Arab International University

Faculty of Informatics and Communication Engineering

Senior Project Report on

Songs Generator

Submitted to

Department of Informatics Engineering

In partial fulfillment of the requirement for the Degree of Bachelor in

Informatics Engineering

Submitted by

Aous Abdulhamid **Amjad Almrai**

Under the Supervision of

Dr. Nada Ghneim **Eng. Massa Baali**

July 2022

July 2022

Faculty of Informatics & Communication Engineering

CERTIFICATE OF APPROVAL

The undersigned certify that they have read and recommended to the Department of Informatics Engineering for acceptance, a project report entitled Songs Generator App in English Submitted by Amjad almrai, Aous Abdulhamid, MHD Bader Al-khoder, Mhd Mouaz Al-Manna'a, Mhd Zaid Arbash and Shade Arbeed in partial fulfilment for the degree of Bachelor of Engineering in Informatics.

Supervisor	**Head of Department**
Dr. Nada Ghneim	**Dr. Said Desouki**

Arab International University

The Arab International University (AIU) is a private Syrian university established in 2005. Its academic plans and the documents issued by it are approved and certified by the Ministry of Higher Education in the Syrian Arab Republic.

The university works to achieve the following goals:

- Preparing a distinguished generation of university graduates who are able to meet and advance the specific needs of society.
- Contributing to theoretical and applied scientific research that serves the purposes of national development. Work is being done to urge professors and academic staff to scientific research and participate in conferences and seminars that organize research.
- Achieving partnership with prestigious Arab and foreign universities with the aim of continuous development and modernization of academic work and conducting joint scientific research.
- Attracting distinguished academic and research competencies by providing the appropriate environment for their work.

The Arab International University is one of the first Syrian universities that have been established and inaugurated. It has been able to attract distinguished educational, research and administrative competencies, to create an integrated edifice from the academic, organizational and administrative aspects. It was able to graduate cadres of distinguished innovators by providing an educational environment based on unique qualitative and material ingredients, including:

- Modern and advanced study plans based on the credit hour system.
- Carefully selected educational cadres.
- Modern scientific laboratories and a laboratory for electronic libraries.
- Physical and moral incentives for students.
- Application of interactive teaching methods.
- Academic and educational guidance and counseling.
- A wide range of scientific cooperation agreements with local, regional and international universities of reputable reputation.
- Multiple agreements and memoranda of understanding with many civil society organizations.
- A proper campus with all the facilities of science, sports and entertainment, which we encourage you to visit and learn about their features.
- Student activities and clubs of all kinds: athletic, cultural, scientific and social.

The Arab International University life years are a time to invest in a student's future. The knowledge and experience that students acquire in the lecture hall and laboratories will help them in developing themselves. They will provide them with reasons for success in the chosen specialty. The student activities will help in expanding students' horizon. The activities of training, clubs and sports will enable students to develop their talents, and may even help in discovering new talents.
Students could invest time, mind and spirit in our university in order to reap the benefits of works and the time devoted in the coming years. We will be by our students in every step of their way.

To our families

For their continued support, love, and patience during all years of study. For being with us every step of the way during these hard times. Without this support none of this effort would be possible.

To our friends

For their support and help during all years of study and for those who helped us with useful expertise to make this project what it is. You were our constant source of energy on this journey. You Are the BEST

To our lecturers at AIU

For providing us with valuable information during all years of study and for their continues care and intensive attention.

Acknowledgements

It gives us immense pleasure to express our deepest sense of gratitude and sincere thanks to our highly respected and esteemed guide Dr. **Nada Ghneim** for her valuable guidance, encouragement, and help in completing this work, her useful suggestions for this whole work, and cooperative behavior are sincerely acknowledged.

We are also grateful to Eng. **Massa Baali** for her constant support and guidance at all stages of this project.

At the end, we would like to express our sincere thanks to all our families, friends and others who helped us directly or indirectly during this project work.

Abstract

Using deep learning models, the songs generator software creates lyrics, music, and voice based on the choices made by the user. The first goal of this software is to generate lyrics; to do this, a deep learning model that has been trained using datasets of lyrics from different artists, but which is not similar to current lyrics, is used to accept a sequence of words as input and output a continuation of that sequence.

With the use of a deep learning model that uses tokenized MIDI files as input and predicts the notes and durations as an output, the second goal of this model aims to create music that is also based on emotion. The result of this model would be a midi file. The last process involved using a trained model to create a voice that combined the output from the first and second models to create a voice that was in tune with the music and lyrics. Anyone can create the best music with our software, depending on their feelings and what they want to listen to.

Contents

Introduction 1
Chapter 1: Project Description 1
1.1 Background 1
1.2 Problem Statement 1
1.3 Project Objective 2
1.4 Project Scope 2
1.5 Project Features 2
1.6 Project Feasibility 2
1.7 System Requirements 2
Chapter 2: Theoretical Study 3
2.1 Artificial intelligence 3
2.2 Symbolic AI 4
2.3 Machine learning 4
2.4 Deep learning 4
2.5 Neural networks 4
2.6 Flask Web framework 4
2.7 Recurrent Neural networks 5
2.8 Long Short-term Memory 6
2.9 MIDI Files 6
Chapter 3: Literature Review 8
3.1 Related Works 8
3.1.1 Automatic Generation of Melodic Accompaniments of lyrics 8
3.1.2 Automagical Composition of Lyrical Songs 8
3.1.3 Magenta 8
3.1.4 GRUV 8
3.1.4 Deep Jazz 9
3.1.5 Mellotron 9
3.1.6 MLP-Singer 9
3.1.7 Diff-Singer 9

3.1.7 Comparison....10
Chapter 4: Lyrics Generation....11
4.1 Description 11
4.2 Data Collection 11
4.3 Pre-processing 11
4.3.1 Preprocessing Steps 11
4.3.2 Preprocessing Functions Used 12
4.4 Algorithm 13
4.4.1 LONG SHORT TERM MEMORY'S GATES 13
4.5 Models 15
4.5.1 Character Level Model 15
4.5.2 Architecture of the model 17
4.6 Result 17
Chapter 5: Music Generation....21
5.1 Description 21
5.1.1 Representing Music....21
5.1.2 Discrete Representation....21
5.2 Data Collection 23
5.2.1 Why use dataset pop909....23
5.2.2 Tasks of the dataset....24
5.2.3 Existing Datasets....24
5.2.4 Dataset Description 25
5.2.5 Data Folder Structure 25
5.2.6 How we used Dataset....26
5.3 Pre-processing 26
5.4 Algorithm 27
5.4.1 Long Short Term Memory (LSTM) 27
5.5 Model 27
5.6 Results 31
Chapter 6: Speech Synthesis....35

6.1 Description 35
6.1.1 Contextual Factors 36
6.1.2 Time-lag Modeling 36
6.2 dataset 37
6.3 Preprocessing 38
6.3.1 Preprocessing the midi file 38
6.3.2 Preprocessing the text file 38
6.4 MIDI2XML 38
6.3.1 Attributes 38
6.5 renderize the speech 39
6.6 make a song 40
Chapter 4: System Analysis 41
4.1 Requirements Specification 41
4.2 Functional Requirements 41
4.3 Non-Functional Requirements: 42
4.4 Use cases 43
4.5 Sequence Diagram 45
4.6 Activity Diagram 46
Chapter 5: System Design 47
5.1 System Components Diagram 47
5.2 System Context Diagram 48
5.3 State Diagram 49
5.3 Block Diagram 50
5.4 Mobile application & API 50
5.4 Application pages design 54
..... 54
Chapter 6: Environments and implementation 58
6.1 Technologies Used 58
6.1.1 Tensor Flow 58
6.1.2 Colab 58
6.1.3 Python 58
6.1.4 Postman 58

6.1.5 Keras 59
6.1.6 Flask 59
6.1.7 Numpy 59
6.1.8 Flutter 59
6.1.9 Music21 60
6.1.10 Anaconda 60
6.1.11 Visual Studio Code 61
6.1.12 Muse Score 61
6.1.13 Http library 61
6.1.14 audio players 61
Conclusion 62
Future Works 62
References 63

Table of Figures

Figure 1 RNN 5
Figure 2 LSTM 6
Figure 3 LSTM(2) ..... **Error! Bookmark not defined.**
Figure 4 Character Level Model 16
Figure 5 Character Level Model Overview 16
Figure 6 Sample of the database before cleaning ..... **Error! Bookmark not defined.**
Figure7 Sample of the database after cleaning ..... **Error! Bookmark not defined.**
Figure8 sample of sequences clean tokens ..... **Error! Bookmark not defined.**
Figure 9 sample of dataset after encode into sequence of integers ..... **Error! Bookmark not defined.**
Figure 10 Model Structure lyrics generation 17
Figure 11 Happy lyrics 18
Figure 12 Sad lyrics 19
Figure 13 Romantic lyrics 19
Figure 14 Adventure lyrics 20
Figure 15 Relax lyrics 20
Figure 16 Discrete or symbolic representation of music 22
Figure 17 Different components of music generation and associated list of tasks 22
Figure 18 ROLE OF PIANO ARRANGEMENT IN THE THREE FORMS OF MUSIC COMPOSITION 23
Figure 19 summary of existing datasets 24
Figure 20 An example of the MIDI file in a piano roll view. Different colors denote different tracks (red for MELODY, yellow for BRIDGE, and green for 25
Figure 21 The folder structure of POP909. The blue boxes denote the folder and the orange boxes denote the file. 26
Figure 22 lstm 27
Figure 23 embedding layer 28
Figure 24 dense layer 29
Figure 25 Model Structure songs generation 30
Figure 26 music sheet of a happy song 31
Figure 27 music sheet of a sad song 32
Figure 28 adventure 33
Figure 29 happy 33
Figure 30 romance 34
Figure 31 sad 34
Figure 32 relax 34
Figure 33 Use Case Diagram 43
Figure 34 sequence diagram 45
Figure 35 activity diagram 46
Figure 36 component diagram 47
Figure 37 component diagram 48
Figure 38 State diagram 49
Figure 39 block diagram 49

Abbreviations

AIU	Arab International University
RT	Report Template
AI	Artificial Intelligence
NN	Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-term Memory
MIDI	Musical Instrument Digital Interface
ML	Machine learning
DL	Deep Learning
TTS	Text to Speech
HMM	hidden Markov models

Keywords

Artificial intelligence, Machine, learning, Training, Deep, Neural, Network, LSTM, Music, Midi, HMM,Note, Chords, Melodies, Algorithm, Lyrics

Introduction

The creation of lyrics, music, and voice is a key aspect of human creativity.

Songs, as well as musical and lyrical arrangements, are difficult for human ingenuity to produce.

It goes without saying that songwriting calls for creativity in the two distinct fields of music composition and lyric writing.

Can lyrics and music be written by AI? The goal is to create a deep learning model that can take a word sequence as input and produce the next word in that sequence.

Because they can employ a distributed representation, in which words with similar meanings have similar representations, and because they can use a vast context of recently observed words when making predictions, neural network models are the method of choice for creating statistical language models.

In order to send this collected text to the model to be trained, the dataset for this model is gathered by accessing several websites that have lyrics for all songs.

In terms of the musical component, new songs from many nations and cultures are being produced every day. Many people worldwide are intrigued by the art of music.

Deep learning has developed to the point that creating music is starting to resemble art in its own right. Studies on the creation of music are scarce. However, there have been initiatives to produce new music.

This section makes an attempt to determine whether music can be created using the LSTM model architecture. Utilizing the pop909 dataset of pop music, this section investigates the music generation category.

The speech was the final component. It had to translate text into voice, match the voice's pitch to the melody's notes, and detect the pitch of the melody in the music. In order to keep the pace, it must also attempt to determine exactly where the virtual singer should utter each syllable. The speech was created using the outputs of the music and lyrics

Chapter 1: Project Description

1.1 Background

Depending on the user's preferences and selections, the project will produce words, piano music, and voices. In order for the user to hear a sample of their chosen

preferences—the emotion-based music, voice generation, and lyrics—they must first select their preferred preferences.

1.2 Problem Statement

Users want to stop waiting around for new albums and listen to music that matches their mood. Composers and vocalists also want to stop waiting so long for fresh song ideas. Composing music and coming up with inventive words for songs are difficult chores, Writer's block affects all songwriters and composers at some point in their careers, no matter how experienced and proficient they are.

1.3 Project Objective

The project's goal is to integrate deep learning AI techniques into our song application so that users can listen to new songs that match their mood. The application can be used on computers and smart phones by anyone, regardless of musical background, and it allows users to stream an unlimited amount of generated music.

1.4 Project Scope

Any user, regardless of musical experience, can use an application that is connected to an AI system that uses deep learning and can be utilized on smartphones to listen to an endless supply of created music and voice.

1.5 Project Features

The project's goal is to assist users in creating new lyrics based on emotion, listening to new music that matches their mood without having to wait a long time to hear it, and assisting composers and singers by creating coherent pairs of matching lyrics and music based on a given mood. The music that has been generated has never been heard before. additionally, users can hear the user's chosen song performed by a singer (male or female).

Finally, if a user wants to listen to the generated song later, they can add it to their playlist.

1.6 Project Feasibility

This system is free to use for personal reasons; however, commercial use is prohibited unless registered with the development end.

Nevertheless, it still offers a lot more affordable option for using music than purchasing the rights to a musical band's music.

1.7 System Requirements

understanding the most recent technology applied to AI models that produce lyrics, music, and voice.

Locate a sizable dataset that meets our requirements and has sufficient data.

In order to differentiate our project from other projects, we must also integrate software features that are special.

Chapter 2: Theoretical Study

2.1 Artificial intelligence

Artificial intelligence was born in the 1950s, when a handful of pioneers from the nascent field of computer science started asking whether computers could be made to think a question whose ramifications we‘re still exploring today.

A concise definition of the field would be as follows: the effort to automate intellectual tasks normally performed by humans.

2.2 Symbolic AI

Many experts believed that human-level artificial intelligence could be achieved by having programmers handcraft a sufficiently large set of explicit rules for manipulating knowledge.

It was the dominant paradigm in AI from the 1950s to the late 1980s.

2.3 Machine learning

A machine-learning system is trained rather than explicitly programmed. It‘s presented with many examples relevant to a task, and it finds statistical structure in these examples that eventually allows the system to come up with rules for automating the task. Although machine learning only started to flourish in the 1990s, it has quickly become the most popular and most successful subfield of AI.

2.4 Deep learning

Deep learning is a specific subfield of machine learning: a new take on learning representations from data that puts an emphasis on learning successive layers of increasingly meaningful representations.
The deep in deep learning isn‘t a reference to any kind of deeper understanding achieved by the approach; rather, it stands for this idea of successive layers of representations.

2.5 Neural networks

An artificial neural network (or simply neural network) consists of an input layer of neurons (or nodes, units), one or two (or even three) hidden layers of neurons, and a final layer of output neurons.

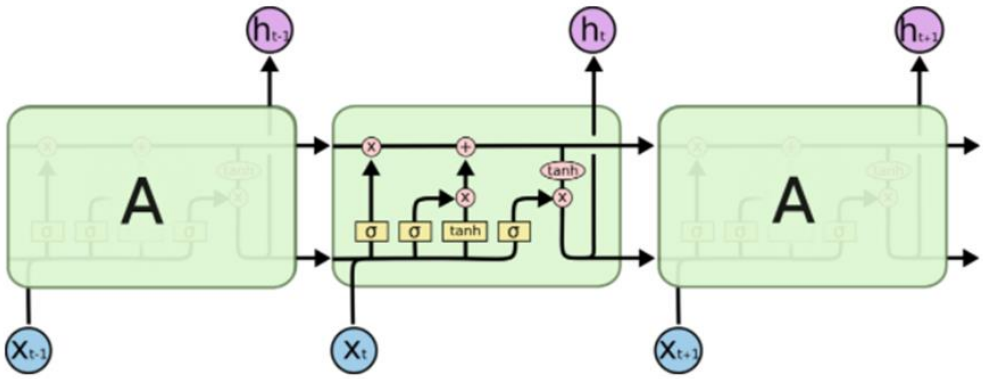
2.6 Flask Web framework

Flask is a web framework written in Python. Flask does not require libraries or specific tools and has no database, abstraction layer, form validation, or any other component and that is why it is considered as a micro-framework. It provides useful tools to create web applications in python.

Flask plays an important role also in building API has to connect python written code with front-end application such as mobile or web application. It is designed as a web framework for Restful API development.

2.7 Recurrent Neural networks

Figure 1 RNN



The repeating module in an LSTM contains four interacting layers.

The recurrent neural network is a class of artificial neural network; it works on the principle of saving the output of a particular layer and feeding this back to the input in order to predict the output of the layer.

The term of “recurrent neural network” is used for the class of network with infinite impulse,

Where the CNN is used for the class of network with finite impulse

The pros and cons of a typical RNN architecture are summed up in the table below:

Advantages	Disadvantages
• Possibility of processing input of any length • Model size not increasing with size of input • Computation takes into account historical information • Weights are shared across time	• Computation being slow • Difficulty of accessing information from a long time ago • Cannot consider any future input for the current state

2.8 Long Short-term Memory

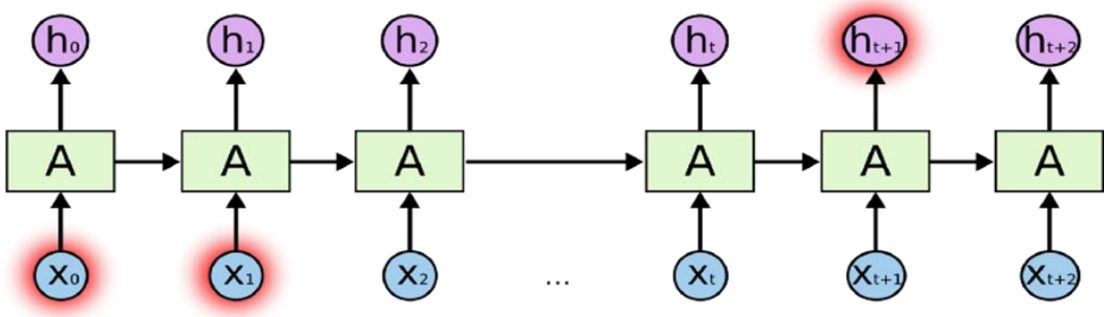


Figure 2 LSTM

Sequence classification is a big problem where there is some sequence of inputs over time and space. The whole idea of this problem is that the sequence can be vary in length and the very large vocabulary that been comprised requires to learn the long-term dependencies between symbols. The LSTM unit is divided into four gates (cell, input gate, output gate and forget gate). The cell remembers values and the three other gates groups the flow of information into and out of the cell.

