UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

A Deep Learning Based Tool For Ear Training

Progress Review Report

David Nogales Pérez

Bachelor Thesis
Specialisation in Computing

Director: Enrique Romero Merino

April, 2022

Contents

List of Figures

List of Tables

CHAPTER 1

Context and Scope

1.1 Introduction

In the following sections, the problem that this project aims to solve will be defined, as well as the theory necessary to understand it.

1.2 Context

The context in which this project is developed is as a Bachelor's thesis of the Computer Engineering Degree specialising in Computer Science, which is imparted at the Facultat d'Informàtica de Barcelona at the Universitat Politècnica de Catalunya. The project is overseen and mentored by Enrique Romero Merino, associate professor at the Department of Computer Science.

1.3 Concepts

Fundamental concepts regarding the scope of the project are defined in the following sections.

1.3.1 Deep Learning

Deep Learning, as defined in [1], is a subset of Machine Learning that provides different kinds of techniques and algorithms that allows computers to "learn" from great amounts of data. Neural Networks are the main algorithms of Deep Learning whose structure of layers of nodes is inspired by the human brain and its neurons [2]. An image of a common structure of a neural network is provided in Figure 1.1

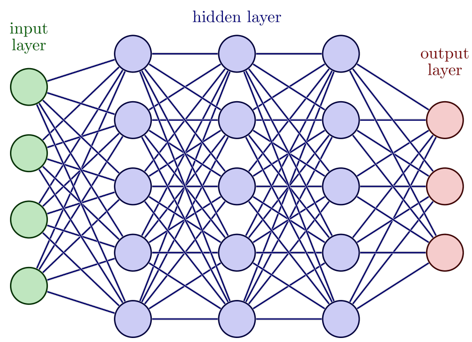

Figure 1.1: Common Neural Network Structure. [3]

Natural Language Processing (NLP)

Natural Language Processing refers to the study of natural language used daily by humans such as text or speech[4] and with the rise of computers and Deep Learning techniques, the study of this field has been greatly broaden.

Common problems tackled inside this field are:

- Text Classification.
- Language Modeling.
- Machine Translation.
- Speech Recognition.

Sequence-to-sequence Models

From an **NLP** perspective, Machine translation is a task mainly used to translate between a source and a target language used by different kinds of people. A **sequence-to-sequence** model is a more generalised approach to translation that, not only includes machine translation but other common tasks such as speech recognition, response generation, and practically anything [5]. A visual example of said tasks can be seen in Figure 1.2

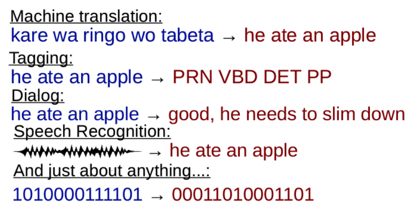

Figure 1.2: Example of Sequence-to-sequence tasks. [5]

Recurrent Neural Networks (RNNs)

RNNs are neural network architectures that differ from the common feedforward neural networks in mainly one aspect, they are oriented to time series or sequential tasks since its internal structure is suited to handle data with relationships through time.

Throughout the years, variations of the same architecture were introduced to tackle different issues that the original one presented, such as the **vanishing gradients** problem[1] which was addressed with the introduction of the **Long Short-Term Memory (LSTM)**[6] or the **Gated Recurrent Unit (GRU)**[7] which tried to maintain the gains of **LSTMs** while presenting a simplified architecture. A simple representation of each architecture can be seen in Figure 1.3.

An empirical comparison of said models was performed in [9], where a prediction task applied to music datasets was used as a benchmark between the models. The findings concluded in that the latter models performed equally well and outperformed the original **RNN** architecture. Some of the results can be seen in the table 1.1 where **GRU** model slightly outperforms the rest of the models.

[1] The vanishing gradients problem arises from the decaying error between the layers when performing backpropagation[6]

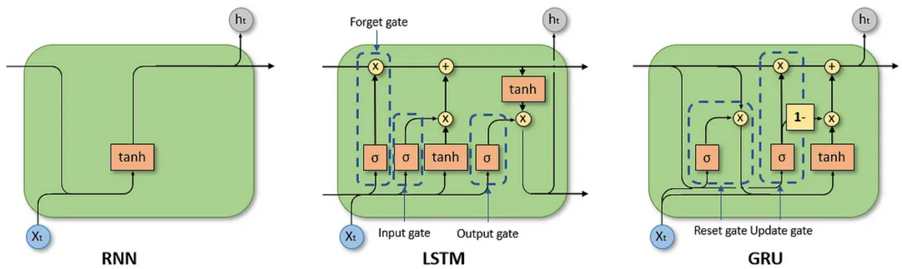

Figure 1.3: RNNs internal architecture. [8]

Music Dataset		tanh	GRU	LSTM
Nottingham	train	3.22	2.79	3.08
	test	**3.13**	3.23	3.20
JSBChorales	train	8.82	6.94	8.15
	test	9.10	**8.54**	8.67
MuseData	train	5.64	4.93	6.49
	test	6.23	**5.99**	6.23
Piano-midi	train	5.64	4.93	6.49
	test	9.03	**8.82**	9.03

Table 1.1: Average negative log-probabilities in music datasets. [9]

Transformers

With the introduction of the transformer architecture[10] (seen in Figure 1.4), models such as **BERT** (Bi-directional Encoder Representations from Transformers)[11] and **GPT** (Generative Pre-Training)[12], achieved **state-of-the-art** results on a wide range of **NLP**-related tasks.

Music Transformer[13] and **MusicLM**[14] are some examples of what can be achieved with this architecture when tackling problems in the music domain such as generating music with long-term structure or generating music from text descriptions.

1.4 Problem to be Solved

As a student at various music schools, I noticed an extensive lack of on-demand exercises that aim to develop one or more necessary basic skills to achieve a certain level of musicianship.

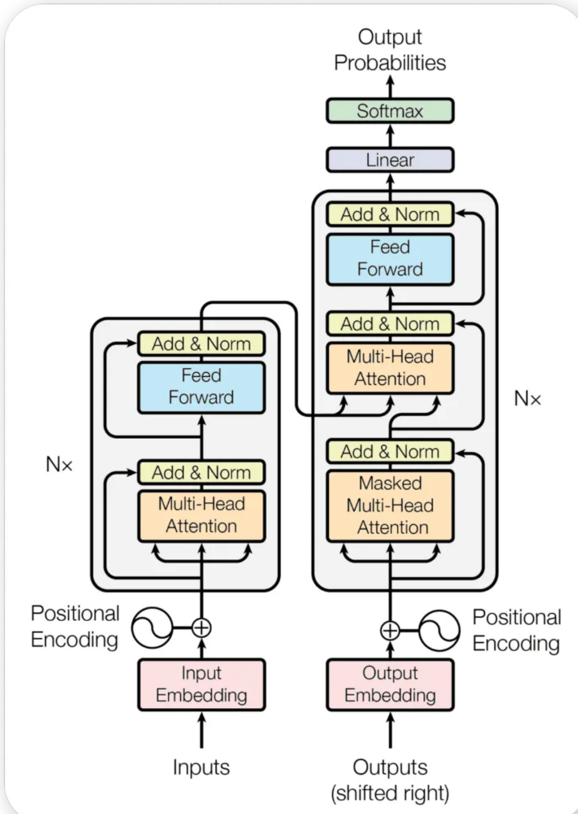

Figure 1.4: Transformer model architecture. [10]

This thesis intends to solve this particular issue by applying Deep Learning techniques to generate meaningful exercises for music teachers and their students. Since providing a technological solution for every type of exercise would not be feasible in the period of this thesis, the project will only focus on providing a solution for Melodic Dictation exercises.

