset. The AI expert is to provide a regex string that is able to extract the filename of the audio file and its corresponding transcript. The file extension of the transcript files within the dataset is also indicated. Finally, all the collated data will be written into JSON files to be used by the trainer.

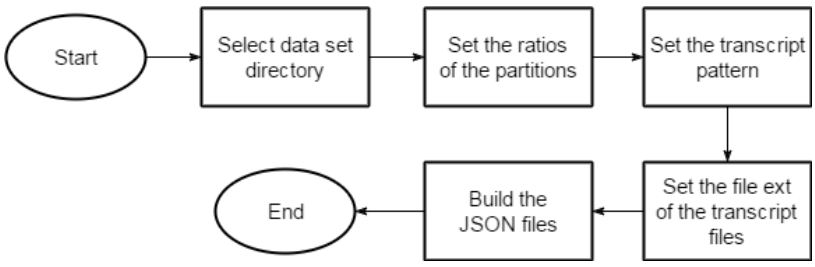


Figure 14: Preparing the Dataset Flowchart

6. **Building the JSON files**

Figure 15 shows the flowchart of the subprocess Building the JSON files in Figure 14. The process involves iterating over all the subdirectories in the dataset directory, opening its transcript files and getting the audio filename and

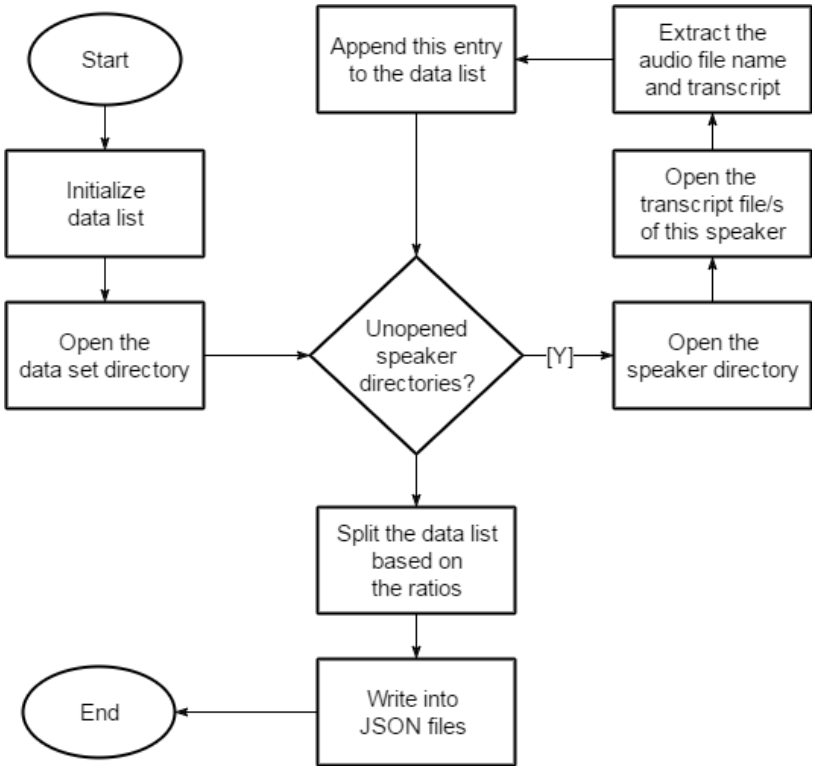


Figure 15: Building the JSON Files Flowchart

its corresponding text transcription. The necessary data is extracted with the use of the provided regex string. After going over all the subdirectories, the data collated is split according to the ratios inputted by the AI expert. Afterwards, the data is written into a JSON file. Three JSON files will be generated by this process: one for training, one for validation, and one for testing.

7. **Creating a new STT model**

Figure 16 shows the flowchart for creating a new STT model. The training and validation JSON files to be used are selected. The AI expert also needs to specify the values of the hyperparameters for the current training session. This includes the number of epochs, learning rate, and the minibatch size. The directory where the models and its other configuration files will be saved also needs to be indicated. Upon starting the training process, the AI expert is able to stop the process whenever.

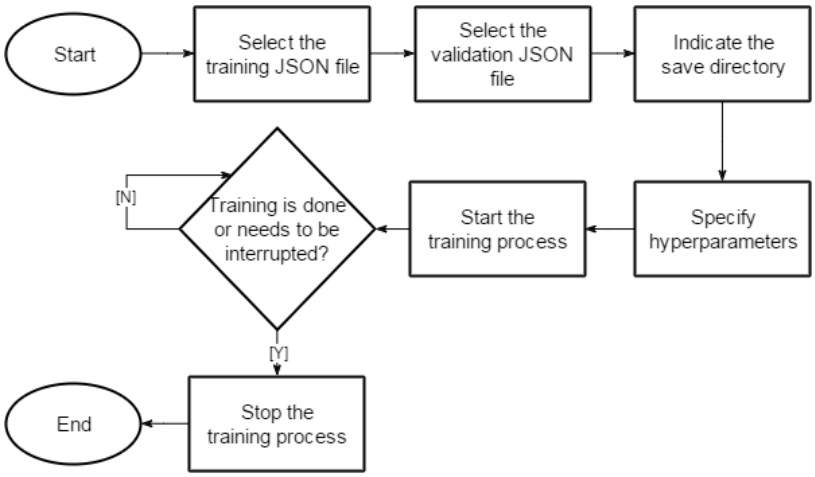


Figure 16: Creating a New STT Model Flowchart

8. **Training process**

   Figure 17 shows the flowchart for the whole training process in Figure 16. Initially, the training and validation JSON files is loaded into the trainer. The training utilities is also initialized; this includes the training function in charge of adjusting the weights, and the data generator in charge of extracting the features of the data. The model is also initialized with the initial weights. The process involves iterating over the number of epochs, and adjusting the weights of the model with the training data repeatedly. Checkpoints of the model is saved in the indicated save directory after every epoch. The validation process is omitted in the flowchart, but it is also performed after every epoch.

9. **Evaluating a STT model**

   Figure 18 shows the flowchart for evaluating a STT model. The testing JSON is loaded into the trainer. All testing utilities is also initialized; this includes the testing function in charge of getting the predictions of the model, and the data generator in charge of extracting the features of the data. The model is also initialized with the weights of the saved model in the indicated directory. The testing process involves iterating over all the testing data, and getting the

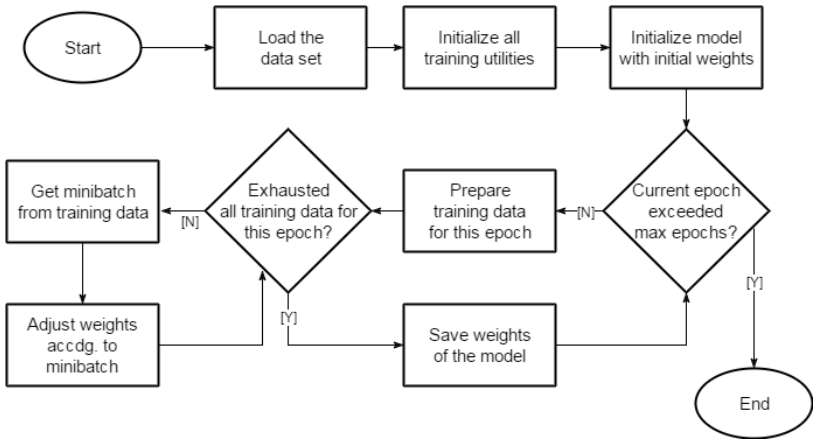


Figure 17: Training process

model's prediction for each of them. The error rate is also calculated for every prediction to quantify the performance of the model.

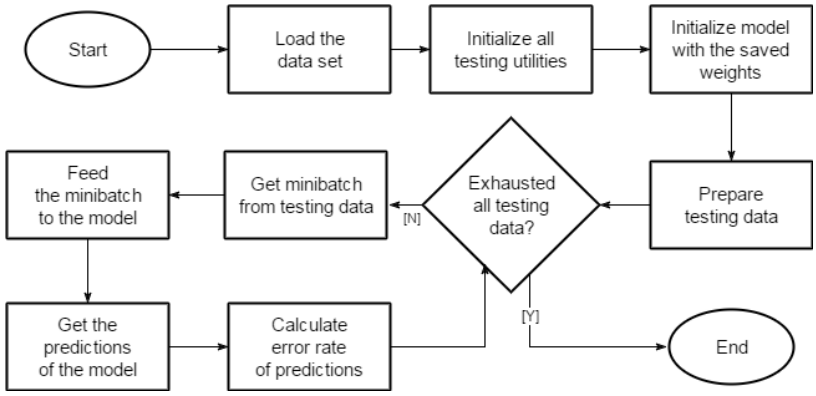


Figure 18: Evaluating a STT Model Flowchart

D. Technical Architecture

The underlying implementation of the desktop application will make use of the Keras framework, a high-level neural networks library written in Python[21]. The Keras framework will serve as a high level wrapper of Theano[22], a key library for deep learning in Python.

1. Mobile application requirements:

   (a) Android Kitkat (4.4) or higher

(b) GNURoot Debian installed with the dependencies

(c) At least 2 GB of free disk space

2. Desktop application requirements:

(a) Ubuntu 16.04

(b) At least 2 GHz CPU

(c) At least 4 GB RAM

(d) At least 2 GB disk free space

3. Dependencies:

(a) Python 2.7, with the following libraries

i. Numpy

ii. Scipy 0.18.1

iii. Futures

iv. Soundfile

v. Matplotlib

vi. H5Py

(b) Theano 0.8.2

(c) Keras 1.1.0, high-level wrapper of Theano

(d) Warp-CTC, a Baidu Research implementation of CTC

(e) NVIDIA Cuda, for systems with an NVIDIA GPU

E. Training Methodology

1. **Data Pre-processing**

For this study, several modifications to the FSC was done. First, all characters with diacritical marks was replaced with its "normalized" form. Also, all

non-letter characters was removed. This includes exclamation points, question marks, commas, periods, and other symbols. This is to simplify the set of characters used by the system into the standard English alphabet.

In addition, text inside grouping symbols such as parentheses and braces was also removed. This is to limit the transcriptions to only include the letters that were actually pronounced in the audio files.

Finally, recordings that contain English words was filtered, so as to have a dataset consisting of only Tagalog words.

2. **Network Architecture**

   The network architecture of the model that was used in the application is a recurrent neural network with three hidden layers, each with 500 nodes. The idea is to make the model lightweight enough to be deployed in the mobile setting.

   The training process comprises of these steps:

   (a) Construct JSON files that references the audio files from the dataset. This is so that transformations such as partitioning or shuffling the dataset will not be done directly to the data files. Since the audio files can be considerably large, loading it all into the system before partitioning or shuffling can be resource-expensive.

   (b) Transform the audio files into spectrogram coefficients. This process converts an audio signal into a sequence of real number values, based on the power at some frequency interval. This will be the input to the network.

   (c) Transform the text transcription into an array of positive integers. This process involves assigning an ID for each letter. As such, a sentence can be represented as a sequence of IDs. These will be used as the output data for training.

(d) Train the model with the given input and output data; model weights are to be adjusted with the loss from CTC through back-propagation.

Such training process is implemented in the Baidu Github DeepSpeech project. The resulting desktop application is an extension of this work. For this system, default values within the framework was used.

V. Results

A. Speech-to-Text Model

The speech-to-text model was trained using the network architecture presented in the Training Methodology. The dataset used was the FSC, which after pre-processing and filtering records from the dataset, resulted in a total of 29 412 entries. These were partitioned into three disjoint subsets, namely for training, testing, and validation. Table 1 shows the record counts for each subset.

Table 1: Partitioning of the Dataset

Dataset	**No. of entries**
Training	17 647
Valid	5 882
Test	5 883
All	29 412

Evaluation of the model was done with the use of Levenshtein distance, also known as Character Error Rate (CER), to measure the deviation between the truth string and the prediction. The "distance" refers to the minimum number of insertions, deletions, or substitutions needed to transform the prediction into the truth string. In which case, a small value for CER is desirable.

Presented in Table 2 are the sample transcriptions produced by the model.

Table 2: Sample Transcriptions from the current model

Truth	**Prediction**
basahing mabuti ang panuto	basahing mabutin ang panuto
ano ba ang mabisang gamot diyan	ano ba ang mamisang kumut yan
kumusta na ang nanay mo	kumusta na ang nanay mo
pamangkin	pamangkina
kapag may isinuksok may madudukot	kapagmay si maksak may marung rokot
pagkat ang salitay isang kahatulan	pagkatang salitay isang kahatulan na
naglunsad ng bagong gimik para sa mga turista	naglunsad nag bagong kimikparas mga tristau
magandang gabi po sa inyo	magandang gabi po sa inyo
kumusta ho kayo	kamustaho kayoa
nasa dugo niyo ba yung ganyang sakit	nasa duguniyo ba at inkan niyang sakitu

The performance of the trained model has been unsatisfactory, with a lot of misspellings in its transcriptions. The current model has rated an average 3.95 CER, which indicates a fairly high misspelling rate from the predictions.

Observations on the test results indicate that the model has been able to predict most of the Tagalog sounds, but is still confused on similar sounding letters such as /n/ and /m/, and /d/ and /r/.

Also, several predictions was also observed to have unnecessary trailing strings appended right after the actual sentence. This can be attributed to the noise by the end of the recordings in the dataset, wherein the model has assigned characters to. Then again, a good model must be able to incorporate this noise to its predictions, which the current model failed to do.

B. Mobile Application

1. **Inbox Screen**

   The inbox screen of the PASABI application is shown in Figure 19. The inbox displays all the recent incoming and outgoing messages of the user. Every entry in the inbox corresponds to a messages thread, or a conversation, with a contact. Upon clicking an inbox entry, the user will be redirected to the Conversation screen, where all the messages sent and received from that particular contact will be displayed. The buttons at the bottom bar are the Compose message button and the Browse button. The Compose message button allows the smartphone user to create a blank conversation. The Browse button allows the smartphone user to select which STT model is the application going to use.

2. **Conversation Screen**

   The conversation screen of the PASABI application is shown in Figure 20. The conversation screen displays all the messages sent and received from a particular

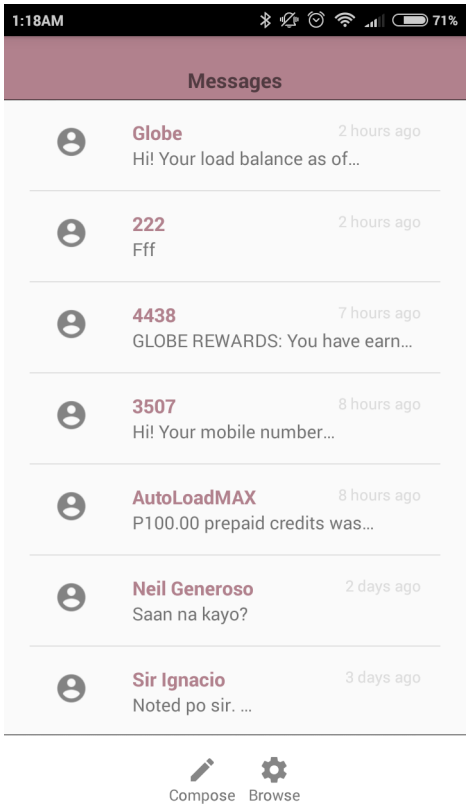


Figure 19: Inbox Screen of the PASABI Application

contact, arranged in chronological order.

The contact information is shown at the top bar, showing the name and the phone number of the contact. On the side is the Contacts button which allows the user to choose whose conversation is to be viewed.

At the bottom bar is the message text field. This is where the user will be typing the text messages with the use of an on-screen keyboard. On its left is the Send button, which will send whatever text is currently in the message text field.

Also in the bottom bar is the microphone icon. Upon clicking the microphone icon, the Speech compartment, containing the option for a speech transcription in composing the message, is opened.

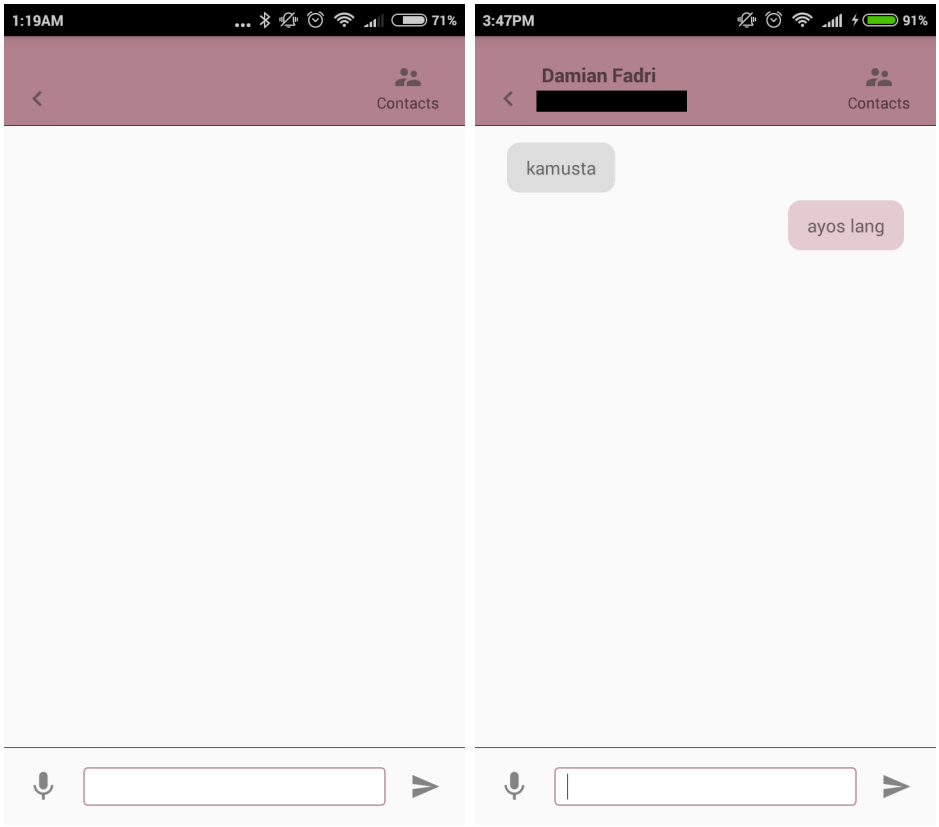


Figure 20: Conversation Screen of the PASABI Application

3. **Speech Compartment**

   The speech compartment of the PASABI application is shown in Figure 21. The compartment shows a big circular button with a microphone symbol, along with some text below to indicate its current state.

   Upon pressing the button, the button state will be shifting to a recording state; the user will be able to provide voice input to the application. Upon pressing the button a second time, the recording will be stopped and the button will be shifting to a transcribing state. At this time, the user must wait until the STT model finishes transcribing the voice input. The transcribing will be done when the transcription shows up on the message text field.

4. **Contacts Screen**

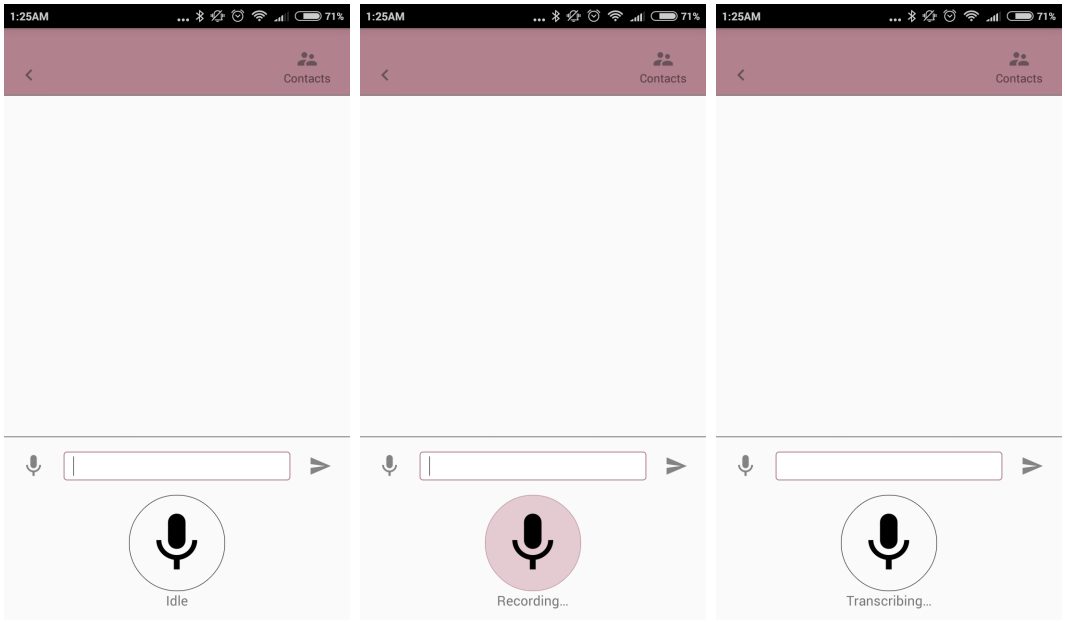


Figure 21: Speech Compartment of the PASABI Application

The contacts screen of the PASABI application is shown in Figure 22. The screen shows all the saved contacts in the user's phone. Upon clicking a name, the contact information is then sent back to the conversation screen and refreshed with the messages relating to the selected contact.

5. **Models Screen**

The models screen of the PASABI application is shown in Figure 23. The screen shows all the saved models in the application directory. The names displayed are the directory names containing the actual STT model files. Below it is the date the model is created. Upon clicking a model name, the model is set to be the default model to be used by the application. The default model is highlighted to differentiate it to other models. When the default model is updated, the STT backend is re-initialized accordingly.

C. Desktop Application

1. **General Layout**

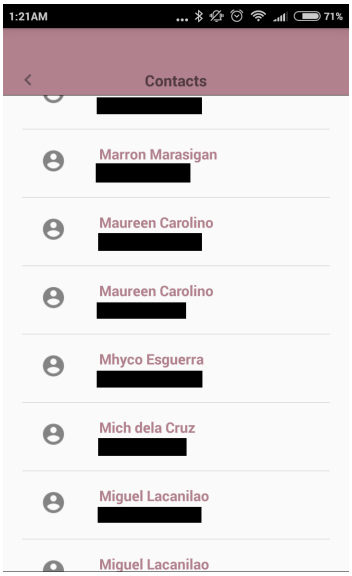


Figure 22: Contacts Screen of the PASABI application

The general layout of the PASABI Trainer is shown in Figure 24. On the left pane are the actions that an AI expert may need divided into tabs. The tabs are as shown; Dataset tab for dataset creation, Train tab for training a STT model, and Test tab for testing a STT model. The right pane holds the logger and the progress bar of the trainer. The logger is in charge of printing whatever the trainer is doing at the moment, along with the current timestamp the action was executed. The progress bar indicates the completion rate of the process the trainer is currently doing. The AI expert is also given the option to export the current log into an external file.

2. **Dataset Pane**

The Dataset pane of the PASABI Trainer is shown in Figure 25. The Dataset pane is in charge of creating a JSON file referencing the dataset. This JSON file will be used on the other parts of the trainer.

The AI expert needs to specify the Dataset directory, or the directory that holds all the audio files along with their text transcriptions. The Transcript

Figure 23: Models Screen of the PASABI application

pattern field requires a regex string input to extract the audio filenames and its corresponding text transcription from the transcription files. The File extension field tells the trainer which files in the dataset directory are to be read as transcription files.

The Dataset ratios allows the AI expert to define how the data will be split, and properly partitioned for training and testing.

3. **Training Pane**

The Training pane of the PASABI Trainer is shown in Figure 26. The Training pane is in charge of the actual training process of STT models.

The AI expert needs to indicate the file paths of the training and validation JSONs generated from the Dataset pane. The files referenced inside those files will be used on the training process. The AI expert also needs to indicate where the STT models will be saved, along with its configuration files.

The AI expert also needs to indicate the hyperparameters. This includes the number of epochs, the learning rate, and the minibatch size to be used for

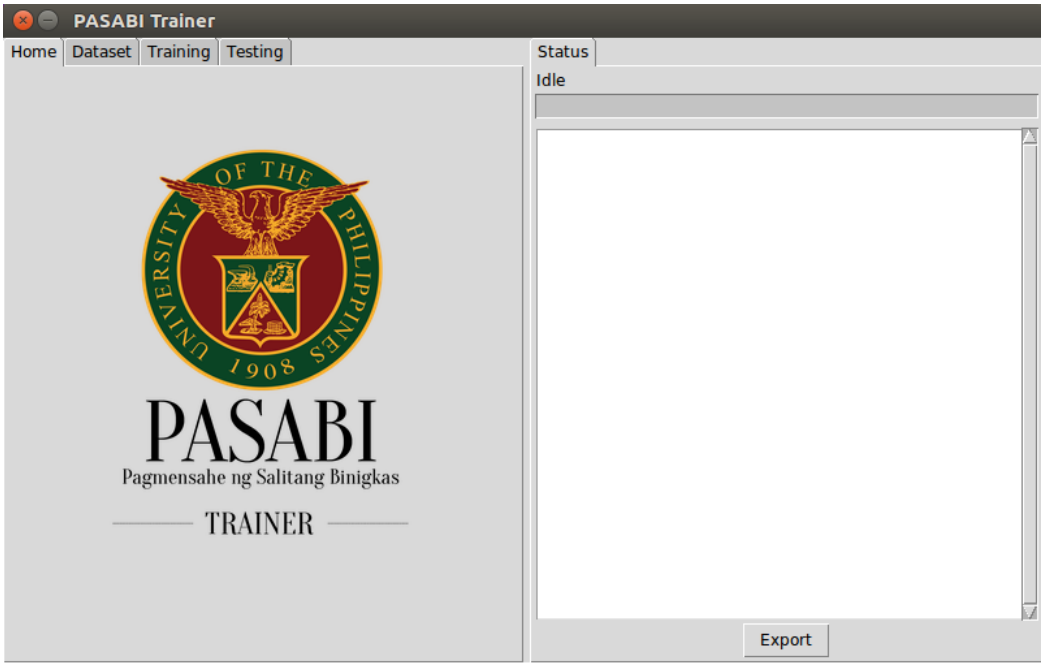


Figure 24: Full view of the PASABI Trainer

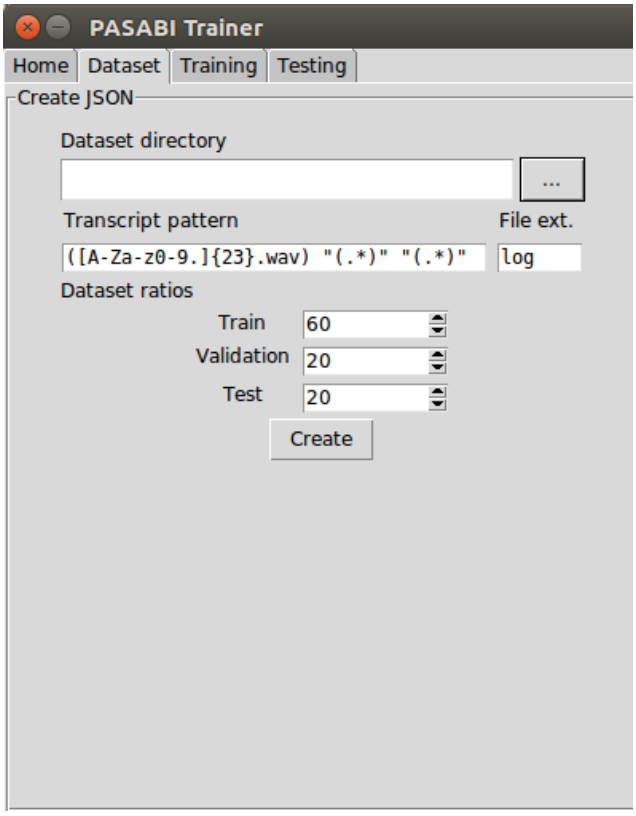


Figure 25: Dataset Pane of the PASABI Trainer

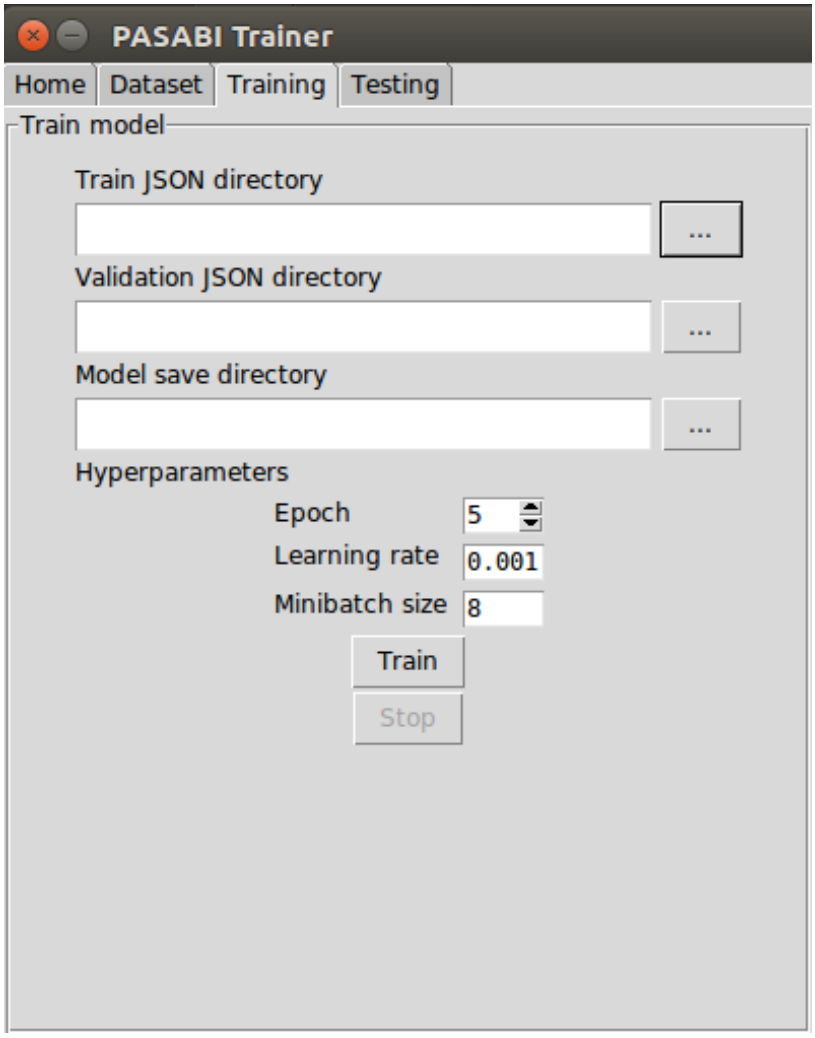


Figure 26: Training Pane of the PASABI Trainer

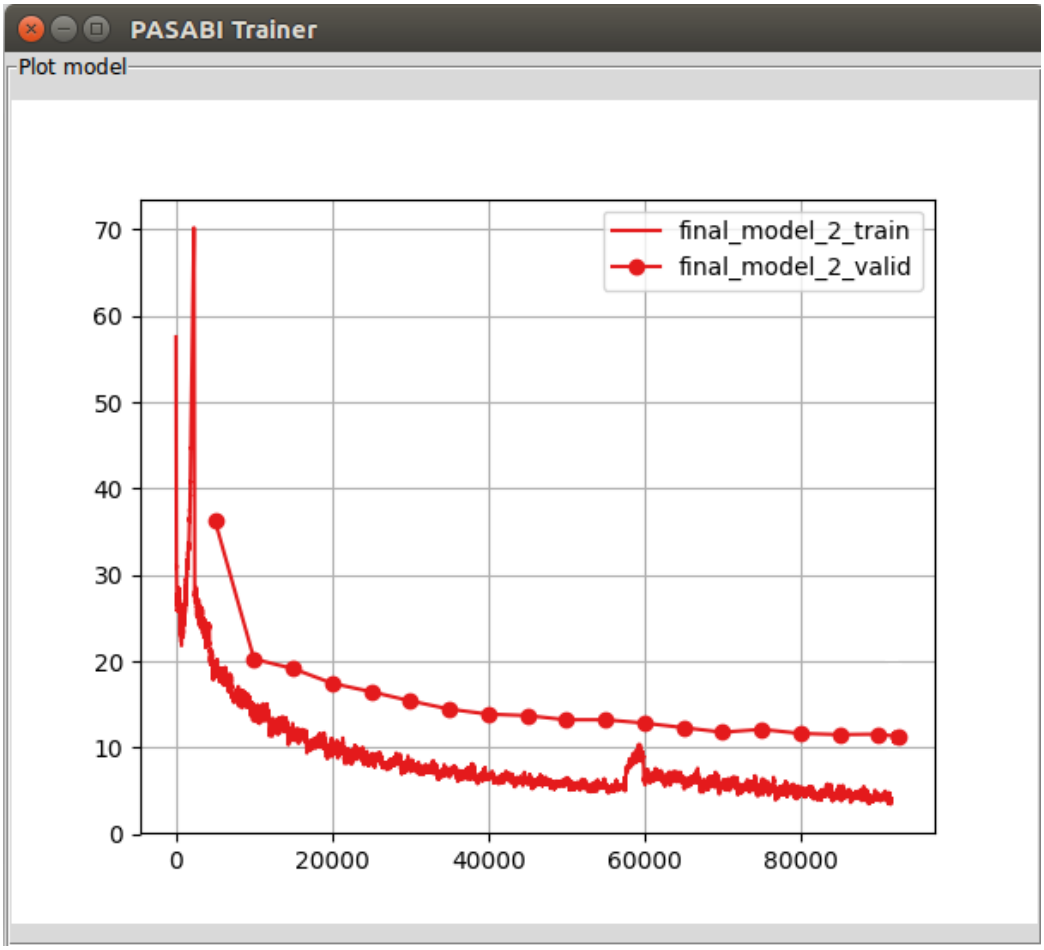


Figure 27: Plot Pane of the PASABI Trainer

training.