2.9 MIDI Files

Musical Instrument Digital Interface (MIDI) is a technical standard that describes a protocol, a digital interface and connectors for interoperability between various electronic musical instruments, software‘s and devices. MIDI carries event messages that specify real-time note performance data as well as control data, MIDI files are the sheet music of the 21st century they tell you the chords and melodies artists wrote for songs and the often even the drum arrangements. MIDI files can contain the inner workings of a complete song.

2.10 Score Generation

A score is a symbolic representation of music that can be used/read by humans or systems to produce music.to draw an analog, we can safely consider the relationship between scores and music to be similar to that between text and speech. Music scores consist of discrete symbols, which can affectivity, convey music. Some works use the term AI Composer to denote models associated with the task of score generation.

2.11 Performance generation

Continuing with text to speech analogy, performance generation is where performers use scores to generate music using their own characteristics of tempo, rhythm and so on. Models associated with the task of performance generation are sometimes termed AI Performers as well

2.12 Chord

A chord, in music, is any harmonic set of pitches/frequencies consisting of multiple notes (also called "pitches") that are heard as if sounding simultaneously. For many practical and theoretical purposes, arpeggios and broken chords (in which the notes of the chord are sounded one after the other, rather than simultaneously), or sequences of chord tones, may also be considered as chords in the right musical context.

2.13 Note

In music, a note is a symbol denoting a musical sound. In English usage, a note is also the sound itself.

Notes can represent the pitch and duration of a sound in musical notation. A note can also represent a pitch class. Notes are the building blocks of much written music: discretization’s of musical phenomena that facilitate performance, comprehension, and analysis

Chapter 3: Literature Review

3.1 Related Works

One of the hardest tasks for AI has been the creation of songs. There have been numerous experiments to decrease the gap between real songs and those produced by computers. Some present implementations use recurrent neural network designs, while others investigate the use of long short-term memory (LSTM) architectures. In this study, only open-source projects are discussed.

3.1.1 Automatic Generation of Melodic Accompaniments of lyrics

Montecito et al. This work creates a system, which can automatically compose melodic accompaniments for any given text. For each given lyrics, it generates hundreds of possibilities for rhythms and pitches and evaluates these possibilities with a number of different metrics in order to select a final output.

3.1.2 Automagical Composition of Lyrical Songs

Toivanen et al. The paper addresses the task of automatically composing lyrical songs with the matching musical and lyrical features, and we present the first prototype. The proposed approach writes lyrics first and then composes music to match the lyrics. The crux is that the music composition sub process has access to the internals of the lyrics writing sub process, so the music can be composed to match the intentions and choices of lyrics writing, rather than just the surface of the lyrics.

3.1.3 Magenta

Magenta is Google‘s open-source deep learning music project. They aim to use machine learning to generate compelling music. The project went open source in June 2016 and currently implements a regular RNN and two LTSM‘s.

3.1.4 GRUV

A Stanford research project that, similar to Wave net, also tries to use audio waveforms as input, but with an LSTM‘s and GRU‘s rather. They have showed their proof of concept to the world in June 2015.

3.1.4 Deep Jazz

The result of a thirty-six-hour hackathon by Ji-Sung Kim. It uses a two-layer LSTM that learns from a midi file as its input source. Deep Jazz has received quite some news coverage in the first six months of its existence .

3.1.5 Mellotron

Is a multi-speaker voice synthesis model based on Tacotron 2 GST that can make a voice emote and sing without emotive or singing training data? By explicitly conditioning on rhythm and continuous pitch contours from an audio signal or music score, Mellotron is able to generate speech in a variety of styles ranging from read speech to expressive speech, from slow drawls to rap and from monotonous voice to singing voice.

3.1.6 MLP-Singer

Recent developments in deep learning have significantly improved the quality of synthesized singing voice audio. However, prominent neural singing voice synthesis systems suffer from slow inference speed due to their autoregressive design. Inspired by MLP-Mixer, a novel architecture introduced in the vision literature for attention-free image classification, we propose MLP Singer, a parallel Korean singing voice synthesis system. To the best of our knowledge, this is the first work that uses an entirely MLP-based architecture for voice synthesis. Listening tests demonstrate that MLP Singer outperforms a larger autoregressive GAN-based system, both in terms of audio quality and synthesis speed. In particular, MLP Singer achieves a real-time factor of up to 200 and 3400 on CPUs and GPUs respectively, enabling order of magnitude faster generation on both environments.

3.1.7 Diff-Singer

DiffSinger is a parameterized Markov chain that iteratively converts the noise into mel-spectrogram conditioned on the music score. By implicitly optimizing variation bound, DiffSinger can be stably trained and generate realistic outputs. To further improve the voice quality and speed up inference, we introduce a shallow diffusion mechanism to make better use of the prior knowledge learned by the simple loss. Specifically, DiffSinger starts generation at a shallow step smaller than the total number of diffusion steps, according to the intersection of the diffusion trajectories of the ground-truth mel-spectrogram and the one predicted by a simple Mel-spectrogram decoder.

3.1.7 Comparison

	Boomy	My Lyrics Maker	Melobytes	Sound raw	Soul Composer
Lyrics Generator	X	From Topic	Based On Your Lyrics	X	✓
Music Generator	✓	✓	✓	✓	✓
Speech Synthesis	Record Your Voice	X	✓	X	✓
Mood Choice	✓	X	X	✓	✓
Save As Play List	X	✓	X	✓	✓

Chapter 4: Lyrics Generation

4.1 Description

The generated lyrics are based on the supplied inputs (Datasets for training the model).

Because they can employ a distributed representation, in which words with similar meanings have similar representations, and because they can use a vast context of recently observed words when making predictions, neural network models are the method of choice for creating statistical language models.

Creating networks that can learn from a given dataset can make it possible to produce original content. Based on the words already seen in the sequence, a language model can forecast the likelihood of the subsequent word in the sequence.

This model's objective is to produce lyrics that are similar to existing lyrics but express the desired mood, such as happy or sad, by training itself using datasets of lyrics from different artists.

The Long Short-Term Memory language model is used to implement this.

The goal is to create a machine learning model that can take a set of words as input and produce the next set of words in that set.

4.2 Data Collection

In order to create lyrics based on seven emotions, we gathered the data for our project (Happy, Sad, Romance, Angry, Dreamy, Relax and Adventure) The top artists for each emotion were gathered using a dataset from All Music.

Each song listed for each of the aforementioned artists was collected using the song's meaning, after which we divided them into around 200 songs for each of the seven text files.

4.3 Pre-processing

4.3.1 Preprocessing Steps

For uniformity, the verse and chorus from the song were omitted.

Before performing any additional preprocessing, the punctuation and line breaks were left exactly as they were.

All of the songs go through this process again; this will serve as the training data.

```
Wouldn't it be nice if we were older?
Then we wouldn't have to wait so long
And wouldn't it be nice to live together
In the kind of world where we belong?

You know it's gonna make it that much better
When we can say goodnight and stay together

Wouldn't it be nice if we could wake up
In the morning when the day is new?
And after having spent the day together
Hold each other close the whole night
```

Figure 3 Sample of the database before cleaning

- First, we need to convert a text file to clean tokens so we remove punctuation, tokens and lowercase all the tokens.

```
['wouldnt', 'it', 'be', 'nice', 'if', 'we', 'were', 'older', 'then', 'we', 'wouldnt', 'have', 'to', 'wait', 'so']
```

Figure4 Sample of the database after cleaning

- Then we organize the clean tokens into sequences of token then we save into file that clean one.

```
['wouldnt it be nice if we were older then we wouldnt have to wait so long and wouldnt it be nice to live
```

Figure5 sample of sequences clean tokens

- In addition, we encode each line into sequence of integers.

```
tokenizer = Tokenizer()
tokenizer.fit_on_texts(lines)
sequences = tokenizer.texts_to_sequences(lines)
print(sequences[:1])
```

```
[[458, 12, 19, 294, 38, 20, 58, 1006, 130, 20, 458, 66, 5, 356, 18, 108, 4, 458, 12, 19, 294, 5, 184, 189,
```

Figure 6 sample of dataset after encode into sequence of integers

- Finally, we will separate the sequence was created to input X and output y and use to_categorical function to Converts a class vector (integers) to binary class matrix, so now our dataset it's ready to pass it to our model .

4.3.2 Preprocessing Functions Used

- You can see that the data consists of various punctuation marks.
  We are going to create a function clean text () to remove all the punctuation marks and special characters from the data.
- We will split the data according to space character and separate each word-using split ().
- make trans() function is used to construct the transition table i.e. specify the list of characters that need to be replaced in the whole string or the characters that need

to be deleted from the string, It returns the translation table which specifies the conversions that can be used by translate().

- String. Punctuation is a pre-initialized string used as string constant, which will give all the sets of punctuation.
- To translate the characters in the string translate () is used to make the translations. This function uses the translation mapping specified using the make Trans ().
- The is alpha () method returns True if all the characters are alphabet letters (a-z). The lower () methods returns the lowercased string from the given string.
- We can see that after passing data to clean text () we get the data in the required format without punctuations and special characters.
- We are going to use a set of previous words to predict the next word in the sentence. To be precise we are going to use a set of 50 words to predict the 51st word.
  Hence, we are going to divide our data in chunks of 51 words and at the last, we will separate the last word from every line.
- The 51st word will the output word used for prediction
- We are going to create a unique numerical token for each unique word in the dataset. fit_on_texts () updates internal vocabulary based on a list of texts. texts_to_sequences() transforms each text in texts to a sequence of integers
- To categorical () converts a class vector (integers) to binary class matrix. Num_classes is the total number of classes, which is vocab_size.

4.4 Algorithm

Algorithm for Generating Lyrics:

Input: Dataset of Lyric

Output: Lyric text file based on emotion

Algorithm steps:

Importing the classes and functions comes first, followed by reading the text collection and extracting unique characters, creating sequences that will be input to the network, creating the next character array that will serve as labels during training, and finally, transforming the character set.

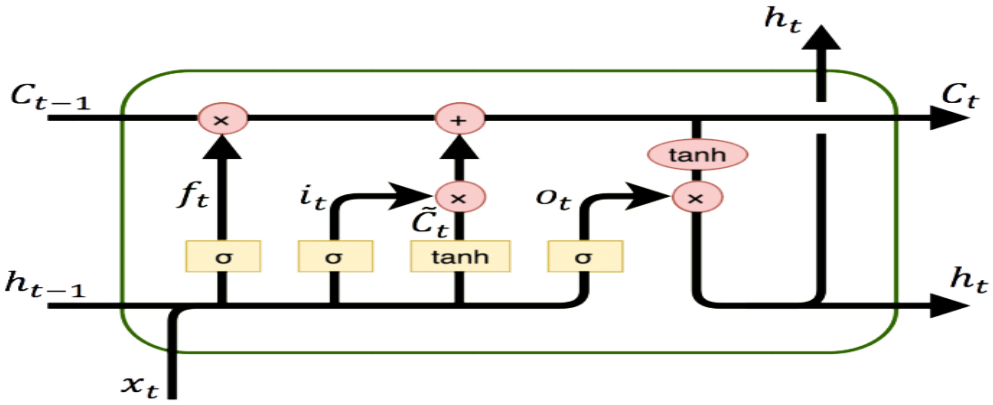
Finally, specify the model's structure and train the model.

4.4.1 LONG SHORT TERM MEMORY'S GATES

The key to LSTMs is the cell state, the horizontal line running through the top of the Figure 3. The cell state is kind of like a conveyor belt. It runs straight down the

entire chain, with only some minor linear interactions. It is very easy for information to just flow along it unchanged.

The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates. Gates are a way to optionally let information through. They are made up of a sigmoid neural net layer and a point wise multiplication operation. The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. A value of zero means let nothing through, while a value of one means let everything through! An LSTM has three of these gates, to protect and control the cell state. The first step in LSTM is to decide what information is to be thrown away from the cell state. This decision is made by a sigmoid layer called the forget gate layer. It looks at ht1 and at, and outputs a number between zero and one for each number in the cell state Ct1. A one represents completely keep this while a zero represents completely get rid of this. The next step is to decide what new information were going to store in the cell state. This has two parts. First, a sigmoid layer called the input gate layer decides which values well update. Next, a tan layer creates a vector of new candidate values, C t that could be added to the state. In the next step, well combine these two to create an update to the state. Finally, what is to be outputted is decided. This output will be based on the cell state, but will be a filtered version. First, a sigmoid layer is ran which decides what parts of the cell state is to be outputted. Then, we put the cell state through tanh (to push the values to be between one and 1) and multiply it by the output of the sigmoid gate, so that the parts that is decided is outputted.



4.5 Models

4.5.1 Character Level Model

A language model can predict the probability of the next word in the sequence, based on the words already observed in the sequence.

The goal of this model is to generate lyrics of the chosen mood say (happy, sad, angry, romantic, etc.), by training based on the respective datasets of lyrics of various artists grouped by emotions, but not identical to existing lyrics.

The benefit of character based language models is their small vocabulary and flexibility in handling any words, punctuation, and other document structure. In the case of a character based language model as mentioned in Figure 4 the input and output sequences must be characters.

The number of characters used as input will also define the number of characters that will need to be provided to the model in order to elicit the first predicted character.

After the first character has been generated, it can be appended to the input sequence and used as input for the model to generate the next character.

The RNN character-level language models were trained That is, the RNN was given a huge chunk of text and asked it to model the probability distribution of the next character in the sequence given a sequence of previous characters.

Concretely, each character is encoded into a vector using one hot encoding (i.e. all zero except for a single one at the index of the character in the vocabulary), and fed into the RNN one at a time. At test time, a character is fed into the LSTM and get a distribution over what characters are likely to come next this distribution is sampled, and fed right back in to get the next letter.

LSTM model uses a specific architecture for hidden transformation defined by the LSTM memory cell. The key feature to the LSTM memory cell is the presence of an input gate, output gate, forget gate, and cell/cell memory, which manifest themselves in the model activation vectors.

Each of these gates/cells has its own bias vector, and the hidden layer at each time-step is now a complex nonlinear combination of gate, cell, and hidden vectors.

LSTMs are explicitly designed to avoid the long-term dependency problem. All recurrent neural networks have the form of a chain of repeating modules of neural network.

The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates. Gates are a way to optionally let information through.

They are composed out of a sigmoid neural net layer and a point wise multiplication operation .

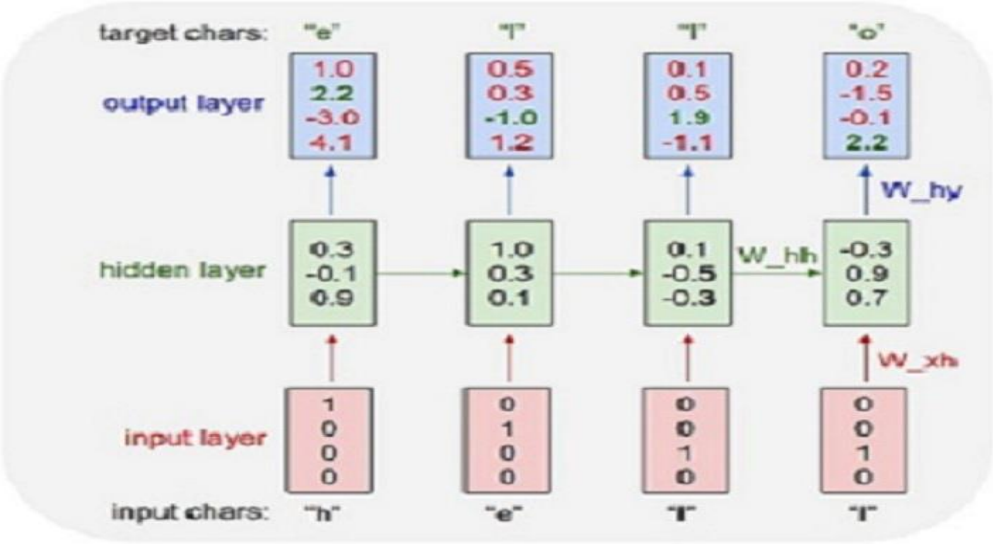


Figure 7 Character Level Model

Basic Character-level model overview

seed length: 20 characters

seed		new_char
Tryna_keep_it_simple	>>>	_
ryna_keep_it_simple_	>>>	i
yna_keep_it_simple_i	>>>	s
na_keep_it_simple_is	>>>	_
a_keep_it_simple_is_	>>>	a
_keep_it_simple_is_a	>>>	_
keep_it_simple_is_a_	>>>	s
eep_it_simple_is_a_s	>>>	t
ep_it_simple_is_a_st	>>>	r
p_it_simple_is_a_str	>>>	u
_it_simple_is_a_stru	>>>	g
it_simple_is_a_strug	>>>	g
t_simple_is_a_strugg	>>>	l
_simple_is_a_struggl	>>>	e
simple_is_a_struggle	>>>	_
imple_is_a_struggle_	>>>	f
mple_is_a_struggle_f	>>>	o
ple_is_a_struggle_fo	>>>	r
le_is_a_struggle_for	>>>	_
e_is_a_struggle_for_	>>>	m
_is_a_struggle_for_m	>>>	e
is_a_struggle_for_me		

Result:

Tryna_keep_it_simple_is_a_struggle_for_me

Figure 8 Character Level Model Overview

4.5.2 Architecture of the model

Long short-term memory (LSTM) Character Level Word Modeling Was used to train the lyric generation.

For LSTM preprocessing, a dictionary of letters from the training data was created based on the conditions that letters are case sensitive, Next, the character sequences was converted into a one-hot encoded array so that each character could be processed separately as a binary feature.
Then, the input vectors was split into batches, to feed into LSTM step by step for the optimizer to minimize MSE by tuning the weights of the LSTM nodes. The optimizer was allowed to run (train the model) until the end of the file is reached. When this happens, an epoch is complete, as the number of epochs increases; the variety in repeated words becomes greater.

```
model = Sequential()
model.add(Embedding(vocab_size, 50, input_length=seq_length))
model.add(LSTM(100, return_sequences=True))
model.add(LSTM(100))
model.add(Dense(100, activation='relu'))
model.add(Dense(vocab_size, activation='softmax'))
```

Figure 9 Model Structure lyrics generation

4.6 Result

We discovered that fine-tuning models is crucial for achieving successful outcomes.