1.5 Stakeholders

The different types of stakeholders involved in this project are listed below.

Primary

Music teachers and **students** are the main stakeholders in the project, as they are whom the outcome of this initiative is aimed at, providing them with tools that improve their workflows, in the case of teachers, or hone their skills, in the case of students.

Secondary

The author of the thesis and **the tutor** are both involved in the development of the project and have a shared interest in completing it successfully.

Tertiary

Music schools and **Conservatoires** could benefit from the outcome of the project but are not directly involved in it. Lastly, the **music community** since it could make use of the research poured into this project.

1.6 Justification

1.6.1 Current Solutions

Throughout my own journey of studying music composition, one of the most recommended websites by music teachers for practising ear training exercises has been **teoria.com**.[2]

The platform hosts different kinds of exercises such as melodic dictation or interval recognition. Looking at the user interface (Fig. 1.5) of one of the aforementioned exercises, one can tell that it allows the user to customise its practice session using a set of high level options.

When selecting different kinds of options a counter is provided to let the user know the number of exercises available that match the selected characteristics. The problem arises when the user selects very few options, leaving it with a small sample of exercises. In other words, the user will always have a **fixed** number of exercises when selecting some desired characteristics.

[2] **teoria.com** is a platform that specialises in providing music theory as well as a diverse range of practical exercises.

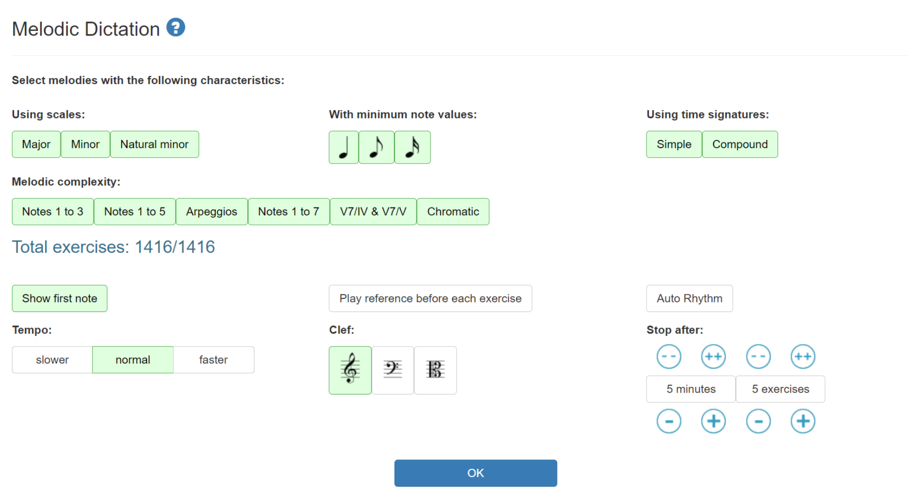

Figure 1.5: Melodic dictation User interface taken from teoria.com. [15]

In fact, the maximum amount of melodic dictation exercises that a user can practice, at least at the time of writing this thesis, are 1416 (See *Total exercises* in figure 1.5), which sounds like a decent amount, but when used regularly in sessions aimed specifically to practice a set of characteristics, it shows its lack of variety.

All in all, **teoria.com** is the better option when it comes to this particular type of exercise. Since it is a free alternative to traditional textbooks full of exercises and alternatives such as see TonedEar whose exercises lack a much needed musicality[3].

1.6.2 Related Studies

In [16] we observed that **GPT-2** can be used to generate **believable music** when working with melodies represented in a text format such as the **ABC Notation**[4].

Additionally it has been demonstrated that **GPT-2** can be fine-tuned to generate music following a **specific structure** dictated by the user by using

[3] **Musicality** in this context meaning the quality of having a pleasant sound or melodiousness.
[4] **ABC Notation** is a system designed to notate music in plain text format.

control codes [17]. This would mean that the model can be steered to create melodies with a particular context such as the generation of melodic dictation exercises.

1.6.3 Solution Justification

As seen previously, the current solutions fail to provide **on-demand** melodic dictation exercises with sufficient variety and musicality when focusing in particular sets of characteristics.

Moreover, in the previous section we saw some studies which proved that **GPT-2** can generate interesting melodies in a controlled manner. This would mean that we could leverage the power of language models to potentially solve our problem.

1.7 Scope

1.7.1 Main Objective

As previously introduced in Section 1.4, the main objective of this thesis is to present a tool capable of generating melodic dictation exercises with an specific structure and sufficient musicality.

1.7.2 Secondary Objectives

To accomplish the main objective, the development of the project has been partitioned into secondary objectives:

- Research transformer based Language Models.
  - Study transformer architecture.
  - Explore different frameworks for manipulating said models.
- Familiarise with **NLP** techniques.
- Train the model on a suitable dataset.
  - Study different music datasets.
  - Fine-tune the model to the dataset
- Deploy the model to a web application.

  - Research web development stacks.
  - Develop user interface.
  - Program API server.

- Empirically evaluate obtained results.

1.7.3 Additional Requirements

Functional Requirements

- The model is capable of generating melodic dictation exercises.
- The website is capable of interacting with the model
- The website provides a music sheet viewer for the generated exercise.

Non Functional Requirements

- The exercises have sufficient musicality.
- The exercises follow the characteristics provided by the user
- The exercises are generated in a timely manner.
- The application is user-friendly.
- The app has a responsive web design.
- The project follows an Agile methodology and good programming practices.

1.7.4 Risks and Obstacles

Throughout the development of the project some problems may arise that have to be taken into account. Potential issues are described as follows:

- **Project deadline.** For each phase of the project there are different deadlines to be met.
- **Inexperience in the domain.** Given that the contents of the project are not studied deeply throughout the degree, the learning curve of the subject has to be considered since more time than the allocated could be needed.

- **Poor planning.** Less time than needed could be assigned to one or more tasks, causing delays in the project.
- **Computational power.** Deep learning models often require a considerable amount of resources that may not be readily available.

1.8 Methodology and Rigour

1.8.1 Work Methodology

For the development of this project I chose to follow an agile methodology since it provides a more flexible environment compared to a waterfall approach.

More specifically the project will use the Scrum framework [18], with *sprints* of one week and with all the roles assigned to myself. At the end of each week the common Scrum events (*Sprint Planning, Review and Retrospective*) will take place. The Daily Scrum will not take place since all the roles are performed by one person.