The Train button initiates the training process, and will be disabled until the training process is stopped or is finished. The Stop button halts the current training progress.

4. **Plot Window**

The Plot Window of the PASABI Trainer is shown in Figure 27. This window will show upon start of the training process. In the plot, the AI expert can see the current state of the model in terms of performance. The plot shows the current training and validation loss against the number of iterations elapsed.

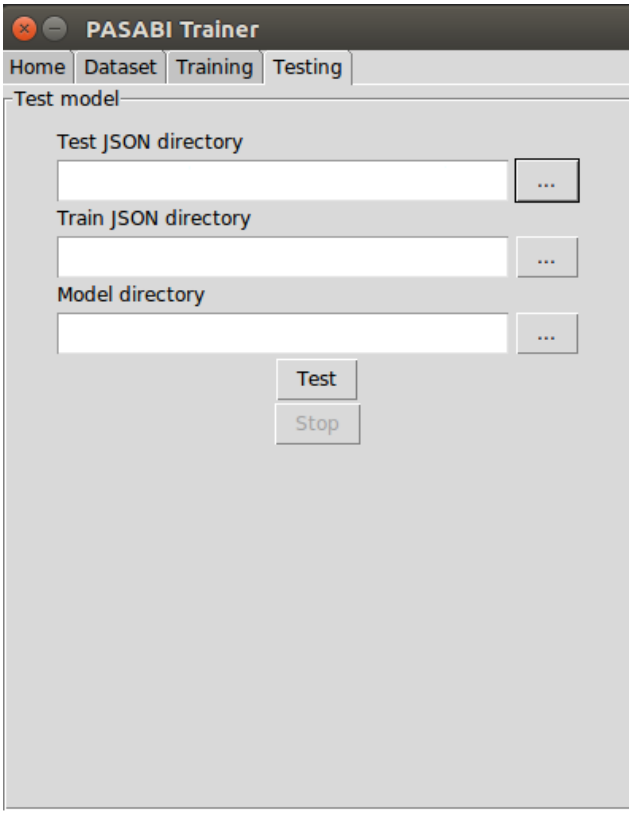


Figure 28: Testing Pane of the PASABI Trainer

5. **Testing Pane**

The Testing Pane of the PASABI Trainer is shown in Figure 28. The Testing pane is in charge of evaluating the performance of the STT model.

The AI expert needs to input the test JSON, which will be used by the trainer to feed into the model and evaluate its predictions. The training JSON input is used to calculate the means and standard deviation of the audio files for normalization. The AI expert also needs to input the directory where the model to be evaluated is saved.

While the testing process is ongoing, the model's predictions along with the true transcriptions are printed alongside in the logger. This is for the AI expert to see the performance of the current model. At the end of the training process, the average character error rate of the model is displayed in the logger.

VI. Discussions

A. PASABI Trainer

A..1 Objectives

The objectives regarding the development of the desktop application trainer has been met. An AI Expert can process the dataset in preparation for training, train models, and test trained models with the developed application.

A..2 Problems Encountered

1. During development, making the base implementation of Baidu's DeepSpeech work required numerous libraries and software for it to function properly. For instance, the CTC loss function implementation is not supported directly by Keras. As such, a third-party implementation, also by Baidu, was used. The limitation is that, it is only compatible with Ubuntu systems, as per their documentation.

   In general, setting up a system for deep learning is straightforward, but quite cumbersome. While they are all available online, checking up on version compatibilities solved most of the dependency errors.

2. Providing a live plot of the training progress was also an issue during development. The plot functionality of the standard Matplotlib library does not provide a way to update an existing plot by just adding the new points. Instead, the approach has been to create an entirely new plot with the updated set of points, and repainting the old plot.

   Such a process can be resource-heavy if done on short intervals (e.g. every iteration during training). To circumvent this, the plot updating has been set to be done after every training epoch. In addition, status logs are also printed

every iteration. This is to compensate on the fact that the user may not be able to closely monitor the model's status with just the slowly-updating plot.

B. PASABI Mobile Application

B..1 Objectives

The primary objectives, that is to regarding the development of the mobile application has been met. In particular, the message sending and message receiving functionalities are provided by extending on the built-in SMS services in the phone; the voice input functionality is provided by the phone's microphone.

B..2 Problems Encountered

1. One of the hurdles of this study is deploying a trained model into the mobile setting. Keras models are currently unsupported with Android's Native Development Kit; in which case, they cannot be imported directly.

   Also, the "transcribing" task is written as a Python script. In this case, the libraries imported in the script must be installed in the system it is currently running at. For Android, there isn't a standard way to do this.

   The solution presented is to use another application, where the scripts will be ran, in parallel with the PASABI mobile application. This is the purpose of GNURoot Debian. The external application serves as an emulated machine where the Keras model is loaded, and transcribe whenever requested.

2. A problem regarding GNURoot Debian is how it will communicate with the PASABI application. Technically, they are two separate applications, but is needed to function as one. Also, there really isn't any direct way to pipe instructions from the PASABI application to GNURoot, and vice-versa.

As such, shared files were introduced for the two applications to communicate. One file serves as a flag file, in which the two applications poll. Whenever the flag indicates that it is an application's "turn," the other application waits until the flag is reset. Another file serves as the message file, in which messages can be "sent" from one application to another.

3. Implementing the SMS functionality also posed some issues during development. First of all, the primary requirement is that it must be able to access the user's actual inbox and send a message with the user's phone number. That is, the PASABI application must not have its own SMS inbox and sending functionality.

   Fortunately, Android provides a high-level framework in accessing the content services of the phone. This includes access to the user's contacts, inbox, outbox, etc. The only requirement is that permissions for such accesses are allowed upon installation.

4. Putting the Keras trained model in the phone posed no problems. The Keras model isn't that memory-heavy, with a filesize of about 50 MB. However, its issues lie when it is used to transcribe. Given a phone's hardware specifications, a prediction task can run for a few minutes until it produces an output. This can be attributed to the numerous calculations being done inside the model, which can run very slowly on a low-end phone.

   Initially, a prediction on a Xiaomi Redmi 2 phone took 5-7 minutes, which was considered extremely slow. An attempted solution is to lessen the number of layers and/or the number of nodes in the RNN. Upon this change, a prediction took about 1-3 minutes. This is still considered slow, but relatively faster than the previous attempts.

C. Significance of PASABI

PASABI serves to provide the means to develop speech-to-text models without the use of phoneme transcriptions. While the presented model may be unsatisfactory, the trainer still exists as a tool for AI Experts to produce far better models using the same concept.

The mobile application serves as a reimplementation of FiliText, which provides a means for hands-free text messaging. Extending on the previous study, the speech-to-text functionality can be utilized to text while driving or outdoors.

VII. Conclusions

PASABI is a system that consists of a desktop trainer and a mobile application. The desktop trainer can be used by AI experts to produce models for speech transcription, which will be deployed onto the mobile application. The mobile application serves as a straightforward text messaging application, offering the standard SMS services. The mobile application extends this by providing a speech-to-text functionality using the trained models.

PASABI is developed to provide the means to create speech-to-text models without the use of phonetic transcriptions. The idea is to eliminate the need to require speech datasets with phonetic transcriptions and/or development of algorithms to generate the phonemes.

Results of the study have shown that a character-based transcription may be insufficient. Further improvements can be implemented for the speech transcription models. This includes training another model, or introducing an alternative methodology in developing these models.

VIII. Recommendations

There are several areas in PASABI that can be improved further or reimplemented. The following recommendations are arranged based on priority:

1. Improve speech-to-text accuracy. Currently, the deployed model is still lacking in terms of its transcription accuracy. This can be done by either training another model, or perhaps changing the methodology fully.

   A possible extension is the introduction of a language model to aid with the proper spelling of words. Since the model uses a character-based prediction, its inaccuracy is primarily due to sound-alikes and unnecessary letters added. A way to circumvent this is a heuristic search for the most likely word the misspelled word represents.

2. Improve speech-to-text transcription speed. With the use of GNURoot, as it is an emulated terminal, the speed of the transcriptions is compromised.

   An alternative approach is to produce another model that is lightweight enough for GNURoot. This includes reducing the number of nodes or layers in the model, or perhaps implementing an entirely new network architecture.

3. Implement an alternative way to deploy a trained model in the mobile setting. The use of GNURoot is functional, but isn't practical since it requires the user to install a lot of dependencies.

IX. Bibliography

[1] R. Gruhn, *Statistical pronunciation modeling for non-native speech processing.* PhD thesis, University of Ulm, 2008.

[2] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," pp. 2673–2681, 1997.

[3] S. Hochreiter and J. Schmidhuber, "LSTM can solve hard long time lag problems," in *Advances in Neural Information Processing Systems 9, NIPS, Denver, aCO, USA, December 2-5, 1996*, pp. 473–479, 1996.

[4] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Ng, "Deep speech: Scaling up end-to-end speech recognition," *CoRR*, 2014.

[5] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning." Book in preparation for MIT Press, 2016.

[6] J. Feng, S. Zhu, R. Hu, and A. Sears, "Speech technology in real world environment," in *Proceedings of the 10th International Conference on Computers and Accessibility - ASSETS '08*, pp. 233–234, 2008.

[7] W. Shudong and M. Higgins, "Limitations of mobile phone learning," in *Proceedings of the IEEE International Workshop on Wireless and Mobile Technologies in Education*, WMTE '05, (Washington, DC, USA), pp. 179–181, IEEE Computer Society, 2005.

[8] F. Ang, M. C. Burgos, and M. De Lara, "Automatic speech recognition for closed-captioning of filipino news broadcasts," in *Natural Language Processing andKnowledge Engineering (NLP-KE), 2011 7th International Conference on*, 2011.

[9] F. Ang, Y. Miyanaga, R. C. Guevara, R. Cajote, and M. G. A. Bayona, "Open domain continuous filipino speech recognition with code-switching," in *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*, 2014.

[10] J. Chua, U. Chua, C. de Padua, J. I. Tan, and D. Cheng, *FiliText: A Filipino Hands-free Text Messaging Application*. Undergraduate thesis, De La Salle University, 2011.

[11] R. Guevara, M. Co, E. Espina, and R. Sagum, "Development of a filipino speech corpus," 2002.

[12] C. M. University, "Cmu sphinx wiki." http://cmusphinx.sourceforge.net/wiki. "Accessed: 2016-11-10".

[13] A. Maas, Z. Xie, D. Jurafsky, and A. Ng, "Lexicon-free conversational speech recognition with neural networks," in *Proceedings the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2015.

[14] A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer, 2012.

[15] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2016, Shanghai, China, March 20-25, 2016*, pp. 4945–4949, 2016.

[16] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*, pp. 6645–6649, 2013.

[17] H. Soltau, H. Liao, and H. Sak, "Phoneme recognition in timit with blstm-ctc," 2008.

[18] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pp. 1764–1772, 2014.

[19] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," 2015.

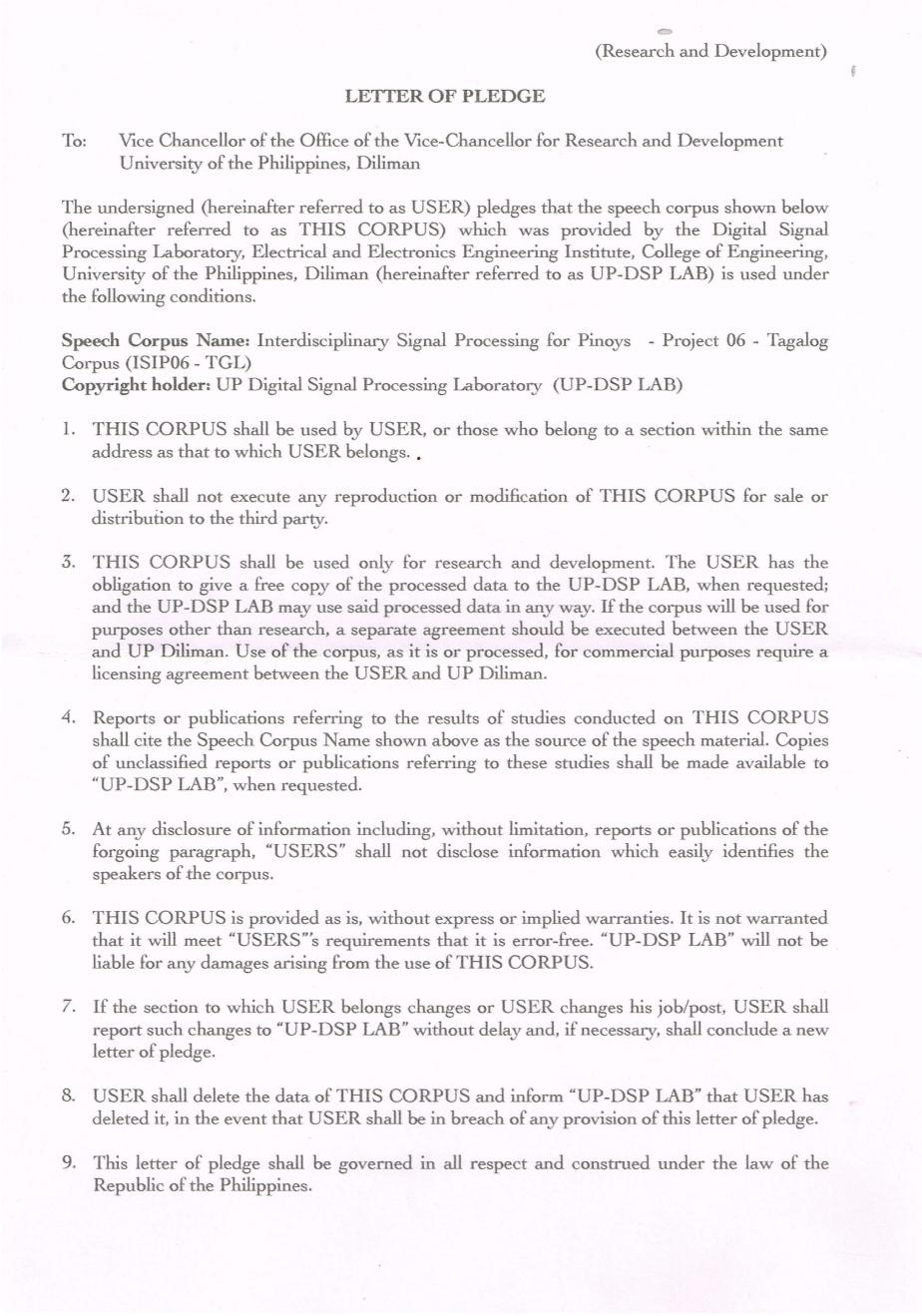
[20] Baidu, "Baidu deepspeech demo." https://github.com/baidu-research/ba-dls-deepspeech, 2017.

[21] F. Chollet, "Keras." https://github.com/fchollet/keras, 2015.

[22] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," vol. abs/1605.02688, 2016.

X. Appendix

A. Forms



(Research and Development)

LETTER OF PLEDGE

To: Vice Chancellor of the Office of the Vice-Chancellor for Research and Development
University of the Philippines, Diliman

The undersigned (hereinafter referred to as USER) pledges that the speech corpus shown below (hereinafter referred to as THIS CORPUS) which was provided by the Digital Signal Processing Laboratory, Electrical and Electronics Engineering Institute, College of Engineering, University of the Philippines, Diliman (hereinafter referred to as UP-DSP LAB) is used under the following conditions.

Speech Corpus Name: Interdisciplinary Signal Processing for Pinoys - Project 06 - Tagalog Corpus (ISIP06 - TGL)
Copyright holder: UP Digital Signal Processing Laboratory (UP-DSP LAB)

1. THIS CORPUS shall be used by USER, or those who belong to a section within the same address as that to which USER belongs.
2. USER shall not execute any reproduction or modification of THIS CORPUS for sale or distribution to the third party.
3. THIS CORPUS shall be used only for research and development. The USER has the obligation to give a free copy of the processed data to the UP-DSP LAB, when requested; and the UP-DSP LAB may use said processed data in any way. If the corpus will be used for purposes other than research, a separate agreement should be executed between the USER and UP Diliman. Use of the corpus, as it is or processed, for commercial purposes require a licensing agreement between the USER and UP Diliman.
4. Reports or publications referring to the results of studies conducted on THIS CORPUS shall cite the Speech Corpus Name shown above as the source of the speech material. Copies of unclassified reports or publications referring to these studies shall be made available to "UP-DSP LAB", when requested.
5. At any disclosure of information including, without limitation, reports or publications of the forgoing paragraph, "USERS" shall not disclose information which easily identifies the speakers of the corpus.
6. THIS CORPUS is provided as is, without express or implied warranties. It is not warranted that it will meet "USERS"'s requirements that it is error-free. "UP-DSP LAB" will not be liable for any damages arising from the use of THIS CORPUS.
7. If the section to which USER belongs changes or USER changes his job/post, USER shall report such changes to "UP-DSP LAB" without delay and, if necessary, shall conclude a new letter of pledge.
8. USER shall delete the data of THIS CORPUS and inform "UP-DSP LAB" that USER has deleted it, in the event that USER shall be in breach of any provision of this letter of pledge.
9. This letter of pledge shall be governed in all respect and construed under the law of the Republic of the Philippines.

Figure 29: Letter of Pledge for the FSC Dataset 1

(Research and Development)

10. UP-DSP LAB and USER shall resolve any problems in good faith and through mutual consultations, unless otherwise specified herein.

11. UP-DSP LAB and USER consent to the jurisdiction of the Quezon City Regional Trial Court for all purposes hereunder.

USER
Signature: _____ Date: Dec. 7, 2016
Printed Name: MARVIN JOHN C. IGNACIO

Section/Division/: Mathematics and Computer Science Unit
Organization: University of the Philippines Manila - College of Arts and Sciences
Address: Padre Faura, Ermita, Manila
Phone: 254 1881 Fax: _____
Email: mcignacio3@up.edu.ph

Section Leader
Signature: _____ Date: Dec. 7, 2016
Printed Name: MARVIN JOHN C. IGNACIO
Title/Designation: Instructor

Figure 30: Letter of Pledge for the FSC Dataset 2

B. Source Code

File: trainer/trainer.py

```
import random
import Tkinter as tk
import ttk
import json

from PIL import Image, ImageTk

import gui.Logger as Logger
import gui.TrainingPanel as TrainingPanel
import gui.TestingPanel as TestingPanel
import gui.DatasetPanel as DatasetPanel
import gui.ProgressPanel as ProgressPanel


root = tk.Tk()

image = Image.open("up.png")
photo = ImageTk.PhotoImage(image)

root.tk.call("wm", "iconphoto", root._w, photo)
root.title("PASABI Trainer")
#root.iconbitmap("up.ico")
root.resizable(0, 0)

# Super main container
main_pane = ttk.PanedWindow(height=480, width
    =800, orient=tk.HORIZONTAL)

# Super subcontainers
left_pane = ttk.PanedWindow(main_pane, width
    =400)

# Tabbed container
ntbk = ttk.Notebook(left_pane)

# First tab
pane_home = ttk.PanedWindow(ntbk)
image = Image.open("pasabi.png")
image = image.resize((400,400), Image.ANTIALIAS)
display = ImageTk.PhotoImage(image)

label_home = tk.Label(pane_home, image=display)
pane_home.add(label_home)

# First tab
DatasetPanel.initialize(ntbk)
pane_dataset = DatasetPanel.get_widget()

# Second tab
TrainingPanel.initialize(ntbk)
pane_training = TrainingPanel.get_widget()

# Third tab
TestingPanel.initialize(ntbk)
pane_testing = TestingPanel.get_widget()

# Add the tabs to the notebook
ntbk.add(pane_home, text="Home")
ntbk.add(pane_dataset, text="Dataset")
ntbk.add(pane_training, text="Training")
ntbk.add(pane_testing, text="Testing")

left_pane.add(ntbk)

# ----------
# Left panel
right_pane = ttk.PanedWindow(main_pane, width
    =300, orient=tk.VERTICAL)

ntbk_right = ttk.Notebook(right_pane)

some_container = tk.PanedWindow(ntbk_right,
    orient=tk.VERTICAL)

ProgressPanel.initialize(right_pane)
pane_progress = ProgressPanel.get_widget()

# Instantiate a logger pane
Logger.initialize(right_pane)
logger = Logger.get_widget()

some_container.add(pane_progress)
some_container.add(logger)

ntbk_right.add(some_container, text="Status")

# Add widgets to the subcontainers
right_pane.add(ntbk_right)

# Add to the super main container
main_pane.add(left_pane)
main_pane.add(right_pane)

# Instantiate main window
main_pane.pack()

root.protocol("WM_DELETE_WINDOW", lambda: root.
    quit())
root.mainloop()
```

File: trainer/code/char_map.py

```
char_map_str = """
' 1
<SPACE> 2
a 3
b 4
c 5
d 6
e 7
f 8
g 9
h 10
i 11
j 12
k 13
l 14
m 15
n 16
o 17
p 18
q 19
r 20
s 21
t 22
u 23
v 24
w 25
x 26
y 27
z 28
"""
char_map = {}
index_map = {}
for line in char_map_str.strip().split('\n'):
    ch, index = line.split()
    char_map[ch] = int(index)
    index_map[int(index)] = ch
index_map[2] = ' '
```

File: trainer/code/create.py

```
from __future__ import absolute_import, division
    , print_function

import re
import os
import wave
import json
import sys
import random

import gui.Logger as Logger
import gui.ProgressPanel as ProgressPanel
import gui.ActionManager as ActionManager
import ttk

# Open dir
        # Open speaker dir
                # Find log file(s)
                # Open log file(s)
                        # Parse using regex
                        # Append to filename (
                            yung folder name (
                            speaker))
                        # Store to some global
                            array
# Convert to JSON
# Write to some file


def get_entry_fields(line, pattern):

        # Extract fields in current line
        matched = re.match(pattern, line)
```

```
        entry = {}
        if matched:
                entry["key"] = matched.group("
                    key")
                entry["text"] = matched.group("
                    text")

        return entry

def get_transcripts(speaker_dir, file_ext):
        transcripts = []

        for i in os.listdir(speaker_dir):
                if i.endswith(file_ext):
                        transcripts.append(i)

        return transcripts

def create_json(args):
        global progress

        try:
                dataset_dir = args["dataset_dir
                    "]
                save_json   = args["save_file"]
                pattern     = args["pattern"]
                file_ext    = args["file_ext"]
                ratios          = args["ratios"]

                # List of all speakers
                speakers = os.listdir(
                    dataset_dir)

                # List of all transcripts
                data_all = []

                ProgressPanel.set_maximum(len(
                    speakers))
                for i, speaker in enumerate(
                    speakers):

                    #Logger.append("(Dataset
                        ) ))
                    ProgressPanel.
                        set_current_value(i
                        + 1)
                    ProgressPanel.
                        set_status_message
                        ("{}/{} speakers
                        processed".format(i
                        + 1, len(speakers)
                        ))

                    # Get path to speaker
                        folder
                    speaker_dir = os.path.
                        join(dataset_dir,
                        speaker)

                    # Get the log files for
                        this speaker
                    transcripts =
                        get_transcripts(
                        speaker_dir,
                        file_ext)

                    for transcript in
                        transcripts:

                        # Get filepath
                            of
                            transcript
                        transcript_dir =
                            os.path.
                            join(
                            speaker_dir
                            ,
                            transcript)

                        # Open
                            transcript
                            file
                        with open(
                            transcript_dir
                            , "r") as
                            trans_f:
                                for line
                                    in

                                        trans_f
                                        :
#
    Remove
    newline
    in
    line
line
    =
    line
    .
    rstrip
    ()

#
    Get
    entry
    fields
entry
    =
    get_entry_fi
    (
    line
    ,
    pattern
    )

if
    (
    len
    (
    entry
    )
    ==
    0)
    :
        continue

#
    File
    name
    of
    the
    audio
    file
audio_file
    =
    os
    .
    path
    .
    join
    (
    speaker_dir
    ,
    entry
    ["
    key
    "])

#
```

```
# Splitting part
ProgressPanel.set_maximum(3)

# Absolute file path of the JSON to be split
filepath = save_json.rstrip()

# Directory where the JSON to be split is
directory = os.path.split(save_json)[0]

# Template for the filename format
template = "{}_{}.json"
filename_all = os.path.splitext(filepath)[0]

# Construct the filenames of the split files
filename_train = os.path.join(directory, template.format(filename_all, "train"))
filename_valid = os.path.join(directory, template.format(filename_all, "valid"))
filename_test  = os.path.join(directory, template.format(filename_all, "test"))

size_all = len(data_all)

# Shuffle the contents
random.shuffle(data_all)

# Split into the proper ratios
size_train = int((float(ratios[0]) / 100) * size_all)
size_valid = int((float(ratios[1]) / 100) * size_all)
size_test  = int((float(ratios[2]) / 100) * size_all)

ProgressPanel.set_current_value(1)
ProgressPanel.set_status_message("{}/{} split files created".format(1, 3))

# Write each to file
with open(filename_train, "w") as handle_train:
    for line in data_all[0:size_train]:
        handle_train.write(json.dumps(line) + "\n")

ProgressPanel.set_current_value(2)
ProgressPanel.set_status_message("{}/{} split files created".format(2, 3))

with open(filename_valid, "w") as handle_valid:
    for line in data_all[size_train:size_train + size_valid]:
        handle_valid.write(json.dumps(line) + "\n")

ProgressPanel.set_current_value(3)
ProgressPanel.set_status_message("{}/{} split files created".format(3, 3))

with open(filename_test, "w") as handle_test:
    for line in data_all[size_train + size_valid:]:
        handle_test.write(json.dumps(line) + "\n")

# Log current results
Logger.append("The split JSON files were saved in {}".
```

```
                    format(directory))
                Logger.append("Training JSON:
                    {}".format(size_train))
                Logger.append("Validation JSON:
                    {}".format(size_valid))
                Logger.append("Test JSON: {}".
                    format(size_all -
                    size_train - size_valid))
        except Exception as e:
                print(e)
                Logger.append("An unexpected
                    error has occurred. The
                    dataset you have inputted
                    is invalid.")

        ProgressPanel.set_idle()
        ActionManager.markAsDone()
```