The quality of the lyrics produced by our first LSTM model before tuning and the quality of the generated lyrics after tweaking were very different. We also require various preprocessing techniques for every dataset.

These are the lyrics generated for every emotion:

bring weve got some joy
in this thing interlude
i feel good and when i feel good
i sing and the joy it brings
makes me feel good
and when i feel good i sing

and the joy it brings it brings
me freedom freedom woah got to
get yourself tothat freedom its freedom
singing freedom freedom woah you deserve
your freedom a smile no missed
in the sound to start

i can staring i got all these
things oh ive also reason now
i could let by as
of where i have done in things
i wanted pulling it im
my protection and i see you

and you are lost but i wont
ever who all ill come back
you see you but dance but
i dont know why i felt
you could hold a dark til
the ones not gonna stare into

Figure 10 Happy lyrics

```
sorrow no tomorrow no tomorrow chorus
and i find it kind of funny i find it kind of sad
these dreams in which im
dying are the best ever had
i find it hard to tell


you i find it hard to
take when people run in circles
a very madworld mad world enlargen
your world mad world floating along
hippopotamus hunting the beast gets what
he deserves fall in to bed


with the daughters of jezebel
and dream of athenians theres a hole
in the sky that the birds
fly through and i want to
fly too oh i want to
fly too theres a hole in


the sky that the birds fly
through and i want to fly
too oh i want to fly
to ya children get out out
of the buildings get and shes
ya talk are called the broken
```

Figure 11 Sad lyrics

```
world still livin for today in
these last days and times
if i ruled the world
if i ruled if i ruled id free
all my sons if i ruled
if i ruled black diamonds and


pearls black diamonds black diamonds could
it be if you could be
mine wed bothshine if i ruled
the world still livin for today
in these last days and times
if i ruled the world


if i ruled if i ruled id
free all my sons black diamonds
i love em love em baby
black diamonds and pearls
if i ruled if i ruled the world
if i ruled the world


i love em love em baby no
matter what aint no cryin you
keep on asking youve knows ooh
ive touch you dont make what
too more in a nice dress
now about it all my little
```

Figure 12 Romantic lyrics

castle inching closer each century crumbling
castle waters rising up thick
and green crumbling castle are we safe
in our citadel look upon our
condition crumbling castle you would not
believe where im from crumbling castle

i dont want to be
a crumbling crumbling crumbling castle dont want
to be acrumbling crumbling crumbling castle
dont want to be a crumbling
crumbling crumbling castle dont want to
be a crumbling crumbling crumbling castle

dont want to personally this care
take me from well be going
for down down down well the
spawn of dust so many reigning
in bloodthirsty odd sign are death
venusi come on a hologram nothing

is upon and i could go
i think of around the days
you dont know if you have
baby california like that i got
old home society home mr style
when that ive got an altered

Figure 13 Adventure lyrics

beautiful like you beautiful beautiful beautiful
like you beautiful beautiful beautiful like
you beautiful beautiful beautiful like you
theres something in the air the
girls are runnin round in summer
dresses with their masks off

and it makes me so happy larchmont
village smells like lilies of the
valley and thebookstore doors are opening
and its finally happening ever since
i fell out of love with
you i fell back in love

with me how were never midnight
cousins waiting for hmm crazy across
the smile you worry be my
friend well i see my key
like you for too empty every
people i have wild friend you

baby my heart im forced to
fake a smile a laugh that
i didnt did ive heard the
lights out in the love that
you want for them ohohoh
i know what youre thinking

Figure 14 Relax lyrics

Chapter 5: Music Generation

5.1 Description

The process of creating music is inherently complex and challenging. It is significantly more difficult to accomplish this with the use of algorithms (machine learning or otherwise). Nevertheless, music generation remains a fascinating field of study with a variety of unresolved issues and intriguing compositions. We will develop a high-level comprehension of this field and comprehend a few crucial and fundamental principles in this section.

Deep music generation, or computer-assisted music generation in more precise terms (owing to the usage of deep learning architectures), is a multi-level learning activity that primarily consists of score generation and performance generation.

In the context of music creation, the following tasks are highlighted in Figure 3:

By concentrating solely on score generation, as demonstrated in this figure, we can work toward objectives like melody generation, melodic hamonization, and music in-painting (associated with filling in missing or lost informaion in a piece or music).

5.1.1 Representing Music

Music is a work of art that represernt mood,rhythm,emotions,and so on.Similar to text,whiche is a collection of alphabets and grammatical rules,music scores have their own notation and set of rules.

Music representation can be categorized into two main classes:continous and discrete.in this section we will focus in dicrete way.

5.1.2 Discrete Representation

Discrete or symbic representation makes makes use of discrete symbols to capture information related to pitch,duration,chorde,and so on.symbloic represention is widely used across different music generation works.this popularity is primmarily due to the ease of understanding and handling such a representation.Figure 4 show cases a sample symplic represntation of a music score.

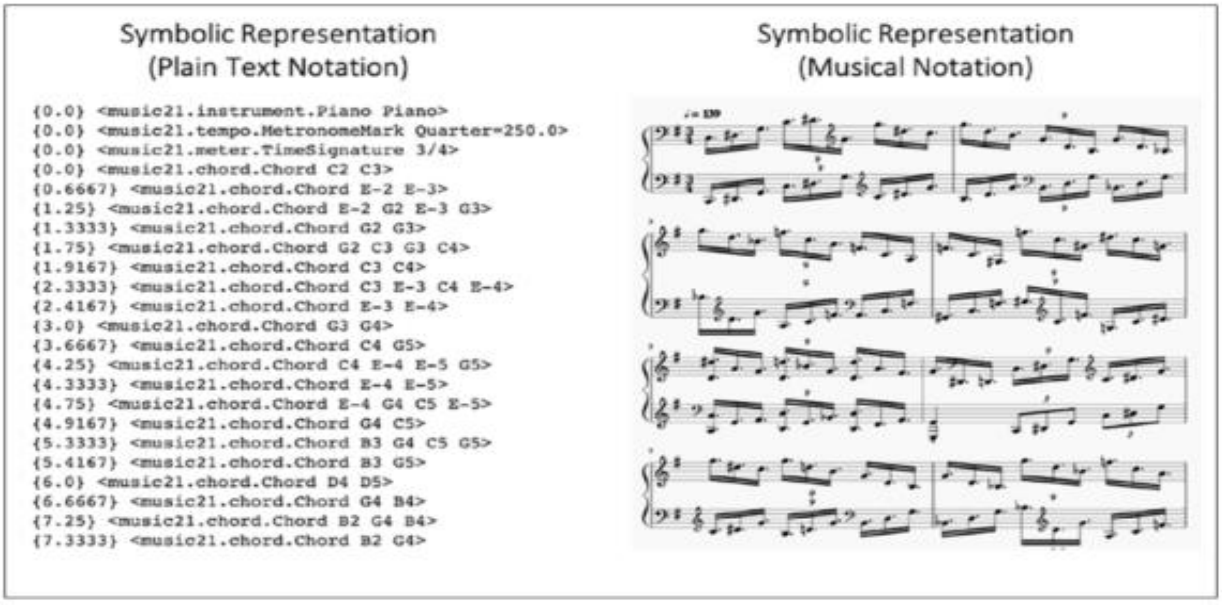


Figure 15 Discrete or symbolic representation of music

As shown, is the figure a symbolic representation captures information using various symbols/positions? MIDI is an interoperable format used by musicians to create, compose and share music.it is a common format used by various electronic musical instruments. Computers, smartphones and even software to read and play files.

The symbolic representation can be designed to capture a whole lot of events such as note-on, note-off, time shift, bar; track .we will focus only on two main events, note-on and note-off. There are a lot of format to use in the music but we choose midi files because they are widely, used, expressive, interoperable, and easy to understand.

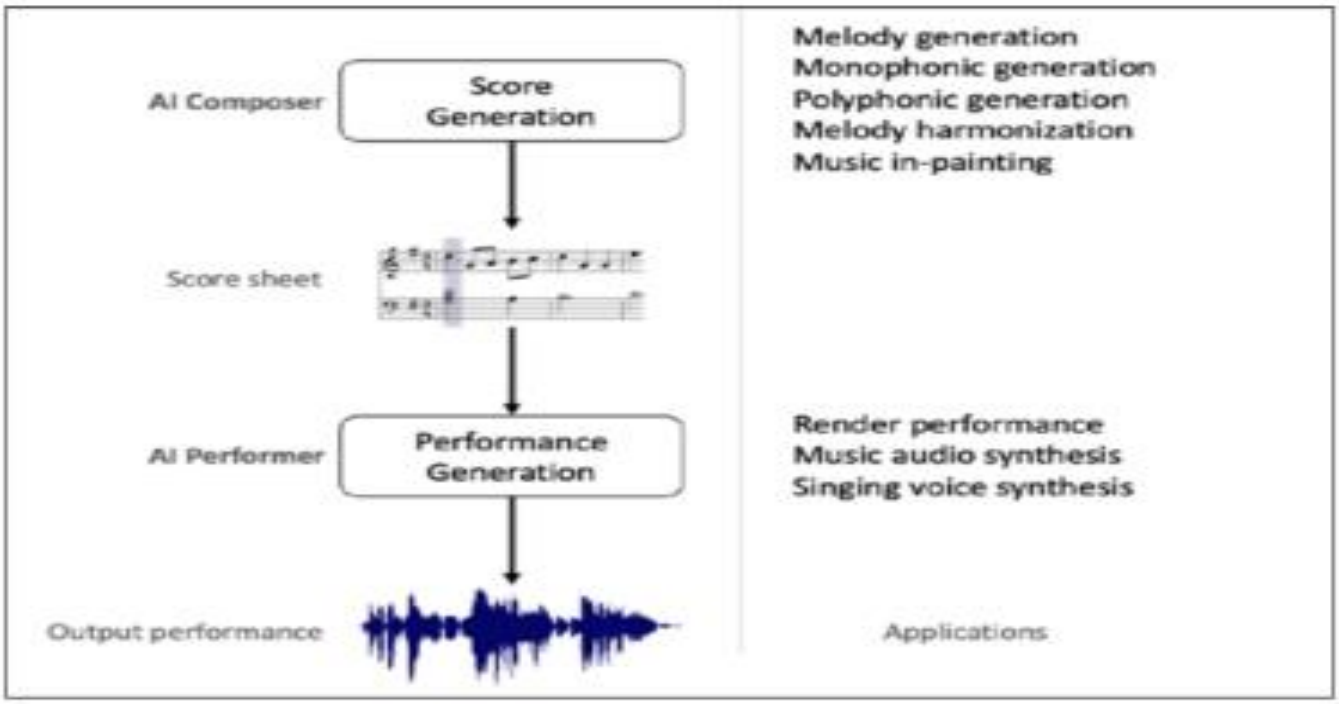


Figure 16 Different components of music generation and associated list of tasks

5.2 Data Collection

The POP909 dataset was the one we used. It is a dataset of pop songs used to create musical arrangements. The collection includes various piano arrangements of 909 well-known songs that were made by musicians for the piano. The vocal melody, lead instrument melody, and piano accompaniment for each song are all included in the dataset's main portion and are all aligned to the original audio files. It includes annotations for the tempo, beat, key, and chords, with MIR algorithms labeling some of the tempo curves and hand labeling others.

5.2.1 Why use dataset pop909

Music arrangement, the process of reconstructing and conceptualizing a piece, can refer to various conditional music generation tasks, which includes accompaniment generation conditioned on a lead sheet (the lead melody with a chord progression), transcription and orchestration conditioned on the original audio, and reduction of a full score so that the piece can be performed by a single (or fewer) instrument(s).

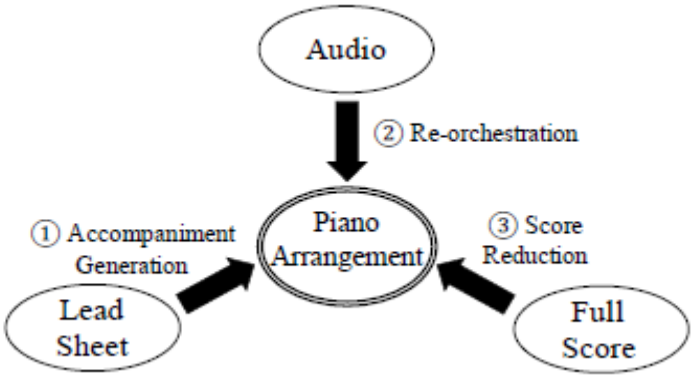


Figure 17 ROLE OF PIANO ARRANGEMENT IN THE THREE FORMS OF MUSIC COMPOSITION

Arrangement acts as a bridge, which connects lead sheet, audio and full score. In particular, piano arrangement is one of the most favored form of music arrangement due to its rich musical expression. With the emergence of player pianos and expressive performance techniques. In the computer music community, despite several promising generative models for arrangement, the lack of suitable datasets becomes one of the main bottlenecks of this research area (as pointed by [one, 2].) A desired arrangement dataset should have three features. First, the arrangement should be a style-consistent re-orchestration, instead of an arbitrary selection of tracks from the original orchestration. Second, the arrangement should be paired with an original form of music (audio, lead sheet, or full score) with precise time alignment, which serves as a supervision for the learning algorithms. Third, the dataset should provide external labels (e.g., chords, downbeat labels), which are commonly used to improve the controllability of the generation process [3]. Until now, we have not seen such a

qualified dataset. Although most existing high-quality datasets (e.g., [4, 5]) contain at least one form of audio, lead melody or full score data, they have less focus on arrangement, lacking accurate alignment and labels.

To this end, we propose POP909 dataset.1 it contains 909 popular songs, each with multiple versions of piano arrangements created by professional musicians. The arrangements are in MIDI format, aligned to the lead melody (also in MIDI format) and the original audios. Furthermore, each song are provided with manually labeled tempo curves and machine-extracted beat, key and chord labels using music information retrieval algorithms

5.2.2 Tasks of the dataset

Task 1: Piano accompaniment generation conditioned on paired melody and auxiliary annotation. This task involves learning the intrinsic relations between melody and accompaniment, including the selection of accompaniment figure, the creation of counterparts and secondary melody, etc.

Task 2: Re-orchestration from audio, i.e., the generation of piano accompaniment based on the audio of a full orchestra.

5.2.3 Existing Datasets

Dataset	Size	Paired Property				Annotation			Modality
		Polyphony	Lead Melody	Audio	Time-alignment	Beat	Key	Chord	
Lakh MIDI [13]	170k	✓		✓	Δ	✓	Δ	Δ	score, perf
JSB Chorales [14]	350+	✓			N/A	✓	✓		score
Maestro [15]	1k	✓		✓					perf
CrestMuse [16]	100	✓		✓	✓	✓	✓		score, perf
RWC-POP [17]	100	✓	✓	✓	Δ	✓	✓	✓	score, perf
Nottingham [18]	1k		N/A		N/A	✓	✓	✓	score
POP909	1k	✓	✓	✓	✓	✓	✓	✓	score, perf

Table 1: A summary of existing datasets.

Figure 18 summary of existing datasets

Table 1 summarizes the existing music datasets, which are the potential resources for the piano arrangement generation tasks. The first column shows the dataset name, and the other columns show some important properties of each dataset.

Lakh MIDI is one of the most popular datasets in symbolic format, containing 176,581 songs in MIDI format from a broad range of genres. Most songs have multiple tracks, most of which are aligned to the original audio. However, the dataset does not mark the lead melody track or the piano accompaniment track and therefore cannot be directly used for piano arrangement.

Maestro and E-piano contains classical piano performances in time-aligned MIDI and audio formats. However, the boundary between the melody and accompaniment is usually ambiguous for classical compositions. Consequently, the dataset is not suitable for the arrangement task 1. Moreover, the MIDI files are transcription rather than re-harmonization of the audio, which makes it inappropriate for the arrangement task 2 either.

Nottingham Database is a high-quality resource of British and Irish folk songs. The database contains MIDI files and ABC notations. One drawback of the dataset is that it only contains monophonic melody without polyphonic texture.

RWC-POP, Crest Muse and JSB-Chorale all contain polyphonic music pieces with rich annotations. However, the sizes of these three datasets are relatively small for training deep generative models

5.2.4 Dataset Description

In POP909, the total duration of 909 arrangements is about 60 hours. 462 artists compose the songs. The release of all songs spans around 60 years (from the earliest in 1950s to the latest around 2010). Each piano arrangement is stored in MIDI format with three tracks. The three tracks are MELODY: the lead (vocal) melody transcription, BRIDGE: the arrangement of secondary melodies or lead instruments. PIANO: the arrangement of main body of the accompaniment, including broken chords, arpeggios, and many other textures

Figure 19 An example of the MIDI file in a piano roll view. Different colors denote different tracks (red for MELODY, yellow for BRIDGE, and green for

5.2.5 Data Folder Structure

The folder structure of POP909. In the root directory, there are 909 folders, corresponding to 909 songs. In each folder, we provide the MIDI format arrangement, text format annotations, and a folder of all arrangement versions produced during the iterative processes. The annotation files contain beat, chord and key annotations in plain text format. Finally, we provide an index file in the root directory containing the song name, artist name, number of modified times and other useful metadata of the

dataset.

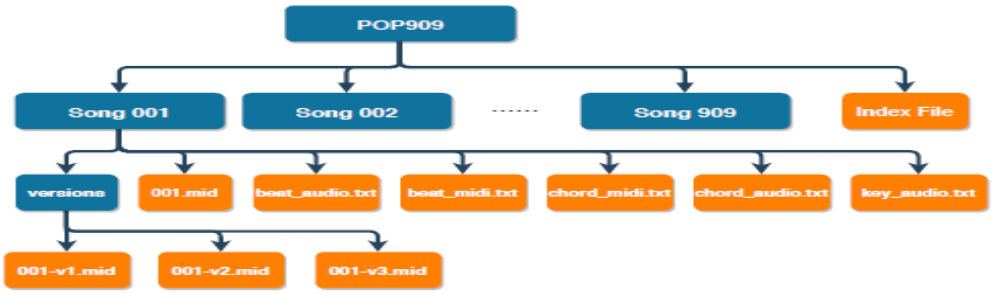


Figure 20 The folder structure of POP909. The blue boxes denote the folder and the orange boxes denote the file.