1.8.2 Monitoring Tools

To monitor the project, we will use **Git** as the main version control software, **Overleaf** to keep track of the report and **GitHub** as the platform for remotely hosting code and as a redundancy measure. Additionally, **GitHub Projects** will be used to keep track of the project following the Scrum Methodology.

Finally, meetings with the tutor will always be scheduled before reaching a milestone, such as passing the project management course, the control meeting stipulated in the project regulations, and before the final presentation. But they will not be regularly scheduled, in an attempt to emulate real-life conditions of the development of a project, aiming to encourage work independence and creativity in finding solutions when facing complex problems.

CHAPTER 2

Project Planning

2.1 Task Definition

The Bachelor thesis of the Computer Science degree (**TFG**) at **UPC** has a work load of 18 **ECTS**, 3 of which belong to the project management course, which according to its syllabus, it is estimated to be 75 working hours.

This would mean that each **ECTS** is worth 25 hours of work, so the overall project should comprise at least 450 working hours, of which 375 hours are allocated to project development and the rest to the project management course.

The course syllabus also proposes to allocate 37.5 hours of its workload to the study of the course material and the synthesis of the final course report, while the remaining 37.5 hours should be dedicated to working on the main tasks of the **TFG**.

Additionally, the project has to meet different deadlines throughout its development. First, the student has to pass the Project Management course, then a follow-up meeting with the tutor has to take place to be allowed to pick a presentation date.

Furthermore, due to personal reasons, the development of this project had to be delayed for a couple of months and therefore the deadlines of a normal term had to be changed accordingly. The deadlines for each phase of the project can be at table 2.1.

Event	Deadline
Project Management Report Delivery	15/03/2023
Progress Review	14/04/2023
Memory Presentation	08/05/23-12/05/2023
Project Defence	15/05/23-19/05/2023

Table 2.1: Project events with their deadlines. [Own Compilation]

In the following sub-sections the tasks will be partitioned according to their nature and assigned an estimation in hours. Moreover, dependencies between tasks will be described as well as human and material resources requirements.

2.1.1 Resources

Material Resources

A wide range of tools will be used throughout the development the project as described down below.

- **Code Editor**. VS Code will be used as the main tool for writing and editing code, given its ease of use and extensibility.
- **Report Editor.** Overleaf will be used since it provides an intuitive interface for writing *LaTex* based documents.
- **Version Control Software.** Git and GitHub will be used to keep track of changes in the project locally and remotely.
- **Project Monitoring Tools.** GitHub Projects will be used to supervise the project and Google Workspace apps will be used to arrange meetings and share progress with the tutor.
- **Development Platform.** Docker will be used to containerise the environment of each aspect of the project.

- **Hardware.** A laptop will be used with the following specifications: 16 GB RAM, Intel Core i7-10870H and NVIDIA GeForce RTX 3060.

Human Resources

During the development of the project different human resources will intervene. Said resources are describe as follows:

- **Project Manager (PM).** In charge of planning the project and ensuring that the objectives are being met.
- **Data Scientist (DS).** Designs and develops the necessary components for an artificial intelligence project. Some responsibilities of a data scientist are gathering data, building pipelines to use said data, and the final model evaluation.
- **Full Stack Developer (FSD).** Is in charge of developing both client and server software for a website.
- **AI Consultant (AIC).** Provides expert knowledge in the domain. This role will be performed by the tutor

In table 2.3, one can observe the different roles assigned to its corresponding task.

2.1.2 Project Management Tasks (T1)

The tasks related to the overall planning of the project are described below:

- **Tutor Meetings (T1.1).** The meetings will only take place to present meaningful progress in the project and occasionally to solve doubts. **(10h)**
- **Contextualisation and Scope (T1.2).** Defines the objectives of the project, its relevance, and the context of the study. **(15h)**
- **Project Planning (T1.3).** Describes the tasks to be solved throughout the project and all the necessary resources for its completion. **(12h)**
- **Economic Management (T1.4).** Analyses the economic cost of undertaking a project of this nature. **(12h)**
- **Sustainability Report (T1.5).** Analyses the sustainability of the project given the resources poured into it. **(12h)**

- **Final Document Synthesis (T1.6).** Verifies the redaction quality of the final report to make sure it gathers all the requirements of a **TFG**. **(40h)**

- **Development Monitoring (T1.7).** Comprises all the events of the Scrum framework necessary to correctly monitor the development of the project. **(35h)**

2.1.3 General Tasks

The objectives described in section 1.7.2 are described down below.

Research (T2)

- **Study transformer Architecture (T2.1).** Familiarise myself with the theory behind the architecture and some popular transformer-based language models. **(60h)**

- **Familiarise with NLP techniques (T2.2).** Study different techniques on how to handle text will help to treat the data in a more efficient way. **(50h)**

- **Analyse Music Datasets (T2.3).** Select a suitable dataset for the project since a proper dataset should avoid doing more work than necessary. **(50h)**

- **Explore Deep Learning Frameworks (T2.4).** Pick an appropriate framework is an important task given that it could greatly reduce the development time since they are aimed to provide different layers of abstraction to solve a variety of problems. **(20h)**

- **Investigate web development stacks (T2.5).** Evaluate the strengths and weaknesses of different libraries when deploying a Deep Learning application. **(20h)**

Development and Experimentation (T3)

- **Create Development Environments (T3.1).** The necessary software for the project has to be defined and installed in order to proceed with the rest of tasks. **(5h)**

- **Exploratory Data Analysis (EDA) (T3.2).** This task is necessary to gain valuable insights on the selected dataset and to pre-process it correctly. **(10h)**
- **Build Experimentation Pipeline (T3.3).** Common Machine Learning Pipelines involve the development of functions with different objectives, such as a data loader, data pre-processor, model trainer, model fine-tuner and a model evaluator. All of these are necessary for creating a model, so it is essential to build a robust pipeline. **(20h)**
- **Train Model (T3.4).** This process consumes a considerable amount of time and utilises the experimentation pipeline. **(25h)**
- **Fine-tune Model (T3.5).** Once trained, the model is evaluated and fine-tuned to generate the best possible results. **(10h)**
- **Design User Interface (T3.6).** A sketch of the interface is necessary to proceed with its implementation. **(10h)**
- **Implement User Interface (T3.7).** The front-end is implemented following a concrete design. **(10h)**
- **Program API Server (T3.8).** The back-end is implemented to allow interaction with the trained model. **(5h)**
- **Deploy web application (T3.9).** The full application is deployed using a suitable platform. **(5h)**

Testing and Evaluation (T4)

- **Evaluate Model (T4.1).** Empirically evaluate the obtained results and draw a conclusion. **(20h)**
- **Test User Interface Design (T4.2).** Assess interface usability. **(20h)**

2.2 Time Estimate

As said previously in Section 2.1, different workloads and deadlines have to be taken into account in order to develop the project successfully.

In order to have balanced workloads every week, the amount of work will be evenly distributed between weeks starting at **02/01/2023** until the memory presentation (See 2.1) which roughly comprises 17 weeks.

The work in hours per week and day is shown in table 2.2, where it is estimated that a typical day will need **4 hours** of work to reach (and surpass) the 450 hours stipulated by the university normative.