File: trainer/code/data_generator.py

```
"""
Defines a class that is used to featurize audio
    clips, and provide
them to the network for training or testing.
"""

from __future__ import absolute_import, division
    , print_function
from functools import reduce

import json
import logging
import numpy as np
import random
import os
import sys

from concurrent.futures import
    ThreadPoolExecutor, wait

from core.utils import calc_feat_dim,
    spectrogram_from_file, text_to_int_sequence
import gui.Logger as Logger
import gui.ProgressPanel as ProgressPanel

RNG_SEED = 123
#logger = logging.getLogger(__name__)


class DataGenerator(object):
    def __init__(self, step=10, window=20,
        max_freq=8000, desc_file=None):
        """
        Params:
            step (int): Step size in
                milliseconds between windows
            window (int): FFT window size in
                milliseconds
            max_freq (int): Only FFT bins
                corresponding to frequencies
                between
                [0, max_freq] are returned
            desc_file (str, optional): Path to a
                JSON-line file that contains
                labels and paths to the audio
                files. If this is None,
                then
                load metadata right away
        """
        self.feat_dim = calc_feat_dim(window,
            max_freq)
        self.feats_mean = np.zeros((self.
            feat_dim,))
        self.feats_std = np.ones((self.feat_dim
            ,))
        self.rng = random.Random(RNG_SEED)
        if desc_file is not None:
            self.load_metadata_from_desc_file(
                desc_file)
        self.step = step
        self.window = window
        self.max_freq = max_freq

    def featurize(self, audio_clip):
        """ For a given audio clip, calculate
            the log of its Fourier Transform
        Params:
            audio_clip(str): Path to the audio
                clip
        """
        return spectrogram_from_file(
            audio_clip, step=self.step, window=
                self.window,
            max_freq=self.max_freq)

    def load_metadata_from_desc_file(self,
        desc_file, partition='train',
                                     max_duration
                                         =10.0)
                                         :
        """ Read metadata from the description
            file
            (possibly takes long, depending on
                the filesize)
        Params:
            desc_file (str):  Path to a JSON-
                line file that contains labels
                and
                paths to the audio files
            partition (str): One of 'train', '
                validation' or 'test'
            max_duration (float): In seconds,
                the maximum duration of
                utterances to train or test on
        """
        #Logger.append("Reading description file
            : {} for partition: {}".format(
            desc_file, partition))

        audio_paths, durations, texts = [], [],
            []
        with open(desc_file) as json_line_file:
            for line_num, json_line in enumerate
                (json_line_file):
                try:
                    spec = json.loads(json_line)
                    if float(spec['duration']) >
                        max_duration:
                        continue

                    audio_paths.append(spec['key
                        '])
                    durations.append(float(spec
                        ['duration']))
                    texts.append(spec['text'])
                except Exception as e:
                    # Change to (KeyError,
                        ValueError) or
                    # (KeyError,json.decoder.
                        JSONDecodeError),
                        depending on
                    # json module version
                    Logger.append("Error reading
                        line #{}: {}".format(
                        line_num, json_line))
                    Logger.append(str(e))

        if partition == 'train':
            self.train_audio_paths = audio_paths
            self.train_durations = durations
            self.train_texts = texts
        elif partition == 'validation':
            self.val_audio_paths = audio_paths
            self.val_durations = durations
            self.val_texts = texts
        elif partition == 'test':
            self.test_audio_paths = audio_paths
            self.test_durations = durations
            self.test_texts = texts
        else:
            raise Exception("Invalid partition
                to load metadata. "
                            "Must be train/
                                validation/test
                                ")

    def load_train_data(self, desc_file):
        self.load_metadata_from_desc_file(
            desc_file, 'train', max_duration
            =10.0)

    def load_test_data(self, desc_file):
        self.load_metadata_from_desc_file(
            desc_file, 'test', max_duration
            =10.0)

    def load_validation_data(self, desc_file):
        self.load_metadata_from_desc_file(
            desc_file, 'validation',
            max_duration=10.0)

    @staticmethod
    def sort_by_duration(durations, audio_paths,
        texts):
        return zip(*sorted(zip(durations,
```

```
            audio_paths, texts)))

    def normalize(self, feature, eps=1e-14):
        return (feature - self.feats_mean) / (
            self.feats_std + eps)

    def prepare_minibatch(self, audio_paths,
        texts):
        """ Featurize a minibatch of audio, zero
            pad them and return a dictionary
        Params:
            audio_paths (list(str)): List of
                paths to audio files
            texts (list(str)): List of texts
                corresponding to the audio
                files
        Returns:
            dict: See below for contents
        """
        assert len(audio_paths) == len(texts),\
            "Inputs and outputs to the network
                must be of the same number"
        # Features is a list of (timesteps,
            feature_dim) arrays
        # Calculate the features for each audio
            clip, as the log of the
        # Fourier Transform of the audio
        features = [self.featurize(a) for a in
            audio_paths]
        input_lengths = [f.shape[0] for f in
            features]
        max_length = max(input_lengths)
        feature_dim = features[0].shape[1]
        mb_size = len(features)
        # Pad all the inputs so that they are
            all the same length
        x = np.zeros((mb_size, max_length,
            feature_dim))
        y = []
        label_lengths = []
        for i in range(mb_size):
            feat = features[i]
            feat = self.normalize(feat)  #
                Center using means and std
            x[i, :feat.shape[0], :] = feat
            label = text_to_int_sequence(texts[i
                ])
            y.append(label)
            label_lengths.append(len(label))
        # Flatten labels to comply with warp-CTC
            signature
        y = reduce(lambda i, j: i + j, y)
        return {
            'x': x,  # (0-padded features of
                shape(mb_size,timesteps,
                feat_dim)
            'y': y,  # list(int) Flattened
                labels (integer sequences)
            'texts': texts,  # list(str)
                Original texts
            'input_lengths': input_lengths,  #
                list(int) Length of each input
            'label_lengths': label_lengths  #
                list(int) Length of each label
        }

    def iterate(self, audio_paths, texts,
        minibatch_size,
                source, max_iters=None):
        if max_iters is not None:
            k_iters = max_iters
        else:
            k_iters = int(np.ceil(len(
                audio_paths) / minibatch_size))

        #Logger.append("Iters: {}".format(
            k_iters))
        if source is "tr" or source is "te":
                ProgressPanel.set_maximum(
                    k_iters)

        pool = ThreadPoolExecutor(1)  # Run a
            single I/O thread in parallel
        future = pool.submit(self.
            prepare_minibatch,
                             audio_paths[:
                                 minibatch_size
                                 ],
                             texts[:
                                 minibatch_size
                                 ])
        start = minibatch_size
        for i in range(k_iters - 1):
            wait([future])
            minibatch = future.result()
            # While the current minibatch is
                being consumed, prepare the
                next
            future = pool.submit(self.
                prepare_minibatch,
                                 audio_paths[
                                     start:
                                     start +
                                     minibatch_size
                                     ],
                                 texts[start:
                                     start +
                                     minibatch_size
                                     ])

            yield minibatch
            start += minibatch_size
        # Wait on the last minibatch
        wait([future])
        minibatch = future.result()
        yield minibatch

    def iterate_train(self, minibatch_size=16,
        sort_by_duration=False,
                      shuffle=True):
        if sort_by_duration and shuffle:
            shuffle = False
            Logger.append("Both sort_by_duration
                and shuffle were set to True.
                "
                          "Setting shuffle to
                              False")
        durations, audio_paths, texts = (self.
            train_durations,
                                         self.
                                             train_audio_paths
                                             ,
                                         self.
                                             train_texts
                                             )
        if shuffle:
            temp = zip(durations, audio_paths,
                texts)
            self.rng.shuffle(temp)
            durations, audio_paths, texts = zip
                (*temp)
        if sort_by_duration:
            durations, audio_paths, texts =\
                DataGenerator.sort_by_duration(
                    durations, audio_paths,
                    texts)
        return self.iterate(audio_paths, texts,
            minibatch_size, source="tr")

    def iterate_test(self, minibatch_size=16):
        return self.iterate(self.
            test_audio_paths, self.test_texts,
                            minibatch_size,
                                source="te")

    def iterate_validation(self, minibatch_size
        =16):
        return self.iterate(self.val_audio_paths
            , self.val_texts,
                            minibatch_size,
                                source="va")

    def fit_train(self, k_samples=100):
        """ Estimate the mean and std of the
            features from the training set
        Params:
            k_samples (int): Use this number of
                samples for estimation
        """
        k_samples = min(k_samples, len(self.
            train_audio_paths))
        samples = self.rng.sample(self.
            train_audio_paths, k_samples)
        feats = [self.featurize(s) for s in
            samples]
        feats = np.vstack(feats)
        self.feats_mean = np.mean(feats, axis=0)
        self.feats_std = np.std(feats, axis=0)
```

File: trainer/code/model.py

```
"""
Define functions used to construct a multilayer
    GRU CTC model, and
```

```
functions for training and testing it.
"""

import ctc
import logging
import keras.backend as K

from keras.layers import (BatchNormalization,
     Convolution1D, Dense,
                          Input, GRU,
                               TimeDistributed,
                               LSTM,
                               Bidirectional)
from keras.models import Model
from keras.optimizers import SGD, Adadelta
# import lasagne

from utils import conv_output_length


def compile_train_fn(model, learning_rate=2e-4):
    """ Build the CTC training routine for
         speech models.
    Args:
        model: A keras model (built=True)
             instance
    Returns:
        train_fn (theano.function): Function
             that takes in acoustic inputs,
            and updates the model. Returns
                 network outputs and ctc cost
    """
    acoustic_input = model.inputs[0]
    network_output = model.outputs[0]
    output_lens = K.placeholder(ndim=1, dtype='
         int32')
    label = K.placeholder(ndim=1, dtype='int32')
    label_lens = K.placeholder(ndim=1, dtype='
         int32')
    network_output = network_output.dimshuffle
         ((1, 0, 2))

    ctc_cost = ctc.cpu_ctc_th(network_output,
         output_lens, label, label_lens).mean()

    trainable_vars = model.trainable_weights
    optimizer = SGD(nesterov=True, lr=
         learning_rate, momentum=0.9, clipnorm
         =100)
    #optimizer = Adadelta(lr=learning_rate)
    updates = optimizer.get_updates(
         trainable_vars, [], ctc_cost)
    #trainable_vars = model.trainable_weights
    #grads = K.gradients(ctc_cost,
         trainable_vars)
    #grads = lasagne.updates.
         total_norm_constraint(grads, 100)
    #updates = lasagne.updates.nesterov_momentum
         (grads, trainable_vars,
    #
         learning_rate, 0.99)
    train_fn = K.function([acoustic_input,
         output_lens, label, label_lens,
                           K.learning_phase()],
                          [network_output,
                               ctc_cost],
                          updates=updates)
    return train_fn


def compile_test_fn(model):
    """ Build a testing routine for speech
         models.
    Args:
        model: A keras model (built=True)
             instance
    Returns:
        val_fn (theano.function): Function that
             takes in acoustic inputs,
            and calculates the loss. Returns
                 network outputs and ctc cost
    """
    acoustic_input = model.inputs[0]
    network_output = model.outputs[0]
    output_lens = K.placeholder(ndim=1, dtype='
         int32')
    label = K.placeholder(ndim=1, dtype='int32')
    label_lens = K.placeholder(ndim=1, dtype='
         int32')
    network_output = network_output.dimshuffle
         ((1, 0, 2))
    ctc_cost = ctc.cpu_ctc_th(network_output,
         output_lens,
                              label, label_lens)
                                   .mean()
    val_fn = K.function([acoustic_input,
         output_lens, label, label_lens,
                         K.learning_phase()],
                         [network_output,
                              ctc_cost])
    return val_fn


def compile_output_fn(model):
    """ Build a function that simply calculates
         the output of a model
    Args:
        model: A keras model (built=True)
             instance
    Returns:
        output_fn (theano.function): Function
             that takes in acoustic inputs,
            and returns network outputs
    """
    acoustic_input = model.inputs[0]
    network_output = model.outputs[0]
    network_output = network_output.dimshuffle
         ((1, 0, 2))

    output_fn = K.function([acoustic_input, K.
         learning_phase()],
                           [network_output])
    return output_fn


def compile_gru_model(input_dim=161, output_dim
    =29, recur_layers=3, nodes=1024,
                      conv_context=11,
                           conv_border_mode='
                           valid', conv_stride
                           =2,
                      initialization='
                           glorot_uniform',
                           batch_norm=True,
                           archi="GRU"):
    """ Build a recurrent network (CTC) for
         speech with GRU units """

    print("Building RNN with {} layers and {}
         nodes".format(recur_layers, nodes))
    # Main acoustic input
    acoustic_input = Input(shape=(None,
         input_dim), name='acoustic_input')

    # Setup the network
    #conv_1d = Convolution1D(nodes, conv_context
         , name='conv1d',
    #                        border_mode=
         conv_border_mode,
    #                        subsample_length=
         conv_stride, init=initialization,
    #                        activation='relu')(
         acoustic_input)

    conv_1d = acoustic_input
    if batch_norm:
        #output = BatchNormalization(name='
             bn_conv_1d', mode=2)(conv_1d)
        output = BatchNormalization(name="
             bn_conv_1d")(conv_1d)
    else:
        output = conv_1d


    # output = acoustic_input
    for r in range(recur_layers):
        output = GRU(nodes, activation='relu',
                     name='rnn_{}'.format(r + 1),
                          init=initialization,
                     return_sequences=True)(output)

        if batch_norm:
            #bn_layer = BatchNormalization(name
                 ='bn_rnn_{}'.format(r + 1),
                 mode=2)
            bn_layer = BatchNormalization(name="
                 bn_rnn_{}".format(r + 1))
            output = bn_layer(output)

    # We don't softmax here because CTC does
         that
    network_output = TimeDistributed(Dense(
         output_dim, name='dense', activation='
```

```
            linear', init=initialization,
    ))(output)
    model = Model(input=acoustic_input, output=
        network_output)

    model.conv_output_length = lambda x: x #
        conv_output_length(x, conv_context,
        conv_border_mode, conv_stride)

    # model.conv_output_length = lambda x: x
    return model
```

File: trainer/code/split.py

```
import os
import random

import gui.ProgressPanel as ProgressPanel
import gui.ActionManager as ActionManager
import gui.Logger as Logger

def split_json(args):
        ProgressPanel.set_maximum(3)
        ProgressPanel.set_status_message("
            Processing files...")

        json_file = args["json_file"]
        ratios = args["ratios"]

        # Absolute file path of the JSON to be
            split
        filepath = json_file.rstrip()

        # Directory where the JSON to be split
            is
        directory = os.path.split(json_file)[0]

        # Template for the filename format
        template = "{}-{}.json"
        filename_all = os.path.splitext(filepath
            )[0]

        # Construct the filenames of the split
            files
        filename_train = os.path.join(directory,
             template.format(filename_all, "
            train"))
        filename_valid = os.path.join(directory,
             template.format(filename_all, "
            valid"))
        filename_test  = os.path.join(directory,
             template.format(filename_all, "
            test"))

        # Load contents in array
        data_all = []
        with open(filepath, "r") as handle_all:
            # Store all the data in a list
            data_all = handle_all.read().
                split("\n")
        size_all = len(data_all)

        # Shuffle the contents
        random.shuffle(data_all)

        # Split into the proper ratios
        size_train = int((float(ratios[0]) /
            100) * size_all)
        size_valid = int((float(ratios[1]) /
            100) * size_all)
        size_test  = int((float(ratios[2]) /
            100) * size_all)

        ProgressPanel.set_current_value(1)
        ProgressPanel.set_status_message("{}/{}
            split files created".format(1, 3))

        # Write each to file
        with open(filename_train, "w") as
            handle_train:
                for line in data_all[0:
                    size_train]:
                        handle_train.write(line
                            + "\n")

        ProgressPanel.set_current_value(2)
        ProgressPanel.set_status_message("{}/{}
            split files created".format(2, 3))

        with open(filename_valid, "w") as
            handle_valid:
                for line in data_all[size_train:
                    size_train + size_valid]:
                        handle_valid.write(line
                            + "\n")

        ProgressPanel.set_current_value(3)
        ProgressPanel.set_status_message("{}/{}
            split files created".format(3, 3))

        with open(filename_test, "w") as
            handle_test:
                for line in data_all[size_train
                    + size_valid:]:
                        handle_test.write(line +
                            "\n")

        # Log current results
        ProgressPanel.set_idle()
        ActionManager.markAsDone()
        Logger.append("(Dataset) The split JSON
            files were saved in {}".format(
            directory))
        Logger.append("(Dataset) Training JSON:
            {}".format(size_train))
        Logger.append("(Dataset) Validation JSON
            : {}".format(size_valid))
        Logger.append("(Dataset) Test JSON: {}".
            format(size_test))
```

File: trainer/code/test.py

```
"""
Test a trained speech model over a dataset
"""

from __future__ import absolute_import, division
    , print_function
from Levenshtein import distance
import argparse
import numpy as np

from core.data_generator import DataGenerator
from core.model import compile_test_fn
from core.utils import argmax_decode,
    conv_output_length, load_model,
    load_hyperparams

import gui.Logger as Logger
import gui.ActionManager as ActionManager
import gui.ProgressPanel as ProgressPanel


def _is_locked():
        global lock
        if not lock.isLocked():
                Logger.append("Stopped testing")
                ProgressPanel.set_idle()
                ActionManager.markAsDone()
                return False
        return True

def _test(model, test_fn, datagen, mb_size=16,
    conv_context=11,
                        conv_border_mode='valid
                            ', conv_stride=2):
        """ Testing routine for speech-models
        Params:
                model (keras.model): Constructed
                     keras model
                test_fn (theano.function): A
                    theano function that
                    calculates the cost over a
                    test set
                datagen (DataGenerator)
                mb_size (int): Size of each
                    minibatch
                conv_context (int): Convolution
                    context
                conv_border_mode (str):
                    Convolution border mode
                conv_stride (int): Convolution
                    stride
        Returns:
                test_cost (float): Average test
                    cost over the whole test
                    set
                """

        global lock

        total_cer = 0.0
        total_count = 0
        ProgressPanel.set_status_message("
            Ongoing testing of the model")
```

```
        for i, batch in enumerate(datagen.
            iterate_test(mb_size)):

            # Abort testing immediately
            if not _is_locked():
                return

            ProgressPanel.set_current_value(
                i)

            inputs = batch['x']
            labels = batch['y']
            input_lengths = batch['
                input_lengths']
            label_lengths = batch['
                label_lengths']
            ground_truth = batch['texts']

            # Due to convolution, the number
                of timesteps of the output
            # is different from the input
                length. Calculate the
                resulting
            # timesteps
            output_lengths = [
                conv_output_length(l,
                conv_context,
```

```
            predictions, ctc_cost = test_fn
                ([inputs, output_lengths,
                labels, label_lengths, True
                ])
            predictions = np.swapaxes(
                predictions, 0, 1)

            for j, prediction in enumerate(
                predictions):

                current_pred =
                    argmax_decode(
                    prediction)
                current_truth =
                    ground_truth[j].
                    encode("utf8")
                cer = distance(
                    current_truth,
                    current_pred)

                line = "CER: {}\nTruth:
                    {}\nPrediction: {}\
                    n".format(cer,
                    current_truth,
                    current_pred)
                Logger.append(line)

                total_count =
                    total_count + 1
                total_cer = total_cer +
                    cer

    if total_count == 0:
            return 0.0
    return (total_cer / total_count)


# test_desc_file, train_desc_file, load_dir
def start_test(args, shared_lock):
        # Lock
        global lock
        lock = shared_lock
        lock.lock()

        Logger.set_directory(args["model_dir"],
            "test_log.txt")

        # Prepare the data generator
        datagen = DataGenerator()
```

```
        # Load the JSON file that contains the
            dataset
        Logger.append("Loading testing dataset")
        datagen.load_train_data(args["train_json
            "])
        datagen.load_test_data(args["test_json
            "])

        # Use a few samples from the dataset, to
            calculate the means and variance
        # of the features, so that we can center
            our inputs to the network
        datagen.fit_train(100)

        # Compile a Recurrent Network with 1 1D
            convolution layer, GRU units
        # and 1 fully connected layer
        Logger.append("Compiling the RNN")
        model = load_model(args["model_dir"])

        if not _is_locked():
            return

        # Compile the testing function
        Logger.append("Compiling the testing
            function")
        test_fn = compile_test_fn(model)

        if not _is_locked():
            return
        # Test the model
        test_loss = _test(model, test_fn,
            datagen)

        # Log the calculated test loss
        ProgressPanel.set_idle()
        ActionManager.markAsDone()

        if(not test_loss == None):
            Logger.append("Test CER: {}".
                format(test_loss))
```

File: trainer/code/train.py

```
"""
Train an end-to-end speech recognition model
    using CTC.
Use $python train.py --help for usage
"""

from __future__ import absolute_import, division
    , print_function

import argparse
import os
import sys
import threading
import Tkinter as tk

from core.data_generator import DataGenerator
from core.model import compile_gru_model,
    compile_train_fn, compile_test_fn
from core.utils import save_model, load_model,
    load_costs, Lock
from core.char_map import index_map

import gui.Plotter as Plotter
import gui.Logger as Logger
import gui.ProgressPanel as ProgressPanel
import gui.ActionManager as ActionManager

        # args:
        #       lrate           learning rate
        #       mb_size         minibatch size
        #       epochs          number of epochs
        #       archi           architecture
        #       sortagrad       sortagrad
        #       train_json      train dataset
            json file
        #       valid_json      validation
            dataset json file
        #       save_dir        save model
            directory

def _validation(mb_size, datagen, model, val_fn)
    :
        """ Validation routine for speech-models
        Params:
            model (keras.model): Constructed
                keras model
```

```
            val_fn (theano.function): A theano
                 function that calculates the
                 cost
                over a validation set
            datagen (DataGenerator)
            mb_size (int): Size of each
                 minibatch
        Returns:
            val_cost (float): Average validation
                 cost over the whole validation
                 set
        """

        avg_cost = 0.0
        j = 0
        for i, batch in enumerate(datagen.
             iterate_validation(mb_size)):
            if not lock.isLocked():
                    if i == 0:
                            return 0.0
                    return avg_cost / i

            ProgressPanel.set_status_message
                 ("Ongoing validation for
                 current epoch")

            inputs = batch['x']
            labels = batch['y']
            input_lengths = batch['
                 input_lengths']
            label_lengths = batch['
                 label_lengths']
            # Due to convolution, the number
                  of timesteps of the output
            # is different from the input
                 length. Calculate the
                 resulting
            # timesteps
            output_lengths = [model.
                 conv_output_length(l) for l
                  in input_lengths]
            output_lengths = [l for l in
                 input_lengths]
            _, ctc_cost = val_fn([inputs,
                 output_lengths, labels,
                 label_lengths, True])

            if i % 10 == 0:
                    Logger.append("(
                         Validating)
                         Iteration: {}, Loss
                         : {}".format(i,
                         ctc_cost))
            avg_cost += ctc_cost
            j += 1

        if j == 0:
                return 0.0

        print (avg_cost / j)
        return (avg_cost / j)

def _is_locked():
        global lock
        if not lock.isLocked():
                Logger.append("Stopped training
                     ")
                ProgressPanel.set_idle()
                Plotter.close_graph()
                ActionManager.markAsDone()
                return False
        return True

def start_train2(args, shared_lock):
        Logger.set_directory(args["save_dir"])

        # Plot window
        Plotter.initialize(args["save_dir"])

        # Lock
        global lock
        lock = shared_lock
        lock.lock()

        # Prepare the data generator
        datagen = DataGenerator()

        # Load the JSON file that contains the
             dataset
        Logger.append("Loading training dataset
             ")
        datagen.load_train_data(args["train_json
             "])

        Logger.append("Loading validation
             dataset")
        datagen.load_validation_data(args["
             valid_json"])

        if (not _is_locked()):
                return

        # Use a few samples from the dataset, to
              calculate the means and variance
        # of the features, so that we can center
              our inputs to the network
        datagen.fit_train(100)

        # Compile a Recurrent Network with 1 1D
             convolution layer, GRU units
        # and 1 fully connected layer
        Logger.append("Compiling the RNN")
        model = load_model(args["save_dir"])

        if (not _is_locked()):
                return

        train_costs, val_costs = load_costs(args
             ["save_dir"])

        if (not _is_locked()):
                return

        # Compile the CTC training function
        Logger.append("Compiling the training
             function")
        train_fn = compile_train_fn(model, args
             ["lrate"])

        if (not _is_locked()):
                return

        # Compile the validation function
        Logger.append("Compiling the validation
             function")
        val_fn = compile_test_fn(model)

        if (not _is_locked()):
                return

        # Start training
        _train(args, datagen, model, train_fn,
             val_fn, do_sortagrad=False,
             train_costs=train_costs, val_costs=
             val_costs)


def start_train(args, shared_lock):
        Logger.set_directory(args["save_dir"])

        # Plot window
        Plotter.initialize(args["save_dir"])

        # Lock
        global lock
        lock = shared_lock
        lock.lock()

        if not os.path.exists(args["save_dir"]):
            os.makedirs(args["save_dir"])

        # Configure logging
        #configure_logging(file_log_path=os.path
             .join(args["save_dir"], 'train_log.
             txt'))

        # Prepare the data generator
        datagen = DataGenerator()

        # Load the JSON file that contains the
             dataset
        Logger.append("Loading training dataset
             ")
        datagen.load_train_data(args["train_json
             "])

        Logger.append("Loading validation
             dataset")
        datagen.load_validation_data(args["
             valid_json"])

        if (not _is_locked()):
                return
```

```
        # Use a few samples from the dataset, to
              calculate the means and variance
        # of the features, so that we can center
              our inputs to the network
        datagen.fit_train(100)

        # Compile a Recurrent Network with 1 1D
             convolution layer, GRU units
        # and 1 fully connected layer
        Logger.append("Compiling the RNN")
        model = compile_gru_model(recur_layers
             =3, nodes=512, batch_norm=True)

        if (not _is_locked()):
                return

        # Compile the CTC training function
        Logger.append("Compiling the training
             function")
        train_fn = compile_train_fn(model, args
             ["lrate"])

        if (not _is_locked()):
                return

        # Compile the validation function
        Logger.append("Compiling the validation
             function")
        val_fn = compile_test_fn(model)

        if (not _is_locked()):
                return

        # Start training
        _train(args, datagen, model, train_fn,
             val_fn)


def _train(args, datagen, model, train_fn,
     val_fn, do_sortagrad=True, train_costs=[],
     val_costs=[]):
        """ Main training routine for speech-
             models
        Params:
            model (keras.model): Constructed
                 keras model
            train_fn (theano.function): A theano
                 function that takes in
                 acoustic
                inputs and updates the model
            val_fn (theano.function): A theano
                 function that calculates the
                 cost
                over a validation set
            datagen (DataGenerator)
            save_dir (str): Path where model and
                 costs are saved
            epochs (int): Total epochs to
                 continue training
            mb_size (int): Size of each
                 minibatch
            do_sortagrad (bool): If true, we
                 sort utterances by their length
                 in the
                first epoch
        """

        save_dir = args["save_dir"]
        epochs = args["epochs"]
        mb_size = args["mb_size"]

        iters = 0

        ProgressPanel.set_maximum(epochs)
        for e in range(epochs):

                if do_sortagrad:
                        shuffle = (e != 0)
                        sortagrad = (e == 0)
                else:
                        shuffle = True
                        sortagrad = False

                ProgressPanel.set_status_message
                     ("Ongoing training for
                     current epoch {}".format(e
                     + 1))
                for i, batch in enumerate(
                     datagen.iterate_train(
                     mb_size, shuffle=shuffle,
                     sort_by_duration=sortagrad)
                     ):

                        # Abort training
                             immediately
                        if (not _is_locked()):
                                return

                        ProgressPanel.
                             set_current_value(i
                             )

                        inputs = batch['x']
                        labels = batch['y']
                        input_lengths = batch['
                             input_lengths']
                        label_lengths = batch['
                             label_lengths']
                        # Due to convolution,
                             the number of
                             timesteps of the
                             output
                        # is different from the
                             input length.
                             Calculate the
                             resulting
                        # timesteps
                        output_lengths = [model.
                             conv_output_length(
                             l) for l in
                             input_lengths]

                        _, ctc_cost = train_fn([
                             inputs,
                             output_lengths,
                             labels,
                             label_lengths, True
                             ])
                        train_costs.append(
                             ctc_cost)

                        if i % 10 == 0:
                                Logger.append("(
                                     Training)
                                     Epoch: {},
                                     Iteration:
                                     {}, Loss:
                                     {}".format(
                                     e + 1, i,
                                     ctc_cost))
                        iters += 1

                # End of an epoch. Check
                     validation cost and save
                     costsS
                val_cost = _validation(mb_size,
                     datagen, model, val_fn)
                val_costs.append(val_cost)
                save_model(save_dir, model,
                     train_costs, val_costs,
                     iters, args)

                Plotter.update_graph()

        ProgressPanel.set_idle()
        ActionManager.markAsDone()
        Plotter.close_graph()
        Logger.append("Training done")
```