5.2.6 How we used Dataset

The five categories we sorted the dataset into—adventure, happy, romance, sad, and relax—represent the five different human emotions. We categorize it correctly by identifying the midi files contained within the dataset and by taking a look at the annotation files and the index file. to play the listener a significant piece of music

5.3 Pre-processing

By transforming all dataset types into music21 objects, such as music21.stream score, we were able to read all of the midi files. After reading the data, we must use codification to extract the chords from the songs, which enables us to condense a complicated score with several parts into a series of chords in a single part that perfectly captures the action in the score.

Once we have the list of scores, the next step is to obtain the notes and their corresponding timing data (Duration Information) using pitch and duration from music21. We then obtain the key for each song in the scores list and determine whether the element is a note or a chord before adding it and its time duration to the lists.

In order to make the task manageable and the model training requirements reasonable, the next step is to lower the dimensionality. We are free to choose any key, so we choose with the C major key.

After pre-processing the data, the notes/chords and duration-related data must now be converted into a usable form.

Making a mapping of integer symbologies is the most straightforward way to accomplish this.

After that, we changed into integers.

They can be fed into the model's embedding layer, which is tweaked throughout the training procedure.

5.4 Algorithm

5.4.1 Long Short Term Memory (LSTM)

An LSTM network is specifically designed to work with a data sequence as it takes into consideration all of the previous inputs while generating an output. LSTMs are actually a type of neural network called Recurrent Neural Network, but RNNs are not known to be effective for dealing with the Long term dependencies in the input sequence because of a problem called the Vanishing gradient problem.

LSTMs were developed to overcome the vanishing gradient and so LSTM cell can remember context for long input sequences.

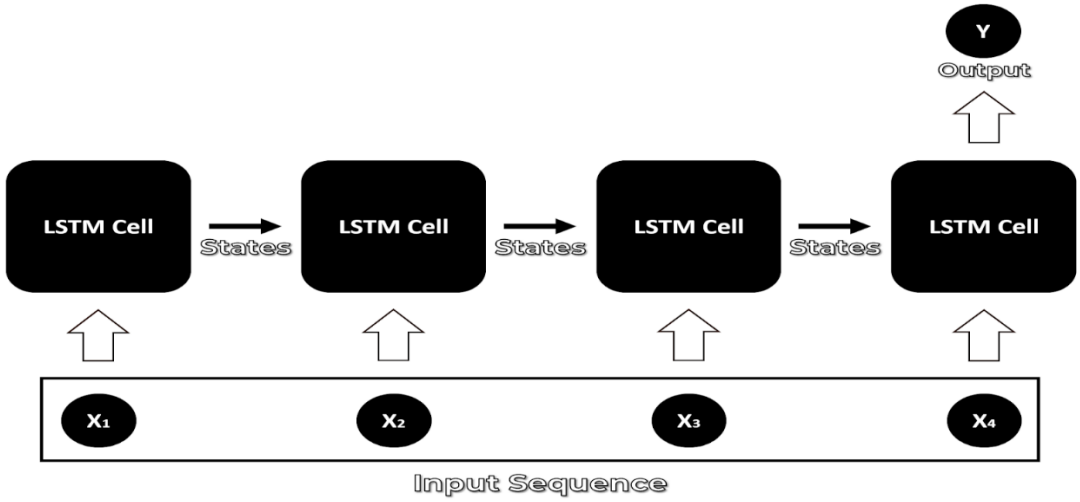


Figure 21 lstm

This makes an LSTM more capable of solving problems involving sequential data such as time series prediction, speech recognition, language translation, or music composition. However, for now, we will only explore the role of LSTMs in developing better action recognition models.

5.5 Model

We create the training dataset as 32-token sequences with the next token representing the matching target. For each time step, the model requires two inputs: notes and duration. At each time step, the model should be able to accept the notes and duration data as inputs and produce an output note with the corresponding duration. To do this, we set up a multi-input multi-output architecture using the

useful tensorflow.keras API. In comparison to networks with a single LSTM layer, stacked LSTMs can learn features that are significantly more sophisticated.

Additionally, attention methods aid in resolving problems with RNNs, such as problems addressing long-range dependencies. In music, there exist both local and global patterns that might be perceived as rhythm and coherence, therefore there is little doubt that attention mechanisms can have an effect. The following step is to use embedding layers to convert each input into a vector.

- The model take two input notes and duration
- The next step, is to transform each input into vectors using embedding layers
    - ❖ Embedding layer enables us to convert each word into a fixed length vector of defined size. The resultant vector is a dense one with having real values instead of just 0's and 1's. The fixed length of word vectors helps us to represent words in a better way along with reduced dimensions

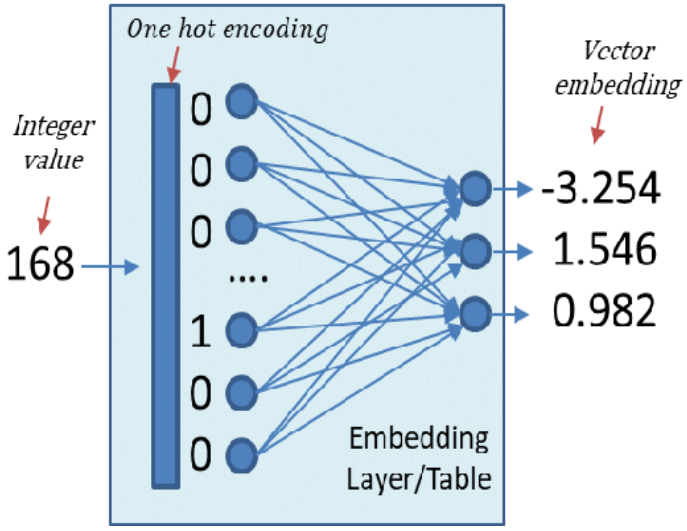


Figure 22 embedding layer

- Both inputs are concatenated
- The concatenated inputs are then passed through a couple of LSTM layers
    - ❖ Each lstm layer takes a sequence as an input (256 units) and can return either sequences (return sequences=True) or a matrix
- The Dense layer is a fully connected neural network layer where each input node is connected to each output node, which works for changing the dimension of the output by performing matrix-vector multiplication. Therefore, we put the units to one and we use the activation function tanh because this function takes any real value as input and outputs values between the ranges -1 to 1.
    - ❖ It is a layer that is deeply connected with its preceding layer, which means the neurons of the layer are connected to every neuron of its

preceding layer. This layer is the most commonly used layer in artificial neural network

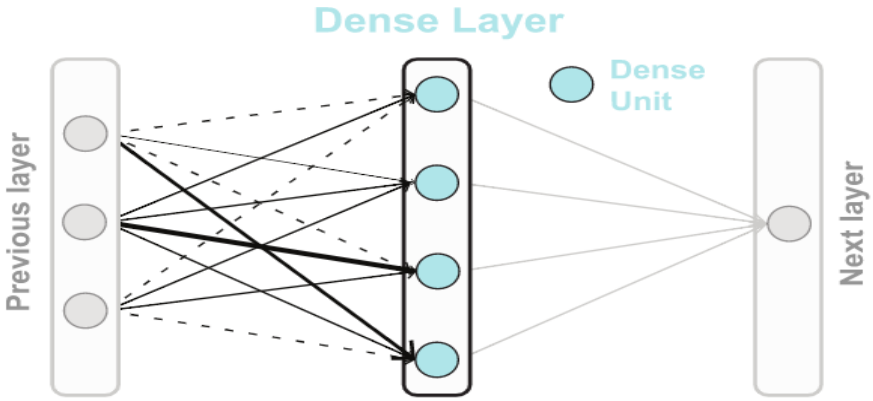


Figure 23 dense layer

- The next step is to reshape the output of the last layer by -one (simply means that it is an unknown dimension), so the NumPy will calculate and figure the dimensions by itself. And NumPy will figure this by looking at the 'length of the array and remaining dimensions' and making sure it satisfies the above-mentioned criteria.

- After the reshape, we used the activation softmax to obtain probability or probability distribution as the output. the network is configured to output N values, one for each class in the classification task, and the softmax function is used to normalize the outputs, converting them from weighted sum values into probabilities that sum to one. Each value in the output of the softmax function is interpreted as the probability of membership for each class.

- The use of axis is throughout Numpy. It's an important concept because you can avoid using loops, thus resulting in better optimization (or vectorized operations).
  And it will sum up lower dimensions in vector operation (i.e., adding all the n-dim arrays together).
  the model again diverges into two outputs (one for the next note and the other for the duration of that note). we use the activation softmax to make predictions for each note and for each duration.

- finally we use the loss sparse_categorical_crossentropy Used as a loss function for multi-class classification model where the output label is assigned integer value (0, 1, 2, 3…) and we use the optimizer RMSprop with learning rate 0.0001 because the optimizer Maintain a moving (discounted) average of the square of gradients and Divide the gradient by the root of this average.

```
def create_network(n_notes, n_durations, embed_size = 100, rnn_units = 256):
    """ create the structure of the neural network """

    notes_in = Input(shape = (None,))
    durations_in = Input(shape = (None,))

    x1 = Embedding(n_notes, embed_size)(notes_in)
    x2 = Embedding(n_durations, embed_size)(durations_in)

    x = Concatenate()([x1,x2])

    x = LSTM(rnn_units, return_sequences=True)(x)

    x = LSTM(rnn_units, return_sequences=True)(x)

    # attention
    e = Dense(1, activation='tanh')(x)
    e = Reshape([-1])(e)
    alpha = Activation('softmax')(e)

    alpha_repeated = Permute([2, 1])(RepeatVector(rnn_units)(alpha))

    c = Multiply()([x, alpha_repeated])
    c = Lambda(lambda xin: K.sum(xin, axis=1), output_shape=(rnn_units,))(c)


    notes_out = Dense(n_notes, activation = 'softmax', name = 'pitch')(c)
    durations_out = Dense(n_durations, activation = 'softmax', name = 'duration')(c)

    model = Model([notes_in, durations_in], [notes_out, durations_out])
    model.compile(loss=['sparse_categorical_crossentropy',
                        'sparse_categorical_crossentropy'], optimizer=RMSprop(lr = 0.001))

    return model
```

Figure 24 Model Structure songs generation

5.6 Results

Results sample from sheets of music:

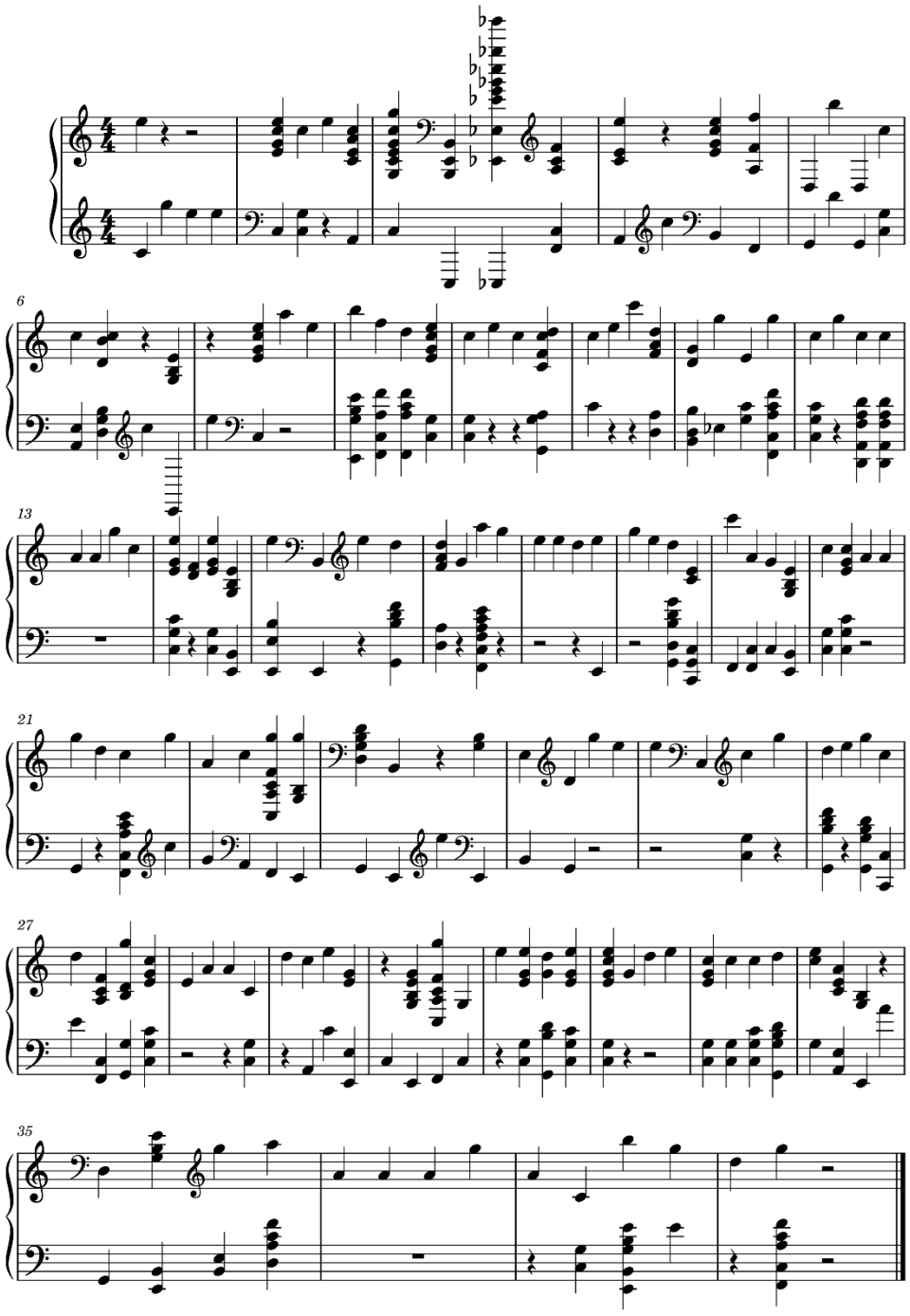


Figure 25 music sheet of a happy song

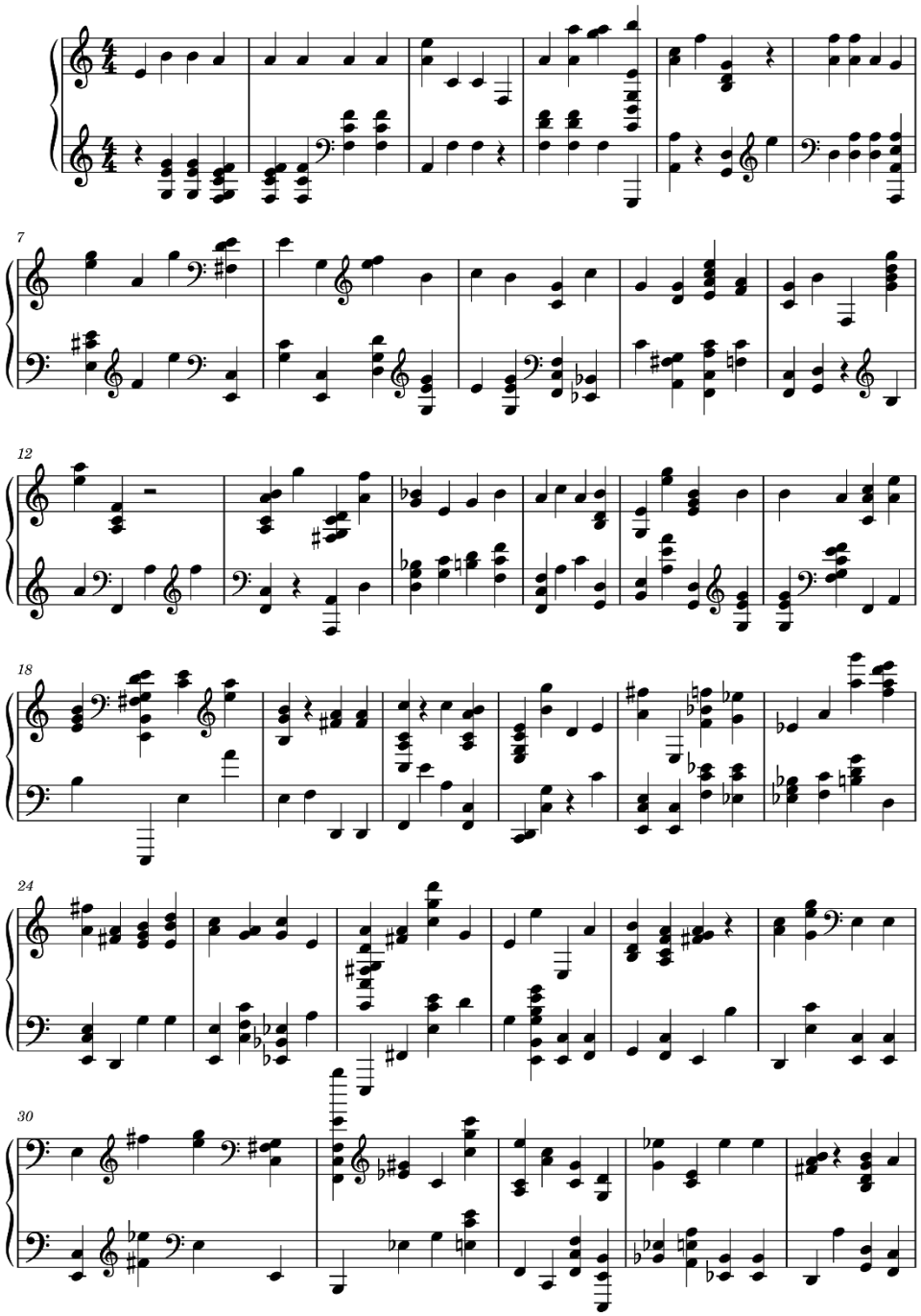


Figure 26 music sheet of a sad song

Each emotion's accuracy and loss as a result of training are as follows:

Figure 27 adventure

Figure 28 happy

Figure 29 romance

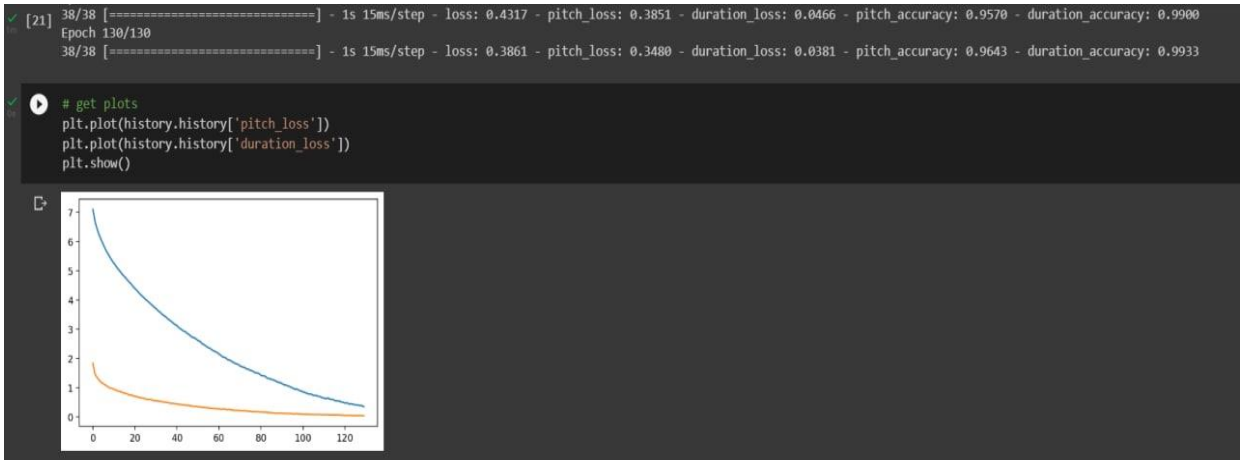


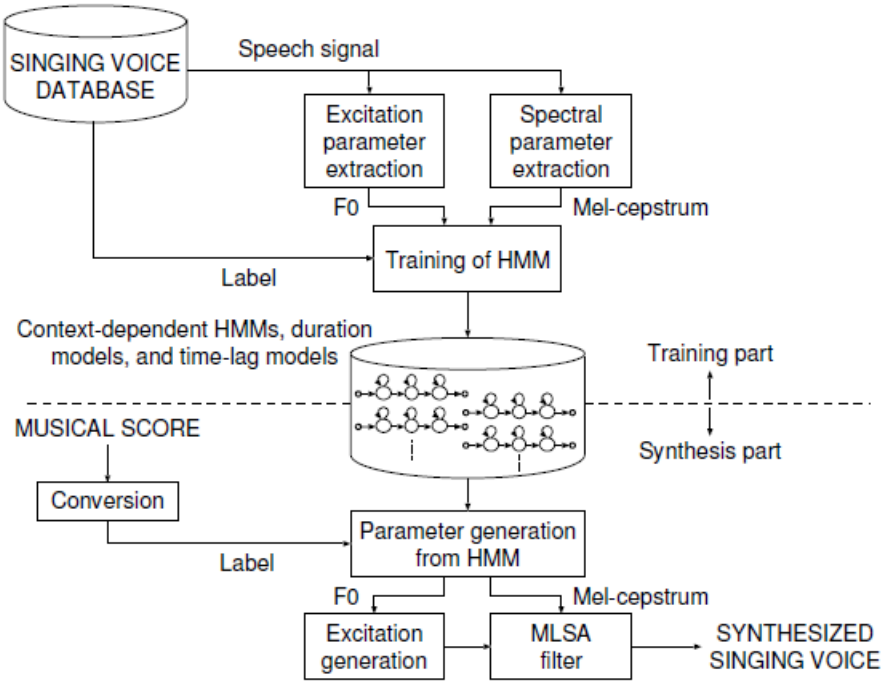
Figure 30 sad

Figure 31 relax

Chapter 6: Speech Synthesis

6.1 Description

One of the most difficult things was the singing. It must translate text into speech, determine the melody's pitch, and then adjust the voice to the melody's notes. Additionally, in order to keep the rhythm, it must attempt to determine where the virtual singer should precisely place each phrase. Therefore, we utilized a hidden Markov model-based singing voice synthesis system (HMMs) This system uses speech synthesis that is HMM-based to create singing voices. Lyrics, tones, and durations of music are jointly described in a single framework of the context-dependent HMM.
It can mimic the voice quality and singing style of the original singer. Results of a singing voice synthesis experiment Show that the proposed system can synthesize smooth and Natural-sounding singing voice.
Currently, there are two main paradigms in the corpus-based speech synthesis area: sample-based approach and statistical approach.
Therefore, in the proposed system, contextual factors, which may affect singing speech (i.e. tones, lyrics, and durations), are used.
HMM-based singing voice synthesis. It consists of training and synthesis parts.
In the training part, first we extract spectral (e.g., mel-cepstral coefficients) and excitation (e.g., fundamental frequencies) parameters from a singing voice database and then they are modeled by context-dependent HMMs. Context-dependent state duration models and time-lag models are estimated.
In the synthesis part, first an arbitrarily given musical score in-cluding lyric to be synthesized is converted to a context-dependent label sequence. Secondly, according to the label sequence, a song HMM is constructed by concatenating the context-dependent HMMs. Thirdly; state durations of the song HMM are determined
With respect not only to the state duration models but also to the time-lag models. Fourthly, spectral and excitation parameters are generated by the speech parameter generation algorithm. Finally, a speech waveform is synthesized directly from the generated spectral and excitation parameters using Mel Log Spectrum, in the next Figure showed *the overview of the HMM-based singing voice synthesis.*



6.1.1 Contextual Factors

In the HMM-based contextual factors that affect singing voice, the following contextual factors were considered:

Phoneme: The preceding, current, and succeeding phonemes.

Tone: The musical tones of the preceding, current, and succeeding musical notes (e.g. "A4", "C5#", "B3_", etc.).

Duration: The durations of the preceding, current, and succeeding musical notes (in 100 me unit).

Position: The positions of the preceding, current, and succeeding musical notes in the corresponding musical bar (in triplet thirty-second note).

These contexts can automatically be determined from the musical score including lyric.

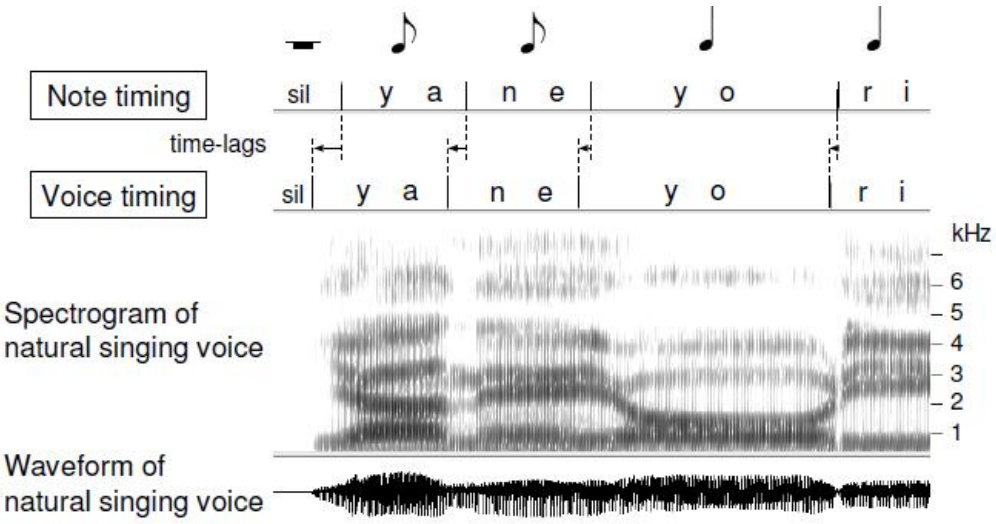
6.1.2 Time-lag Modeling

Another unique feature in the proposed system is time-lag modeling.

In the case of singing voice synthesis, we have to obey rhythm or tempo of the music. Therefore, start timing of musical notes or phoneme durations in each musical note should be determined according to the musical score.

To model this phenomenon, we introduce "time-lag models." After training context-dependent HMMs, the forced alignment is performed to the training data to obtain time lags between musical notes and real human speech.

In the synthesis stage, first we determine the duration of each musical note from the given score including lyric to be synthesized. Then, the time lags of musical notes and state durations of the song HMM, we showed that in the next Figure.



6.2 Dataset

Since we input the outputs of the earlier models (lyrics and music) into a trained model of speech that was trained on multi-speaker speech and MIDI to construct the speech, there is no dataset for our voice.

To create singer voices, we used a model called midi2voice.

Table 1: *Singing voice database.*

Singer	1 male (non-professional)
Songs	60 Japanese children's songs (about 72minutes in total)
Sampling Rate	44.1kHz
Quantization	16bit

Table 2: *Mel-cepstral analysis condition.*

Sampling Rate	16kHz
Frame Shift	5ms
Window Length	25ms
Window Function	Blackman Window
Spectral Feature	24 mel-cepstral analysis [8]

6.3 Preprocessing

6.3.1 Preprocessing the midi file

We specify the kind of instrument we employ, which is the piano.

Only vocals may be present in the midi file, hence there should only be one channel.

We assign the appropriate note to each element after determining whether it is a note or chord in the midi file.

To get the values of the notes, we finally advance the mathematics.

6.3.2 Preprocessing the text file

The system looks for uppercase characters and deletes some punctuation.

The algorithm will then split each phrase by the character (|), replace each ('-') with a (|), and split each word onto a new line.

The function vocals then verify that the text matches the number of notes.

We may now assign the appropriate notes or chords to each word as a result.

6.4 MIDI2XML

MusicXML consists of two main types of elements. One set of elements is used to represent how a piece of music is notated. The other set of elements are used to represent a sonic realization of the score, and are commonly used for MIDI playback.

```
<attributes>
  <divisions>24</divisions>
  <key>
    <fifths>-3</fifths>
    <mode>minor</mode>
  </key>
  <time>
    <beats>3</beats>
    <beat-type>4</beat-type>
  </time>
</attributes>
```

6.3.1 Attributes

element contains information about time signatures, key signatures, transpositions, clefs, and other musical data that is usually specified at the beginning of a piece or at the start of a measure.

6.3.1 Chords

The duration elements in MusicXML move a musical counter. To play chords, we need to indicate that a note should start at the same time as the previous note, rather than following the previous note. To do this in MusicXML, add a chord element to the note.

6.3.1.1 Divisions

Musical durations are commonly referred to as fractions: whole notes, half notes, quarter notes, and the like. While each musical note could have a fraction associated with it, MusicXML instead follows MIDI by specifying the number of divisions per quarter note at the start of a musical part, and then specifying note durations in terms of these divisions.

6.3.1.2 Key

Standard key signatures are represented like MIDI key signatures. The <fifths> element specifies the number of flats or sharps in the key signature - negative for flats, positive for sharps. The fifths name indicates that this value represents the key signature's position on the circle of fifths. MusicXML uses the <mode> element to indicate major or minor key signatures.

6.3.1.3 Time

Standard time signatures are represented more directly in MusicXML than in MIDI 1.0. The beats element represents the time signature numerator, and the beat-type element represents the time signature denominator. Many variations are possible, including composite and interchangeable time signatures.

6.3.1.4 Duration

The duration element is an integer that represents a note's duration in terms of divisions per quarter note. Since our example has 24 divisions per quarter note, a quarter note has a duration of 24. The eighth-note triplets have a duration of 8, while the eighth notes have a duration of 12.

```
<note>
  <rest measure="yes"/>
  <duration>72</duration>
</note>
```

6.5 renderize the speech

We need to determine whether a user prefers a male or female singer based on the spoken language.

If the speaker is rated 10, she should be a girl, and if she is rated 11, he should be a guy.

The purpose of this stage is simply to differentiate between voice genders.

The Model is then given the lyrics, midi, gender (male or female), and Temp.

Finally, we have the speech that is based on the midi and lyrics.

6.6 make a song

The final step was to combine that voice file with the original MIDI music that we generated to produce the song.

We did this by converting the MIDI file to WAV, and then using ffmpeg's Package To have a full Song.

Chapter 4: System Analysis

4.1 Requirements Specification

Requirement specification provide an in-depth description of the proposed application and provide a full picture of the product that is under development. The main point behind requirement specification is to determine the system functionality. In other words, it will describe what the proposed application is supposed to do and how it is expected to perform these functions.

Furthermore, these requirements play a vital role in verification stage where the developed product can be verified that it has fulfilled all of the requirements. For the purpose of this project, we can determine that the user's needs are to generate songs with lyrics. Thus, the process involved in the project is mainly generating songs with lyrics.

Based on the previous we can set the functional and non-functional requirements of the system.

4.2 Functional Requirements

AI:

1- Receive input from mobile application
2- Process the input as per programmed Moodle for songs or dynamic data set
3- Return prediction data
4- Receive text from mobile application
5- Divide the text to words and return it as an array

The users should be able to do the following using the mobile application

1. User should be able to open application and define his preferred type of songs and instruments
2. User should be able to get generating results based on his demands

3. User should be able to play the song and see it's lyrics
4. User should be able to make his own playlist

4.3 Non-Functional Requirements:

1. **Security:** Security is considered one of the most important requirements for any system. The developed application must be secure. In addition, the developed application must be available at all times and keep the user's data safe.
2. **Performance:** The developed application must response to user's actions fast under different workload. This means that the user must not wait for a long time before the task (such as translation time) is completed. This also include all background processes that the user cannot see which must perform under these requirements too.
3. **Usability:** Usability refers to the term of user's experience in interacting with the system. This means that the developed application will be easy to use where users will be able to achieve their tasks efficiently and effectively. In addition, the aim here is to build an application design that is familiar to all users from different age group or computer skills background in order to allow them to use the application easily without help.
4. **Reliability:** The proposed system must have consistency in performing its functions without failure. This means that the application or any of its component should run without any errors and the probability of failure during a specified period should be low.
5. **Scalability:** The proposed application must be scalable which means any new additional feature or security enhancement must be add easily in order to keep the system performance at best. This requirement is considered very important to allow the application to add more features such as different languages for translation.

4.4 Use cases

Use case diagram is a graphical representation of the user's possible interactions with the proposed system. The use case diagram consists of actors, use cases, and different relations that define each interaction. For the proposed system, the use case diagram is shown in figure 5 below; we can identify the following actors:

- User: considered as any individual using the develop application for translation.
- AI system: is considered as a secondary actor that is triggered when user interact with the system and request choices to generate songs
- Admin: is also considered as a secondary actor that control the accounts in this application and modify them.

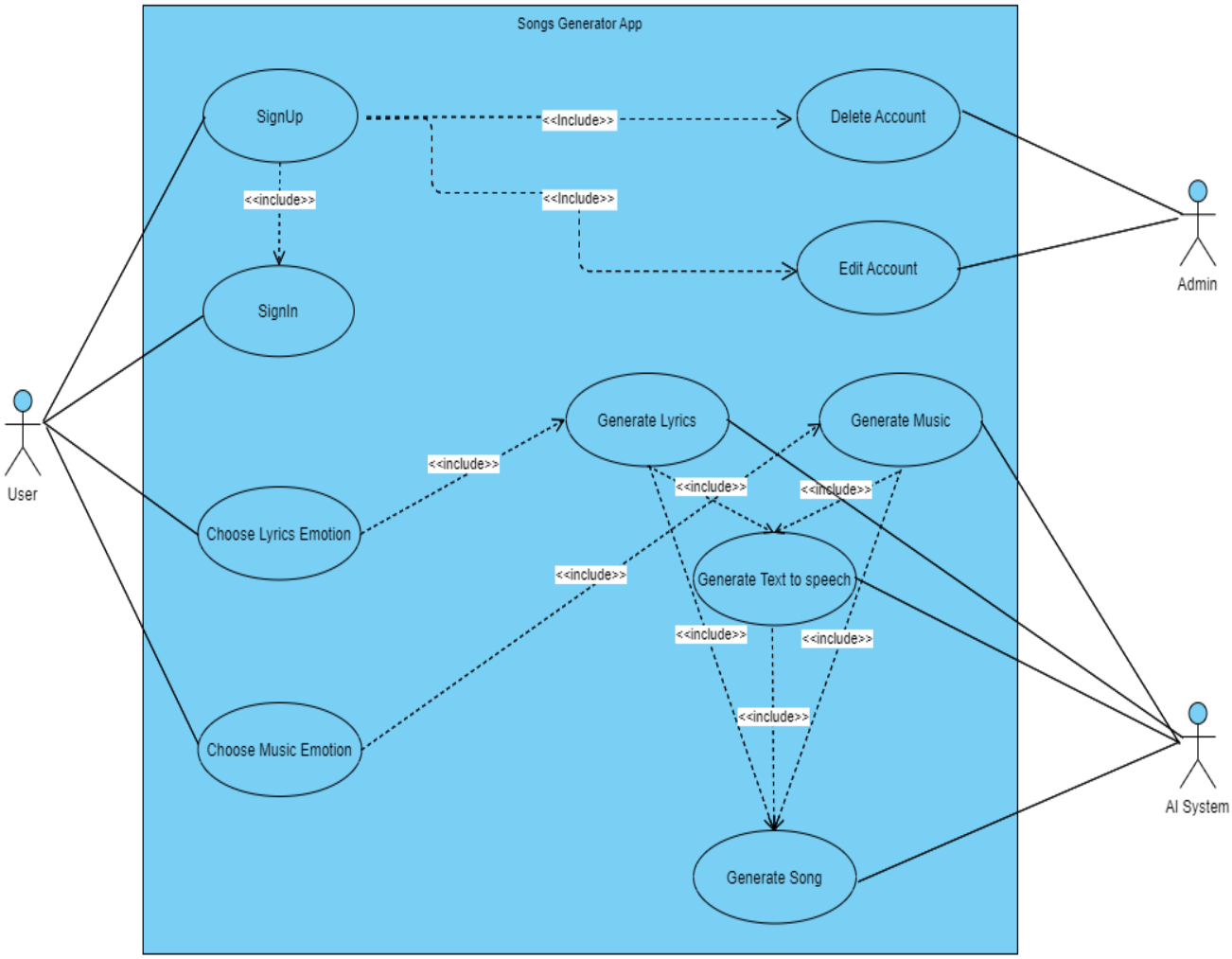


Figure 32 Use Case Diagram

Use Case Name: select types of preferred song	**ID: 1**	**Importance Level: High**
Primary Actor: User	**Use case Type**: Essential	
Pre-conditions: None		
Post-conditions: UI is shown.		
Brief Description: through this use case, user should be able to pick		
Relationships: This use case is related to choose song type, genre, choose singer, create a playlist and choose music instrument to generate songs with lyrics.		
Normal Flow of Events: 1- User will launch the application 2- Choose what he wants to hear 3- Get the translation results.		
Sub Flows: 1- The data sent to AI model through API. 2- The response for the request will return the results after getting it from the AI system		