It is worth noticing that the rounded estimation is an approximation of the project objective and that the hours allocated for the project management course are there to reflect that some weeks may need approximately **5 hours** to be dedicated in order to do both at the same time.

	Total (h)	**Working Weeks**	**Hours per Week (h)**	**Working Days**	**Hours per Day (h)**
Project Objective	412.5	17	24.3	7	3.5
Project Management Course	37.5	4	9.4	7	1.3
Rounded Estimation	476	17	28	7	4

Table 2.2: Time allocation by weeks and days. [Own Compilation]

2.2.1 Workload Estimation

The estimated workload for each task is provided in table 2.3. Additionally a dependency graph is provided (See Figure 2.1) to assess the concurrency of the tasks efficiently.

It is worth noting that tasks T1.1 and T1.7 are missing from the aforementioned graph given that they can be performed in every step of the development.

Name	Task	Time (h)	Dependencies	Human Resources
Project Management	**T1**	**136**		
Tutor Meetings	T1.1	10	-	PM, AIC
Contextualisation and Scope	T1.2	15	-	PM, DS
Project Planning	T1.3	12	T1.2	PM
Economic Management	T1.4	12	T1.3	PM
Sustainability Report	T1.5	12	T1.4	PM
Final Document Synthesis	T1.6	40	T1.5, T4.1, T4.2	PM
Development Monitoring	T1.7	35	-	PM
Research	**T2**	**200**		
Study transformer Architecture	T2.1	60	T1.2	DS
Familiarise with NLP techniques	T2.2	50	T1.2	DS
Analyse Music Datasets	T2.3	50	T1.2	DS
Explore Deep Learning Frameworks	T2.4	20	T2.1, T2.2	DS
Investigate web development stacks	T2.5	20	-	FSD
Development and Experimentation	**T3**	**100**		
Create Development Environments	T3.1	5	T2.4, T2.5	DS
Exploratory Data Analysis (EDA)	T3.2	10	T2.3, T3.1	DS
Build Experimentation Pipeline	T3.3	20	T3.2	DS
Train Model	T3.4	25	T3.3	DS
Fine-tune Model	T3.5	10	T3.4	DS
Design User Interface	T3.6	10	-	FSD
Implement User Interface	T3.7	10	T3.6, T3.1	FSD
Program API Server	T3.8	5	T3.1	FSD
Deploy web application	T3.9	5	T3.8, T3.7	FSD
Testing and Evaluation	**T4**	**40**		
Evaluate Model	T4.1	20	T3.4, T3.5	DS, AIC
Test User Interface Design	T4.2	20	T3.9	FSD

Table 2.3: Task dependencies and resource allocation. [Own compilation]

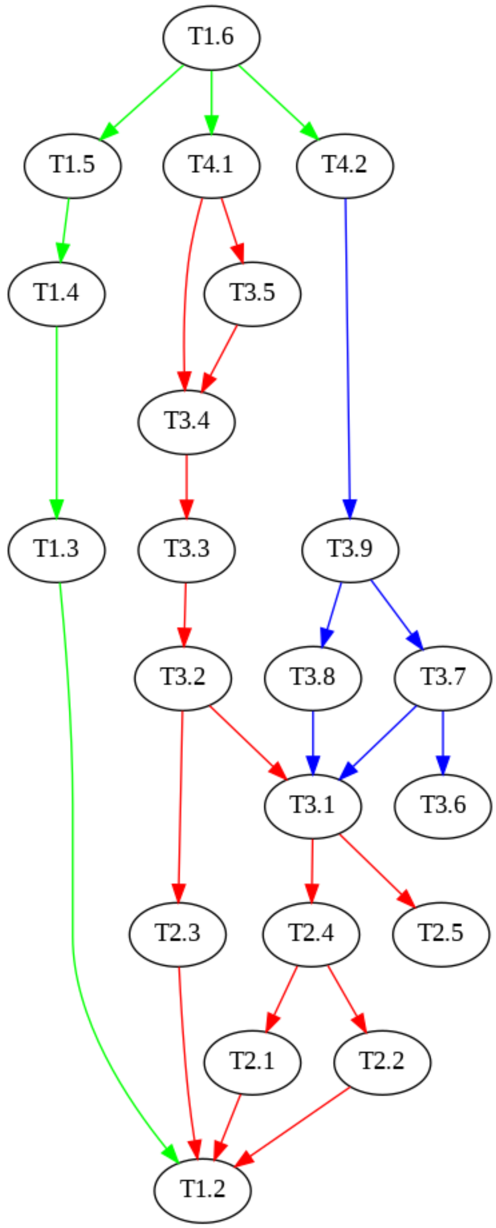

Figure 2.1: Task dependency graph from table 2.3 with resource colour coded (**PM**,green;**DS**,red and **FSD**,blue). [Own Compilation]

2.2.2 Gantt Diagram

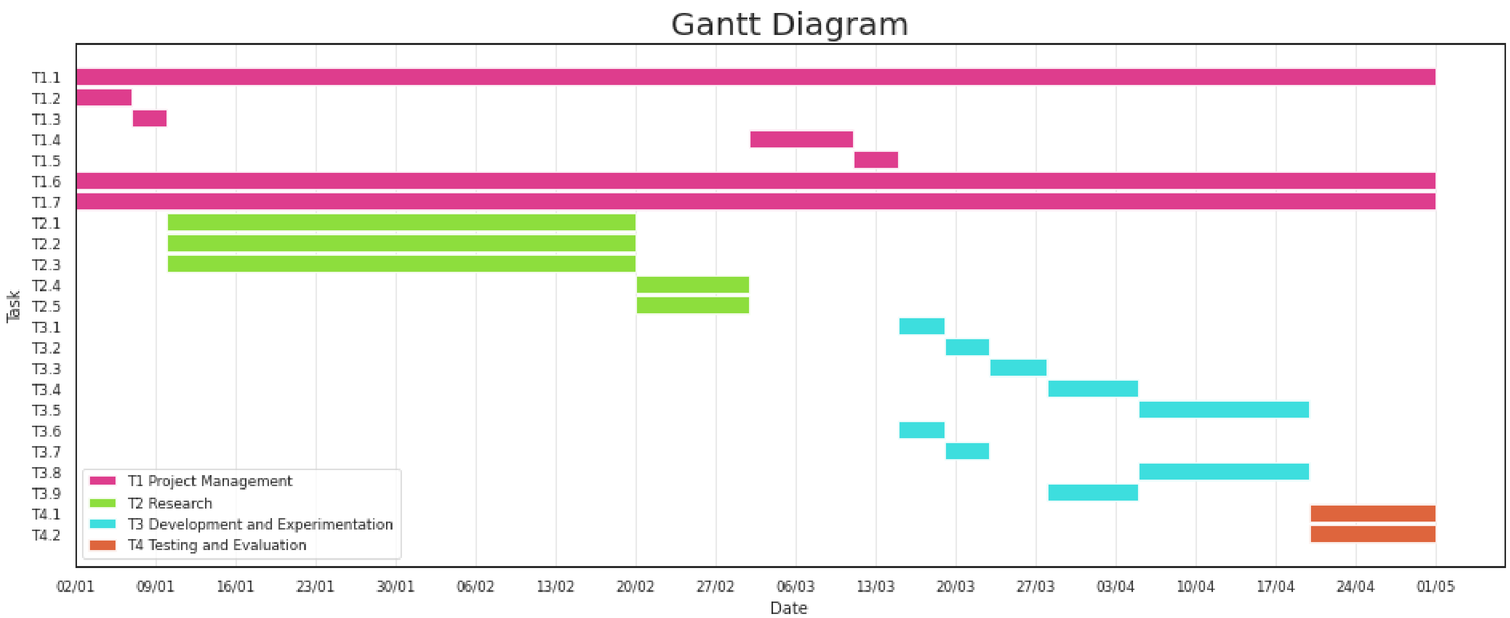

Figure 2.2: Estimated Gantt Diagram [Own Compilation]