File: trainer/code/utils.py

```
import glob
import logging
import os
import numpy as np
import re
import soundfile
import json
from keras.models import model_from_json
from numpy.lib.stride_tricks import as_strided

from core.char_map import char_map, index_map
import gui.Logger as Logger

#logger = logging.getLogger(__name__)


class Lock():
        def __init__(self):
                self.status = True

        def unlock(self):
                self.status = False

        def lock(self):
```

```
                self.status = True

        def isLocked(self):
                return self.status

def calc_feat_dim(window, max_freq):
    return int(0.001 * window * max_freq) + 1


def conv_output_length(input_length, filter_size
    , border_mode, stride,
                       dilation=1):
    """ Compute the length of the output
        sequence after 1D convolution along
        time. Note that this function is in line
             with the function used in
        Convolution1D class from Keras.
    Params:
        input_length (int): Length of the input
            sequence.
        filter_size (int): Width of the
            convolution kernel.
        border_mode (str): Only support `same`
            or `valid`.
        stride (int): Stride size used in 1D
            convolution.
        dilation (int)
    """
    if input_length is None:
        return None
    assert border_mode in {'same', 'valid'}
    dilated_filter_size = filter_size + (
        filter_size - 1) * (dilation - 1)
    if border_mode == 'same':
        output_length = input_length
    elif border_mode == 'valid':
        output_length = input_length -
            dilated_filter_size + 1
    return (output_length + stride - 1) //
        stride


def spectrogram(samples, fft_length=256,
    sample_rate=2, hop_length=128):
    """
    Compute the spectrogram for a real signal.
    The parameters follow the naming convention
        of
    matplotlib.mlab.specgram

    Args:
        samples (1D array): input audio signal
        fft_length (int): number of elements in
            fft window
        sample_rate (scalar): sample rate
        hop_length (int): hop length (relative
            offset between neighboring
            fft windows).

    Returns:
        x (2D array): spectrogram [frequency x
            time]
        freq (1D array): frequency of each row
            in x

    Note:
        This is a truncating computation e.g. if
             fft_length=10,
        hop_length=5 and the signal has 23
            elements, then the
        last 3 elements will be truncated.
    """
    assert not np.iscomplexobj(samples), "Must
        not pass in complex numbers"

    window = np.hanning(fft_length)[:, None]
    window_norm = np.sum(window**2)

    # The scaling below follows the convention
        of
    # matplotlib.mlab.specgram which is the same
        as
    # matlabs specgram.
    scale = window_norm * sample_rate

    trunc = (len(samples) - fft_length) %
        hop_length
    x = samples[:len(samples) - trunc]

    # "stride trick" reshape to include overlap
    nshape = (fft_length, (len(x) - fft_length)
        // hop_length + 1)
    nstrides = (x.strides[0], x.strides[0] *
        hop_length)
    x = as_strided(x, shape=nshape, strides=
        nstrides)

    # window stride sanity check
    assert np.all(x[:, 1] == samples[hop_length
        :(hop_length + fft_length)])

    # broadcast window, compute fft over columns
         and square mod
    x = np.fft.rfft(x * window, axis=0)
    x = np.absolute(x)**2

    # scale, 2.0 for everything except dc and
        fft_length/2
    x[1:-1, :] *= (2.0 / scale)
    x[(0, -1), :] /= scale

    freqs = float(sample_rate) / fft_length * np
        .arange(x.shape[0])

    return x, freqs


def spectrogram_from_file(filename, step=10,
    window=20, max_freq=None,
                              eps=1e-14):
    """ Calculate the log of linear spectrogram
        from FFT energy
    Params:
        filename (str): Path to the audio file
        step (int): Step size in milliseconds
            between windows
        window (int): FFT window size in
            milliseconds
        max_freq (int): Only FFT bins
            corresponding to frequencies
            between
            [0, max_freq] are returned
        eps (float): Small value to ensure
            numerical stability (for ln(x))
    """
    with soundfile.SoundFile(filename) as
        sound_file:
        audio = sound_file.read(dtype='float32')
        sample_rate = sound_file.samplerate
        if audio.ndim >= 2:
            audio = np.mean(audio, 1)
        if max_freq is None:
            max_freq = sample_rate / 2
        if max_freq > sample_rate / 2:
            raise ValueError("max_freq must not
                be greater than half of "
                             " sample rate")
        if step > window:
            raise ValueError("step size must not
                be greater than window size")
        hop_length = int(0.001 * step *
            sample_rate)
        fft_length = int(0.001 * window *
            sample_rate)
        pxx, freqs = spectrogram(
            audio, fft_length=fft_length,
                sample_rate=sample_rate,
            hop_length=hop_length)
        ind = np.where(freqs <= max_freq)[0][-1]
            + 1
    return np.transpose(np.log(pxx[:ind, :] +
        eps))


def save_model(save_dir, model, train_costs,
    validation_costs, index=None, args=None):
    """ Save the model and costs into a
        directory
    Params:
        save_dir (str): Directory used to store
            the model
        model (keras.models.Model)
        train_costs (list(float))
        validation_costs (list(float))
        index (int): If this is provided, add
            this index as a suffix to
            the weights (useful for
                checkpointing during training)
    """
    Logger.append("(Train) Checkpointing model
        to: {}".format(save_dir))
    model_config_path = os.path.join(save_dir, '
        model_config.json')
    with open(model_config_path, 'w') as
```

```
            model_config_file:
            model_json = model.to_json()
            model_config_file.write(model_json)

    hyperparams_path = os.path.join(save_dir, "
        hyperparams.json")
    if not args == None:
        with open(hyperparams_path, "w") as
            hyperparams_file:
            line = json.dumps({"lrate" : args["
                lrate"], "epochs" : args["
                epochs"], "mb_size" : args["
                mb_size"]})
            hyperparams_file.write(line)

    if index is None:
        weights_format = 'model_weights.h5'
    else:
        weights_format = 'model_{}_weights.h5'.
            format(index)
    model_weights_file = os.path.join(save_dir,
        weights_format)
    model.save_weights(model_weights_file,
        overwrite=True)

    np.savez(os.path.join(save_dir, 'costs.npz')
        , train=train_costs,
            validation=validation_costs)


def load_hyperparams(load_dir):
    if (load_model(load_dir, check=True)):
        hyperparams_path = os.path.join(load_dir
            , "hyperparams.json")

        hyperparams = None
        with open(hyperparams_path, "r") as
            hyperparams_file:
            line = hyperparams_file.readline()
            hyperparams = json.loads(line)

        return hyperparams

def load_costs(load_dir):
    path = os.path.join(load_dir, 'costs.npz')

    if (not os.path.isfile(path)):
        return ([], [])

    try:
        # Load npz file for the costs per
            iteration
        costs = np.load(path)
    except Exception as e:
        return ([], [])
    return (costs['train'].tolist(), costs['
        validation'].tolist())

def load_model(load_dir, weights_file=None,
    check=False):

    """ Load a model and its weights from a
        directory
    Params:
        load_dir (str): Path the model directory
        weights_file (str): If this is not
            passed in, try to load the latest
            model_*weights.h5 file in the
                directory
    Returns:
        model (keras.models.Model)
    """
    def atoi(text):
        return int(text) if text.isdigit() else
            text

    def natural_keys(text):
        # From http://stackoverflow.com/
            questions/5967500
        return [atoi(c) for c in re.split('(\d+)
            ', text)]

    try:
        model_config_file = os.path.join(
            load_dir, 'model_config.json')
        model_config = open(model_config_file).
            read()
        model = model_from_json(model_config)

        if weights_file is None:
            # This will find all files of name
                model_*weights.h5
            # We try to use the latest one saved
            weights_files = glob.glob(os.path.
                join(load_dir, 'model_*weights.
                h5'))
            weights_files.sort(key=natural_keys)
            model_weights_file = weights_files
                [-1]  # Use the latest model
        else:
            model_weights_file = weights_file

        model.load_weights(model_weights_file)
        if check:
            return True

        model.conv_output_length = lambda x:
            conv_output_length(x, 11, "valid",
            2)
        return model

    except Exception as e:
        if check:
            return False
        return None

def argmax_decode(prediction):
    """ Decode a prediction using the highest
        probable character at each
        timestep. Then, simply convert the
            integer sequence to text
    Params:
        prediction (np.array): timestep *
            num_characters
    """
    int_sequence = []
    for timestep in prediction:
        int_sequence.append(np.argmax(timestep))
    tokens = []
    c_prev = -1
    for c in int_sequence:
        if c == c_prev:
            continue
        if c != 0:  # Blank
            tokens.append(c)
        c_prev = c

    text = ''.join([index_map[i] for i in tokens
        ])
    return text


def text_to_int_sequence(text):
    """ Use a character map and convert text to
        an integer sequence """
    int_sequence = []
    for c in text:
        if c == ' ':
            ch = char_map['<SPACE>']
        else:
            ch = char_map[c]
        int_sequence.append(ch)
    return int_sequence
```

File: trainer/gui/Action.py

```
class Action():
        CREATE_JSON = 1
        SPLIT_JSON = 2
        TRAIN_MODEL = 3
        TEST_MODEL = 4
        CONTINUE_TRAIN_MODEL = 5
        STOP_TRAIN_MODEL = 6
        STOP_TEST_MODEL = 7

        def __init__(self, type, args):
                self.type = type
                self.args = args

        def getType(self):
                return self.type

        def getArgs(self):
                return self.args
```

File: trainer/gui/ActionManager.py

```
from core.utils import Lock
from gui.Action import Action


import threading

global IDLE, RUNNING
```

```
    global current_state, current_action

    # Constants
    IDLE = 1
    RUNNING = 2

    current_state = IDLE
    current_action = None
    interrupt_lock = None

    def setAction(action):
            global IDLE, RUNNING
            global current_state

            if (_isActionInterrupt(action)):
                    global interrupt_lock
                    if (interrupt_lock == None):
                            return

                    # Interrupt currently
                         interruptible action
                    interrupt_lock.unlock()
                    return

            if (current_state == RUNNING):
                    return

            # Set current action and state
            _startAction(action)

    def _isActionInterrupt(action):
            type = action.getType()
            return (type == Action.STOP_TRAIN_MODEL
                 or
                            type == Action.
                                 STOP_TEST_MODEL)

    def _isActionInterruptible(action):
            type = action.getType()
            return (type == Action.TRAIN_MODEL or
                            type == Action.
                                 CONTINUE_TRAIN_MODEL
                                   or
                            type == Action.
                                 TEST_MODEL)

    def _startAction(action):
            from core.create import create_json
            from core.split import split_json
            from core.train import start_train,
                 start_train2
            from core.test import start_test

            type = action.getType()
            args = action.getArgs()

            # Action thread
            th = None

            if (not _isActionInterruptible(action)):
                    if (type == Action.CREATE_JSON):
                            th = threading.Thread(
                                 target=create_json,
                                   args=(args, ))

                    elif (type == Action.SPLIT_JSON)
                         :
                            th = threading.Thread(
                                 target=split_json,
                                 args=(args, ))

            else:   # Interruptible
                    global interrupt_lock
                    interrupt_lock = Lock()
                    interrupt_lock.lock()

                    if (type == Action.TRAIN_MODEL):
                            th = threading.Thread(
                                 target=start_train,
                                   args=(args,
                                 interrupt_lock))

                    elif (type == Action.TEST_MODEL)
                         :
                            th = threading.Thread(
                                 target=start_test,
                                 args=(args,
                                 interrupt_lock))

                    elif (type == Action.
                         CONTINUE_TRAIN_MODEL):
                            th = threading.Thread(
                                 target=start_train2
                                 , args=(args,
                                 interrupt_lock))

            # Invalid type
            if (th == None):
                    return

            # Start action
            global current_state
            current_state = RUNNING
            th.start()

            # Disable buttons
            _disableButtons(type)

    def markAsDone():
            global IDLE
            global current_state
            global interrupt_lock

            interrupt_lock = None
            current_state = IDLE

            _enableButtons()

    def _disableButtons(type):
            import gui.TrainingPanel as
                 TrainingPanel
            import gui.TestingPanel as TestingPanel
            import gui.DatasetPanel as DatasetPanel

            TrainingPanel.disable_buttons(type)
            TestingPanel.disable_buttons(type)
            DatasetPanel.disable_buttons(type)

    def _enableButtons():
            import gui.TrainingPanel as
                 TrainingPanel
            import gui.TestingPanel as TestingPanel
            import gui.DatasetPanel as DatasetPanel

            TrainingPanel.enable_buttons()
            TestingPanel.enable_buttons()
            DatasetPanel.enable_buttons()
```

File: trainer/gui/DatasetPanel.py

```
import ttk
import Tkinter as tk
import tkMessageBox
import tkFileDialog

import os
import random
import threading

import gui.Logger as Logger
import gui.ActionManager as ActionManager

from gui.Action import Action
from gui.DirectoryText import DirectoryText

def initialize(parent):
        global pane

        pane = tk.PanedWindow(parent, orient=tk.
             VERTICAL)

        _initialize_pane_create()

def get_widget():
        global pane
        return pane

def _initialize_pane_create():
        global pane
        global progress_bar
        global old_ratios
        global value_spin_train,
             value_spin_valid, value_spin_test

        # Create JSON pane
        frame_create_json = tk.LabelFrame(pane,
             text="Create JSON", height=300,
             padx=30, pady=10)

        # Container for the buttons
        label_datadir = tk.Label(
             frame_create_json, text="Dataset
             directory", anchor="w")
        pane_selecting = DirectoryText(
             frame_create_json)
```

```
#pane_selecting.setText("/media/iantot/
    Happy/ba-dls-deepspeech-fil/22
    DEC2016/ToUPMCS")

# Container for labels
pane_labels = tk.PanedWindow(
    frame_create_json)

# Subcontainers
pane_labels_l = tk.PanedWindow(
    pane_labels, width=270)
pane_labels_r = tk.PanedWindow(
    pane_labels)

# Labels
label_pattern = tk.Label(pane_labels_l,
    text="Transcript pattern", anchor="
    w")
label_file_ext = tk.Label(pane_labels_r,
     text="File ext.", anchor="w")

# Add labels to their subcontainers
pane_labels_l.add(label_pattern)
pane_labels_r.add(label_file_ext)

# Add subcontainers to the label
    container
pane_labels.add(pane_labels_l)
pane_labels.add(pane_labels_r)

# Container for text widgets
pane_texts = tk.PanedWindow(
    frame_create_json)

# Subcontainers
pane_texts_l = tk.PanedWindow(pane_texts
    , width=270)
pane_texts_r = tk.PanedWindow(pane_texts
    )

# Text widgets
text_pattern = tk.Text(pane_texts_l,
    height=1, width=20)
text_pattern.insert(tk.END, '(?P<key>[A-
    Za-z0-9.]{23}.wav) "(.*)" "(?P<text
    >.*)"')
text_file_ext = tk.Text(pane_texts_r,
    height=1)
text_file_ext.insert(tk.END, "log")

# Add widgets to the subcontainers
pane_texts_l.add(text_pattern)
pane_texts_r.add(text_file_ext)

# Add subcontainers to the widget
    container
pane_texts.add(pane_texts_l)
pane_texts.add(pane_texts_r)

label_ratio = tk.Label(frame_create_json
    , text="Dataset ratios", anchor="w
    ")

# Container for canvas
pane_spins = tk.PanedWindow(
    frame_create_json, width=165)

pane_spins_l = tk.PanedWindow(pane_spins
    , orient=tk.VERTICAL)
pane_spins_r = tk.PanedWindow(pane_spins
    , orient=tk.VERTICAL)

label_train = tk.Label(pane_spins_l,
    text="Train")
label_valid = tk.Label(pane_spins_l,
    text="Validation")
label_test = tk.Label(pane_spins_l, text
    ="Test")

# Spinners value monitors
old_ratios = [60, 20, 20]

value_spin_train = tk.IntVar()
value_spin_train.set(old_ratios[0])

value_spin_valid = tk.IntVar()
value_spin_valid.set(old_ratios[1])

value_spin_test = tk.IntVar()
value_spin_test.set(old_ratios[2])

# Spinners
spin_train = tk.Spinbox(pane_spins_r,
    from_=5, to=90, increment=5, width
    =5, state="readonly",
    readonlybackground="white",
        command=_trigger_spinbox_change,
            textvariable=
            value_spin_train)
spin_valid = tk.Spinbox(pane_spins_r,
    from_=5, to=90, increment=5, width
    =5, state="readonly",
    readonlybackground="white",
        command=_trigger_spinbox_change,
            textvariable=
            value_spin_valid)
spin_test = tk.Spinbox(pane_spins_r,
    from_=5, to=90, increment=5, width
    =5, state="readonly",
    readonlybackground="white",
        command=_trigger_spinbox_change,
            textvariable=
            value_spin_test)

pane_spins_l.add(label_train)
pane_spins_l.add(label_valid)
pane_spins_l.add(label_test)

pane_spins_r.add(spin_train)
pane_spins_r.add(spin_valid)
pane_spins_r.add(spin_test)

pane_spins.add(pane_spins_l)
pane_spins.add(pane_spins_r)

# Create JSON button
global button_create
button_create = tk.Button(
    frame_create_json, text="Create",
        command=lambda:
            _trigger_create_json(
            text_pattern.get("1.0", tk.
            END), text_file_ext.get
            ("1.0", tk.END),
            pane_selecting.getText()))

# Arrange UI elements in order
label_datadir.pack(fill="both")
pane_selecting.pack(fill="both")

pane_labels.pack(fill="both")
pane_texts.pack(fill="both")

label_ratio.pack(fill="both")
# pane_labels.pack()
pane_spins.pack()


button_create.pack()

frame_create_json.pack(fill="both",
    expand="yes")

def _trigger_spinbox_change():
    # Retrieve previous values
    global old_ratios
    global value_spin_train,
        value_spin_valid, value_spin_test

    # Get ratios
    current_ratio_train = value_spin_train.
        get()
    current_ratio_valid = value_spin_valid.
        get()
    current_ratio_test  = value_spin_test.
        get()

    old_ratio_train = old_ratios[0]
    old_ratio_valid = old_ratios[1]
    old_ratio_test  = old_ratios[2]

    # Get differences from old and current
        values
    diff_train = current_ratio_train -
        old_ratio_train
    diff_valid = current_ratio_valid -
        old_ratio_valid
    diff_test  = current_ratio_test  -
        old_ratio_test

    # Check if train spinbox has increased
        in value
    if (diff_train > 0):
```

```
            # If validation spinbox is
                larger than test spinbox
            if (current_ratio_valid >=
                current_ratio_test):

                # Decrease the
                    validation spinbox
                new_val = max(5,
                    current_ratio_valid
                    - diff_train)
                value_spin_valid.set(
                    new_val)

            # If test spinbox is larger than
                validation spinbox
            else:
                # Decrease the test
                    spinbox
                new_val = max(5,
                    current_ratio_test
                    - diff_train)
                value_spin_test.set(
                    new_val)

    # Check if train spinbox has decreased
        in value
    elif (diff_train < 0):

            # If validation spinbox is
                smaller than test spinbox
            if (current_ratio_valid <=
                current_ratio_test):
                # Increase the
                    validation spinbox
                new_val = min(95,
                    current_ratio_valid
                    - diff_train)
                value_spin_valid.set(
                    new_val)

            # If test spinbox is smaller
                than test spinbox
            else:
                # Increase the
                    validation spinbox
                new_val = min(95,
                    current_ratio_test
                    - diff_train)
                value_spin_test.set(
                    new_val)

    # Check if validation spinbox has
        increased in value
    elif (diff_valid > 0):

            # Decrease the test spinbox
            new_val = max(5,
                current_ratio_test -
                diff_valid)
            value_spin_test.set(new_val)

    # Check if validation spinbox has
        decreased in value
    elif (diff_valid < 0):

            # Decrease the test spinbox
            new_val = min(95,
                current_ratio_test -
                diff_valid)
            value_spin_test.set(new_val)

    # Check if test spinbox has increased in
        value
    elif (diff_test > 0):

            # Decrease the valid spinbox
            new_val = max(5,
                current_ratio_valid -
                diff_test)
            value_spin_valid.set(new_val)

    # Check if test spinbox has decreased in
        value
    elif (diff_test < 0):

            # Increase the valid spinbox
            new_val = min(95,
                current_ratio_valid -
                diff_test)
            value_spin_valid.set(new_val)

    # The actual current values
            current_ratio_train = value_spin_train.
                get()
            current_ratio_valid = value_spin_valid.
                get()
            current_ratio_test  = value_spin_test.
                get()

            # Check if for some reason things doesnt
                add up
            if (not (current_ratio_train +
                current_ratio_valid +
                current_ratio_test) == 100):

                # Revert the ratios
                value_spin_train.set(
                    old_ratio_train)
                value_spin_valid.set(
                    old_ratio_valid)
                value_spin_test.set(
                    old_ratio_test)

            else:
                # Update old ratios with the
                    current
                old_ratios = [
                    current_ratio_train,
                    current_ratio_valid,
                    current_ratio_test]

    def _trigger_create_json(pattern, file_ext,
        dataset_dir):
            global progress_bar
            global old_ratios

            # Get pattern for the transcripts
            pattern = pattern.rstrip()

            # Get file extension of the transcript
                files
            file_ext = file_ext.rstrip()

            # Get directory of the dataset
            dataset_dir = dataset_dir.rstrip()

            # Validate input
            if ((not pattern) or (not file_ext) or (
                not dataset_dir)):
                # Display warning if not all
                    fields are filled up
                tkMessageBox.showwarning(title="
                    Unfilled fields", message="
                    Please fill up all the
                    fields")
            else:
                # Create a new file at the
                    selected save location
                save_file = tkFileDialog.
                    asksaveasfilename(
                    initialdir=dataset_dir,
                        defaultextension=".json
                            ", filetypes=[("
                            JSON file", ".json
                            ")])

                if (save_file):
                        Logger.append("Creating
                            JSON file from
                            dataset")

                        args = {
                                "dataset_dir" :
                                    dataset_dir
                                    ,
                                "save_file"   :
                                    save_file,
                                "pattern"     :
                                    pattern,
                                "file_ext"    :
                                    file_ext,
                                "ratios"
                                    :
                                    old_ratios
                        }

                        action = Action(Action.
                            CREATE_JSON, args)
                        ActionManager.setAction(
                            action)

    def disable_buttons(type):
            global button_create
            global button_split
```

```
        button_create.config(state=tk.DISABLED)

def enable_buttons():
        global button_create

        button_create.config(state=tk.NORMAL)
```

File: trainer/gui/DirectoryText.py

```
import Tkinter as tk
import tkFileDialog

class DirectoryText(tk.PanedWindow):

        default_ext = "json"
        file_types = [("JSON file", ".json")]

        # type
        #       d               directory
        #       f               file
        #       nf              new file (write)

        def __init__(self, parent, callback=None
             , type="d"):
                tk.PanedWindow.__init__(self,
                     parent)

                # Type to filepath to browse
                self.type = type

                # Textfield for the directory
                self.path = tk.Text(self, height
                     =1, width=40)

                # Set as read only
                self.path.config(state=tk.
                     DISABLED)

                # Browse directory button
                button_browse = tk.Button(self,
                     text="...", command=self.
                     __browse)

                self.add(self.path)
                self.add(button_browse)

                self.open_model = callback

        def getText(self):
                return (self.path.get("1.0", tk.
                     END))

        def setText(self, text):
                # Enable text field
                self.path.config(state=tk.NORMAL
                     )

                if (self.path.get("1.0", tk.END)
                     ):
                        # Erase contents
                        self.path.delete("1.0",
                             tk.END)

                # Insert the new content
                self.path.insert("1.0", text)

                # Disable text field
                self.path.config(state=tk.
                     DISABLED)


        # Search for the directory containing
             the dataset
        def __browse(self):
                # Ask user to select the
                     directory
                if (self.type == "f"):
                        dir_path = tkFileDialog.
                             askopenfilename(
                             initialdir="/",
                                 title="Browse
                                      ...",
                                      defaultextension
                                      =self.
                                      default_ext
                                      , filetypes
                                      =self.
                                      file_types)
                elif (self.type == "nf"):
                        dir_path = tkFileDialog.
                             asksaveasfilename(
                             initialdir="/",
                                 title="Browse
                                      ...",
                                      defaultextension
                                      =self.
                                      default_ext
                                      , filetypes
                                      =self.
                                      file_types)
                else:
                        dir_path = tkFileDialog.
                             askdirectory(
                             initialdir="/",
                                 title="Browse
                                      ...",
                                      mustexist=
                                      True)

                # If user selected a valid
                     directory
                if (dir_path):
                        self.setText(dir_path)
                        if callable(self.
                             open_model):
                                self.open_model(
                                     self.
                                     getText().
                                     rstrip())
```

File: trainer/gui/Logger.py

```
import Tkinter as tk
import tkFileDialog
import time
import os

def initialize(parent):
        global pane
        global filepath

        filepath = ""
        pane = tk.PanedWindow(parent, orient=tk.
             VERTICAL)

        _initialize_pane()

def _is_full():
        if (len(get_log()) >= 10000):
                return True
        return False

def get_log():
        global log
        return log.get("1.0", tk.END)

def append(message):
        global log
        global filepath

        if _is_full():
                mode = "w"
                if (filepath):
                        if os.path.exists(
                             filepath):
                                mode = "a"

                with open(filepath, mode
                     ) as export_file:
                        export_file.
                             write(
                             get_log())

                log.configure(state=tk.
                     NORMAL)
                log.delete("1.0", tk.END
                     )
                log.configure(state=tk.
                     DISABLED)

        # Get timestamp
        timestamp = time.strftime("%Y-%m-%d %H:%
             M:%S", time.localtime())

        # Compose the log entry
        log_entry = "{}: {}\n".format(timestamp,
              message)

        # Enable writing
        log.configure(state=tk.NORMAL)

        # Insert to the log pane
        log.insert(tk.END, log_entry)
```

```
        # Disable writing
        log.configure(state=tk.DISABLED)

        # Scroll to the latest log
        log.see(tk.END)

def get_widget():
        global pane
        return pane

def set_directory(model_dir, filename="train_log
    .txt"):
        global filepath

        directory = model_dir
        filepath = os.path.join(model_dir,
            filename)

def _initialize_pane():
        global pane
        global log

        pane_text = tk.PanedWindow(pane, height
            =400)

        # Instantiate log pane
        log = tk.Text(pane_text, height=400,
            padx=10, pady=5, wrap=tk.WORD)

        pane_text.add(log)

        # Instantiate scrollbar
        scroll = tk.Scrollbar(pane_text, orient
            ="vertical", command=log.yview)

        # Set scrollbar for the log pane
        log.configure(yscrollcommand=scroll.set)
        scroll.pack(side="right", fill="y")

        # Disable writing
        log.config(state=tk.DISABLED)

        button_export = tk.Button(pane, text="
            Export", command=_trigger_export)

        pane_text.pack(fill="both", expand="yes
            ")
        button_export.pack()

def _trigger_export():
        global log

        # Create a new file at the selected save
             location
        save_file = tkFileDialog.
            asksaveasfilename(initialdir="/",
                defaultextension=".txt",
                    filetypes=[("Text file", ".
                    txt")])

        if (save_file):
                with open(save_file, "w") as
                    log_handle:
                        log_handle.write(log.get
                            ("1.0", tk.END))
                append("Exported activity log")


"""
class Logger(tk.PanedWindow):

        # Instance variables
        #       log                             Tkinter
            Text instance

        def __init__(self, parent):
                # Main container
                tk.PanedWindow.__init__(self,
                    parent)

                # Instantiate log pane
                self.log = tk.Text(parent, padx
                    =10, pady=5, wrap="none")

                # Instantiate scrollbar
                scroll = tk.Scrollbar(parent,
                    orient="vertical", command=
                    self.log.yview)

                # Set scrollbar for the log pane
                self.log.configure(
                    yscrollcommand=scroll.set)
                scroll.pack(side="right", fill="
                    y")

                # Disable writing
                self.log.config(state=tk.
                    DISABLED)

                # Add the log pane to the
                    container
                self.add(self.log)


        def append(self, message):
                # Get timestamp
                timestamp = time.strftime("%Y-%m
                    -%d %H:%M:%S", time.
                    localtime())

                # Compose the log entry
                log_entry = "{}: {}\n".format(
                    timestamp, message)

                # Enable writing
                self.log.configure(state=tk.
                    NORMAL)

                # Insert to the log pane
                self.log.insert(tk.END,
                    log_entry)

                # Disable writing
                self.log.configure(state=tk.
                    DISABLED)

                # Scroll to the latest log
                self.log.see(tk.END)
"""
```