4.5 Sequence Diagram

Sequence Diagram represent the interactions in particular process or operation as a flow of events. It describes how the actors are interacting with each other and with other entities. For the purpose of our application, we have the following sequence diagrams:

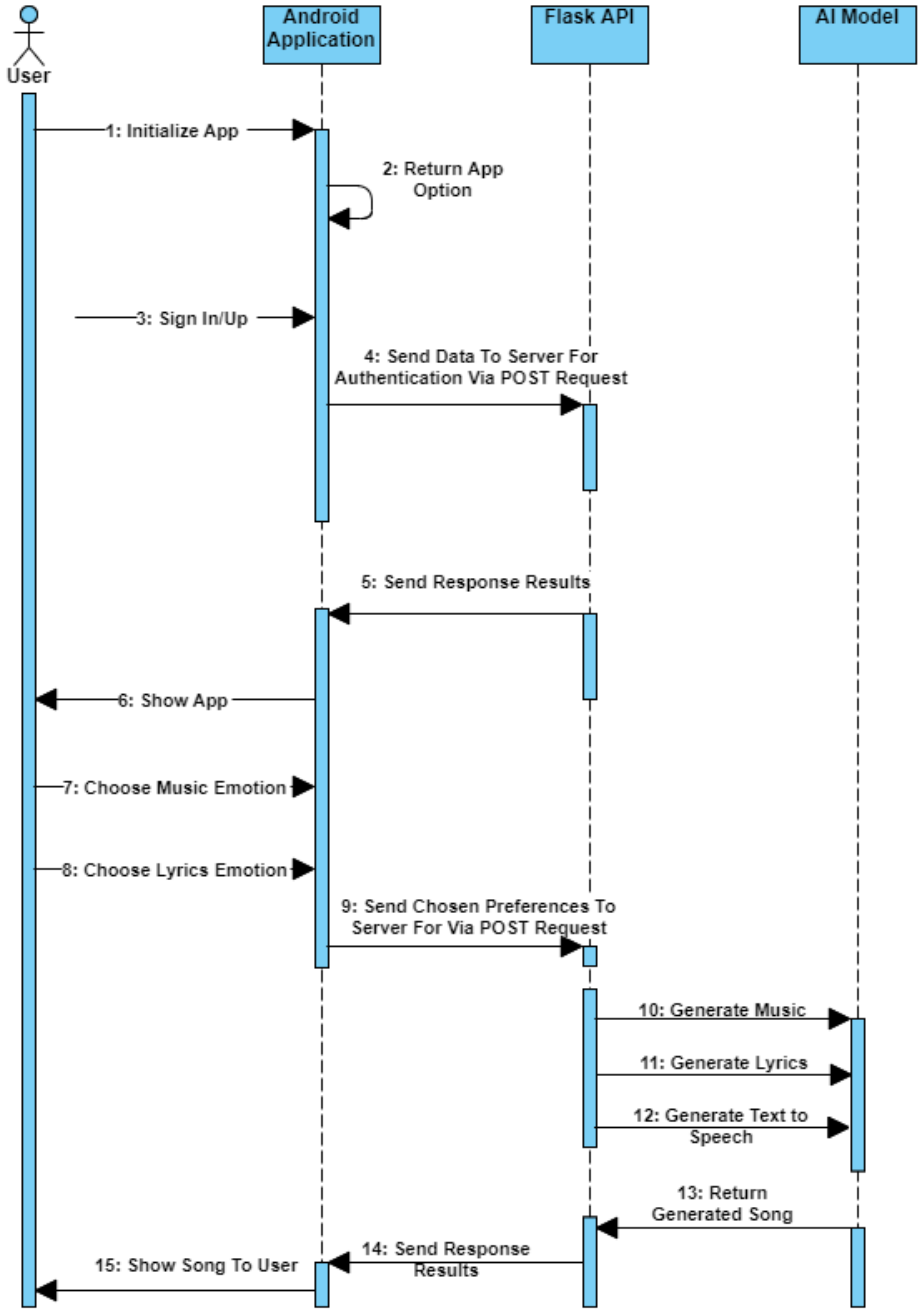


Figure 33 sequence diagram

4.6 Activity Diagram

Activity UML diagram is behavioral diagram that describe system behavior in a way similar to flowchart. It represents the system action graphically in steps from the beginning of the action to the end of it. It contains processes, selection, iteration, and many others between actors and different components of the system.

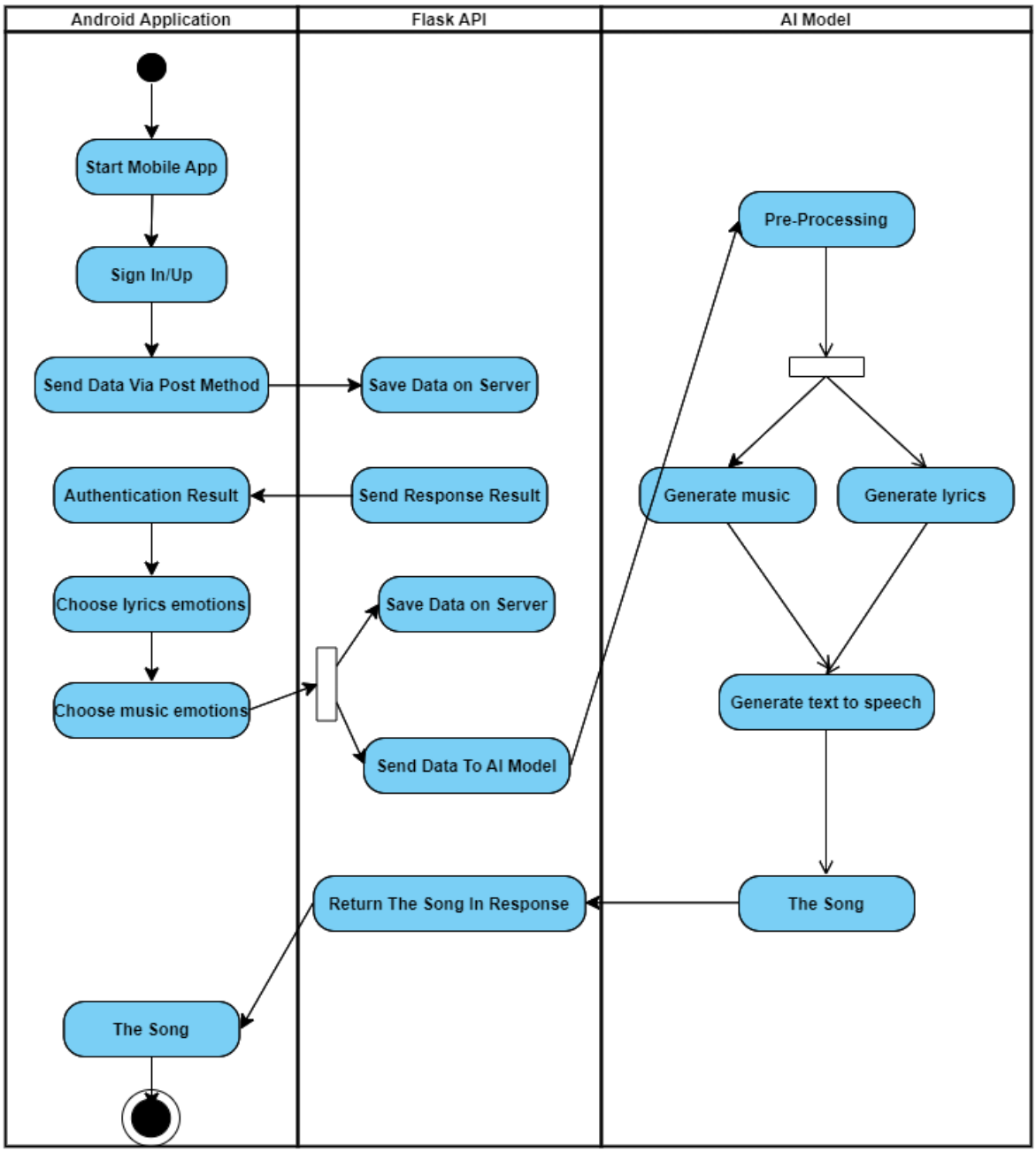


Figure 34 activity diagram

Chapter 5: System Design

5.1 System Components Diagram

A component diagram allows verification that a system has required functionality is acceptable. These diagrams are also used as a communication tool between the developer and stakeholders of the system. Programmers and developers use the diagrams to formalize a roadmap for the implementation, allowing for better decision-making about task assignment or needed skill improvements. System administrators can use component diagrams to plan, using the view of the logical software components and their relationships on the system

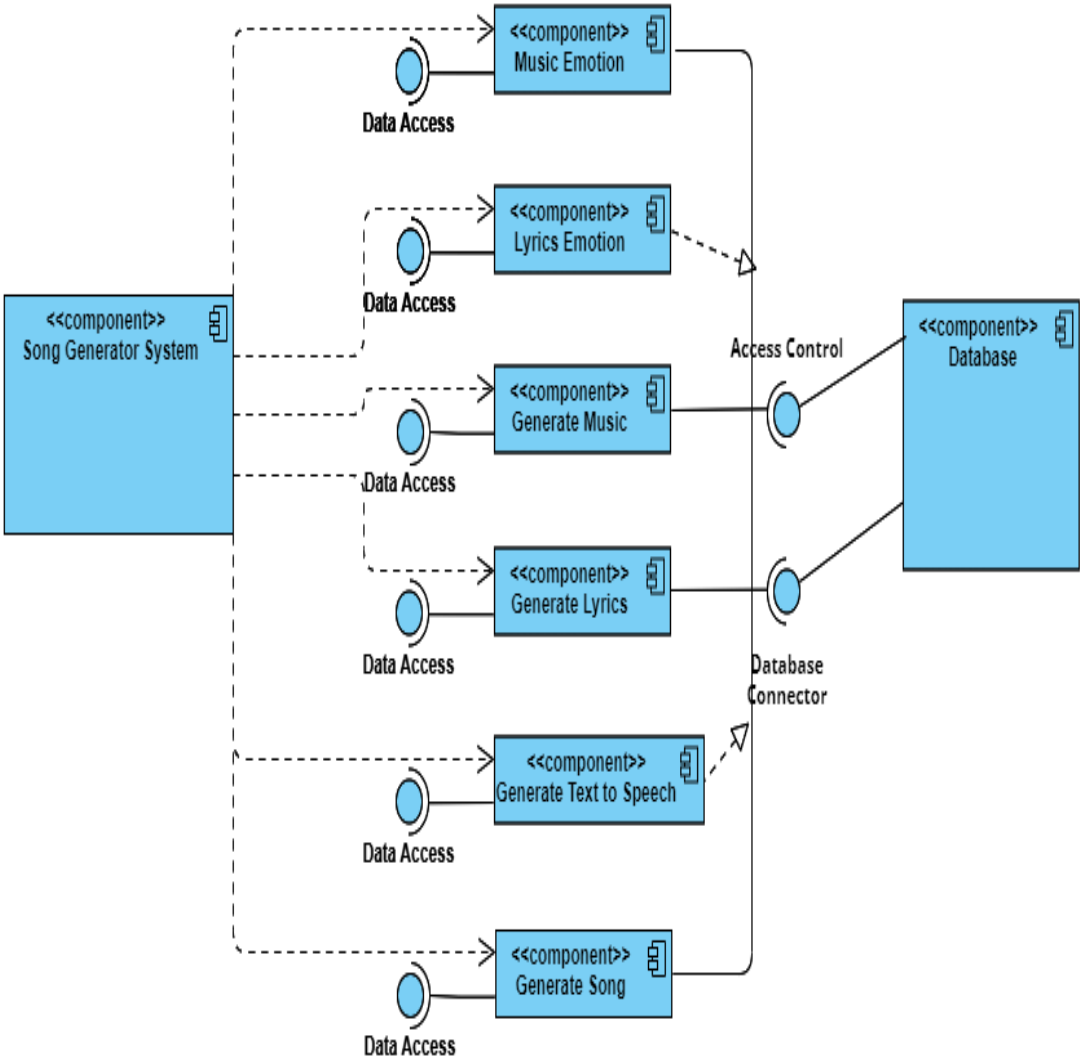


Figure 35 component diagram

5.2 System Context Diagram

System Context Diagrams ... represent all external entities that may interact with a system ... Such a diagram pictures the system at the center, with no details of its interior structure, surrounded by all its interacting systems, environments and activities. The objective of the system context diagram is to focus attention on external factors and events that should be considered in developing a complete set of systems requirements and constraints.

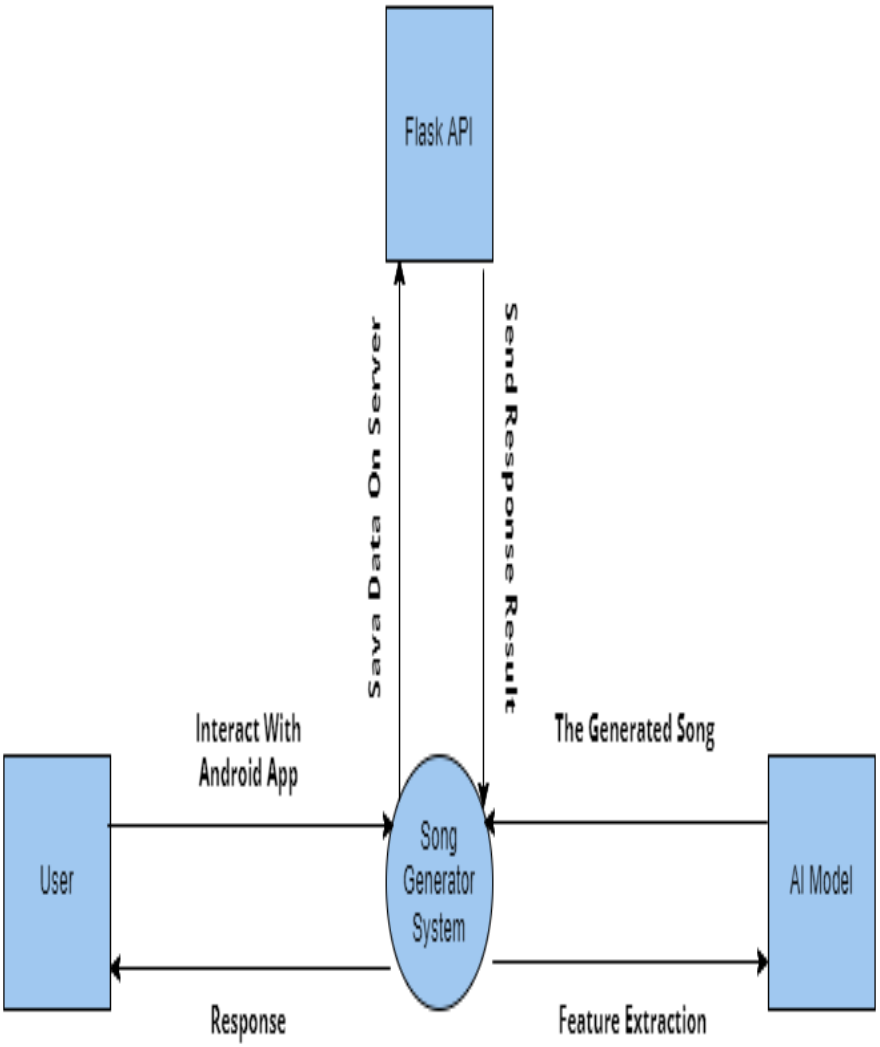


Figure 36 component diagram

5.3 State Diagram

A state diagram, sometimes known as a state machine diagram, is a type of behavioral diagram In the Unified Modeling Language (UML) that shows transitions between various objects.

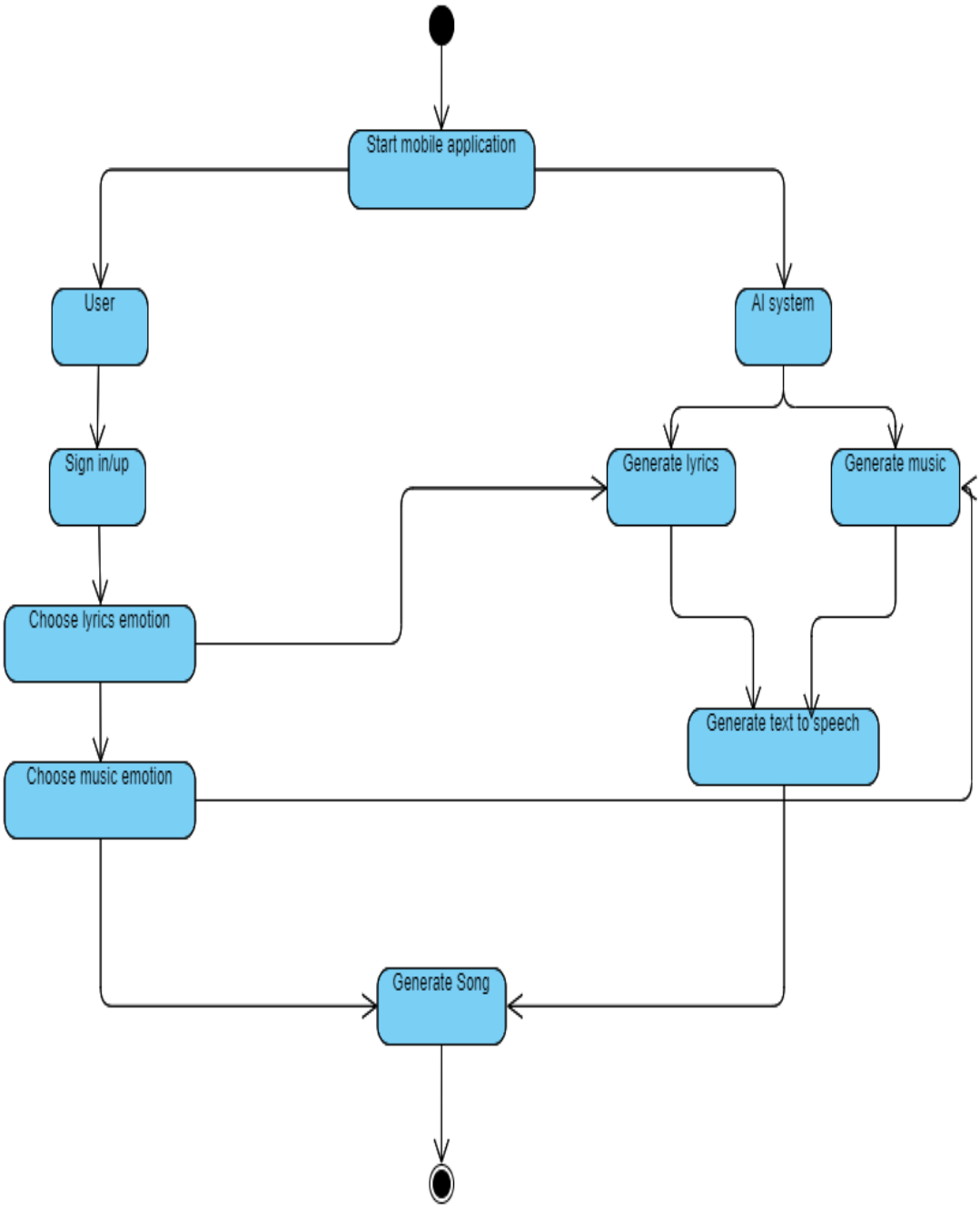


Figure 37 State diagram

5.3 Block Diagram

is a system diagram where the main components or functions are depicted by blocks connected by lines that highlight the connections between the blocks.