2.3 Risk Management

As previously discussed in Section 1.7.4, one has to contemplate possible ways to solve such problems. An assessment of the impact of each problem is provided below, accompanied by a recommended solution and a time estimation of resolution based on a risk percentage ranging from 10 to 20 percent of the total time of the task according to its risk.

- **Project deadline [Low Impact].** If a deadline is approaching, it is possible to increase the number of hours per day for any task, as all planning has been calculated with this option in mind. Furthermore, this issue is constantly evaluated at the end of each sprint to re-evaluate the workloads. (**Task 1.7** estimated delay **3.5-7h**)
- **Inexperience in the domain [High Impact].** More hours can be allocated to research if necessary, but this would mean that some non-functional requirements would not be developed or even that the quality of the project itself would be lowered, resulting in a reduction of the project scope. (**Task 2.1** estimated delay **6-12h**)
- **Poor planning [Low Impact].** By following an Agile Methodology this problem is greatly mitigated as the workload of each task is reviewed in a timely manner to avoid delays. (**Task 1.7** estimated delay **3.5-7h**)
- **Computational power [Medium Impact].** If a deadline is approaching, one can evaluate the possibility of moving the local training to a paid cloud-based environment to accelerate the model training. But in addition to this being expensive in some cases, we would also have to adapt the code to run on the remote instance. (**Task 3.4** estimated delay **2.5-5h**)

CHAPTER 3

Budget

3.1 Cost Identification

This section analyses the budget necessary to carry out the project in its entirety.

3.1.1 Staff Costs

In section 2.1.1, we have already identified the different roles in the project and also assigned them their corresponding tasks (See 2.3).

The different salaries for the roles are shown at the table 3.1. The annual salary and hourly fees are described with and without taking into account the social security percentage. It is worth noting that the hourly rate was calculated considering 40 hours of work each week and 52 weeks in a year. The full year has been taken into account as it includes: working days, government holidays and the annual paid holiday period [19].

Role	Annual Salary (€)	Annual Salary with Social Security (€)	Hourly Rate (€/h)
Project Manager (PM)	48452	65410.20	31.45
Data Scientist (DS)	35666	48149.10	23.15
Full Stack Developer (FSD)	22481	30349.35	14.59
AI Consultant (AIC)	53117	71707.95	34.47

Table 3.1: Salaries by role with 35% social security percentage.[20]

In table 3.2 we can observe the cost per activity (**CPA**) based on the hourly fees of each role while taking into account the social security payments.

Name	Task	Time (h)	PM (h)	DS (h)	FSD (h)	AIC (h)	Cost Per Task (€)
Project Management	**T1**	**136**					
Tutor Meetings	T1.1	10	10	0	0	10	659.22
Contextualisation and Scope	T1.2	15	15	15	0	0	818.94
Project Planning	T1.3	12	12	0	0	0	377.37
Economic Management	T1.4	12	12	0	0	0	377.37
Sustainability Report	T1.5	12	12	0	0	0	377.37
Final Document Synthesis	T1.6	40	40	0	0	0	1257.89
Development Monitoring	T1.7	35	35	0	0	0	1100.65
Research	**T2**	**200**					
Study transformer Architecture	T2.1	60	0	60	0	0	1388.92
Familiarise with NLP techniques	T2.2	50	0	50	0	0	1157.43
Analyse Music Datasets	T2.3	50	0	50	0	0	1157.43
Explore Deep Learning Frameworks	T2.4	20	0	20	0	0	462.97
Investigate web development stacks	T2.5	20	0	0	20	0	291.82
Development and Experimentation	**T3**	**100**					
Create Development Environments	T3.1	5	0	5	0	0	115.74
Exploratory Data Analysis (EDA)	T3.2	10	0	10	0	0	231.49
Build Experimentation Pipeline	T3.3	20	0	20	0	0	462.97
Train Model	T3.4	25	0	25	0	0	578.72
Fine-tune Model	T3.5	10	0	10	0	0	231.49
Design User Interface	T3.6	10	0	0	10	0	145.91
Implement User Interface	T3.7	10	0	0	10	0	145.91
Program API Server	T3.8	5	0	0	5	0	72.96
Deploy web application	T3.9	5	0	0	5	0	72.96
Testing and Evaluation	**T4**	**40**					
Evaluate Model	T4.1	20	0	20	0	20	1152.47
Test User Interface Design	T4.2	20	0	0	20	0	291.82
		Hours per Role (h)	136	285	70	30	Total Cost (€)
		Cost per Role (€)	4276.82	6597.35	1021.37	1034.25	**12929.80**

Table 3.2: Human cost per activity (CPA). [Own Compilation]

3.1.2 General Costs

In order to reflect the costs of working remotely, internet and electricity costs are calculated as follows:

- **Internet Cost.** Internet with fiber optic connection which will be used exclusively for the project has a monthly invoice that costs around 55€. Therefore, 5 months worth of internet will cost 275€
- **Electricity Cost.** The power consumption of a laptop is around 0.2 kWh. Electricity cost in Spain is around 0.228 €/h, given that the laptop will be used for 476 hours, the estimated electricity cost will be 21.71 €.
- **Rent Cost.** Given that the project is being developed in a remote manner, we will include in our budget the mean rent in Catalonia in 2023 which is around 800€[21].

Additionally, the laptop depreciates by the passing of time and by its use so we have to take into account its amortisation. Table 3.3 shows the laptop amortisation.

Hardware	Initial Cost (€)	Life Expectancy (Years)	Yearly Usage (h)	Hours Used (h)	Amortisation (€)
Laptop	1200	5	2080	476	**54.92**

Table 3.3: Laptop amortization calculation. [Own Compilation]

One can see the General Costs (GC) calculated in table 3.4 for hardware and cost of electricity, software is not taken into account since all the software described in section 2.1.1 is free to use or at least, a free tier is available.

Type	Cost (€)
Hardware	
Laptop amortization	54.92
Space	
Internet	275.00
Electricity	21.71
Rent	800
Total GC	**1096.71**

Table 3.4: General Costs Summary. [Own Compilation]

3.1.3 Contingencies and Incidentals

In order to lessen the impact of unexpected events, a 10% contingency will be applied to the CPA, as it is common practice to do so in project of this nature.