File: trainer/gui/Plotter.py

```
from __future__ import division
from __future__ import print_function

import os
import argparse
import matplotlib
import numpy as np
import Tkinter as tk

import matplotlib
matplotlib.use("TkAgg")

import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import
    FigureCanvasTkAgg
from matplotlib.figure import Figure

global CHECKPOINT
CHECKPOINT = 5000

def initialize(model_dir):
        global top
        global pane

        top = tk.Toplevel()
        pane = tk.PanedWindow(top, orient=tk.
            VERTICAL)

        global model_directory
        model_directory = model_dir

        _initialize_pane()

        pane.pack()

def close_graph():
        global top

        plt.close("all")
        top.destroy()

def update_graph():
        global model_directory
        global ax
        global fig_canvas
        global CHECKPOINT

        average_window = 100
        average_filter = np.ones(average_window)
            / float(average_window)

        # Colors
```

```
        colors_raw = [(228, 26, 28)]
        colors = [(r / 255, g / 255, b / 255)
            for r, g, b in colors_raw]

        # Model directory
        name = os.path.basename(os.path.abspath(
            model_directory))

        # Color of the line graph
        color = colors[0]

        path = os.path.join(model_directory, '
            costs.npz')

        if (not os.path.isfile(path)):
                return

        # Load npz file for the costs per
            iteration
        costs = np.load(path)

        # Get costs for training and validation
        train_costs = costs['train']
        valid_costs = costs['validation'].tolist
            ()

        iters = train_costs.shape[0]
        valid_range = [CHECKPOINT * (i + 1) for
            i in range(iters // CHECKPOINT)]

        if len(valid_range) != len(valid_costs):
                valid_range.append(iters)
        if train_costs.ndim == 1:
                train_costs = np.convolve(
                    train_costs, average_filter
                    , mode='valid')

        # Clear current content
        ax.clear()

        # Plot training and validation costs
        ax.plot(train_costs, color=color, label=
            name + '_train', lw=1.5)
        ax.plot(valid_range, valid_costs[:len(
            valid_range)], '-o', color=color,
            label=name + '_valid')

        # Enable grid lines
        ax.grid(True)
        ax.legend(loc='best')

        # Update the canvas
        fig_canvas.draw()

def _initialize_pane():
        global pane
        global fig
        global ax
        global fig_canvas

        # Create JSON pane
        frame_plot_model = tk.LabelFrame(pane,
            text="Plot model", pady=10)

        # Textfield for dataset directory
        label_canvas = tk.Label(frame_plot_model
            , text="Plot", anchor="w")
        #pane_model = DirectoryText(lf_plotModel
            )

        # Setup graph labels

        fig = Figure()

        ax = fig.add_subplot(111)
        ax.set_xlabel('Iters')
        ax.set_ylabel('Loss')

        # Instantiate figure canvas
        fig_canvas = FigureCanvasTkAgg(fig, pane
            )

        # Pane containing the graph
        pane_canvas = tk.PanedWindow(
            frame_plot_model, width=600)

        # Tk widget for the graph
        canvas = fig_canvas.get_tk_widget()

        # Add the graph to the container
        pane_canvas.add(canvas)

        pane_canvas.pack(fill="both", expand=
            True)
        frame_plot_model.pack(fill="both",
            expand="yes")

        # Enable grid lines
        ax.grid(True)
        ax.legend(loc='best')

        fig_canvas.draw()
        update_graph()
"""
class PlotManager(tk.PanedWindow):

        def __init__(self, parent, model_dir):

                # Main container
                tk.PanedWindow.__init__(self,
                    parent, orient=tk.VERTICAL)

                # Save an instance of the logger
                #self.logger = logger

                # Directory of the model to be
                    plotted
                self.model_dir = model_dir

                # Initialize pane for create
                    JSON
                self.__initPane_plotModel__()

                # Plot the fucking graph
                #self.plot_graph()

        def plot_graph(self):
                average_window = 100
                average_filter = np.ones(
                    average_window) / float(
                    average_window)

                # Colors
                colors_raw = [(228, 26, 28)]
                colors = [(r / 255, g / 255, b /
                    255) for r, g, b in
                    colors_raw]

                # Model directory
                name = os.path.basename(os.path.
                    abspath(self.model_dir))

                # Color of the line graph
                color = colors[0]

                # Load npz file for the costs
                    per iteration
                costs = np.load(os.path.join(
                    self.model_dir, 'costs.npz
                    '))

                # Get costs for training and
                    validation
                train_costs = costs['train']
                valid_costs = costs['validation
                    '].tolist()

                iters = train_costs.shape[0]
                valid_range = [500 * (i + 1) for
                    i in range(iters // 500)]

                if len(valid_range) != len(
                    valid_costs):
                        valid_range.append(iters
                            )
                if train_costs.ndim == 1:
                        train_costs = np.
                            convolve(
                            train_costs,
                            average_filter,
                            mode='valid')

                # Clear current content
                self.ax.clear()

                # Plot training and validation
                    costs
                self.ax.plot(train_costs, color=
                    color, label=name + '_train
                    ', lw=1.5)
                self.ax.plot(valid_range,
                    valid_costs[:len(
                    valid_range)], '-o', color=
```

```
                    color, label=name + '_valid
                    ')

                # Enable grid lines
                self.ax.grid(True)
                self.ax.legend(loc='best')

                # Update the canvas
                self.fig_canvas.draw()

        def __initPane_plotModel__(self):
                # Create JSON pane
                lf_plotModel = tk.LabelFrame(
                    self, text="Plot model",
                    pady=10)

                # Textfield for dataset
                    directory
                label_canvas = tk.Label(
                    lf_plotModel, text="Plot",
                    anchor="w")
                #pane_model = DirectoryText(
                    lf_plotModel)

                # Setup graph labels
                self.fig = Figure()

                self.ax = self.fig.add_subplot
                    (111)
                self.ax.set_xlabel('Iters')
                self.ax.set_ylabel('Loss')

                # Instantiate figure canvas
                self.fig_canvas =
                    FigureCanvasTkAgg(self.fig,
                     self)

                # Pane containing the graph
                pane_canvas = tk.PanedWindow(
                    lf_plotModel, width=600)

                # Tk widget for the graph
                canvas = self.fig_canvas.
                    get_tk_widget()

                # Add the graph to the container
                pane_canvas.add(canvas)

                # Plot button
                #button_plot = tk.Button(
                    lf_plotModel, text="Plot",
                    command=lambda: self.
                    plot_graph())

                pane_canvas.pack(fill="both",
                    expand=True)

                #button_clear.pack()
                #button_plot.pack()

                lf_plotModel.pack(fill="both",
                    expand="yes")
"""
```

File: trainer/gui/ProgressPanel.py

```
import Tkinter as tk
import ttk


def initialize(parent):
        global pane
        pane = tk.PanedWindow(parent)

        _initialize_pane()

def get_widget():
        global pane
        return pane

def get_maximum():
        global progress_bar
        return progress_bar["maximum"]

def get_current_value():
        global progress_bar
        return progress_bar["value"]

def set_status_message(message):
        global status
        status["text"] = message

def set_maximum(max_value):
        global progress_bar

        # Reset value
        progress_bar["value"] = 0
        progress_bar["maximum"] = max_value

def set_current_value(value):
        global progress_bar
        progress_bar["value"] = value

def set_idle():
        set_maximum(0)
        set_status_message("Idle")

def _initialize_pane():
        global pane
        global progress_bar
        global status

        status = tk.Label(pane, text="Idle",
            anchor="w")
        pane_progress = tk.PanedWindow(pane)

        progress_bar = ttk.Progressbar(
            pane_progress, orient="horizontal",
             length=200, mode="determinate")
        pane_progress.add(progress_bar)

        status.pack(fill="both")
        pane_progress.pack(fill="both")
```

File: trainer/gui/StatusPanel.py

```
import Tkinter as tk
import ttk


def initialize(parent):
        global pane

        pane = tk.PanedWindow(parent)

        _initialize_pane()

def _initialize_pane():
        global pane

        label_status = tk.Label(pane, text="",
            anchor="w")

        progress_bar = ttk.Progressbar(
            pane_progress, orient="horizontal",
             length=200, mode="determinate")
```

File: trainer/gui/TestingPanel.py

```
import Tkinter as tk
import threading as th
import tkMessageBox

from gui.DirectoryText import DirectoryText
from gui.Action import Action
from core.utils import load_model

import gui.ActionManager as ActionManager
import gui.Logger as Logger

def initialize(parent):
        global pane
        pane = tk.PanedWindow(parent, orient=tk.
            VERTICAL)
        _initialize_pane()

def get_widget():
        global pane
        return pane


def _initialize_pane():
        global pane

        # Create JSON pane
        frame_test_model = tk.LabelFrame(pane,
            text="Test model", padx=30, pady
            =10)

        label_json_test = tk.Label(
            frame_test_model, text="Test JSON
            directory", anchor="w")
        pane_json_test = DirectoryText(
            frame_test_model, type="f")
```

```
        #pane_json_test.setText("/home/iantot/
            Desktop/test2_test.json")

        label_json_train = tk.Label(
            frame_test_model, text="Train JSON
            directory", anchor="w")
        pane_json_train = DirectoryText(
            frame_test_model, type="f")
        #pane_json_train.setText("/home/iantot/
            Desktop/test2_train.json")

        label_model = tk.Label(frame_test_model,
             text="Model directory", anchor="w
            ")
        pane_model = DirectoryText(
            frame_test_model)
        #pane_model.setText("/home/iantot/
            Desktop/new model")

        # Train button
        global button_test
        global button_stop
        button_test = tk.Button(frame_test_model
            , text="Test",
                command=lambda:
                    _trigger_start_testing(
                    pane_json_test.getText(),
                    pane_json_train.getText(),
                    pane_model.getText()))

        button_stop = tk.Button(frame_test_model
            , text="Stop",
                command=_trigger_stop_testing)
        button_stop.config(state=tk.DISABLED)

        label_json_test.pack(fill="both")
        pane_json_test.pack(fill="both")

        label_json_train.pack(fill="both")
        pane_json_train.pack(fill="both")

        label_model.pack(fill="both")
        pane_model.pack(fill="both")

        button_test.pack()
        button_stop.pack()

        frame_test_model.pack(fill="both",
            expand="yes")

def _trigger_start_testing(test_json, train_json
    , model_dir):

        # Remove whitespaces
        test_json = test_json.rstrip()
        train_json = train_json.rstrip()
        model_dir = model_dir.rstrip()

        if ((not test_json) or (not train_json)
            or (not model_dir)):
                tkMessageBox.showwarning(title="
                    Unfilled fields",
                        message="Please fill up
                            all the fields!")
                return

        if (not load_model(model_dir, check=True
            )):
                tkMessageBox.showwarning(title="
                    Invalid model",
                        message="Please input a
                            trained model
                            directory!")
                return

        Logger.append("Testing the model in {}".
            format(model_dir))

        args = {
                "test_json" : test_json,
                "train_json" : train_json,
                "model_dir" : model_dir
        }

        # Thread the testing process
        action = Action(Action.TEST_MODEL, args)
        ActionManager.setAction(action)


def _trigger_stop_testing():
        action = Action(Action.STOP_TEST_MODEL,
            {})
        ActionManager.setAction(action)

def disable_buttons(type):
        global button_test
        global button_stop

        button_test.config(state=tk.DISABLED)
        if (type == Action.TEST_MODEL):
                button_stop.config(state=tk.
                    NORMAL)
        else:
                button_stop.config(state=tk.
                    DISABLED)

def enable_buttons():
        global button_test
        global button_stop

        button_test.config(state=tk.NORMAL)
        button_stop.config(state=tk.DISABLED)
```

File: trainer/gui/TrainingPanel.py

```
import Tkinter as tk
import threading as th
import numpy as np

import os
import ttk
import json
import json
import time
import tkMessageBox

import gui.ActionManager as ActionManager
from gui.DirectoryText import DirectoryText
from gui.Action import Action
from core.char_map import char_map as cmap
from core.utils import load_hyperparams

import gui.Logger as Logger


def initialize(parent):
        global pane
        pane = tk.PanedWindow(parent, orient=tk.
            VERTICAL)

        # Construct contents of pane
        _initialize_pane()

        # Update char map to include spaces
        global char_map
        char_map = cmap
        char_map[" "] = 2

def get_widget():
        global pane
        return pane

def _has_foreign_chars(line):
        global char_map

        for c in line:
                if (not c in char_map):
                        return True
        return False

def _is_json(json_str):
        try:
                json_object = json.loads(
                    json_str)
        except ValueError, e:
                return False
        return True

def _check_json_file(filepath):

        with open(filepath, "r") as json_f:
                for line in json_f:

                        # Remove whitespaces
                        line = line.rstrip()

                        # Check if current line
                            is a valid JSON
                        if (not _is_json(line)):
                                tkMessageBox.
                                    showwarning
                                    (title="
                                    Invalid
                                    JSON file",
```

```
                    message
                        ="                                                  characters
                        The
                                                                            unrecognized
                        JSON                                                 by
                        file                                                the
                        contains                                            trainer
                                                                            !")
                        lines                                   return False
                                        return True
                        that
                                def _prepare_training_args(train_json,
                        are             valid_json, save_dir, hyperparams):
                                    # Remove whitespaces/newlines
                        not         train_json = train_json.rstrip()
                         in         valid_json = valid_json.rstrip()
                                    save_dir = save_dir.rstrip()
                        valid
                                    # Check if hyperparams values are within
                        JSON              bounds
                                    epoch = hyperparams[0].rstrip()
                        format      lrate = hyperparams[1].rstrip()
                        .")         mb_size = hyperparams[2].rstrip()
            return False
                                    # Check if all fields are filled up
    # Convert current line          if ((not train_json) or (not valid_json)
        to a JSON                         or (not save_dir) or (not lrate)
    json_obj = json.loads(                or (not epoch) or (not mb_size)):
        line)                           tkMessageBox.showwarning(title="
                                              Unfilled fields", message="
    # Check if a valid JSON                   Please fill up all the
        recognized by the                     fields!")
        system                          return None
    if (not all(key in
        json_obj.keys() for         # Check if valid format for
         key in ["key", "               hyperparameters
        text", "duration"])         try:
        ):                              lrate_val = float(lrate)
            tkMessageBox.               epoch_val = int(epoch)
                 showwarning            mbsize_val = int(mb_size)
                 (title="           except ValueError:
                 Unrecognized           tkMessageBox.showwarning(title="
                  JSON                        Invalid hyperparameters",
                 format",                    message="Invalid values for
                     message                  the hyperparameters!")
                         ="             return None
                         The
                                    # Check contents of train JSON
                         JSON       if (not _check_json_file(train_json)):
                                        return None
                         file
                                    # Check contents of validation JSON
                         does       if (not _check_json_file(valid_json)):
                                        return None
                         not
                                    # Log current
                         contain    Logger.append("(Train) Starting training
                                          for model saved in {}".format(
                         JSON             save_dir))

                         keys       args = {
                                        "lrate" : lrate_val,
                         recognized     "epochs" : epoch_val,
                          by            "mb_size" : mbsize_val,
                                        "train_json" : train_json,
                         the            "valid_json" : valid_json,
                                        "save_dir" : save_dir
                         trainer    }
                         !")
            return False            return args
                                #
    # Check if current line     #   train_json              filepath of
        contains invalid            train json
        chars                   #   valid_json              filepath of
    if _has_foreign_chars(          valid_json
        json_obj["text"]):      #   save_dir                directory where
            tkMessageBox.           the model will be saved
                 showwarning    #   hyperparams             list of
                 (title="           hyperparameters
                 Invalid        #           epoch
                 JSON file",    #           lrate
                     message    #           archi
                         ="     #
                         The        def _trigger_start_training(train_json,
                                        valid_json, save_dir, hyperparams, resume=
                         JSON           False):
                                        args = _prepare_training_args(train_json
                         file                , valid_json, save_dir, hyperparams
                                            )
                         contains       if args == None:
```

```
                return

        if not resume:
                action = Action(Action.
                    TRAIN_MODEL, args)
        else:
                action = Action(Action.
                    CONTINUE_TRAIN_MODEL, args)

        ActionManager.setAction(action)

def _trigger_stop_training():
        action = Action(Action.STOP_TRAIN_MODEL,
            {})
        ActionManager.setAction(action)

def _trigger_open_model(model_dir):
        global button_train

        args = load_hyperparams(model_dir)
        if not args == None:
                button_train["text"] = "Continue
                    "
                _load_params(args)

        else:
                button_train["text"] = "Train"

def _load_params(args):
        global epoch_value
        global text_mbsize
        global text_lr

        epoch_value.set(args["epochs"])

        if (text_mbsize.get("1.0", tk.END)):
                text_mbsize.delete("1.0", tk.END
                    )
        text_mbsize.insert("1.0", args["mb_size
            "])

        if (text_lr.get("1.0", tk.END)):
                text_lr.delete("1.0", tk.END)
        text_lr.insert("1.0", args["lrate"])


def _initialize_pane():
        global pane

        # Create JSON pane
        frame_train_model = tk.LabelFrame(pane,
                text="Train model", padx=30,
                    pady=10)

        # Textfield for train JSON directory
        label_json_train = tk.Label(
            frame_train_model,
                text="Train JSON directory",
                    anchor="w")

        pane_json_train = DirectoryText(
            frame_train_model, type="f")

        # Textfield for validation JSON
            directory
        label_json_valid = tk.Label(
            frame_train_model, text="Validation
             JSON directory", anchor="w")
        pane_json_valid = DirectoryText(
            frame_train_model, type="f")

        # Textfield for model save directory
        label_save_model = tk.Label(
            frame_train_model, text="Model save
             directory", anchor="w")
        pane_save_model = DirectoryText(
            frame_train_model, callback=
            _trigger_open_model)

        # Hyperparameters pane
        label_hyperp = tk.Label(
            frame_train_model, text="
            Hyperparameters", anchor="w")
        pane_hyperp = tk.PanedWindow(
            frame_train_model, width=140)

        pane_hyperp_l = tk.PanedWindow(
            pane_hyperp, orient=tk.VERTICAL)
        pane_hyperp_r = tk.PanedWindow(
            pane_hyperp, orient=tk.VERTICAL)

        # Epochs
        global epoch_value
        epoch_value = tk.IntVar()
        label_epoch = tk.Label(pane_hyperp_l,
            text="Epoch", anchor="w")
        spin_epoch = tk.Spinbox(pane_hyperp_r,
            textvariable=epoch_value, from_=1,
            to=100, increment=1, width=5)

        pane_hyperp_l.add(label_epoch)
        pane_hyperp_r.add(spin_epoch)

        # Learning rate
        global text_lr
        label_lr = tk.Label(pane_hyperp_l, text
            ="Learning rate", anchor="w")
        text_lr = tk.Text(pane_hyperp_r, height
            =1)
        text_lr.insert(tk.END, "0.001")

        pane_hyperp_l.add(label_lr)
        pane_hyperp_r.add(text_lr)

        # Minibatch size
        global text_mbsize
        label_mbsize = tk.Label(pane_hyperp_l,
            text="Minibatch size", anchor="w")
        text_mbsize = tk.Text(pane_hyperp_r,
            height=1)
        text_mbsize.insert(tk.END, "2")

        pane_hyperp_l.add(label_mbsize)
        pane_hyperp_r.add(text_mbsize)

        pane_hyperp.add(pane_hyperp_l)
        pane_hyperp.add(pane_hyperp_r)


        global button_train
        global button_stop
        # Train button
        button_train = tk.Button(
            frame_train_model, text="Train")
        button_train.configure(command=lambda:
                _trigger_start_training(
                        pane_json_train.getText
                            (),
                        pane_json_valid.getText
                            (),
                        pane_save_model.getText
                            (),
                        (spin_epoch.get(),
                            text_lr.get("1.0",
                            tk.END),
                            text_mbsize.get
                            ("1.0", tk.END)),
                        button_train["text"] ==
                            "Continue"))

        button_continue = tk.Button
        button_stop = tk.Button(
            frame_train_model, text="Stop",
                command=_trigger_stop_training)
        button_stop.config(state=tk.DISABLED)

        label_json_train.pack(fill="both")
        pane_json_train.pack(fill="both")

        label_json_valid.pack(fill="both")
        pane_json_valid.pack(fill="both")

        label_save_model.pack(fill="both")
        pane_save_model.pack(fill="both")

        label_hyperp.pack(fill="both")
        pane_hyperp.pack()

        button_train.pack()
        button_stop.pack()

        frame_train_model.pack(fill="both",
            expand="yes")

def disable_buttons(type):
        global button_train
        global button_stop

        button_train.config(state=tk.DISABLED)
        if (type == Action.TRAIN_MODEL or type
            == Action.CONTINUE_TRAIN_MODEL):
                button_stop.config(state=tk.
                    NORMAL)
        else:
```

```
                button_stop.config(state=tk.
                    DISABLED)

def enable_buttons():
        global button_train
        global button_stop

        button_train.config(state=tk.NORMAL)
        button_stop.config(state=tk.DISABLED)
```

File: app/src/main/java/ph/edu/up/pasabi/activity/ContactsActivity.java

```
package ph.edu.up.pasabi.activity;

import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.provider.ContactsContract;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;

import ph.edu.up.pasabi.R;
import ph.edu.up.pasabi.sms.ContactAdapter;
import ph.edu.up.pasabi.sms.SMSConvoAdapter;
import ph.edu.up.pasabi.sms.SMSGrandObserver;
import ph.edu.up.pasabi.sms.SMSManager;

/**
 * Created by Iantot on 4/16/2017.
 */

public class ContactsActivity extends Activity
{
    private String[] columnNames;
    private String query;
    private String[] argsNames;
    private String sortOrder;

    private ListView list;
    private ContactAdapter adapter;

    @Override
    protected void onCreate(Bundle
        savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.
            activity_contacts);

        columnNames = new String[]{
            "_id",
            "display_name",
            "data4"
        };

        query = null;
        argsNames = null;
        sortOrder = "display_name asc";

        Cursor cursor = getContentResolver().
            query(ContactsContract.
            CommonDataKinds.Phone.CONTENT_URI,
            columnNames, query, argsNames,
            sortOrder);

        list = (ListView) findViewById(R.id.
            contacts_list);
        adapter = new ContactAdapter(this,
            cursor);

        list.setAdapter(adapter);
        list.setOnItemClickListener(new
            AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView
                <?> adapterView, View view, int
                i, long l) {
                String contactName = (String)
                    view.getTag(R.id.
                    tag_contactName);
                String contactNum = (String)
                    view.getTag(R.id.
                    tag_contactNum);
                String threadId = (String) view.
                    getTag(R.id.tag_threadId);

                Intent intent = new Intent();
                intent.putExtra("contactName",
                    contactName);
                intent.putExtra("contactNum",
                    contactNum);
                intent.putExtra("threadId",
                    threadId);

                setResult(SMSManager.
                    GET_CONTACT_NUMBER, intent)
                    ;
                finish();
            }
        });
    }

    @Override
    public void onBackPressed()
    {
        Intent intent = new Intent();
        intent.putExtra("contactName", "");
        intent.putExtra("contactNum", "");
        intent.putExtra("threadId", "");

        setResult(SMSManager.GET_CONTACT_NUMBER,
            intent);
        finish();
    }

    public void goBack(View view)
    {
        onBackPressed();
    }
}
```

File: app/src/main/java/ph/edu/up/pasabi/activity/ContactsActivity.java

```
package ph.edu.up.pasabi.activity;

import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.provider.ContactsContract;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;

import ph.edu.up.pasabi.R;
import ph.edu.up.pasabi.sms.ContactAdapter;
import ph.edu.up.pasabi.sms.SMSConvoAdapter;
import ph.edu.up.pasabi.sms.SMSGrandObserver;
import ph.edu.up.pasabi.sms.SMSManager;

/**
 * Created by Iantot on 4/16/2017.
 */

public class ContactsActivity extends Activity
{
    private String[] columnNames;
    private String query;
    private String[] argsNames;
    private String sortOrder;

    private ListView list;
    private ContactAdapter adapter;

    @Override
    protected void onCreate(Bundle
        savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.
            activity_contacts);

        columnNames = new String[]{
            "_id",
            "display_name",
            "data4"
        };

        query = null;
        argsNames = null;
        sortOrder = "display_name asc";

        Cursor cursor = getContentResolver().
            query(ContactsContract.
            CommonDataKinds.Phone.CONTENT_URI,
            columnNames, query, argsNames,
            sortOrder);

        list = (ListView) findViewById(R.id.
            contacts_list);
```

```
        adapter = new ContactAdapter(this,
            cursor);

        list.setAdapter(adapter);
        list.setOnItemClickListener(new
            AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView
                <?> adapterView, View view, int
                i, long l) {
                String contactName = (String)
                    view.getTag(R.id.
                    tag_contactName);
                String contactNum = (String)
                    view.getTag(R.id.
                    tag_contactNum);
                String threadId = (String) view.
                    getTag(R.id.tag_threadId);

                Intent intent = new Intent();
                intent.putExtra("contactName",
                    contactName);
                intent.putExtra("contactNum",
                    contactNum);
                intent.putExtra("threadId",
                    threadId);

                setResult(SMSManager.
                    GET_CONTACT_NUMBER, intent)
                    ;
                finish();
            }
        });
    }

    @Override
    public void onBackPressed()
    {
        Intent intent = new Intent();
        intent.putExtra("contactName", "");
        intent.putExtra("contactNum", "");
        intent.putExtra("threadId", "");

        setResult(SMSManager.GET_CONTACT_NUMBER,
            intent);
        finish();
    }

    public void goBack(View view)
    {
        onBackPressed();
    }
}
```