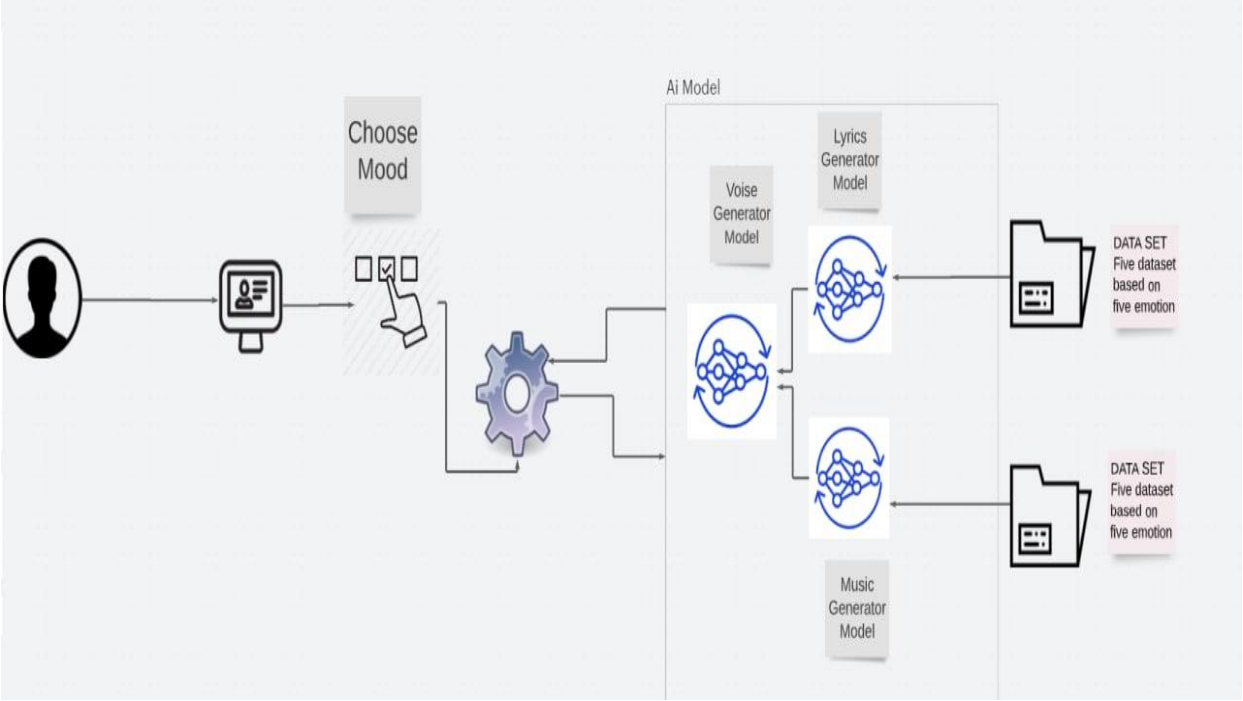


Figure 39 block diagram

5.4 Mobile application & API

The connection between the front end and AI models was developed using custom API that is specifically made to send the music or lyrics throw the API and it starts like that:

- from the frontend side the user sends a request to the api and that is one of the six emotions that is shown in the frontend
- the flask Api takes the request and converts it from Jason to a string to searches for the

```python
@app.route('/lyrics', methods=['GET', 'POST'])
def lyricsRequest():

    time.sleep(1)
    global response
    global response2

    if(request.method == 'POST'):

        request_data = request.data
        request_data = json.loads(request_data.decode('utf-8'))
        feeling = request_data['feelings']
```

- then takes the string and searches for the emotion and gives the file to the response

  The generated lyrics and sends them back to the flutter side, and it saves the lyrics as a text file to pass it for the TTS (text to speech) as the voice to generate

```python
if(feeling == 'relax'):
    response = relax.lyrics_relax.generated
    with open('D:/senior/finalAPI/API/TTS/relax.txt', 'w') as f:
        sys.stdout = f
        sys.stdout =relax.lyrics_relax.generated1()

if(feeling == 'sad'):
    response = sad.Lyrics_sad.generated
    with open('D:/senior/finalAPI/API/TTS/sad.txt', 'w') as f:
        sys.stdout = f
        sys.stdout =sad.Lyrics_sad.generated1()

if(feeling == 'happy'):
    response = happy.lyrics_happy.generated
    with open('D:/senior/finalAPI/API/TTS/happy.txt', 'w') as f:
        sys.stdout = f
        sys.stdout =happy.lyrics_happy.generated1()
```

- it returns to the front end the lyrics to show it

```
    if(feeling == 'advanture'):
        response = adventure.lyrics_advanture.generated
        with open('D:/senior/finalAPI/API/TTS/advanture.txt', 'w') as f:
            sys.stdout = f
            sys.stdout =adventure.lyrics_advanture.generated1()


    if(feeling == 'romance'):
        response = romance.lyrics_romance.generated
        with open('D:/senior/finalAPI/API/TTS/romance.txt', 'w') as f:
            sys.stdout = f
            sys.stdout =romance.lyrics_romance.generated1()


    if(feeling == 'dreamy'):
        response = dreamy.lyrics_dreamy.generated
        with open('D:/senior/finalAPI/API/TTS/dreamy.txt', 'w') as f:
            sys.stdout = f
            sys.stdout =dreamy.Lyrics_sad.generated1()



    return 'ok'
else:
    return jsonify({'feelings': response})
```

- the api generates the music based on the same request from the front-end the emotion

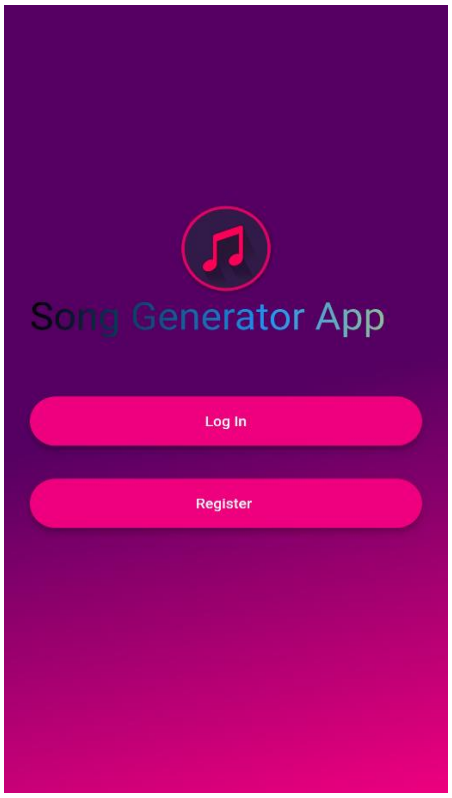
  And takes the generated music to the assets of the flutter to play it in the app

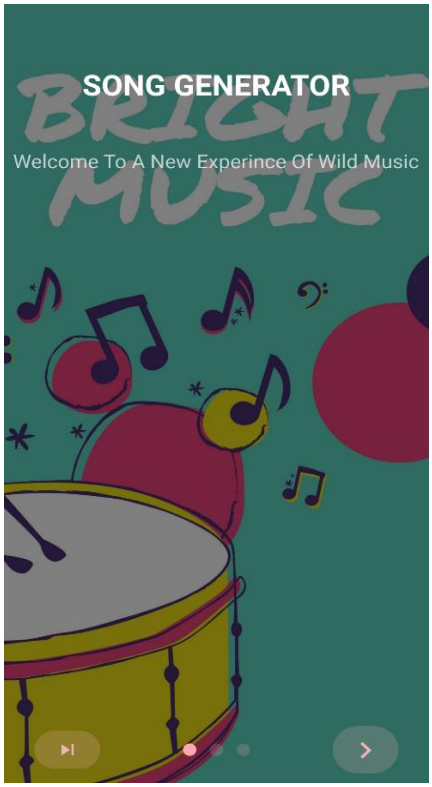
- and takes the generated tts (Text To Speech) and save it in the flutter assets

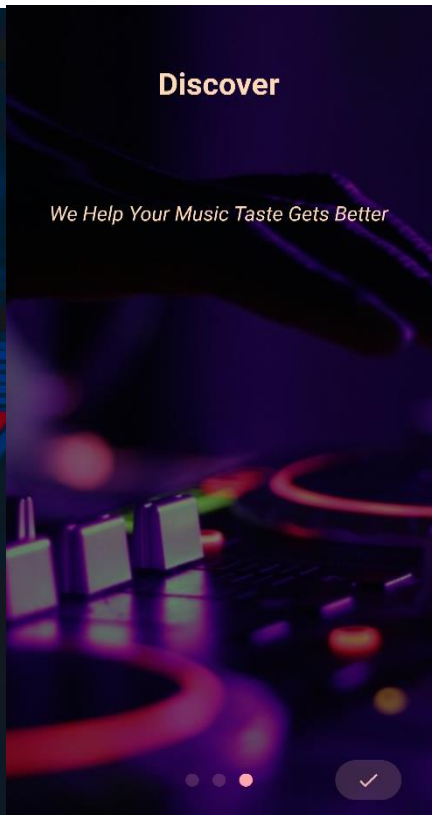
```
@app.route('/music', methods=['GET', 'POST'])
def musicRequest():

    time.sleep(1)
    global response
    global response2

    if(request.method == 'POST'):

        request_data = request.data
        request_data = json.loads(request_data.decode('utf-8'))
        feeling = request_data['feelings']

        if(feeling == 'happy'):
            happy_music.generated_stream.write('midi', fp='D:/senior/finalAPI/API/TTS/happy.mid')
            happy_music.generated_stream.write('midi', fp='D:/fluttersdks/semior_demo/senior_demo/assets/sf2/happy.mp3')
            response2='ok'
            renderize_voice("D:/senior/finalAPI/API/TTS/happy.txt",'D:/senior/finalAPI/API/TTS/happy.mid','female',121)

        if(feeling == 'advanture'):
            adventure_music.generated_stream.write('midi', fp='D:/senior/finalAPI/API/TTS/advanture.mid')
            adventure_music.generated_stream.write('midi', fp='D:/fluttersdks/semior_demo/senior_demo/assets/sf2/advanture.mp3')
            response2='ok'
            renderize_voice("D:/senior/finalAPI/API/TTS/advanture.txt",'D:/senior/finalAPI/API/TTS/advanture2.mid','female',121)

        if(feeling == 'romance'):
            romance_music.generated_stream.write('midi', fp='D:/senior/finalAPI/API/TTS/romance.mid')
            romance_music.generated_stream.write('midi', fp='D:/fluttersdks/semior_demo/senior_demo/assets/sf2/romance.mp3')
            response2='ok'
            renderize_voice("D:/senior/finalAPI/API/TTS/romance.txt",'D:/senior/finalAPI/API/TTS/romance2.mid','female',121)
```

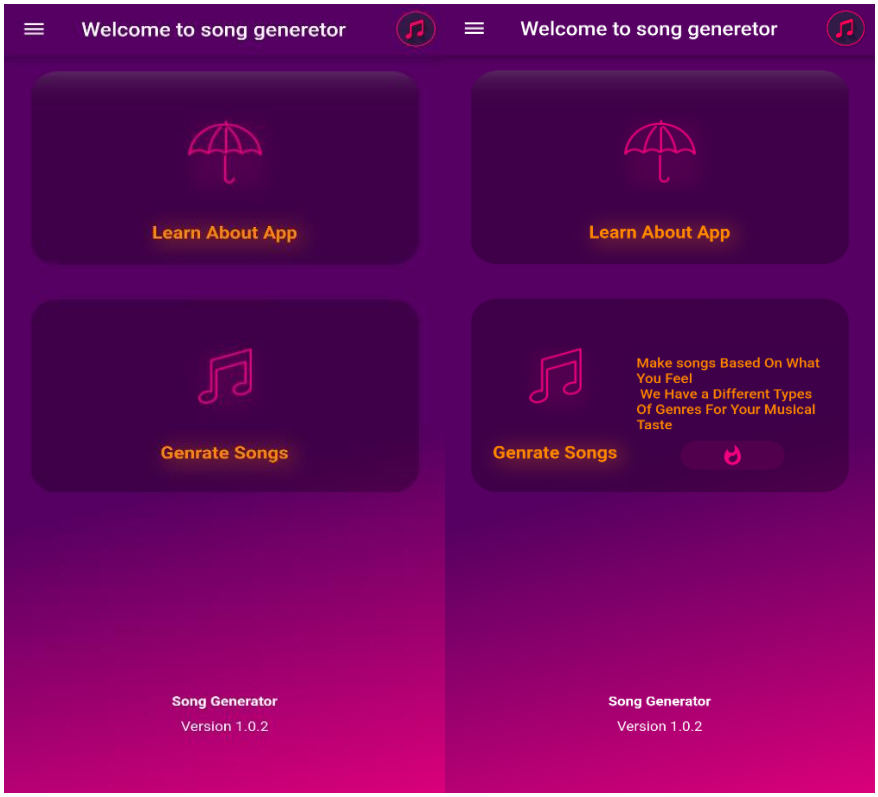
```
        if(feeling == 'sad'):
            sad_music.generated_stream.write('midi', fp='D:/senior/finalAPI/API/TTS/sad.mid')
            sad_music.generated_stream.write('midi', fp='D:/fluttersdks/semior_demo/senior_demo/assets/sf2/sad.mp3')
            response2='ok'
            renderize_voice("D:/senior/finalAPI/API/TTS/sad.txt",'D:/senior/finalAPI/API/TTS/sad2.mid','female',121)

        if(feeling == 'relax'):
            relax_music.generated_stream.write('midi', fp='D:/senior/finalAPI/API/TTS/relax.mid')
            relax_music.generated_stream.write('midi', fp='D:/fluttersdks/semior_demo/senior_demo/assets/sf2/relax.mp3')
            response2='ok'
            renderize_voice("D:/senior/finalAPI/API/TTS/relax.txt",'D:/senior/finalAPI/API/TTS/relaxvoice.mid','female',121)

        return 'ok'
    else:
        return jsonify({'feelings': response2})




if __name__ == "__main__":
    app.run(host="169.254.209.151", port=40222)
    #http_server = WSGIServer(("192.168.1.104", 4040), app)
    #http_server.serve_forever()
    #app.run(thread=True)
```