Additionally, incidentals like those seen in section 1.7.4 can occur. For this, the risk of this events has to be calculated and also has to be provided with part of the budget. These calculations can be seen in table 3.5.

	Estimated Cost (€)	Risk (%)	Cost (€)
Project Deadline	1100.65	10	110.07
Inexperience in the Domain	1388.92	20	277.78
Poor Planning	1100.65	10	110.07
Computational Power	578.72	15	86.81
		Total Cost	**584.72**

Table 3.5: Incidental Cost Calculation. [Own Compilation]

3.2 Cost Estimates

An estimation for the entirety of the project is shown in table 3.6 as a summary of all the previous tables.

Activity	**Cost (€)**
Total GC	1096.71
Total CPA	12929.80
CPA Contingency	1939.47
Total Incidentals	584.72
Total Budget	**16550.70**

Table 3.6: Project Total Budget. [Own Compilation]

3.3 Management Control

In order to control possible budget deviations, a set of formulas is presented with the aim of providing a way to detect possible imprecisions in our estimations.

- **CPA Deviation (CPAD).** $CPAD = (estimated\_cost\_per\_hour - real\_cost\_per\_hour) * total\_hours\_consumed$
- **CG Deviation (CGD).** $CGD = ED + AD$
- **Electricity Deviation (ED).** $ED = (estimated\_usage\_per\_hour - real\_usage\_per\_hour) * price\_per\_hour$
- **Amortisation Deviation (AD).** $AD = (estimated\_hours\_used - real\_hours\_used) * price\_per\_hour$
- **Contingency and Incidental Deviation (CID).** $CID = (estimated\_incidental\_hours - real\_incidental\_hours) * total\_incidental\_hours$
- **Cost Deviation (CD).** $CD = estimated\_cost - (CPAD + CGD + CID)$

CHAPTER 4

Sustainability Analysis

In the following sections, the questions regarding the different dimensions of sustainability applied to the current project are answered and ending this analysis with a self-evaluation concerning the subject.

4.1 Environmental Dimension

Have you estimated the environmental impact of undertaking the project? Have you considered how to minimise the impact, for example by reusing resources?

Given that this project only needs a laptop for its development and taking into account the electricity consumption denoted in section 3.1.2, one could estimate that throughout the project development the computer will consume 95.2 **kWh** which according to [22] would yield approximately 0.041 Metric Tons of Carbon Dioxide (CO_2), which roughly equals to 17.41 litres of gasoline. It is worth mentioning that the aforementioned estimations can greatly deviate from reality since electricity costs may vary due to a wide range of circumstances (time of the day, time of the year, weather, war, etc.).

Using transfer learning has been contemplated as a way of reusing resources in the context of the project since it would avoid to train a model from the ground up and potentially spending more resources along the way.

How is the problem that you wish to address resolved currently (state of the art)? In what ways will your solution environmentally improve existing solutions?

As previously seen in section 1.6.2 there are some related studies that reused **GPT-2** for their own tasks, so one way of reusing resources in this context would clearly be to apply **transfer learning** where possible.

My solution proposes to use these pre-trained language models to offer a better alternative to current solutions, while avoiding training a model from scratch, thus saving resources.

4.2 Economic Dimension

Have you estimated the cost of undertaking the project (human and material resources)

Yes, in section 2.1.1 the necessary resources are defined and chapter 3 is devoted to estimate different kinds of costs for the project.

How is the problem that you wish to address resolved currently (state of the art)? In what ways will your solution economically improve existing solutions?

My solution would offer a free alternative to paid content, such as books full of exercises. It would also save time within the classroom, as the teacher would not have to spend time producing new kinds of exercises.

Consequently, avoiding wasting time on a task that can be automated would allow that time to be better spent doing more fruitful things and therefore making the most of the limited time the student has with the teacher, thus saving money as well from the perspective of the student since studying music can be quite expensive.

4.3 Social Dimension

What do you think undertaking the project has contributed to you personally

As a music student myself and failing in exactly the area that this tool is intended to help, I believe that when it is finished it will not only help me, but other students as well. Furthermore, finding this intersection between music and technology has made me appreciate both fields even more.

How is the problem that you wish to address resolved currently (state of the art)? In what ways will your solution socially improve (quality of life) existing? Is there a real need for the project?

I think the project could potentially help other music students besides myself, as it is an issue I struggle with, I think at least someone else may have the same problem as me so yes, I think there is a real need for the project.

4.4 Self-assessment

Having answered the survey provided by the project management course that aims to assess the knowledge of Sustainable Development Goals (**SDGs**) and sustainability skills across environmental, economic, and social dimensions, I have come to realise how limited knowledge I possess when discussing subjects concerning the environment.

Even though throughout the Computer Science Degree at the Barcelona School of Informatics (**FIB**), there are courses like Computer Architecture (**AC**), Social and Environmental Issues Of Information Technologies (**ASMI**) and Business and Economic Environment (**EEE**) where the student is expected to develop competences in assessing the impact across the different dimensions mentioned earlier.

The first (**AC**) is a mandatory course in the degree where a small portion of the syllabus is dedicated to assessing the impact on power consumption while being aware of the underlying computer architecture (memory hierarchy, processor design, instruction set architecture design, etc.). The second (**ASMI**) is an elective course, which I have not taken, but according to its syllabus is designed to provide knowledge on the Social and Environmental aspects of information technologies. The third (**EEE**) is a mandatory course whose main objective is to work on the social and economic dimensions of sustainability by studying the socioeconomic environment of companies from an economics and business perspective.

On the one hand, the university is aware of the importance of a subject such as sustainability, given that it provides courses aimed at that. On the other hand, said courses have little content regarding environmental importance or are optional for the student to take. Consequently, this would mean that when finalising the degree, the student may or may not have sufficient skills to write a report on a complex matter such as sustainability in an environmental context.

In my personal opinion, I think it is high time for the university to broaden the scope of the different subjects within the degree in order to fit in environmental topics or make it mandatory to take courses like **ASMI** to raise awareness among the students on such matters and to provide them with necessary conditions for them to develop skills regarding the topic.

Lastly, I think having a mandatory course on the subject, such as **EEE**, would make a fine addition to the curriculum of the informatics degree since critical times like the ones we are living in demand new professionals to be aware of the possible environmental ramifications of executing projects inside their field of expertise.

CHAPTER 5

Further Examining the Problem

As previously stated in sections 1.4 and 1.7.1, our main task is to develop a tool to generate melodic dictation exercises with minimal input from the user. Some specific characteristics that may be inputted by the user in order to control the exercise generation are formally described as follows:

- **Key:** A specific key in which the melody is presented (A,B,C...).
- **Mode:** The specific mode in which the melody is presented.
- **Meter:** or time signature to provide a sense of rhythm and manage the different rhythmic values that notes can have (♩, ♪, ♬, etc.).
- **Structure:** Such as the length of the exercise, chord progression, both, or even none.

This set of requirements poses several challenges when picking a suitable dataset, the way it is represented and the strategies used to feed the model with the dataset.