File: app/src/main/java/ph/edu/up/pasabi/activity/ConversationActivity.java

```
package ph.edu.up.pasabi.activity;

import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.ContentValues;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.database.ContentObserver;
import android.database.Cursor;
import android.graphics.Typeface;
import android.media.AudioRecord;
import android.media.MediaRecorder;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Handler;
import android.support.v4.content.
    LocalBroadcastManager;
import android.telephony.SmsManager;
import android.util.Log;
import android.view.View;
import android.widget.CursorAdapter;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.ListView;
import android.widget.RelativeLayout;
import android.widget.RelativeLayout.
    LayoutParams;
import android.widget.TextView;

import java.io.IOException;
import java.util.Locale;

import ph.edu.up.pasabi.sms.SMSGrandObserver;
import ph.edu.up.pasabi.sms.SMSManager;
import ph.edu.up.pasabi.sms.SMSMessageAdapter;
import ph.edu.up.pasabi.sms.SMSObserver;
import ph.edu.up.pasabi.utils.ExtAudioRecorder;
import ph.edu.up.pasabi.utils.PASABIUtils;
import ph.edu.up.pasabi.R;

/**
 * Created by Iantot on 4/8/2017.
 */

public class ConversationActivity extends
    Activity implements SMSObserver
{
    // Audio recorder
    private ExtAudioRecorder recorder;

    private String threadId;
    private String contactName;
    private String contactNum;

    private String[] columnNames;
    private String query;
    private String[] argsNames;
    private String sortOrder;

    private ListView list;
    private CursorAdapter adapter;



    @Override
    protected void onCreate(Bundle
        savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.
            activity_conversation);

        SMSGrandObserver.getInstance().
            registerObserver(this);

        // Instantiate the recorder
        recorder = null;

        Intent intent = getIntent();
        contactName = intent.getStringExtra("
            contactName");
        contactNum = intent.getStringExtra("
            contactNum");
        threadId = intent.getStringExtra("
            threadId");

        // Set the heading
        setConversationHeading(contactNum,
            contactName);

        columnNames = new String[]{
            "_id",
            "thread_id",
            "body",
            "date",
            "type"
        };

        query = "thread_id = ? AND (type = ? OR
            type = ?)";
        argsNames = new String[]{
            String.format(Locale.US, "%s",
                threadId),
            SMSManager.INCOMING_SMS,
            SMSManager.OUTGOING_SMS
        };
        sortOrder = "date asc";

        setListContents(threadId);

        // Start initializing thread
        Intent initializeIntent = new Intent(
            getApplicationContext(),
            SpeechInitializerService.class);
        startService(initializeIntent);

        IntentFilter filter = new IntentFilter("
            go-intent");
        filter.addAction("error-intent");
        filter.addAction("transcribe-intent");
        filter.addAction("transcribing-intent");
        filter.addAction("initializing-intent");

        LocalBroadcastManager.getInstance(this).
            registerReceiver(
```

```
new BroadcastReceiver()
{
    @Override
    public void onReceive(Context
         context, Intent intent)
    {
        String action = intent.getAction
            ();
        ImageButton speechButton = (
            ImageButton) findViewById(R
            .id.convo_speech_button);
        TextView statusField = (TextView
            ) findViewById(R.id.
            convo_speech_status);

        if (action.equals("go-intent"))
        {
            // Apply transcript to the
                text message field
            statusField.setText(R.string
                .
                convo_speech_status_idle
                );
            speechButton.
                setImageResource(R.
                drawable.
                speech_button_big_normal
                );
            speechButton.setEnabled(true
                );
        }
        else if (action.equals("error-
            intent"))
        {
            // Apply transcript to the
                text message field
            statusField.setText(R.string
                .
                convo_speech_status_error
                );
            speechButton.
                setImageResource(R.
                drawable.
                speech_button_big_off);
            speechButton.setEnabled(
                false);
        }
        else if (action.equals("
            initializing-intent"))
        {
            // Apply transcript to the
                text message field
            statusField.setText(R.string
                .
                convo_speech_status_initializing
                );
            speechButton.
                setImageResource(R.
                drawable.
                speech_button_big_off);
            speechButton.setEnabled(
                false);
        }
        else if (action.equals("
            transcribing-intent"))
        {
            statusField.setText(R.string
                .
                convo_speech_status_transcribing
                );
            speechButton.
                setImageResource(R.
                drawable.
                speech_button_big_normal
                );
            speechButton.setEnabled(
                false);
        }
        else if (action.equals("
            transcribe-intent"))
        {
            // Extract transcript from
                broadcasted intent
            String transcript = intent.
                getStringExtra("
                transcript");

            // Apply transcript to the
                text message field
            TextView messageField = (
                TextView) findViewById(
                R.id.convo_edittext);
            messageField.setText(
                transcript);

            statusField.setText(R.string
                .
                convo_speech_status_idle
                );
            speechButton.
                setImageResource(R.
                drawable.
                speech_button_big_normal
                );
            speechButton.setEnabled(true
                );
        }
    }
}, filter);

// Set listener for voice recording
final ImageButton toggleButton = (
    ImageButton) findViewById(R.id.
    convo_button_toggle);

OnToggleListener compartmentListener =
    new OnToggleListener()
{
    @Override
    public void onOpen(View view)
    {
        RelativeLayout hiddenCompartment
            = (RelativeLayout)
            findViewById(R.id.
            convo_hidden_compartment);
        RelativeLayout botBar = (
            RelativeLayout)
            findViewById(R.id.
            convo_botbar);

        hiddenCompartment.setVisibility(
            View.VISIBLE);

        LayoutParams hcParams = (
            LayoutParams)
            hiddenCompartment.
            getLayoutParams();
        hcParams.addRule(RelativeLayout.
            ALIGN_PARENT_BOTTOM);
        hiddenCompartment.
            setLayoutParams(hcParams);

        LayoutParams bbParams = (
            LayoutParams) botBar.
            getLayoutParams();
        bbParams.removeRule(
            RelativeLayout.
            ALIGN_PARENT_BOTTOM);
        botBar.setLayoutParams(bbParams)
            ;
    }

    @Override
    public void onClose(View view)  //
    {
        RelativeLayout hiddenCompartment
            = (RelativeLayout)
            findViewById(R.id.
            convo_hidden_compartment);
        RelativeLayout botBar = (
            RelativeLayout)
            findViewById(R.id.
            convo_botbar);

        hiddenCompartment.setVisibility(
            View.GONE);

        LayoutParams hcParams = (
            LayoutParams)
            hiddenCompartment.
            getLayoutParams();
        hcParams.removeRule(
            RelativeLayout.
            ALIGN_PARENT_BOTTOM);
        hiddenCompartment.
            setLayoutParams(hcParams);

        LayoutParams bbParams = (
            LayoutParams) botBar.
            getLayoutParams();
        bbParams.addRule(RelativeLayout.
            ALIGN_PARENT_BOTTOM);
        botBar.setLayoutParams(bbParams)
            ;
```

```
    }
};
toggleButton.setOnClickListener(
    compartmentListener);

final ImageButton speechButton = (
    ImageButton) findViewById(R.id.
    convo_speech_button);
final TextView speechStatus = (TextView)
     findViewById(R.id.
    convo_speech_status);

OnToggleListener speechButtonListener =
    new OnToggleListener()
{
    @Override
    public void onOpen(View view)
    {
        Log.i("MainActivity", "onOpen
            call!");
        if (recorder == null)
            recorder = ExtAudioRecorder.
                getInstance(false);

        if (recorder.getState() !=
            ExtAudioRecorder.State.
            RECORDING)
        {
            // Start recording
            recorder.setOutputFile(
                PASABIUtils.
                getAudioFilePath());
            recorder.prepare();
            recorder.start();

            speechButton.
                setImageResource(R.
                drawable.
                speech_button_big_pressed
                );
            speechStatus.setText(R.
                string.
                convo_speech_status_recording
                );
        }
    }

    @Override
    public void onClose(View view)
    {
        Log.i("MainActivity", "onClose
            call!");
        if (null != recorder)
        {
            // Stop recording
            recorder.stop();
            recorder.reset();

            recorder = ExtAudioRecorder.
                getInstance(false);

            speechStatus.setText(R.
                string.
                convo_speech_status_transcribing
                );

            // Call speech-to-text
                service after recording
            Intent transcribeIntent =
                new Intent(
                getApplicationContext()
                ,
                SpeechTranscriberService
                .class);
            startService(
                transcribeIntent);

            speechButton.
                setImageResource(R.
                drawable.
                speech_button_big_normal
                );
            speechButton.setEnabled(
                false);
        }
    }
};

speechButton.setOnClickListener(
    speechButtonListener);
speechStatus.setText(R.string.
    convo_speech_status_initializing);
    speechButton.setImageResource(R.drawable
        .speech_button_big_off);
    speechButton.setEnabled(false);
}

private void setListContents(String threadId
    )
{
    argsNames = new String[]{
            String.format(Locale.US, "%s",
                threadId),
            SMSManager.INCOMING_SMS,
            SMSManager.OUTGOING_SMS
    };

    Cursor cursor = getContentResolver().
        query(SMSManager.SMS_ALL,
        columnNames, query, argsNames,
        sortOrder);
    adapter = new SMSMessageAdapter(this,
        cursor);

    list = (ListView) findViewById(R.id.
        convo_messages);
    list.setAdapter(adapter);
    list.setSelection(adapter.getCount() -
        1);

    list.invalidateViews();
}
public void selectContact(View view)
{
    Intent intent = new Intent(
        getApplicationContext(),
        ContactsActivity.class);
    startActivityForResult(intent,
        SMSManager.GET_CONTACT_NUMBER);
}

public void goBack(View view)
{
    super.onBackPressed();
}

@Override
public void onActivityResult(int request,
    int result, Intent data)
{
    if (request == SMSManager.
        GET_CONTACT_NUMBER)
    {
        Bundle bundle = data.getExtras();
        contactName = bundle.getString("
            contactName");
        contactNum = bundle.getString("
            contactNum");
        threadId = bundle.getString("
            threadId");

        setConversationHeading(contactNum,
            contactName);
        setListContents(threadId);
    }
}

private void setConversationHeading(String
    contactNum, String contactName)
{
    ImageButton sendButton = (ImageButton)
        findViewById(R.id.convo_button_send
        );
    if ("".equals(contactName) && "".equals(
        contactNum))
    {
        // disable send
        sendButton.setEnabled(false);
        return;
    }

    sendButton.setEnabled(true);

    TextView contactNumView = (TextView)
        findViewById(R.id.
        convo_topbar_contact_num);
    contactNumView.setText(contactNum);

    if ("".equals(contactName)) {
        contactNumView.setTypeface(null,
            Typeface.BOLD);
        return;
    }
```

```
        TextView contactNameView = (TextView)
            findViewById(R.id.
            convo_topbar_contact_name);
        contactNameView.setText(contactName);
    }

    /**
     * onClick method for sending a text message
        .
     * @param view
     */
    public void sendTextMessage(View view)
    {
        Log.i("MainActivity", "Sending message
            !");
        EditText inputField = (EditText)
            findViewById(R.id.convo_edittext);
        String message = inputField.getText().
            toString().trim();

        if (message.isEmpty())
            return;


        SmsManager sms = SmsManager.getDefault()
            ;
        sms.sendTextMessage(contactNum, null,
            message, null, null);

        ContentValues values = new ContentValues
            ();
        values.put("address", contactNum);
        values.put("body", message);

        getContentResolver().insert(SMSManager.
            SMS_SENT, values);
        getContentResolver().notifyChange(
            SMSManager.SMS_ALL, null, true);

        // Scroll to the bottom
        //CursorAdapter adapter = (CursorAdapter
            ) list.getAdapter();
        //list.setSelection(adapter.getCount() -
             1);

        // Set to an empty field
        inputField.setText("");
    }

    @Override
    public void onPause()
    {
        super.onPause();
        getContentResolver().
            unregisterContentObserver(
            SMSGrandObserver.getInstance());
    }

    @Override
    public void onResume()
    {
        getContentResolver().
            registerContentObserver(SMSManager.
            SMS_ALL, true, SMSGrandObserver.
            getInstance());
        super.onResume();
    }

    @Override
    public void onChange(Uri uri)
    {
        Log.i("ConversationActivity", "onReceive
            !");
        Cursor cursor = getContentResolver().
            query(uri, null, null, null, null);
        if (cursor.getCount() <= 0) {
            cursor.close();
            return;
        }

        cursor.moveToFirst();
        String type = cursor.getString(cursor.
            getColumnIndexOrThrow("type"));
        Log.i("ConversationActivity", "onReceive
            : " + type);
        cursor.close();

        if (SMSManager.INCOMING_SMS.equals(type)
             || SMSManager.OUTGOING_SMS.equals(
            type))
        {
            Cursor cursor1 = getContentResolver
                ().query(SMSManager.SMS_ALL,
                columnNames, query, argsNames,
                sortOrder);

            Cursor cursor2 = adapter.swapCursor(
                cursor1);
            if (cursor2 != null)
                cursor2.close();

            CursorAdapter adapter = (
                CursorAdapter) list.getAdapter
                ();
            list.setSelection(adapter.getCount()
                 - 1);
        }
    }
}
```

File: app/src/main/java/ph/edu/up/pasabi/activity/MainActivity.java

```
package ph.edu.up.pasabi.activity;

import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.res.AssetManager;
import android.database.Cursor;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Environment;
import android.os.Handler;
import android.support.v4.content.
    LocalBroadcastManager;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.CursorAdapter;
import android.widget.ImageButton;
import android.widget.ListView;
import android.widget.RelativeLayout;
import android.widget.TextView;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

import ph.edu.up.pasabi.R;
import ph.edu.up.pasabi.sms.SMSConvoAdapter;
import ph.edu.up.pasabi.sms.SMSGrandObserver;
import ph.edu.up.pasabi.sms.SMSManager;
import ph.edu.up.pasabi.sms.SMSObserver;
import ph.edu.up.pasabi.utils.PASABIUtils;

public class MainActivity extends Activity
    implements SMSObserver {
    private ListView list;
    private CursorAdapter adapter;

    private String[] columnNames;
    private String query;
    private String[] argsNames;
    private String sortOrder;

    @Override
    protected void onCreate(Bundle
        savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        columnNames = new String[]{
                "_id",
                "address",
                "body",
                "thread_id",
                "date"
        };

        query = "thread_id IS NOT NULL) GROUP BY
            (thread_id";
        argsNames = null;
        sortOrder = "date desc";

        SMSGrandObserver.getInstance().
            registerObserver(this);

        Cursor cursor = getContentResolver().
            query(SMSManager.SMS_ALL,
```

```
            columnNames, query, argsNames,
            sortOrder);

        adapter = new SMSConvoAdapter(this,
            cursor);

        list = (ListView) findViewById(R.id.
            contacts_list);
        list.setAdapter(adapter);
        list.setOnItemClickListener(new
            AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView
                <?> adapterView, View view, int
                 i, long l) {
                String threadId = (String) view.
                    getTag(R.id.tag_threadId);
                String contactName = (String)
                    view.getTag(R.id.
                    tag_contactName);
                String contactNum = (String)
                    view.getTag(R.id.
                    tag_contactNum);

                Intent conversationIntent = new
                    Intent(
                    getApplicationContext(),
                    ConversationActivity.class)
                    ;
                conversationIntent.putExtra("
                    threadId", threadId);
                conversationIntent.putExtra("
                    contactName", contactName);
                conversationIntent.putExtra("
                    contactNum", contactNum);

                startActivity(conversationIntent
                    );
            }
        });

        ImageButton composeButton = (ImageButton
            ) findViewById(R.id.compose_button)
            ;
        ImageButton directoriesButton = (
            ImageButton) findViewById(R.id.
            directories_button);

        composeButton.setEnabled(false);
        directoriesButton.setEnabled(false);


        new AssetsCopier().execute();
    }

    private class AssetsCopier extends AsyncTask
        <Void, Void, Void>
    {


        @Override
        protected Void doInBackground(Void...
            voids)
        {
            Log.i("Pasabi", "Copying assets");
            if (!copyAssets("pasabi", ""))
                return null;

            // Enable buttons when done
            runOnUiThread(new Runnable()
            {
                @Override
                public void run() {
                    ImageButton composeButton =
                        (ImageButton)
                        findViewById(R.id.
                        compose_button);
                    ImageButton
                        directoriesButton = (
                        ImageButton)
                        findViewById(R.id.
                        directories_button);

                    composeButton.setEnabled(
                        true);
                    directoriesButton.setEnabled
                        (true);

                    // Hide loading
                    RelativeLayout loading = (
                        RelativeLayout)
                        findViewById(R.id.
                            loadingPanel);
                    loading.setVisibility(View.
                        GONE);
                }
            });
            return null;
        }

        private boolean copyAssets(String
            filepath, String filename)
        {
            String[] subFilepaths;
            String currentPath = filepath + (
                filename.isEmpty() ? "" : "/" +
                 filename);
            Log.i("Pasabi", "Currently in " +
                currentPath);

            try
            {
                subFilepaths = getAssets().list(
                    currentPath);
                if (subFilepaths.length > 0)
                    // filepath/name is a
                    directory
                {
                    // Create directory in
                        sdcard
                    File currDirectory = new
                        File(PASABIUtils.
                        getSDCardPath(),
                        currentPath);
                    if (!currDirectory.exists())
                        currDirectory.mkdir();

                    // Iterate through
                        subdirectories
                    for (String name :
                        subFilepaths)
                        if (!copyAssets(
                            currentPath, name))
                            return false;
                }
                else
                {
                    InputStream in;
                    OutputStream out;

                    // copy to sdcard
                    in = getAssets().open(
                        currentPath);

                    File outFile = new File(
                        PASABIUtils.
                        getSDCardPath(),
                        currentPath);
                    if (outFile.exists())
                        return true;

                    out = new FileOutputStream(
                        outFile);

                    copyFile(in, out);

                    in.close();
                    out.close();
                }
            }
            catch (Exception e)
            {
                return false;
            }
            return true;
        }

        private void copyFile(InputStream in,
            OutputStream out) throws
            IOException {

            byte[] buffer = new byte[1024];
            int read;
            while((read = in.read(buffer)) !=
                -1){
                out.write(buffer, 0, read);
            }
        }
    }

    @Override
    public void onPause()
    {
        super.onPause();
```

```
        getContentResolver().
                unregisterContentObserver(
                SMSGrandObserver.getInstance());
    }

    @Override
    public void onResume()
    {
        getContentResolver().
                registerContentObserver(SMSManager.
                SMS_ALL, true, SMSGrandObserver.
                getInstance());
        super.onResume();
    }


    public void browseModels(View view)
    {
        Intent browseIntent = new Intent(
                getApplicationContext(),
                ModelDirectoryActivity.class);

        startActivity(browseIntent);
    }
    public void composeMessage(View view)
    {
        Intent conversationIntent = new Intent(
                getApplicationContext(),
                ConversationActivity.class);
        conversationIntent.putExtra("threadId",
                "");
        conversationIntent.putExtra("contactName
                ", "");
        conversationIntent.putExtra("contactNum
                ", "");

        startActivity(conversationIntent);
    }

    @Override
    public void onChange(Uri uri)
    {
        Log.i("MainActivity", "onReceive!");
        Cursor cursor = getContentResolver().
                query(uri, null, null, null, null);
        if (cursor.getCount() <= 0) {
            cursor.close();
            return;
        }

        cursor.moveToFirst();
        String type = cursor.getString(cursor.
                getColumnIndexOrThrow("type"));
        Log.i("MainActivity", "onReceive: " +
                type);

        if (SMSManager.INCOMING_SMS.equals(type)
                || SMSManager.OUTGOING_SMS.equals(
                type))
        {
            Cursor cursor1 = getContentResolver
                    ().query(SMSManager.SMS_ALL,
                    columnNames, query, argsNames,
                    sortOrder);

            Cursor cursor2 = adapter.swapCursor(
                    cursor1);
            if (cursor2 != null)
                cursor2.close();

            list.setAdapter(adapter);
            list.invalidateViews();
        }
    }
}
```

File: app/src/main/java/ph/edu/up/pasabi/activity/ModelDirectoryActivity.java

```
package ph.edu.up.pasabi.activity;

import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;
import android.graphics.Path;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import java.io.File;
import java.io.FilenameFilter;
import java.util.ArrayList;
import java.util.Date;

import ph.edu.up.pasabi.R;
import ph.edu.up.pasabi.sms.DirectoryAdapter;
import ph.edu.up.pasabi.sms.DirectoryEntryData;
import ph.edu.up.pasabi.sms.SMSConvoAdapter;
import ph.edu.up.pasabi.sms.SMSGrandObserver;
import ph.edu.up.pasabi.sms.SMSManager;
import ph.edu.up.pasabi.utils.FileHandler;
import ph.edu.up.pasabi.utils.PASABIUtils;

/**
 * Created by Iantot on 4/16/2017.
 */

public class ModelDirectoryActivity extends
    Activity
{
    private ListView list;
    private DirectoryAdapter adapter;

    @Override
    protected void onCreate(Bundle
        savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.
            activity_browse_directory);

        File PASABIDir = new File(PASABIUtils.
            getAbsolutePath(), PASABIUtils.
            PASABI_MODELS_DIRECTORY);
        String[] directories = PASABIDir.list(
            new FilenameFilter()
        {
            @Override
            public boolean accept(File file,
                String s) {
                return new File(file, s).
                    isDirectory();
            }
        });

        ArrayList<DirectoryEntryData> dataset =
            new ArrayList<>();
        for (String name : directories)
        {
            File currDirectory = new File(
                PASABIDir.getAbsolutePath(),
                name);
            Date date = new Date(currDirectory.
                lastModified());

            dataset.add(new DirectoryEntryData(
                name, date));
        }

        list = (ListView) findViewById(R.id.
            directory_list);

        adapter = new DirectoryAdapter(this,
            dataset);
        list.setAdapter(adapter);
        list.setOnItemClickListener(new
            AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView
                <?> adapterView, View view, int
                i, long l)
            {
                // Set as default
                String directoryName = (String)
                    view.getTag(R.id.
                    tag_directoryName);

                FileHandler handler = new
                    FileHandler(PASABIUtils.
                    PASABI_DIRECTORY,
                    PASABIUtils.
                    PASABI_CONFIG_FILE);
                handler.writeToFile(
                    directoryName);

                adapter.updateAdapter();
                list.invalidateViews();
            }
        });
    }
```

```
    public void goBack(View view)
    {
        super.onBackPressed();
    }


}
```

File: app/src/main/java/ph/edu/up/pasabi/activity/OnToggleListener.java

```
package ph.edu.up.pasabi.activity;

import android.view.View;

/**
 * Created by Iantot on 4/13/2017.
 */

public abstract class OnToggleListener
    implements View.OnClickListener
{
    private static final int STATE_OPEN = 1;
    private static final int STATE_CLOSE = 2;

    private int state;

    public OnToggleListener()
    {
        // Start with a CLOSE state
        state = STATE_CLOSE;
    }

    @Override
    public void onClick(View view)
    {
        switch (state)
        {
            case STATE_CLOSE:
                onOpen(view);
                break;
            case STATE_OPEN:
                onClose(view);
                break;
        }

        toggle();
    }

    public abstract void onOpen(View view);
    public abstract void onClose(View view);

    private void toggle()
    {
        switch (state)
        {
            case STATE_CLOSE:
                state = STATE_OPEN;
                break;
            case STATE_OPEN:
                state = STATE_CLOSE;
                break;
        }
    }
}
```

File: app/src/main/java/ph/edu/up/pasabi/activity/SpeechInitializerService.java

```
package ph.edu.up.pasabi.activity;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.support.annotation.Nullable;
import android.support.v4.content.
    LocalBroadcastManager;
import android.util.Log;

import java.util.Timer;
import java.util.TimerTask;

import ph.edu.up.pasabi.sms.SMSManager;
import ph.edu.up.pasabi.utils.FileHandler;
import ph.edu.up.pasabi.utils.PASABIUtils;
import ph.edu.up.pasabi.utils.PlatformLock;

/**
 * Created by Iantot on 4/15/2017.
 */

public class SpeechInitializerService extends
    Service
{
    public static final int POLL_INTERVAL = 1;
    public static final int POLL_START_DELAY =
        0;

    private FileHandler handler;
    private Timer timer;

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
    public void onCreate()
    {
        Log.i("Pasabi", "SpeechInitializer alive
            ");
        handler = new FileHandler(PASABIUtils.
            PASABI_DIRECTORY, PASABIUtils.
            PASABI_COMM_FILE);
        timer = new Timer();
    }

    @Override
    public int onStartCommand(Intent intent, int
        flags, int startId)
    {
        // Start polling
        timer.scheduleAtFixedRate(new
            SpeechInitializerService.
            InitializeTranscriberTask(),
            POLL_START_DELAY * 1000,
            POLL_INTERVAL * 1000);

        return Service.START_STICKY;
    }

    @Override
    public void onDestroy()
    {
        Log.i("Pasabi", "SpeechInitializer daed
            ");
        timer.cancel();
    }

    public void broadcastGoSignal()
    {
        Intent goIntent = new Intent("go-intent
            ");
        LocalBroadcastManager.getInstance(this).
            sendBroadcast(goIntent);
    }

    public void broadcastErrorSignal()
    {
        Intent errorIntent = new Intent("error-
            intent");
        LocalBroadcastManager.getInstance(this).
            sendBroadcast(errorIntent);
    }

    public void broadcastTranscribingSignal()
    {
        Intent transcribeIntent = new Intent("
            transcribing-intent");
        LocalBroadcastManager.getInstance(this).
            sendBroadcast(transcribeIntent);
    }

    public void broadcastInitializingSignal()
    {
        Intent initializingIntent = new Intent("
            initializing-intent");
        LocalBroadcastManager.getInstance(this).
            sendBroadcast(initializingIntent);
    }

    public void broadcastTranscribeSignal(String
        transcript)
    {
        Intent transcriptIntent = new Intent("
            transcribe-intent");
        transcriptIntent.putExtra("transcript",
            transcript);


        LocalBroadcastManager.getInstance(this).
            sendBroadcast(transcriptIntent);
```

```
    }


    public class InitializeTranscriberTask
         extends TimerTask
    {
        @Override
        public void run()
        {
            // Resolve activity
            String message = handler.
                 readFromFile();

            if (message.equals(PASABIUtils.
                 GO_SIGNAL))
            {
                broadcastGoSignal();
                stopSelf();
            }
            else if (message.equals(PASABIUtils.
                 ERROR_SIGNAL))
            {
                // send error message
                broadcastErrorSignal();
                stopSelf();
            }
            else if (message.equals(PASABIUtils.
                 TRANSCRIBE_SIGNAL))
            {
                broadcastTranscribingSignal();
                stopSelf();
            }
            else if (message.equals(PASABIUtils.
                 INITIALIZING_SIGNAL))
            {
                broadcastInitializingSignal();
            }
            else
            {
                // There is a  transcription in
                     the file
                // Delegate transcription to
                     SpeechTranscriber
                stopSelf();
            }
        }
    }

}
```

File: app/src/main/java/ph/edu/up/pasabi/sms/ContactAdapter.java

```
package ph.edu.up.pasabi.sms;

import android.content.Context;
import android.database.Cursor;
import android.provider.ContactsContract;
import android.telephony.PhoneNumberUtils;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.CursorAdapter;
import android.widget.TextView;

import java.util.ArrayList;

import ph.edu.up.pasabi.R;

/**
 * Created by Iantot on 4/16/2017.
 */

public class ContactAdapter extends
     CursorAdapter
{
    private ArrayList<Integer> cache;
    public ContactAdapter(Context context,
         Cursor cursor)
    {
        super(context, cursor, 0);
        cache = new ArrayList<>();
    }

    @Override
    public View newView(Context context, Cursor
         cursor, ViewGroup viewGroup) {
        return LayoutInflater.from(context).
             inflate(R.layout.
             layout_sms_contact_item, viewGroup,
              false);
    }

    @Override
    public void bindView(View view, Context
         context, Cursor cursor)
    {
        String contactName = cursor.getString(
             cursor.getColumnIndexOrThrow("
             display_name"));
        String contactNum = cursor.getString(
             cursor.getColumnIndexOrThrow("data4
             "));

        TextView contactNameView = (TextView)
             view.findViewById(R.id.
             contacts_entry_contact_name);
        TextView contactNumView = (TextView)
             view.findViewById(R.id.
             contacts_entry_number);

        Cursor localCursor = context.
             getContentResolver().query(
                 SMSManager.SMS_ALL,
                 new String[]{"thread_id", "
                      address", "date"},
                 "thread_id IS NOT NULL) GROUP BY
                       (thread_id",
                 null,
                 "date desc"
        );

        String threadId = "";
        localCursor.moveToFirst();
        do {
            String address = localCursor.
                 getString(localCursor.
                 getColumnIndexOrThrow("address
                 "));
            if (PhoneNumberUtils.compare(address
                 , contactNum))
                threadId = localCursor.getString
                     (localCursor.
                     getColumnIndexOrThrow("
                     thread_id"));
        } while (localCursor.moveToNext());

        localCursor.close();

        contactNameView.setText(contactName);
        contactNumView.setText(contactNum);

        view.setTag(R.id.tag_contactName,
             contactName);
        view.setTag(R.id.tag_contactNum,
             contactNum);
        view.setTag(R.id.tag_threadId, threadId)
             ;
    }
}
```

File: app/src/main/java/ph/edu/up/pasabi/sms/DirectoryAdapter.java

```
package ph.edu.up.pasabi.sms;


import android.content.Context;
import android.support.v4.content.ContextCompat;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Locale;

import ph.edu.up.pasabi.R;
import ph.edu.up.pasabi.utils.FileHandler;
import ph.edu.up.pasabi.utils.PASABIUtils;

/**
 * Created by Iantot on 4/16/2017.
 */

public class DirectoryAdapter extends
     ArrayAdapter<DirectoryEntryData>
{
    private String defaultModel;
    private Context context;
```

```
    public DirectoryAdapter(Context context,
            ArrayList<DirectoryEntryData> data)
    {
        super(context, 0, data);

        this.context = context;

        FileHandler handler = new FileHandler(
                PASABIUtils.PASABI_DIRECTORY,
                PASABIUtils.PASABI_CONFIG_FILE);
        defaultModel = handler.readFromFile();
    }

    public void updateAdapter()
    {
        FileHandler handler = new FileHandler(
                PASABIUtils.PASABI_DIRECTORY,
                PASABIUtils.PASABI_CONFIG_FILE);
        defaultModel = handler.readFromFile();
    }

    @Override
    public View getView(int position, View
            convertView, ViewGroup parent)
    {
        // Get the data item for this position
        DirectoryEntryData data = getItem(
                position);

        // Check if an existing view is being
                reused, otherwise inflate the view
        if (convertView == null) {
            convertView = LayoutInflater.from(
                    getContext()).inflate(R.layout.
                    layout_directory_item, parent,
                    false);
        }

        String modelName = data.getName();
        SimpleDateFormat dateFormat = new
                SimpleDateFormat("dd/MM/yyyy",
                Locale.US);
        String formattedDate = dateFormat.format
                (data.getDateModified());

        if (modelName.equals(defaultModel))
        {
            convertView.setBackgroundColor(
                    ContextCompat.getColor(context,
                    R.color.lightMaroon));
        }
        else
        {
            convertView.setBackgroundColor(
                    ContextCompat.getColor(context,
                    R.color.transparent));
        }

        // Lookup view for data population
        TextView directoryName = (TextView)
                convertView.findViewById(R.id.
                directory_name);
        TextView directoryLastModified = (
                TextView) convertView.findViewById(
                R.id.directory_last_access);

        // Populate the data into the template
                view using the data object
        directoryName.setText(modelName);
        directoryLastModified.setText(
                formattedDate);

        convertView.setTag(R.id.
                tag_directoryName, modelName);

        // Return the completed view to render
                on screen
        return convertView;
    }
}
```

File: app/src/main/java/ph/edu/up/pasabi/sms/DirectoryEntryData.java

```
package ph.edu.up.pasabi.sms;

import java.util.Date;

public class DirectoryEntryData
{
    private String name;
    private Date dateModified;

    public DirectoryEntryData(String name, Date
            dateModified)
    {
        this.name = name;
        this.dateModified = dateModified;
    }

    public String getName(){
        return this.name;
    }

    public Date getDateModified()
    {
        return this.dateModified;
    }

}
```

File: app/src/main/java/ph/edu/up/pasabi/sms/SMSConvoAdapter.java

```
package ph.edu.up.pasabi.sms;

import android.app.Activity;
import android.content.Context;
import android.database.Cursor;
import android.net.Uri;
import android.provider.ContactsContract;
import android.text.format.DateUtils;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.CursorAdapter;
import android.widget.TextView;
import ph.edu.up.pasabi.R;

import java.util.Date;
import java.util.List;

/**
 * Created by Iantot on 4/12/2017.
 */

public class SMSConvoAdapter extends
        CursorAdapter
{
    private static final int
            SMS_CONVO_ENTRY_LAYOUT = R.layout.
            layout_sms_convo_item;


    public SMSConvoAdapter(Context context,
            Cursor cursor)
    {
        super(context, cursor, 0);
    }

    @Override
    public View newView(Context context, Cursor
            cursor, ViewGroup viewGroup)
    {
        return LayoutInflater.from(context).
                inflate(SMS_CONVO_ENTRY_LAYOUT,
                viewGroup, false);
    }

    @Override
    public void bindView(View view, Context
            context, Cursor cursor)
    {
        String contactName = "";
        String contactNum = cursor.getString(
                cursor.getColumnIndexOrThrow("
                address"));
        String SMSBody = cursor.getString(cursor
                .getColumnIndexOrThrow("body"));
        String threadId = cursor.getString(
                cursor.getColumnIndexOrThrow("
                thread_id"));
        long timestamp = cursor.getLong(cursor.
                getColumnIndexOrThrow("date"));

        String relative = (String) DateUtils.
                getRelativeTimeSpanString(timestamp
                , new Date().getTime(), DateUtils.
                MINUTE_IN_MILLIS, DateUtils.
                FORMAT_ABBREV_ALL);

        TextView timestampView = (TextView) view
```

```
            .findViewById(R.id.
                convo_entry_timestamp);
        TextView contactNumView = (TextView)
                view.findViewById(R.id.
                convo_entry_contact_name);

        timestampView.setText(relative);

        // Get name
        Uri personUri = Uri.withAppendedPath(
                ContactsContract.PhoneLookup.
                CONTENT_FILTER_URI, contactNum);
        Cursor localCursor = context.
                getContentResolver().query(
                personUri, new String[]{
                ContactsContract.Contacts.
                DISPLAY_NAME}, null, null, null);

        if (localCursor.getCount() != 0)
        {
            localCursor.moveToFirst();
            contactName = localCursor.getString(
                    localCursor.getColumnIndex(
                    ContactsContract.Contacts.
                    DISPLAY_NAME));
            contactNumView.setText(contactName);
        }
        else
        {
            contactNumView.setText(contactNum);
        }

        localCursor.close();

        TextView SMSBodyView = (TextView) view.
                findViewById(R.id.
                convo_entry_message);
        SMSBodyView.setText(SMSBody);

        view.setTag(R.id.tag_threadId, threadId)
                ;
        view.setTag(R.id.tag_contactName,
                contactName);
        view.setTag(R.id.tag_contactNum,
                contactNum);
    }
}
```