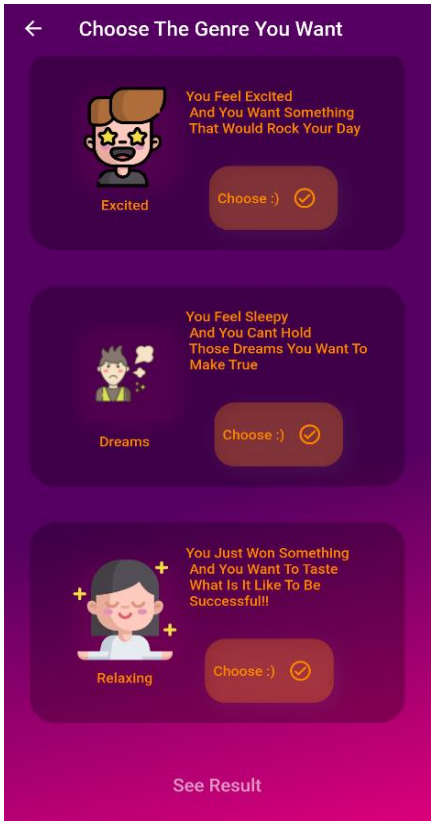
5.4 Application pages design

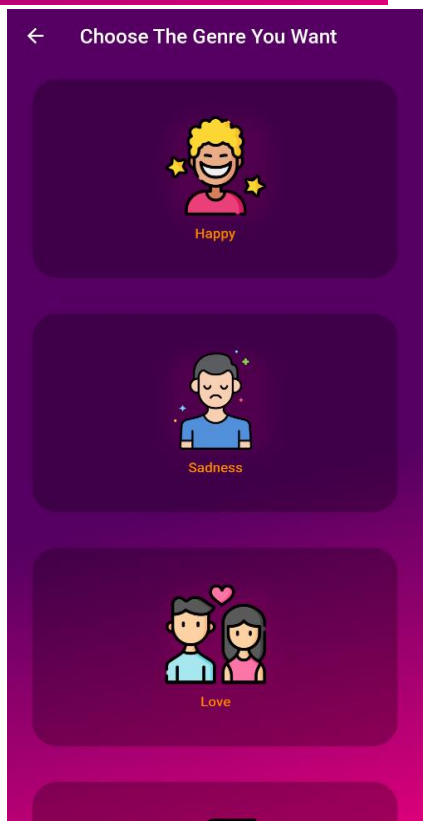


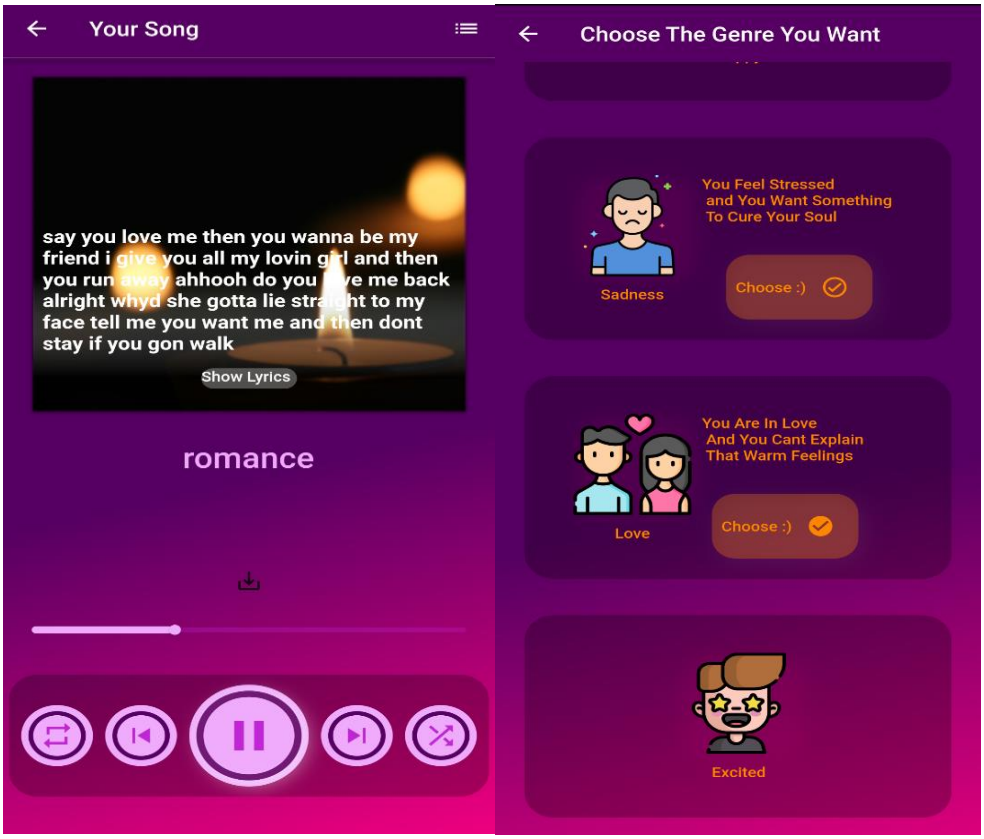


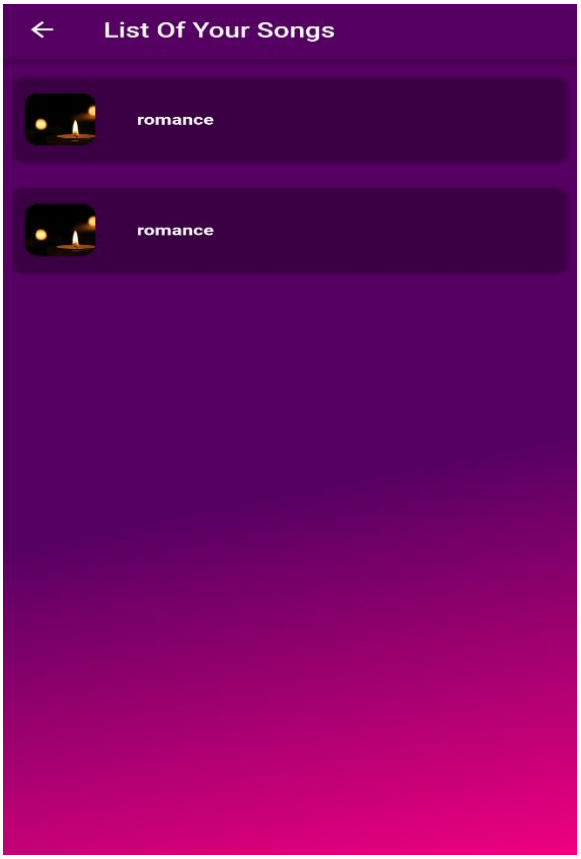



Welcome to song generetor
Learn About App
Genrate Songs
Song Generator
Version 1.0.2
Welcome to song generetor
Learn About App
Make songs Based On What You Feel
We Have a Different Types Of Genres For Your Musical Taste
Genrate Songs
Song Generator
Version 1.0.2


Choose The Genre You Want
You Feel Excited
And You Want Something
That Would Rock Your Day
Excited
Choose :)
You Feel Sleepy
And You Cant Hold
Those Dreams You Want To
Make True
Dreams
Choose :)
You Just Won Something
And You Want To Taste
What Is It Like To Be
Successful!!
Relaxing
Choose :)
See Result


Choose The Genre You Want
Happy
Sadness
Love


Your Song
say you love me then you wanna be my friend i give you all my lovin girl and then you run away ahhooh do you ve me back alright whyd she gotta lie straight to my face tell me you want me and then dont stay if you gon walk
Show Lyrics
romance
Choose The Genre You Want
You Feel Stressed and You Want Something To Cure Your Soul
Sadness
Choose :)
You Are In Love And You Cant Explain That Warm Feelings
Love
Choose :)
Excited


List Of Your Songs
romance
romance

Chapter 6: Environments and implementation

6.1 Technologies Used

6.1.1 Tensor Flow

Tensor Flow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. Tensor Flow is a symbolic math library based on dataflow and differentiable programming.

6.1.2 Colab

Colab is free Jupyter notebook environment that runs entirely in the cloud. Most importantly, it does not require your team members can simultaneously edit a setup and the notebooks that you create. Colab supports many popular machine-learning libraries, which can be easily loaded
In your notebook.

6.1.3 Python

Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation.

6.1.4 Postman

Postman is an API client that makes it easy for Developers to create, share, test and document APIs. This is done by allowing users to create and Save simple and complex HTTP/s requests, as well As read their responses.
The result - more efficient and less tedious work.

6.1.5 Keras

Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the Tensor Flow library.

6.1.6 Flask

Flask is a micro web framework written in Python. It is classified as a micro framework because it does not require particular tools or libraries.
It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

6.1.7 Numpy

NumPy is a library for the Python programming language, Adding support for large, multi-dimensional arrays and Matrices, along with a large collection of high-level Mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

6.1.8 Flutter

Flutter is a cross-platform UI toolkit that is Designed to allow code reuse across operating Systems such as is and Android, while also Allowing applications to interface directly with Underlying platform services. The goal is to Enable developers to deliver high-performance Apps that feel natural on different platforms, Embracing differences where they exist while Sharing as much code as possible.
During development, Flutter apps run in a VM that offers tasteful hot reload of changes without needing a full recompile. For release, Flutter apps are compiled directly to machine code, whether Intel x64 or ARM instructions, or to JavaScript if targeting the web. The framework is open source, with a permissive BSD license, and has a thriving ecosystem of third-party packages that supplement the core library functionality.

Benefits of using Flutter:

- Whether you are developing an enterprise application or just a prototype, there are tons of benefits to using Flutter.
- Less development time
- Everyone wants to develop high-quality apps as quickly as possible. Flutter enables faster app development using its hot reload and hot restart features. Both of these help to see changes instantly so that one can quickly iterate the design while working.
- Hot reload provides a sub-second time between when you save the code and when the change appears on the same screen that you were. Even if it is 30 pages into the navigation of your app. Hot, reload came out of co-experimentation with lots of engineers at Google, all of who cared about just making developers happy and productive and get home on time.

6.1.9 Music21

Music21 is a set of tools for helping scholars and other active listeners answer questions about music quickly and simply. If you have ever asked, yoursMusic21 is a Python-based toolkit for computer-aided musicology.
People use music21 to answer questions from musicology using computers, to study large datasets of music, to generate musical examples, to teach fundamentals of music theory, to edit musical notation, study music and the brain, and to compose music (both algorithmically and directly).

6.1.10 Anaconda

Anaconda is an open-source distribution of
The Python and R programming languages
For data science that aims to simplify package
Management and deployment. Package versions
In Anaconda are managed by the package management system, condominium, which analyzes the current environment before executing an installation to avoid disrupting other frameworks and packages.
The Anaconda distribution comes with over 250 packages automatically installed. Over 7500 additional open-source packages can be installed from PyPI as well as the conda package and virtual environment manager. It also includes a GUI (graphical user interface), Anaconda Navigator, as a graphical alternative to the command line interface. Anaconda Navigator is included in the Anaconda distribution, and allows users to launch applications and manage conda packages, environments and channels without using command-line commands. Navigator can search for packages, install them in an environment, run the packages and update them.

6.1.11 Visual Studio Code

Visual Studio Code (famously known as VS Code)
Is a free open source text editor by Microsoft?
VS Code is available for Windows, Linux, and
macOS. Although the editor is relatively lightweight,
It includes some powerful features that have
made VS Code one of the most popular development
environment tools in recent times .The Anaconda
distribution comes with over 250 packages automaticall¸
additional open-source packages can be installed from PyPI as well as the conda package and virtual environment manager. It also includes a GUI (graphical user interface), Anaconda Navigator, as a graphical alternative to the command line interface. Anaconda Navigator is included in the Anaconda distribution, and allows users to launch applications and manage conda packages, environments and channels without using command-line commands. Navigator can search for packages, install them in an environment, run the packages and update them.

6.1.12 Muse Score

Muse Score is open source music notation software that
Runs on Windows, MacOS, and Linux, and is available
In over forty different languages. It features an easy
To use WYSIWYG editor with audio score playback for
Results that look and sound beautiful,
Rivaling commercial offerings like Finale and Sibelius.

6.1.13 Http library

A compostable, Future-based library for
Making HTTP requests. This package
Contains a set of high-level functions and
Classes that make it easy to consume HTTP resources.
It's multi-platform, and supports mobile,
Desktop and the browser.

6.1.14 audio players

A Flutter plugin to play multiple simultaneously audio files,
Works for Android, IOS, Linux, macOS, Windows, and web.

Conclusion

The deep learning model is capable of producing a certain kind of music and lyrics. A system that creates straightforward lyrical art songs and music based on a music note dataset was created as the first step toward a generative music-lyrics model. As a result, this tutorial describes how to use Python's Keras and LSTM recurrent neural networks to generate a text model character by character. Based on the inputs, lyrics are generated.

This project will result in a collection of brand-new lyrics based on the provided feeling. The music is distinctive and enjoyable, and generally, the model's scores have a high opinion value. Therefore, it is possible to create musical compositions that are visually beautiful by employing a high-level representation of our model.

Future research will examine these characteristics and the use of a larger and more complete dataset to make music that is more diverse than the style or genre it was trained on. The experimental results showed that the proposed system could mimic the voice quality and singing style of the original singer and that it was possible to synthesize a singing voice with a natural sound. The voice that was generated was using a model called midi2voice that helps us with generation voice that takes lyrics and music as input to produce Speech.

Future Works

The application can be enhanced in many ways to benefit the user.

Create lyrics in several languages (Arabic, French,) Additionally, by training on a much larger dataset and exploring the idea of using different instruments and several recordings, we can produce a certain type or genre of music.

Additionally, create voices by selecting the singer of the song and creating a voice in various dialects and styles.

References

[1] Venkat Chadalavada “LSTM based Lyrics and Music Generation Model”, medium, Mar 16, 2020.

[2] Aishwarya Sathya Kumar “Lyrics generator using LSTM”, medium, Feb 29, 2020.

[3] George Seif “You can now speak using someone else’s voice with Deep Learning”, towardsdatascience, Jul 2, 2019.

[4] Allan Souza “Text to Speech with Real-time Voice Cloning”, medium, Aug 10, 2020.

[5] Yi-Hsuan Yang“MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment”, paperswithcode19 Sep 2017.

[6] Szu-Yu Chou, “A Convolutional Generative Adversarial Network for Symbolic-domain Music Generation, 31 Mar 2017.

[7] Sean Vasquez “A Generative Model for Audio in the Frequency Domain”, paperswithcode, 4 Jun 2019.

[8] Allen Huang “Deep Learning for Music”, paperswithcode ,15 Jun 2016.

[9] Swetapadma Das “Music Generation Using Deep Learning”, medium,Jun 30 2021.

[10] Joachims “Greek Lyrics Generation”, ncbi.nlm.nih.gov2020 , May 6.

[11] Philippe Petit “Piano Instrumental Music Generation using Deep Neural Networks”, towardsdatascience ,May 27, 2021.

[12] Viet Pham “Jazz music generation using GPT, Jan 6, 2021.

[13] Stefania Cristina “The Attention Mechanism from Scratch”, towardsdatascience , September 20, 2021.

[14] Allen Nie “Understand Numpy Reshape, Transpose, and Theano Dimshuffle", anie.me ,December 13, 2015.

[15] Jason Brownlee “Softmax Activation Function with Python", anie.me ,October 19, 2020.

ملخص

في هذه الدراسة التي تتمحور حول إمكانية انشاء نظام قادر على توليد الأغاني الجديدة(كلمات وموسيقا وصوت) بالاعتماد على مشاعر الانسان باستخدام خوارزميات الذكاء الصنعي, حيث يمكن للمستخدم سماع الأغاني الجديدة التي تناسب مزاجه ويتم ذلك من خلال توليد الكلمات الجديدة باستخدام بنية الشبكات العصبونية التي تأخذ دخل نصي والخرج أيضا نصي , وفي مرحلة توليد الموسيقا تعتمد أيضا على الشبكات العصبونية تأخذ ملفات بصيغة موسيقية معينة كمدخلات وتتوقع الملاحظات والمدة الزمنية لهم كمخرجات .
وأخيرا عندما يتم توليد الكلمات والموسيقا سوف يتم توليد صوت المغني بما يتناسب مع الموسيقى والكلمات وبذلك يكون تم انتاج اغنية جديدة كليا بعيدا عن التدخل البشري ومع الاعتماد على عدة شبكات عصبونية مختلفة التي تعتمد على معادلات رياضية وخوارزميات تعمل بشكل معين على المعلومات التي لدينا للحصول على أفضل نتيجة ممكنة التي تعطي الشعور بالرضا والسعادة للمستخدمين ويتم توظيف ذلك في تطبيق للأجهزة الذكية ، حيث يمكنهم الاستماع على الاغنية الذي يرغبون في سماعها حسب اختيارهم المزاجي .

الجامعة العربية الدولية

الجامعة العربية الدولية AIU جامعة سورية خاصة أُحدثت عام 2005، خططها الدراسية والوثائق الصادرة عنها معتمدة ومصدقة من قبل وزارة التعليم العالي في الجمهورية العربية السورية.

تعمل الجامعة على تحقيق الأهداف الآتية:

- إعداد جيل متميز من الخريجين الجامعيين القادرين على تلبية الحاجات النوعية للمجتمع والنهوض به.
- الإسهام في البحوث العلمية النظرية والتطبيقية التي تخدم أغراض التنمية الوطنية، ويتم العمل على حث الأساتذة والعاملين الأكاديميين على البحث العلمي والمشاركة في المؤتمرات والندوات التي تنظم الأبحاث.
- تحقيق الشراكة مع الجامعات العربية والأجنبية المرموقة بهدف التطوير والتحديث المستمرين للعمل الأكاديمي والقيام ببحوث علمية مشتركة.
- استقطاب الكفاءات الأكاديمية والبحثية المتميزة عن طريق توفير البيئة المناسبة لعملها.

الجامعة العربية الدولية من الجامعات السورية الأولى التي جرى تأسيسها وافتتاحها، وقد تمكنت من اجتذاب الكفاءات التعليمية والبحثية والإدارية المتميزة، لإنشاء صرح متكامل من النواحي الأكاديمية والتنظيمية والإدارية. وتمكنت من تخريج كوادر من المبدعين والمتميزين من خلال توفير بيئة تعليمية ترتكز إلى مقومات نوعية ومادية فريدة منها:

- الخطط الدراسية الحديثة والمتطورة المستندة إلى نظام الساعات المعتمدة.
- الأطر التعليمية المنتقاة بعناية كبيرة.
- المختبرات العلمية الحديثة، ومختبر للمكتبات الإلكترونية.
- المحفزات المادية والمعنوية للطلبة.
- تطبيق طرائق التدريس التفاعلي.
- التوجيه والإرشاد الأكاديمي والتربوي.
- مجموعة كبيرة من اتفاقيات التعاون العلمي مع جامعات محلية وإقليمية ودولية ذات سمعة مرموقة.
- اتفاقيات ومذكرات تفاهم متعددة مع العديد من مؤسسات المجتمع المدني.
- الحرم الجامعي اللائق والمزود بكافة المرافق العلمية والرياضية والترفيهية، والذي نشجعك على زيارته والتعرف على مزاياه.
- الأنشطة والأندية الطلابية بمختلف أنواعها: الرياضية والثقافية والعلمية والاجتماعية.

في الجامعة العربية الدولية سنوات الحياة الجامعية هي وقت للاستثمار في مستقبل الطالب. فالمعارف والخبرات التي يحصلها في قاعة المحاضرات والمختبرات ستساعده في تطوير ذاته، وستمنحه أسباب النجاح في التخصص الذي اختاره، والنشاط الطلابي الذي يمارسه سيساعده في توسيع أفقه، وفعاليات التدريب والأندية والرياضة ستمكنه من تطوير مواهبه، ولربما تساعده في اكتشاف مواهب جديدة.
ليستثمر وقته وذهنه وروحه في جامعتنا كي يجني فوائد عمله والوقت الذي كرسه في السنين القادمة. ونحن سوف نكون بجانب طلبتنا في كل خطوة على دربهم.

الجامعة العربية الدولية

كلية الهندسة المعلوماتية والاتصالات

مشروع التخرج

مولد الأغاني

تم تقديمه إلى

قسم الهندسة المعلوماتية

تقديم

أمجد المرعي **أوس عبد الجميد**

بإشراف

د. ندى غنيم **م. ماسة بعلي**

تموز 2022