5.1 Applicable Laws and Regulations

Given the nature of the project, we must abide by the current copyright regulations in Spain, and the European Union meaning that the dataset should be open source in nature.

Although, there has been debates around training AI models with copyrighted data as well as the ethical implications [23], there is still a thin line between copyright infringement and the originality of AI-generated music, since the nature of basically every AI model is to identify and replicate the patterns inside the data it was trained on.

Therefore, in order to mitigate the risk of legal action, it would be advisable to steer away from such datasets.

5.2 Picking a Dataset

In this section we will analyse the benefits and drawbacks of popular datasets in the literature.

In this instance we will only focus in symbolic music datasets given that our goal is to render the exercise in music sheet format, starting with a symbolic dataset is a more suitable choice. Additionally, this will facilitate the acquisition of noteworthy characteristics and its posterior analysis.

Dataset	**Format**	**Hours**	**Songs**	**Genre**	**Melody**	**Chords**	**Multitrack**
Lakh MIDI Dataset	MIDI	>5000	174,533	misc	*	*	*
MAESTRO Dataset	MIDI	201.21	1,282	classical			
Wikifonia Lead Sheet Dataset	MusicXML	198.40	6,405	misc	O	O	
Essen Folk Song Dataset	ABC	56.62	9,034	folk	O	O	
NES Music Database	MIDI	46.11	5,278	game	O		O
MusicNet Dataset	MIDI	30.36	323	classical			*
Hymnal Tune Dataset	MIDI	18.74	1,756	hymn	O		
Hymnal Dataset	MIDI	17.50	1,723	hymn			
music21's Corpus	misc	16.86	613	misc	*		*
EMOPIA Dataset	MIDI	10.98	387	pop			
Nottingham Database	ABC	10.54	1,036	folk	O	O	
music21's JSBach Corpus	MusicXML	3.46	410	classical			O
JSBach Chorale Dataset	MIDI	3.21	382	classical			O
Haydn Op.20 Dataset	Humdrum	1.26	24	classical		O	

Table 5.1: Symbolic Music Datasets supported by MusPy.[24]

At first, the JSBach Chorale Dataset was considered as a good starting point, given its extensive use in various projects [25]. Furthermore, from a didactic point of view, works by Bach are studied from the very first years of classical musical training because of their rich harmony and exceptional use of counterpoint.

The main issue with utilising this dataset to address our task was the necessity to conduct a comprehensive harmonic analysis on the entire dataset which would take a considerable amount of time and effort. Although some techniques can be applied to automatically label the dataset[26], we have no way of proving the correctness of the technique, so the lack of ground truth leads us to discard the dataset.

In table 5.1, we can observe several characteristics of symbolic datasets and if we take a look to the Melody and Chords columns we can instantly see three datasets that would fit our needs.

- **Wikifonia Lead Sheet Dataset**: This dataset offers a quite large amount of songs, but the genres are not specified which would difficult the cleaning of the dataset. Additionally, for interacting with the dataset, third party software would be needed to visualise and transform the files if necessary.
- **Essen Folk Song Dataset and Nottigham Database**: Both are datasets with defined genres and between the two provide an abundant collection of instances. Furthermore both datasets are in ABC notation which is a plain text representation, this would greatly ease the manipulation of the datasets (See Fig. 5.1).

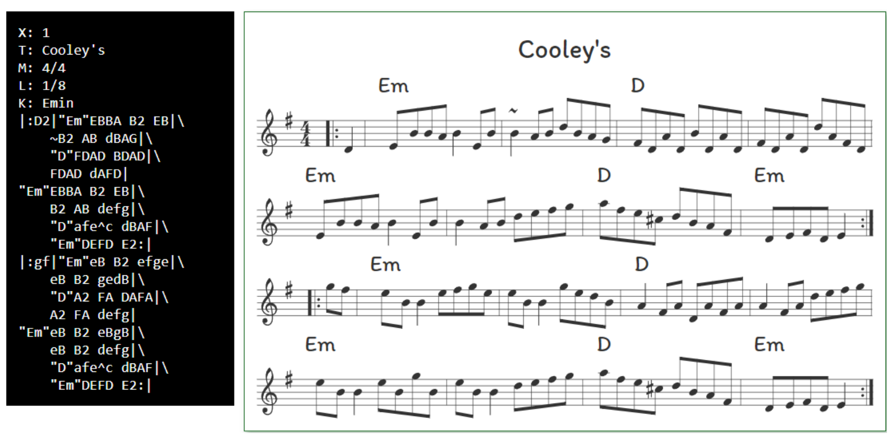

Figure 5.1: ABC notation lead sheet (left) and its music sheet (right). [27]

After skimming through the Wikifonia Lead Sheet Dataset (obtained by MusPy), some songs were found that may be subjected to copyright such as: 'He's a Pirate' by Klaus Badelt, 'Bob-Omb Battlefield' by Koji Kondo, 'Love' by Yiruma, and many more. So, due to the difficulty of verifying copyright compliance for each file this dataset is being discarded.

Regarding the Essen Folk Song Dataset, this dataset will be discarded in favour of the Nottigham Database given that the latter will require less computational resources to be treated due to its compactness.

CHAPTER 6

Exploratory Data Analysis

6.1 Song Format

First, the dataset was downloaded using the MusPy library [24] in order to be pre-processed. The data is downloaded as **TuneBooks** where each TuneBook is a file with a set of consecutive songs separated by two newline characters.

The structure of a common song is divided into a header and a body where the header contains metadata of the song and the body contains the song itself. Some of the header metadata is described as follows:

- '**X**': This is a unique number assigned to the tune for reference purposes.
- '**T**': This is the title of the tune.
- 'S': This indicates where the tune was collected or transcribed from.
- '**M**': This specifies the time signature of the tune, indicating how many beats are in each measure and which note value receives one beat.
- '**L**': This specifies the default duration of a note in the tune.
- '**R**': This indicates the rhythm or style of the tune, such as a jig or reel.

- '**P**': This specifies the structure of the tune by indicating which parts are repeated and in what order.
- '**K**': This specifies the key signature of the tune, indicating which notes should be played sharp or flat.
- '**F**': This is the name of the file containing the ABC notation for the tune.
- '**N**': This field can contain any additional notes or comments about the tune.

Additionally, according to the ABC standard[28] '**X**' should always be at the beginning of the header and '**K**' at the end of the header, with the rest of the fields between them and without a specific order.

6.2 Pre-processing and Visualisation

6.2.1 Visualising the original Dataset

In order visualise the data, each tunebook was parsed and concatenated into a single dataframe, where each field of the header represents a single feature and the body is also counted as a feature. Yielding around 1000 songs as seen in figure 6.1.