File: app/src/main/java/ph/edu/up/pasabi/sms/SMSGrandObserver.java

```
package ph.edu.up.pasabi.sms;

import android.content.ContentResolver;
import android.database.ContentObserver;
import android.database.Cursor;
import android.net.Uri;
import android.os.Handler;
import android.util.Log;
import android.widget.CursorAdapter;

import java.util.ArrayList;

/**
 * Created by Iantot on 4/15/2017.
 */

public class SMSGrandObserver extends
        ContentObserver
{
    private static SMSGrandObserver instance;
    private ArrayList<SMSObserver> observers;

    public static SMSGrandObserver getInstance()
    {
        if (instance == null)
            instance = new SMSGrandObserver(new
                    Handler());

        return instance;
    }

    private SMSGrandObserver(Handler handler)
    {
        super(handler);

        observers = new ArrayList<>();
    }

    public void registerObserver(SMSObserver
            observer)
    {
        observers.add(observer);
    }

    @Override
    public void onChange(boolean selfChange, Uri
            uri)
    {
        Log.i("GrandObserver", "Woot");

        for (SMSObserver obs : observers)
            obs.onChange(uri);
    }
}
```

File: app/src/main/java/ph/edu/up/pasabi/sms/SMSManager.java

```
package ph.edu.up.pasabi.sms;

import android.content.Context;
import android.net.Uri;
import android.provider.ContactsContract;
import android.util.Log;

import java.util.ArrayList;
import java.util.List;

/**
 * Created by Iantot on 4/14/2017.
 */
public class SMSManager
{

    public static final String PASABI_AUDIO_FILE
            = "record.wav";

    public static final Uri SMS_ALL = Uri.parse
            ("content://sms/");
    public static final Uri SMS_INBOX = Uri.
            parse("content://sms/inbox");
    public static final Uri SMS_SENT = Uri.parse
            ("content://sms/sent");
    public static final Uri SMS_CONTACTS =
            ContactsContract.Contacts.CONTENT_URI;

    public static final int
            EVENT_RECEIVED_MESSAGE = 12;
    public static final int EVENT_SENT_MESSAGE =
            11;

    public static final String OUTGOING_SMS =
            "2";
    public static final String INCOMING_SMS =
            "1";

    public static final int GET_CONTACT_NUMBER =
            1;

    private static SMSManager instance;


    public static SMSManager getInstance()
    {
        if (instance == null)
            instance = new SMSManager();

        return instance;
    }

    private SMSManager()
    {
    }
}
```

File: app/src/main/java/ph/edu/up/pasabi/sms/SMSMessageAdapter.java

```
package ph.edu.up.pasabi.sms;

import android.app.Activity;
import android.content.Context;
import android.database.Cursor;
import android.graphics.Color;
import android.support.v4.content.ContextCompat;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AbsListView;
import android.widget.CursorAdapter;
import android.widget.RelativeLayout;
import android.widget.RelativeLayout.
    LayoutParams;
```

```
import android.widget.TextView;

import ph.edu.up.pasabi.R;

/**
 * Created by Iantot on 4/13/2017.
 */

public class SMSMessageAdapter extends
    CursorAdapter
{
    private static final int
        SMS_MESSAGE_ENTRY_LAYOUT = R.layout.
        layout_sms_message_item;

    public SMSMessageAdapter(Context context,
        Cursor cursor) {
        super(context, cursor, 0);
    }

    @Override
    public View newView(Context context, Cursor
        cursor, ViewGroup viewGroup)
    {
        return LayoutInflater.from(context).
            inflate(SMS_MESSAGE_ENTRY_LAYOUT,
            viewGroup, false);
    }

    @Override
    public void bindView(View view, Context
        context, Cursor cursor)
    {
        TextView SMSBodyView = (TextView) view.
            findViewById(R.id.
            text_message_content);
        RelativeLayout SMSContainer = (
            RelativeLayout) view.findViewById(R
            .id.text_message_container);

        int offset = (int) context.getResources
            ().getDimension(R.dimen.
            text_message_offset);
        String type = cursor.getString(cursor.
            getColumnIndexOrThrow("type"));
        String SMSBody = cursor.getString(cursor
            .getColumnIndexOrThrow("body"));

        LayoutParams messageParams = (
            LayoutParams) SMSBodyView.
            getLayoutParams();
        LayoutParams containerParams = (
            LayoutParams) SMSContainer.
            getLayoutParams();

        if (SMSManager.OUTGOING_SMS.equals(type)
            )
        {
            //params.addRule(RelativeLayout.
                END_OF, R.id.convo_margin_left)
                ;
            SMSContainer.setPadding(offset, 0,
                0, 0);

            messageParams.addRule(RelativeLayout
                .ALIGN_PARENT_END);
            containerParams.addRule(
                RelativeLayout.ALIGN_PARENT_END
                );
            SMSBodyView.setBackgroundResource(R.
                drawable.balloon_outgoing);

            // Clean old properties
            messageParams.removeRule(
                RelativeLayout.
                ALIGN_PARENT_START);
            containerParams.removeRule(
                RelativeLayout.
                ALIGN_PARENT_START);
        }
        else if (SMSManager.INCOMING_SMS.equals(
            type))
        {
            //params.addRule(RelativeLayout.
                START_OF, R.id.
                convo_margin_right);
            SMSContainer.setPadding(0, 0, offset
                , 0);
            messageParams.addRule(RelativeLayout
                .ALIGN_PARENT_START);
            containerParams.addRule(
                RelativeLayout.
                ALIGN_PARENT_START);
            SMSBodyView.setBackgroundResource(R.
                drawable.balloon_incoming);

            // Clean old properties
            messageParams.removeRule(
                RelativeLayout.ALIGN_PARENT_END
                );
            containerParams.removeRule(
                RelativeLayout.ALIGN_PARENT_END
                );
        }

        SMSContainer.setLayoutParams(
            containerParams);

        SMSBodyView.setLayoutParams(
            messageParams);
        SMSBodyView.setText(SMSBody);
    }
}
```

File: app/src/main/java/ph/edu/up/pasabi/sms/SMSObserver.java

```
package ph.edu.up.pasabi.sms;

import android.net.Uri;

/**
 * Created by Iantot on 4/15/2017.
 */

public interface SMSObserver
{
    public void onChange(Uri uri);
}
```

File: app/src/main/java/ph/edu/up/pasabi/utils/ExtAudioRecorder.java

```
package ph.edu.up.pasabi.utils;

/**
 * Created by Iantot on 4/5/2017.
 */
import java.io.File;
import java.io.IOException;
import java.io.RandomAccessFile;

import android.media.AudioFormat;
import android.media.AudioRecord;
import android.media.MediaRecorder;
import android.media.MediaRecorder.AudioSource;
import android.util.Log;

public class ExtAudioRecorder
{
    private final static int[] sampleRates =
        {44100, 22050, 11025, 8000};

    public static ExtAudioRecorder getInstance(
        Boolean recordingCompressed)
    {
        ExtAudioRecorder result = null;

        if (recordingCompressed)
        {
            result = new ExtAudioRecorder(false,
                AudioSource.MIC, sampleRates
                [3], AudioFormat.
                CHANNEL_CONFIGURATION_MONO,
                AudioFormat.ENCODING_PCM_16BIT)
                ;
        }
        else
        {
            int i=0;
            do
            {
                result = new ExtAudioRecorder(
                    true, AudioSource.MIC,
                    sampleRates[i], AudioFormat
                    .CHANNEL_CONFIGURATION_MONO
                    , AudioFormat.
                    ENCODING_PCM_16BIT);
            }
            while((++i < sampleRates.length) &
                !(result.getState() ==
                ExtAudioRecorder.State.
                INITIALIZING));
        }
```

```
    return result;
}

/**
 * INITIALIZING : recorder is initializing;
 * READY : recorder has been initialized,
      recorder not yet started
 * RECORDING : recording
 * ERROR : reconstruction needed
 * STOPPED: reset needed
 */
public enum State {INITIALIZING, READY,
     RECORDING, ERROR, STOPPED};

public static final boolean
     RECORDING_UNCOMPRESSED = true;
public static final boolean
     RECORDING_COMPRESSED = false;

// The interval in which the recorded
     samples are output to the file
// Used only in uncompressed mode
private static final int TIMER_INTERVAL =
     120;

// Toggles uncompressed recording on/off;
     RECORDING_UNCOMPRESSED /
     RECORDING_COMPRESSED
private boolean rUncompressed;

// Recorder used for uncompressed recording
private AudioRecord audioRecorder = null;

// Recorder used for compressed recording
private MediaRecorder mediaRecorder = null;

// Stores current amplitude (only in
     uncompressed mode)
private int cAmplitude= 0;

// Output file path
private String filePath = null;

// Recorder state; see State
private State state;

// File writer (only in uncompressed mode)
private RandomAccessFile randomAccessWriter;

// Number of channels, sample rate, sample
     size(size in bits), buffer size, audio
     source, sample size(see AudioFormat)
private short nChannels;
private int sRate;
private short bSamples;
private int bufferSize;
private int aSource;
private int aFormat;

// Number of frames written to file on each
     output(only in uncompressed mode)
private int framePeriod;

// Buffer for output(only in uncompressed
     mode)
private byte[] buffer;

// Number of bytes written to file after
     header(only in uncompressed mode)
// after stop() is called, this size is
     written to the header/data chunk in the
      wave file
private int payloadSize;

/**
 *
 * Returns the state of the recorder in a
      RehearsalAudioRecord.State typed
      object.
 * Useful, as no exceptions are thrown.
 *
 * @return recorder state
 */
public State getState()
{
    return state;
}

/*
*
* Method used for recording.
*
*/
private AudioRecord.
     OnRecordPositionUpdateListener
     updateListener = new AudioRecord.
     OnRecordPositionUpdateListener()
{
    public void onPeriodicNotification(
         AudioRecord recorder)
    {
        audioRecorder.read(buffer, 0, buffer
             .length); // Fill buffer
        try
        {
            if (randomAccessWriter != null)
            {
                randomAccessWriter.write(
                     buffer); // Write
                     buffer to file
                payloadSize += buffer.length
                     ;
                if (bSamples == 16)
                {
                    for (int i=0; i<buffer.
                         length/2; i++)
                    { // 16bit sample size
                        short curSample =
                             getShort(buffer
                             [i*2], buffer[i
                             *2+1]);
                        if (curSample >
                             cAmplitude)
                        { // Check amplitude
                            cAmplitude =
                                 curSample;
                        }
                    }
                }
                else
                { // 8bit sample size
                    for (int i=0; i<buffer.
                         length; i++)
                    {
                        if (buffer[i] >
                             cAmplitude)
                        { // Check amplitude
                            cAmplitude =
                                 buffer[i];
                        }
                    }
                }
            }
        }
        catch (IOException e)
        {

            Log.e(ExtAudioRecorder.class.
                 getName(), "Error occured
                 in updateListener,
                 recording is aborted", e);
            //stop();
        }
    }

    public void onMarkerReached(AudioRecord
         recorder)
    {
        // NOT USED
    }
};
/**
 *
 *
 * Default constructor
 *
 * Instantiates a new recorder, in case of
      compressed recording the parameters
      can be left as 0.
 * In case of errors, no exception is thrown
      , but the state is set to ERROR
 *
 */
public ExtAudioRecorder(boolean uncompressed
     , int audioSource, int sampleRate, int
     channelConfig, int audioFormat)
{
    try
    {
        rUncompressed = uncompressed;
        if (rUncompressed)
        { // RECORDING_UNCOMPRESSED
            if (audioFormat == AudioFormat.
                 ENCODING_PCM_16BIT)
```

```
            {
                bSamples = 16;
            }
            else
            {
                bSamples = 8;
            }

            if (channelConfig == AudioFormat
                .CHANNEL_CONFIGURATION_MONO
                )
            {
                nChannels = 1;
            }
            else
            {
                nChannels = 2;
            }

            aSource = audioSource;
            sRate   = sampleRate;
            aFormat = audioFormat;

            framePeriod = sampleRate *
                TIMER_INTERVAL / 1000;
            bufferSize = framePeriod * 2 *
                bSamples * nChannels / 8;
            if (bufferSize < AudioRecord.
                getMinBufferSize(sampleRate
                , channelConfig,
                audioFormat))
            { // Check to make sure buffer
                size is not smaller than
                the smallest allowed one
                bufferSize = AudioRecord.
                    getMinBufferSize(
                    sampleRate,
                    channelConfig,
                    audioFormat);
                // Set frame period and
                    timer interval
                    accordingly
                framePeriod = bufferSize / (
                     2 * bSamples *
                    nChannels / 8 );
                Log.w(ExtAudioRecorder.class
                    .getName(), "Increasing
                     buffer size to " +
                    Integer.toString(
                    bufferSize));
            }

            audioRecorder = new AudioRecord(
                audioSource, sampleRate,
                channelConfig, audioFormat,
                 bufferSize);

            if (audioRecorder.getState() !=
                AudioRecord.
                STATE_INITIALIZED)
                throw new Exception("
                    AudioRecord
                    initialization failed")
                    ;
            audioRecorder.
                setRecordPositionUpdateListener
                (updateListener);
            audioRecorder.
                setPositionNotificationPeriod
                (framePeriod);
        } else
        { // RECORDING_COMPRESSED
            mediaRecorder = new
                MediaRecorder();
            mediaRecorder.setAudioSource(
                MediaRecorder.AudioSource.
                MIC);
            mediaRecorder.setOutputFormat(
                MediaRecorder.OutputFormat.
                THREE_GPP);
            mediaRecorder.setAudioEncoder(
                MediaRecorder.AudioEncoder.
                AMR_NB);
        }
        cAmplitude = 0;
        filePath = null;
        state = State.INITIALIZING;
    } catch (Exception e)
    {
        if (e.getMessage() != null)
        {
            Log.e(ExtAudioRecorder.class.
                getName(), e.getMessage());
        }
        else
        {
            Log.e(ExtAudioRecorder.class.
                getName(), "Unknown error
                occured while initializing
                recording");
        }
        state = State.ERROR;
    }
}

/**
 * Sets output file path, call directly
    after construction/reset.
 *
 * @param
 *
 */
public void setOutputFile(String argPath)
{
    try
    {
        if (state == State.INITIALIZING)
        {
            filePath = argPath;
            if (!rUncompressed)
            {
                mediaRecorder.setOutputFile(
                    filePath);
            }
        }
    }
    catch (Exception e)
    {
        if (e.getMessage() != null)
        {
            Log.e(ExtAudioRecorder.class.
                getName(), e.getMessage());
        }
        else
        {
            Log.e(ExtAudioRecorder.class.
                getName(), "Unknown error
                occured while setting
                output path");
        }
        state = State.ERROR;
    }
}

/**
 *
 * Returns the largest amplitude sampled
    since the last call to this method.
 *
 * @return returns the largest amplitude
    since the last call, or 0 when not in
    recording state.
 *
 */
public int getMaxAmplitude()
{
    if (state == State.RECORDING)
    {
        if (rUncompressed)
        {
            int result = cAmplitude;
            cAmplitude = 0;
            return result;
        }
        else
        {
            try
            {
                return mediaRecorder.
                    getMaxAmplitude();
            }
            catch (IllegalStateException e)
            {
                return 0;
            }
        }
    }
    else
    {
        return 0;
    }
}
```

```
/**
 *
 * Prepares the recorder for recording, in
      case the recorder is not in the
      INITIALIZING state and the file path
      was not set
 * the recorder is set to the ERROR state,
      which makes a reconstruction necessary
      .
 * In case uncompressed recording is toggled
      , the header of the wave file is
      written.
 * In case of an exception, the state is
      changed to ERROR
 *
 */
public void prepare()
{
    try
    {
        if (state == State.INITIALIZING)
        {
            if (rUncompressed)
            {
                if ((audioRecorder.getState
                     () == AudioRecord.
                     STATE_INITIALIZED) & (
                     filePath != null))
                {
                    // write file header
                    randomAccessWriter = new
                          RandomAccessFile(
                         filePath, "rw");

                    randomAccessWriter.
                         setLength(0); //
                         Set file length to
                         0, to prevent
                         unexpected behavior
                          in case the file
                         already existed
                    randomAccessWriter.
                         writeBytes("RIFF");
                    randomAccessWriter.
                         writeInt(0); //
                         Final file size not
                          known yet, write 0
                    randomAccessWriter.
                         writeBytes("WAVE");
                    randomAccessWriter.
                         writeBytes("fmt ");
                    randomAccessWriter.
                         writeInt(Integer.
                         reverseBytes(16));
                         // Sub-chunk size,
                         16 for PCM
                    randomAccessWriter.
                         writeShort(Short.
                         reverseBytes((short
                         ) 1)); //
                         AudioFormat, 1 for
                         PCM
                    randomAccessWriter.
                         writeShort(Short.
                         reverseBytes(
                         nChannels));//
                         Number of channels,
                          1 for mono, 2 for
                         stereo
                    randomAccessWriter.
                         writeInt(Integer.
                         reverseBytes(sRate)
                         ); // Sample rate
                    randomAccessWriter.
                         writeInt(Integer.
                         reverseBytes(sRate*
                         bSamples*nChannels
                         /8)); // Byte rate,
                          SampleRate*
                         NumberOfChannels*
                         BitsPerSample/8
                    randomAccessWriter.
                         writeShort(Short.
                         reverseBytes((short
                         )(nChannels*
                         bSamples/8))); //
                         Block align,
                         NumberOfChannels*
                         BitsPerSample/8
                    randomAccessWriter.
                         writeShort(Short.
                         reverseBytes(
                         bSamples)); // Bits
                          per sample
                    randomAccessWriter.
                         writeBytes("data");
                    randomAccessWriter.
                         writeInt(0); //
                         Data chunk size not
                          known yet, write 0

                    buffer = new byte[
                         framePeriod*
                         bSamples/8*
                         nChannels];
                    state = State.READY;
                }
                else
                {
                    Log.e(ExtAudioRecorder.
                         class.getName(), "
                         prepare() method
                         called on
                         uninitialized
                         recorder");
                    state = State.ERROR;
                }
            }
            else
            {
                mediaRecorder.prepare();
                state = State.READY;
            }
        }
        else
        {
            Log.e(ExtAudioRecorder.class.
                 getName(), "prepare()
                 method called on illegal
                 state");
            release();
            state = State.ERROR;
        }
    }
    catch(Exception e)
    {
        if (e.getMessage() != null)
        {
            Log.e(ExtAudioRecorder.class.
                 getName(), e.getMessage());
        }
        else
        {
            Log.e(ExtAudioRecorder.class.
                 getName(), "Unknown error
                 occured in prepare()");
        }
        state = State.ERROR;
    }
}

/**
 *
 *
 *  Releases the resources associated with
      this class, and removes the
      unnecessary files, when necessary
 *
 */
public void release()
{
    if (state == State.RECORDING)
    {
        stop();
    }
    else
    {
        if ((state == State.READY) & (
             rUncompressed))
        {
            try
            {
                randomAccessWriter.close();
                     // Remove prepared file
                randomAccessWriter = null;
            }
            catch (IOException e)
            {
                Log.e(ExtAudioRecorder.class
                     .getName(), "I/O
                     exception occured while
                      closing output file");
            }
            (new File(filePath)).delete();
```

```
        }
    }

    if (rUncompressed)
    {
        if (audioRecorder != null)
        {
            audioRecorder.release();
        }
    }
    else
    {
        if (mediaRecorder != null)
        {
            mediaRecorder.release();
        }
    }
}

/**
 *
 *
 * Resets the recorder to the INITIALIZING
     state, as if it was just created.
 * In case the class was in RECORDING state,
     the recording is stopped.
 * In case of exceptions the class is set to
     the ERROR state.
 *
 */
public void reset()
{
    try
    {
        if (state != State.ERROR)
        {
            release();
            filePath = null; // Reset file
                 path
            cAmplitude = 0; // Reset
                 amplitude
            if (rUncompressed)
            {
                audioRecorder = new
                     AudioRecord(aSource,
                     sRate, nChannels+1,
                     aFormat, bufferSize);
            }
            else
            {
                mediaRecorder = new
                     MediaRecorder();
                mediaRecorder.setAudioSource
                     (MediaRecorder.
                     AudioSource.MIC);
                mediaRecorder.
                     setOutputFormat(
                     MediaRecorder.
                     OutputFormat.THREE_GPP)
                     ;
                mediaRecorder.
                     setAudioEncoder(
                     MediaRecorder.
                     AudioEncoder.AMR_NB);
            }
            state = State.INITIALIZING;
        }
    }
    catch (Exception e)
    {
        Log.e(ExtAudioRecorder.class.getName
             (), e.getMessage());
        state = State.ERROR;
    }
}

/**
 *
 *
 * Starts the recording, and sets the state
     to RECORDING.
 * Call after prepare().
 *
 */
public void start()
{
    if (state == State.READY)
    {
        if (rUncompressed)
        {
            payloadSize = 0;
            audioRecorder.startRecording();
            audioRecorder.read(buffer, 0,
                 buffer.length);
        }
        else
        {
            mediaRecorder.start();
        }
        state = State.RECORDING;
    }
    else
    {
        Log.e(ExtAudioRecorder.class.getName
             (), "start() called on illegal
             state");
        state = State.ERROR;
    }
}

/**
 *
 *
 *  Stops the recording, and sets the state
     to STOPPED.
 * In case of further usage, a reset is
     needed.
 * Also finalizes the wave file in case of
     uncompressed recording.
 *
 */
public void stop()
{
    if (state == State.RECORDING)
    {
        if (rUncompressed)
        {
            audioRecorder.stop();

            try
            {
                randomAccessWriter.seek(4);
                     // Write size to RIFF
                     header
                randomAccessWriter.writeInt(
                     Integer.reverseBytes
                     (36+payloadSize));

                randomAccessWriter.seek(40);
                     // Write size to
                     Subchunk2Size field
                randomAccessWriter.writeInt(
                     Integer.reverseBytes(
                     payloadSize));

                randomAccessWriter.close();

                randomAccessWriter = null;
            }
            catch(IOException e)
            {
                Log.e(ExtAudioRecorder.class
                     .getName(), "I/O
                     exception occured while
                      closing output file");
                state = State.ERROR;
            }
        }
        else
        {
            mediaRecorder.stop();
        }
        state = State.STOPPED;
    }
    else
    {
        Log.e(ExtAudioRecorder.class.getName
             (), "stop() called on illegal
             state");
        state = State.ERROR;
    }
}

/*
 *
 * Converts a byte[2] to a short, in
     LITTLE_ENDIAN format
 *
 */
private short getShort(byte argB1, byte
     argB2)
{
    return (short)(argB1 | (argB2 << 8));
}
```

```
}
```

File: app/src/main/java/ph/edu/up/pasabi/utils/FileHandler.java

```
package ph.edu.up.pasabi.utils;

import android.os.Environment;
import android.util.Log;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;

/**
 * Created by Iantot on 3/26/2017.
 */

public class FileHandler
{
    private FileInputStream input;
    private FileOutputStream output;

    private File filepath;
    public String filename;

    public FileHandler(String directory, String
        filename)
    {
        // get the path to sdcard
        File sdcard = Environment.
            getExternalStorageDirectory();

        // to this path add a new directory path
        this.filepath = new File(sdcard.
            getAbsolutePath() + directory);
        this.filename = filename;

        File file = new File(this.filepath, this
            .filename);
        Log.i("MainActivity", "Filepath na ina-
            access: " +file.getAbsolutePath())
            ;
    }

    public String getFilepath()
    {
        return filepath.getAbsolutePath();
    }

    public String readFromFile()
    {
        File file = new File(filepath, filename)
            ;
        StringBuilder fromFile = new
            StringBuilder();

        try
        {
            // Read the file and store the
                string
            input = new FileInputStream(file);
            int content;

            while ((content = input.read()) !=
                -1)
            {
                fromFile.append((char) content);
            }

            input.close();
        }
        catch (Exception e)
        {
            return null;
        }

        return fromFile.toString();
    }

    public boolean writeToFile(String message)
    {
        // create this directory if not already
            created
        filepath.mkdir();

        // create the file in which we will
            write the contents
        File file = new File(filepath, filename
            );

        try
        {
            output = new FileOutputStream(file);
            output.write(message.getBytes());
            output.close();
        }
        catch (Exception e)
        {
            return false;
        }

        return true;
    }
}
```

File: app/src/main/java/ph/edu/up/pasabi/utils/PASABIUtils.java

```
package ph.edu.up.pasabi.utils;

import android.os.Environment;

import java.io.File;

/**
 * Created by Iantot on 4/8/2017.
 */

public class PASABIUtils
{
    public static final int LOADER_INBOX = 1;
    public static final int LOADER_OUTBOX = 2;

    public static final String AUDIO_FILE_EXT =
        ".wav";
    public static final String AUDIO_FILE_NAME =
        "record";
    public static final String PASABI_DIRECTORY
        = "/pasabi";
    public static final String
        PASABI_MODELS_DIRECTORY = "/models";

    public static final String PASABI_COMM_FILE
        = "comm.gg";
    public static final String PASABI_FLAG_FILE
        = "lock.gg";
    public static final String
        PASABI_CONFIG_FILE = "config.gg";

    public static final String GO_SIGNAL = "G";
    public static final String ERROR_SIGNAL = "E
        ";
    public static final String TRANSCRIBE_SIGNAL
        = "T";
    public static final String
        INITIALIZING_SIGNAL = "I";

    public static String getSDCardPath()
    {
        String externalStorage = Environment.
            getExternalStorageDirectory().
            getPath();
        return externalStorage;
    }

    public static String getAbsolutePath()
    {
        // /sdcard
        String externalStorage = Environment.
            getExternalStorageDirectory().
            getPath();
        File directory = new File(
            externalStorage, PASABI_DIRECTORY);

        // Create PASABI folder if it does not
            exist
        if (!directory.exists())
            directory.mkdirs();

        return directory.getAbsolutePath();
    }

    public static String getAudioFilePath()
    {
        return (PASABIUtils.getAbsolutePath() +
            "/" + AUDIO_FILE_NAME +
            AUDIO_FILE_EXT);
    }
}
```

File: app/src/main/java/ph/edu/up/pasabi/utils/PlatformLock.java

```
package ph.edu.up.pasabi.utils;

import ph.edu.up.pasabi.utils.FileHandler;

/**
 * Created by Iantot on 3/26/2017.
 */

public class PlatformLock
{
    public static final String JAVA = "jv";
    public static final String PYTHON = "py";

    private FileHandler fileHandler;

    public PlatformLock(String directory, String
          filename)
    {
        // Instantiate FileHandler
        fileHandler = new FileHandler(directory,
              filename);
    }

    public boolean isJava()
    {
        String line = fileHandler.readFromFile()
              ;
        return (JAVA.equals(line));
    }

    public boolean isPython()
    {
        String line = fileHandler.readFromFile()
              ;
        return (PYTHON.equals(line));
    }

    public void setJava()
    {
        fileHandler.writeToFile(JAVA);
    }

    public void setPython()
    {
        fileHandler.writeToFile(PYTHON);
    }
}
```

File: app/src/main/res/drawable/balloon_incoming.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<shape xmlns:android="http://schemas.android.com
     /apk/res/android">
    <solid android:color="@color/lightGray" />
    <stroke
        android:width="@dimen/
             balloon_border_size"
        android:color="@color/black" />
    <corners android:radius="@dimen/
         balloon_border_radius" />
</shape>
```

File: app/src/main/res/drawable/balloon_outgoing.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<shape xmlns:android="http://schemas.android.com
     /apk/res/android">
    <solid android:color="@color/lightMaroon" />
    <stroke
        android:width="@dimen/
             balloon_border_size"
        android:color="@color/black" />
    <corners android:radius="@dimen/
         balloon_border_radius" />
</shape>
```

File: app/src/main/res/drawable/convo_edittext.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<shape xmlns:android="http://schemas.android.com
     /apk/res/android">
    <solid android:color="#FFFFFF" />
    <stroke
        android:width="1dip"
        android:color="@color/maroon" />
    <corners android:radius="@dimen/
         convo_edittext_radius" />
    <padding
        android:bottom="0dip"
        android:left="0dip"
        android:right="0dip"
        android:top="0dip" />
</shape>
```

File: app/src/main/res/drawable/convo_pressed.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.
     com/apk/res/android">

    <item
        android:state_accelerated="false"
        android:drawable="@color/lightMaroon" />
    <item
        android:state_pressed="true"
        android:drawable="@color/lightMaroon" />
    <item
        android:state_focused="true"
        android:drawable="@color/lightMaroon" />
    <item
        android:drawable="@color/transparent" />
</selector>
```

File: app/src/main/res/drawable/speech_button_big_normal.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.
     android.com/apk/res/android" >
    <item>
        <shape android:shape="oval">
            <solid android:color="@color/
                 transparent" />
            <stroke android:width="1px"
                    android:color="@color/black
                         "/>
            <size android:width="100dp"
                  android:height="100dp" />
        </shape>
    </item>
    <item>
        <bitmap android:src="@drawable/mic_on"
                android:gravity="center" />
    </item>
</layer-list>
```

File: app/src/main/res/drawable/speech_button_big_off.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.
     android.com/apk/res/android" >
    <item>
        <shape android:shape="oval">
            <solid android:color="@color/
                 transparent" />
            <stroke android:width="1px"
                    android:color="@color/black
                         "/>
            <size android:width="100dp"
                  android:height="100dp" />
        </shape>
    </item>
    <item>
        <bitmap android:src="@drawable/mic_off"
                android:gravity="center" />
    </item>
</layer-list>
```

File: app/src/main/res/drawable/speech_button_big_pressed.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.
     android.com/apk/res/android" >
    <item>
        <shape android:shape="oval">
            <solid android:color="@color/
                 lightMaroon" />
            <stroke android:width="1px"
                    android:color="@color/maroon
                         "/>
            <size android:width="100dp"
                  android:height="100dp" />
        </shape>
    </item>
    <item>
        <bitmap android:src="@drawable/mic_on"
                android:gravity="center" />
    </item>
</layer-list>
```

File: app/src/main/res/layout/activity_browse_directory.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/
         apk/res/android"
    xmlns:tools="http://schemas.android.com/
         tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
```

```
android:layout_height="match_parent"
android:paddingBottom="@dimen/
    activity_vertical_margin"
android:paddingLeft="@dimen/
    activity_horizontal_margin"
android:paddingRight="@dimen/
    activity_horizontal_margin"
android:paddingTop="@dimen/
    activity_vertical_margin"
tools:context="ph.edu.up.pasabi.activity.
    MainActivity">

<!--    Loading
    <RelativeLayout
        android:id="@+id/loadingPanel"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center" >

        <ProgressBar
            android:layout_width="
                wrap_content"
            android:layout_height="
                wrap_content"
            android:indeterminate="true" />
    </RelativeLayout>
-->

<RelativeLayout
    android:id="@+id/directory_topbar"
    android:layout_width="match_parent"
    android:layout_height="@dimen/
        directory_topbar_height"
    android:background="@color/maroon">
    <ImageButton
        android:id="@+id/
            directory_topbar_back_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/back_button"
        android:background="@null"
        android:layout_alignParentStart="
            true"
        android:layout_alignParentBottom="
            true"
        android:layout_marginStart="@dimen/
            convo_topbar_padding_bottom"
        android:layout_marginEnd="@dimen/
            convo_topbar_padding_bottom"
        android:onClick="goBack"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Models"
        android:layout_alignParentBottom="
            true"
        android:layout_centerHorizontal="
            true"
        android:layout_marginBottom="5dp"
        android:textStyle="bold"
        android:textSize="16sp"/>
</RelativeLayout>

<RelativeLayout
    android:background="@color/black"
    android:layout_width="match_parent"
    android:layout_height="@dimen/
        border_height"
    android:layout_above="@+id/
        directory_list">
</RelativeLayout>

<ListView
    android:id="@+id/directory_list"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:paddingLeft="@dimen/
        directory_padding"
    android:paddingRight="@dimen/
        directory_padding"
    android:layout_below="@+id/
        directory_topbar"
    android:layout_above="@+id/
        directory_botbar">
</ListView>

<RelativeLayout
    android:background="@color/black"
    android:layout_width="match_parent"
    android:layout_height="@dimen/
        border_height"
    android:layout_above="@+id/
        directory_botbar">
</RelativeLayout>

<RelativeLayout
    android:id="@+id/directory_botbar"
    android:background="@color/white"
    android:gravity="center_horizontal"

    android:layout_width="match_parent"
    android:layout_height="@dimen/
        directory_botbar_height"
    android:layout_alignParentBottom="true">

    <!--
    <RelativeLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center">
        <ImageButton
            android:id="@+id/
                add_directory_button"
            android:layout_width="
                wrap_content"
            android:layout_height="
                wrap_content"
            android:src="@drawable/
                add_button"
            android:layout_centerInParent="
                true"
            android:background="#FFFFFF"
            android:layout_margin="2dp"/>
        <TextView
            android:layout_width="
                wrap_content"
            android:layout_height="
                wrap_content"

            android:layout_marginTop="2dp"
            android:text="@string/
                directory_add_button"
            android:textSize="12sp"
            android:layout_below="@+id/
                add_directory_button" />
    </RelativeLayout>
    -->

</RelativeLayout>
</RelativeLayout>
```

File: app/src/main/res/layout/activity_contacts.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/
        apk/res/android"
    xmlns:tools="http://schemas.android.com/
        tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/
        activity_vertical_margin"
    android:paddingLeft="@dimen/
        activity_horizontal_margin"
    android:paddingRight="@dimen/
        activity_horizontal_margin"
    android:paddingTop="@dimen/
        activity_vertical_margin"
    tools:context="ph.edu.up.pasabi.activity.
        MainActivity">

    <!--    Loading
        <RelativeLayout
            android:id="@+id/loadingPanel"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:gravity="center" >

            <ProgressBar
                android:layout_width="
                    wrap_content"
                android:layout_height="
                    wrap_content"
                android:indeterminate="true" />
        </RelativeLayout>
    -->

    <RelativeLayout
        android:id="@+id/contacts_topbar"
        android:layout_width="match_parent"
        android:layout_height="@dimen/
            directory_topbar_height"
```

```
            android:background="@color/maroon">
            <ImageButton
                android:id="@+id/
                    contacts_topbar_back_button"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:src="@drawable/back_button"
                android:background="@null"
                android:layout_alignParentStart="
                    true"
                android:layout_alignParentBottom="
                    true"
                android:layout_marginStart="@dimen/
                    convo_topbar_padding_bottom"
                android:layout_marginEnd="@dimen/
                    convo_topbar_padding_bottom"
                android:onClick="goBack"/>

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Contacts"
                android:layout_alignParentBottom="
                    true"
                android:layout_centerHorizontal="
                    true"
                android:layout_marginBottom="5dp"
                android:textStyle="bold"
                android:textSize="16sp"/>
        </RelativeLayout>

        <RelativeLayout
            android:background="@color/black"
            android:layout_width="match_parent"
            android:layout_height="@dimen/
                border_height"
            android:layout_above="@+id/contacts_list
                ">
        </RelativeLayout>

        <ListView
            android:id="@+id/contacts_list"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingLeft="@dimen/
                directory_padding"
            android:paddingRight="@dimen/
                directory_padding"
            android:layout_below="@+id/
                contacts_topbar"
            android:layout_above="@+id/
                contacts_botbar">
        </ListView>

        <RelativeLayout
            android:background="@color/black"
            android:layout_width="match_parent"
            android:layout_height="@dimen/
                border_height"
            android:layout_above="@+id/
                contacts_botbar">
        </RelativeLayout>

        <RelativeLayout
            android:id="@+id/contacts_botbar"
            android:background="@color/white"
            android:gravity="center_horizontal"

            android:layout_width="match_parent"
            android:layout_height="@dimen/
                contacts_botbar_height"
            android:layout_alignParentBottom="true">
        </RelativeLayout>
    </RelativeLayout>
```

File: app/src/main/res/layout/activity_conversation.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/
        apk/res/android"
    xmlns:tools="http://schemas.android.com/
        tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/
        activity_horizontal_margin"
    android:paddingRight="@dimen/
        activity_horizontal_margin"
    tools:context="ph.edu.up.pasabi.activity.
        ConversationActivity">

    <RelativeLayout
        android:id="@+id/convo_topbar"
        android:background="@color/maroon"

        android:layout_width="match_parent"
        android:layout_height="@dimen/
            convo_topbar_height"

        android:paddingStart="@dimen/
            convo_topbar_padding_bottom"
        android:paddingEnd="@dimen/
            convo_topbar_padding"
        android:paddingTop="@dimen/
            convo_topbar_padding"
        android:paddingBottom="5dp"

        android:layout_alignParentTop="true">
        <ImageButton
            android:id="@+id/
                convo_topbar_back_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/back_button"
            android:background="@null"
            android:layout_alignParentStart="
                true"
            android:layout_alignParentBottom="
                true"
            android:layout_marginEnd="@dimen/
                convo_topbar_padding_bottom"
            android:onClick="goBack"/>

        <TextView
            android:id="@+id/
                convo_topbar_contact_name"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_above="@+id/
                convo_topbar_contact_num"
            android:layout_toEndOf="@+id/
                convo_topbar_back_button"
            android:textSize="@dimen/
                convo_topbar_fontsize"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/
                convo_topbar_contact_num"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="
                true"
            android:layout_toEndOf="@+id/
                convo_topbar_back_button"
            android:layout_marginBottom="@dimen/
                convo_topbar_margin_bottom"
            android:textSize="@dimen/
                convo_topbar_fontsize" />

        <RelativeLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentEnd="true"
            android:layout_marginBottom="5dp">
            <ImageButton
                android:layout_width="
                    wrap_content"
                android:layout_height="
                    wrap_content"
                android:src="@drawable/
                    contacts_button"
                android:background="@null"
                android:onClick="selectContact"
                android:layout_centerHorizontal
                    ="true"
                android:layout_above="@+id/
                    convo_topbar_contacts_text
                    "/>
            <TextView
                android:id="@+id/
                    convo_topbar_contacts_text"
                android:layout_width="
                    wrap_content"
                android:layout_height="
                    wrap_content"
                android:text="@string/
                    convo_topbar_contacts_text"
                android:textSize="@dimen/
                    contacts_botbar_text_size"
                android:layout_centerHorizontal
                    ="true"
```

```
            android:layout_alignParentBottom
                ="true"/>
    </RelativeLayout>

</RelativeLayout>

<RelativeLayout
    android:background="@color/black"
    android:layout_width="match_parent"
    android:layout_height="@dimen/
        border_height"
    android:layout_above="@+id/
        convo_messages">
</RelativeLayout>

<ListView
    android:id="@+id/convo_messages"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@+id/convo_topbar"
    android:layout_above="@+id/convo_botbar"

    android:paddingTop="@dimen/
        convo_entry_padding_vert"
    android:paddingBottom="@dimen/
        convo_entry_padding_vert"
    android:clipToPadding="false"

    android:listSelector="@android:color/
        transparent"
    android:divider="@null"
    android:dividerHeight="@dimen/
        convo_divider_height">
</ListView>

<RelativeLayout
    android:background="@color/black"
    android:layout_width="match_parent"
    android:layout_height="@dimen/
        border_height"
    android:layout_above="@+id/convo_botbar
        ">
</RelativeLayout>

<RelativeLayout
    android:id="@+id/convo_botbar"
    android:paddingTop="@dimen/
        convo_botbar_padding_vert"
    android:paddingBottom="@dimen/
        convo_botbar_padding_vert"
    android:paddingLeft="@dimen/
        convo_botbar_padding_hori"
    android:paddingRight="@dimen/
        convo_botbar_padding_hori"

    android:layout_width="wrap_content"
    android:layout_height="@dimen/
        convo_botbar_height"
    android:layout_above="@+id/
        convo_hidden_compartment"
    android:layout_alignParentBottom="true">

    <ImageButton
        android:id="@+id/convo_button_toggle
            "
        android:src="@drawable/
            speech_on_button"
        android:background="@null"

        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="
            true"
        android:layout_centerInParent="true
            "/>

    <EditText
        android:id="@+id/convo_edittext"
        android:inputType="textCapSentences"
        android:textSize="@dimen/
            convo_edittext_fontsize"
        android:paddingTop="@dimen/
            convo_edittext_padding_vert"
        android:paddingBottom="@dimen/
            convo_edittext_padding_vert"
        android:paddingStart="@dimen/
            convo_edittext_padding_hori"
        android:paddingEnd="@dimen/
            convo_edittext_padding_hori"
        android:background="@drawable/
            convo_edittext"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"

        android:layout_marginLeft="@dimen/
            convo_edittext_margin_hori"
        android:layout_marginRight="@dimen/
            convo_edittext_margin_hori"
        android:layout_marginTop="@dimen/
            convo_edittext_margin_vert"
        android:layout_marginBottom="@dimen/
            convo_edittext_margin_vert"

        android:layout_toEndOf="@+id/
            convo_button_toggle"
        android:layout_toStartOf="@+id/
            convo_button_send"/>

    <ImageButton
        android:id="@+id/convo_button_send"
        android:src="@drawable/send_button"
        android:background="@null"

        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

        android:layout_alignParentEnd="true"
        android:layout_centerInParent="true"
        android:onClick="sendTextMessage" />
</RelativeLayout>
<RelativeLayout
    android:id="@+id/
        convo_hidden_compartment"
    android:layout_width="match_parent"
    android:layout_height="@dimen/
        convo_hidden_compartment_height"
    android:visibility="gone">

    <ImageButton
        android:id="@+id/convo_speech_button
            "
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/
            speech_button_big_off"
        android:background="@null"
        android:layout_centerHorizontal="
            true"/>

    <TextView
        android:id="@+id/convo_speech_status
            "
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:layout_below="@+id/
            convo_speech_button"
        android:text="@string/
            convo_speech_status_initializing
            "/>
</RelativeLayout>



</RelativeLayout>
```

File: app/src/main/res/layout/activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/
    res/android"
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/activity_main"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:paddingBottom="@dimen/
    activity_vertical_margin"
android:paddingLeft="@dimen/
    activity_horizontal_margin"
android:paddingRight="@dimen/
    activity_horizontal_margin"
android:paddingTop="@dimen/
    activity_vertical_margin"
tools:context="ph.edu.up.pasabi.activity.
    MainActivity">


    <RelativeLayout
        android:id="@+id/loadingPanel"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center" >
```

```
    <ProgressBar
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:indeterminate="true" />
</RelativeLayout>

<RelativeLayout
    android:id="@+id/contacts_topbar"
    android:layout_width="match_parent"
    android:layout_height="@dimen/
        contacts_topbar_height"
    android:background="@color/maroon">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/
            contacts_topbar_text"
        android:layout_alignParentBottom="
            true"
        android:layout_centerHorizontal="
            true"
        android:layout_marginBottom="@dimen/
            contacts_topbar_margin_bottom"
        android:textStyle="bold"
        android:textSize="@dimen/
            contacts_topbar_font_size"/>
</RelativeLayout>

<RelativeLayout
    android:background="@color/black"
    android:layout_width="match_parent"
    android:layout_height="@dimen/
        border_height"
    android:layout_above="@+id/contacts_list
        ">
</RelativeLayout>

<ListView
    android:id="@+id/contacts_list"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:paddingLeft="@dimen/
        convo_padding"
    android:paddingRight="@dimen/
        convo_padding"
    android:layout_below="@+id/
        contacts_topbar"
    android:layout_above="@+id/
        actions_container">
</ListView>

<RelativeLayout
    android:background="@color/black"
    android:layout_width="match_parent"
    android:layout_height="@dimen/
        border_height"
    android:layout_above="@+id/
        actions_container">
</RelativeLayout>

<RelativeLayout
    android:id="@+id/actions_container"
    android:background="@color/white"
    android:gravity="center_horizontal"

    android:layout_width="match_parent"
    android:layout_height="@dimen/
        bottom_bar_height"
    android:layout_alignParentBottom="true">

    <RelativeLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center">
        <RelativeLayout
            android:id="@+id/
                contacts_botbar_compose_container
                "
            android:layout_width="
                wrap_content"
            android:layout_height="
                wrap_content">
            <ImageButton
                android:id="@+id/
                    compose_button"
                android:layout_width="
                    wrap_content"
                android:layout_height="
                    wrap_content"
                android:src="@drawable/
                    compose_button"
                android:
                    layout_centerInParent="
                    true"
                android:background="@color/
                    white"
                android:layout_margin="
                    @dimen/
                    contacts_botbar_margin_top
                    "
                android:onClick="
                    composeMessage"
                android:enabled="false"/>
            <TextView
                android:layout_width="
                    wrap_content"
                android:layout_height="
                    wrap_content"

                android:layout_marginTop="
                    @dimen/
                    contacts_botbar_margin_top
                    "
                android:text="@string/
                    contacts_botbar_compose_text
                    "
                android:textSize="@dimen/
                    contacts_botbar_text_size
                    "
                android:layout_below="@+id/
                    compose_button"/>
        </RelativeLayout>
        <RelativeLayout
            android:layout_width="
                wrap_content"
            android:layout_height="
                wrap_content"
            android:layout_toEndOf="@+id/
                contacts_botbar_compose_container
                "
            android:layout_marginStart="
                @dimen/
                contacts_botbar_margin_start
                ">

            <ImageButton
                android:id="@+id/
                    directories_button"
                android:layout_width="
                    wrap_content"
                android:layout_height="
                    wrap_content"
                android:src="@drawable/
                    settings_button"
                android:
                    layout_centerInParent="
                    true"
                android:background="@color/
                    white"
                android:layout_margin="
                    @dimen/
                    contacts_botbar_margin_top
                    "
                android:onClick="
                    browseModels"
                android:clickable="false"/>
            <TextView
                android:layout_width="
                    wrap_content"
                android:layout_height="
                    wrap_content"

                android:layout_marginTop="
                    @dimen/
                    contacts_botbar_margin_top
                    "
                android:text="@string/
                    contacts_botbar_browse_text
                    "
                android:textSize="@dimen/
                    contacts_botbar_text_size
                    "
                android:layout_below="@+id/
                    directories_button" />
        </RelativeLayout>
    </RelativeLayout>

</RelativeLayout>
```

File: app/src/main/res/layout/layout_directory_item.xml

```
<RelativeLayout
```

```
    xmlns:android="http://schemas.android.com/
        apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="@dimen/
        convo_entry_height"
    android:padding="@dimen/
        convo_entry_padding_hori">

    <ImageView
        android:id="@+id/directory_image"
        android:layout_width="@dimen/
            directory_avatar_height"
        android:layout_height="@dimen/
            directory_avatar_height"
        android:layout_marginEnd="@dimen/
            directory_avatar_margin"
        android:layout_alignParentStart="true"
        android:background="@drawable/
            convo_placeholder"/>

    <TextView
        android:id="@+id/directory_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toEndOf="@+id/
            directory_image"
        android:textSize="@dimen/
            directory_font_size"
        android:textColor="@color/maroon"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/directory_last_access"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/
            directory_name"
        android:layout_toEndOf="@+id/
            directory_image"
        android:maxLines="1"
        android:ellipsize="end" />
</RelativeLayout>
```

File: app/src/main/res/layout/layout_sms_contact_item.xml

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/
        apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="@dimen/
        convo_entry_height"
    android:padding="@dimen/
        convo_entry_padding_hori">

    <ImageView
        android:id="@+id/contacts_entry_image"
        android:layout_width="@dimen/
            contact_avatar_height"
        android:layout_height="@dimen/
            contact_avatar_height"
        android:layout_marginEnd="@dimen/
            contact_avatar_margin"
        android:layout_alignParentStart="true"
        android:background="@drawable/
            convo_placeholder"/>

    <TextView
        android:id="@+id/
            contacts_entry_contact_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toEndOf="@+id/
            contacts_entry_image"
        android:textSize="@dimen/
            convo_entry_font_size"
        android:textColor="@color/maroon"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/contacts_entry_number"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/
            contacts_entry_contact_name"
        android:layout_toEndOf="@+id/
            contacts_entry_image"
        android:maxLines="1"
        android:ellipsize="end" />
</RelativeLayout>
```

File: app/src/main/res/layout/layout_sms_convo_item.xml

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/
        apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="@dimen/
        convo_entry_height"
    android:padding="@dimen/
        convo_entry_padding_hori">

    <ImageView
        android:id="@+id/convo_entry_image"
        android:layout_width="@dimen/
            contact_avatar_height"
        android:layout_height="@dimen/
            contact_avatar_height"
        android:layout_marginEnd="@dimen/
            contact_avatar_margin"
        android:layout_alignParentStart="true"
        android:background="@drawable/
            convo_placeholder"/>

    <TextView
        android:id="@+id/
            convo_entry_contact_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toEndOf="@+id/
            convo_entry_image"
        android:textSize="@dimen/
            convo_entry_font_size"
        android:textColor="@color/maroon"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/convo_entry_timestamp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:textSize="@dimen/
            convo_entry_timestamp_font_size"
        android:textColor="@color/lightGray" />

    <TextView
        android:id="@+id/convo_entry_message"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/
            convo_entry_contact_name"
        android:layout_toEndOf="@+id/
            convo_entry_image"
        android:maxLines="1"
        android:ellipsize="end" />
</RelativeLayout>
```

File: app/src/main/res/layout/layout_sms_message_item.xml

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/
        apk/res/android"
    android:id="@+id/text_message_padding"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">

    <RelativeLayout
        android:id="@+id/text_message_container"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="@dimen/
            convo_padding"
        android:layout_marginEnd="@dimen/
            convo_padding">

        <TextView
            android:id="@+id/
                text_message_content"

            android:layout_marginStart="@dimen/
                text_message_horizontal_margin"
            android:layout_marginEnd="@dimen/
                text_message_horizontal_margin"
            android:layout_marginTop="@dimen/
                text_message_vertical_margin"
            android:layout_marginBottom="@dimen/
                text_message_vertical_margin"

            android:paddingTop="@dimen/
                convo_entry_padding_vert"
            android:paddingBottom="@dimen/
                convo_entry_padding_vert"
            android:paddingStart="@dimen/
                convo_entry_padding_hori"
            android:paddingEnd="@dimen/
                convo_entry_padding_hori"
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content
                "/>
    </RelativeLayout>

</RelativeLayout>
```

File: app/src/main/res/values/colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</
        color>
    <color name="colorAccent">#FF4081</color>
    <color name="white">#FFFFFF</color>
    <color name="black">#000000</color>
    <color name="maroon">#b1818d</color>
    <color name="lightMaroon">#e4cbd1</color>
    <color name="lightGray">#dddddd</color>
    <color name="pink">#FCD0C7</color>
    <color name="transparent">#00000000</color>

    <color name="button_onPress">#CCB9B9B9</
        color>
</resources>
```

File: app/src/main/res/values/dimens.xml

```
<resources>
    <!-- Default screen margins, per the Android
         Design guidelines. -->
    <dimen name="activity_horizontal_margin">0dp
        </dimen>
    <dimen name="activity_vertical_margin">0dp</
        dimen>
    <dimen name="convo_padding">20dp</dimen>


    <dimen name="input_font_size">20sp</dimen>
    <dimen name="button_height">40dp</dimen>

    <dimen name="text_message_font_size">15sp</
        dimen>
    <dimen name="text_message_horizontal_margin
        ">5dp</dimen>
    <dimen name="text_message_vertical_margin">3
        dp</dimen>
    <dimen name="text_message_padding">18dp</
        dimen>
    <dimen name="text_message_horizontal_padding
        ">27dp</dimen>
    <dimen name="text_message_offset">50dp</
        dimen>

    <dimen name="contacts_topbar_height">50dp</
        dimen>
    <dimen name="contacts_topbar_margin_bottom
        ">5dp</dimen>
    <dimen name="contacts_topbar_font_size">16sp
        </dimen>

    <dimen name="contacts_botbar_margin_top">2dp
        </dimen>
    <dimen name="contacts_botbar_margin_start
        ">10dp</dimen>
    <dimen name="contacts_botbar_text_size">12sp
        </dimen>
    <dimen name="contacts_botbar_height">50dp</
        dimen>

    <dimen name="convo_entry_font_size">15sp</
        dimen>
    <dimen name="convo_entry_timestamp_font_size
        ">12sp</dimen>
    <dimen name="convo_entry_height">70dp</dimen
        >
    <dimen name="convo_entry_padding_vert">10dp
        </dimen>
    <dimen name="convo_entry_padding_hori">15dp
        </dimen>
    <dimen name="convo_entry_border_height">1dip
        </dimen>

    <dimen name="bottom_bar_height">70dp</dimen>
    <dimen name="
        bottom_bar_compose_button_height">50dp
        </dimen>
    <dimen name="bottom_bar_compose_button_width
        ">50dp</dimen>
    <dimen name="directory_topbar_height">70dp</
        dimen>
    <dimen name="directory_botbar_height">70dp</
        dimen>
    <dimen name="directory_padding">20dp</dimen>

    <dimen name="convo_topbar_height">70dp</
        dimen>
    <dimen name="convo_topbar_margin_bottom">5dp
        </dimen>
    <dimen name="convo_topbar_padding">20dp</
        dimen>
    <dimen name="convo_topbar_padding_bottom">10
        dp</dimen>
    <dimen name="convo_topbar_fontsize">15sp</
        dimen>
    <dimen name="convo_topbar_edittext_margin
        ">30dp</dimen>
    <dimen name="
        convo_topbar_edittext_padding_vert">2dp
        </dimen>
    <dimen name="
        convo_topbar_edittext_padding_hori">5dp
        </dimen>

    <dimen name="convo_divider_height">0dp</
        dimen>
    <dimen name="convo_botbar_height">60dp</
        dimen>
    <dimen name="convo_botbar_padding_vert">10dp
        </dimen>
    <dimen name="convo_botbar_padding_hori">15dp
        </dimen>

    <dimen name="convo_edittext_fontsize">15sp</
        dimen>
    <dimen name="convo_edittext_padding_hori">10
        dp</dimen>
    <dimen name="convo_edittext_padding_vert">5
        dp</dimen>
    <dimen name="convo_edittext_margin_hori">15
        dp</dimen>
    <dimen name="convo_edittext_margin_vert">5dp
        </dimen>
    <dimen name="convo_edittext_radius">3dp</
        dimen>

    <dimen name="convo_hidden_compartment_height
        ">130dp</dimen>
    <dimen name="convo_speech_button_padding">20
        dp</dimen>

    <dimen name="directory_avatar_height">35dp</
        dimen>
    <dimen name="directory_avatar_margin">30dp</
        dimen>
    <dimen name="directory_font_size">15sp</
        dimen>

    <dimen name="contact_avatar_height">35dp</
        dimen>
    <dimen name="contact_avatar_margin">30dp</
        dimen>
    <dimen name="border_height">1px</dimen>

    <dimen name="balloon_border_size">0px</dimen
        >
    <dimen name="balloon_border_radius">10dp</
        dimen>
</resources>
```

File: app/src/main/res/values/strings.xml

```
<resources>
    <string name="app_name">PASABI</string>
    <string name="convo_speech_status_idle">Idle
        </string>
    <string name="convo_speech_status_recording
        ">Recording&#8230;</string>
    <string name="
        convo_speech_status_transcribing">
        Transcribing&#8230;</string>
    <string name="
        convo_speech_status_initializing">
        Initializing&#8230;</string>
    <string name="convo_speech_status_error">
        Error</string>
    <string name="convo_topbar_contacts_text">
        Contacts</string>
    <string name="contacts_topbar_text">Messages
        </string>
    <string name="contacts_botbar_compose_text">
        Compose</string>
```

```
    <string name="contacts_botbar_browse_text">
        Browse</string>
    <string name="directory_add_button">Add
        directory</string>
    <item type="id" name="tag_threadId" />
    <item type="id" name="tag_contactName" />
    <item type="id" name="tag_contactNum" />
    <item type="id" name="tag_directoryName" />
</resources>
```

File: app/src/main/res/values/styles.xml

```
<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.
        AppCompat.Light.NoActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/
            colorPrimary</item>
        <item name="colorPrimaryDark">@color/
            colorPrimaryDark</item>
        <item name="colorAccent">@color/
            colorAccent</item>
    </style>
</resources>
```

XI. Acknowledgement

Hi all. Sa panahong sinusulat ko ito, hinahabol ko ang deadline na sinet ni Ma'am Eden. Haha. Hi Ma'am. Anyways, g.

Unang-una sa lahat, gusto kong pasalamatan ang aking dakilang adviser, si Sir Marvin Ignacio, sa suporta, sa tulong, at sa pag-encourage sakin nung medyo nag-aalangan na ako. Haha. Iba ka sir! Sa pag-introduce samin ng ML, sa pagcheck ng documents kahit medyo malalim na yung gabi (haha sorry sir sa abala), at sa paulit-ulit na pagpapaalala sa amin ng deadlines, etc. Di mabilang ang tulong ni sir, swear. Muli, maraming maraming salamat po!

Sa mga #TeamLearning diyan, kaway-kaway hahaha. Yung original horsemen, malupet kayo. Kayo ang naging guide naming horsebabbies HAHA. Di nyo lang alam pero super helpful ng mga pagtuturo nyo sa akin. Thank you boys!

Shoutout din sa Everwing Support Group (na isang group na TOTALLY not about Everwing)! Hahaha. Habang sinusulat ko to, overflow na ko yare haha pero SP is life, ika nga. Psst, Otogi > CR.

Alfred! Andrei! Haha. Thanks mga bes. Presensya nyo lang, sapat na para makalma ako. Kingina nyo. Nag-aaway pa kayo kung sino mas lamang, e pareho naman kayong malupet. < 3 AHAHA KappaPride

Marron! Salamat since Day 1. Alamoyan. Ewan ko, pero nung first class natin, akala ko mabait ka HAHA. Tas dank memer pala. Tas you unironically like Kuzu no Honkai pala. Tas admin pala ng Dextah 3.0 (4.0 incoming!). Y A R E. HAHAHA pero de tanggap parin kita haha. More movie sesh pls!

Reinier! Boy, salamat sa lahat! Sa pinagsamahan natin all throughout the year. Sa mga kwento mo na napakalupet na mala-series ang dating hahaha. Sa tawanan, sa mga seryoso, sa mga kabaitan, sa mga "kabaitan", sa mga kain every after 174 o 142 o kahit na anong night class, sa mga tambay sa school kasi wala lang, sa mga suggestion mo ng wholesome vids, sa mga lecture a la prof mo, lahat na. Salamat

talaga! Shoutout sa mga future students ni Sir Maristela! Boom.

At sa iyong nagbabasa nito. Siguro dahil naghahanap ka ng topic, o kaya trip mo lang magbasa ng mga SP, o naghahanap ng inspirasyon (at pang-angat ng loob) sa mga acknowledgement (tulad ko!). Ako na nagsasabi sayo, kaya mo yan. Teka, isa pa. KAYA MO YAN. Marahil medyo nahihirapan ka ngayon, siguro medyo di tugma ang panahon sayo, pero de. Mararating mo rin ang pinakamasarap sa sablay, shurbol. Ika nga ng chatwheel sa Dota, > Don't give up!

Ayun na. Late na ko kay Ma'am Eden. Sana mapakiusapan onti hehe. Babush.

Sa wakas. Narito na ang wakas.