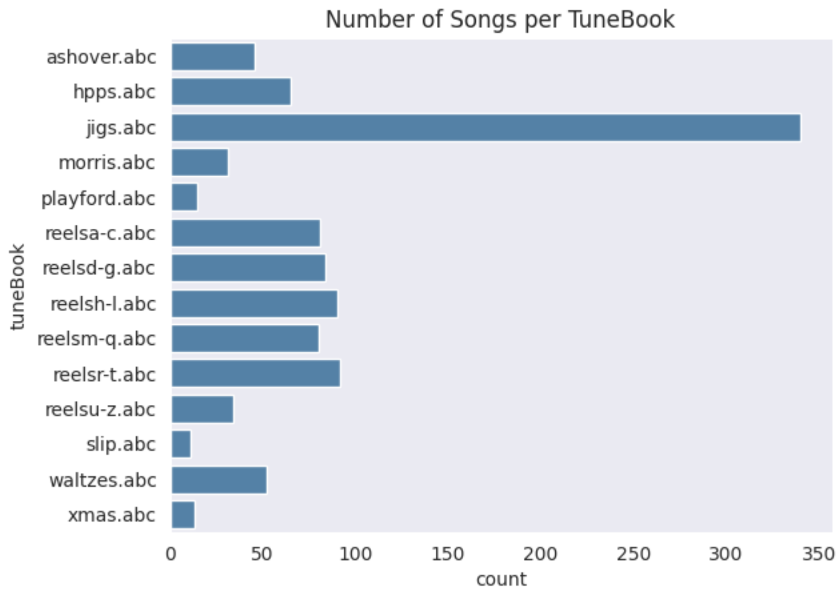

Figure 6.1: Tune Count per TuneBook. [Own Compilation]

6.2.2 Song Header Visualisation

There was a lot of noise in the metadata, such as the field '**R**' which was mostly empty, some comments were parsed as meaningful data, the unit length ('**L**') was missing in almost half of the dataset given that the ABC standard has default values that are calculated according to the meter ('**M**') value.

In figure 6.2, we can see the key counts of the thousand songs present in the dataset. As expected, the number of Major keys present in the data are more frequent than the Minor ones. Additionally, we can see that not every key is present in the figure and that the data is heavily unbalanced.

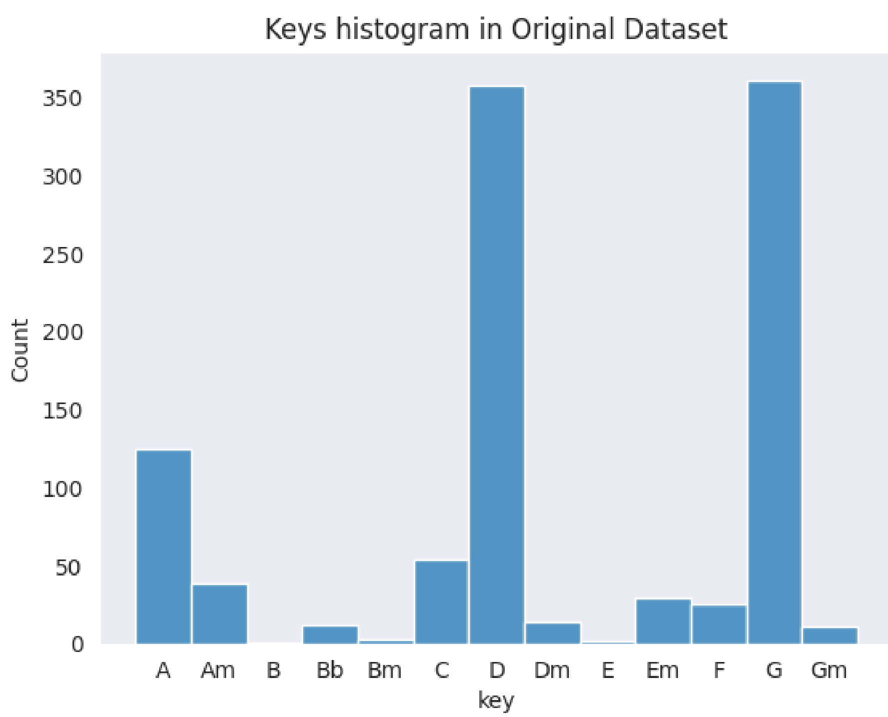

Figure 6.2: Key Count in original dataset. [Own Compilation]

As an interesting fact we can notice that C Major is not the most common key in the dataset as it is in general [29], one could draw the conclusion that Irish folk songs vastly prefer D and G major keys over other and may be a core characteristic of the genre.

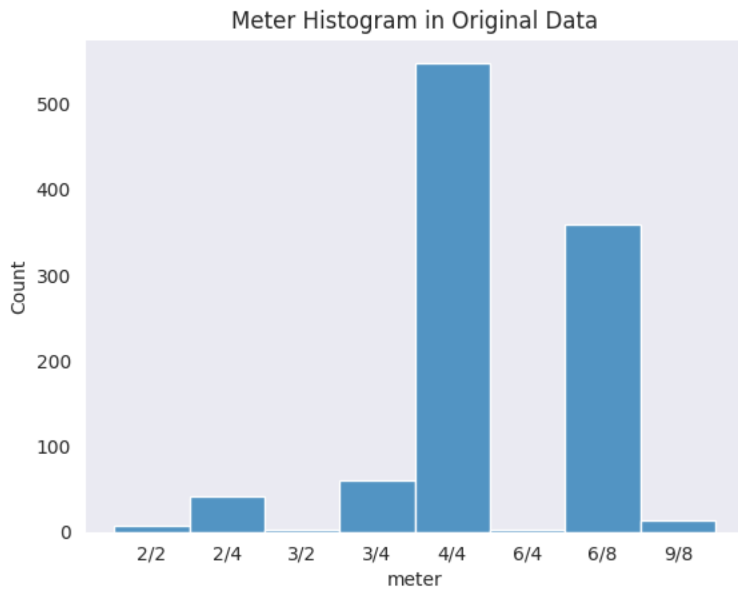

Figure 6.3: Meter Count in original dataset. [Own Compilation]

Looking at the meter values in the original data (See Fig. 6.3) we can see that the most common time signatures are 4/4 and 6/8 which in general are rather typical in western music. Moreover, the rest of the keys have less than a hundred occurrences causing an uneven distribution in the feature.

Observing the different note lengths (Fig. 6.4) present in the dataset we find two values that indicate mostly crotchets ♩(1/4) and quavers ♪(1/8), but also one can notice that more than 300 values are missing. This is due to the ABC standard that infers some note resolutions according to the meter.

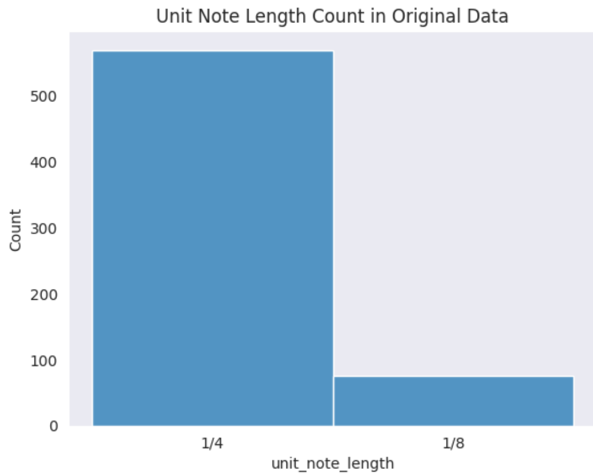

Figure 6.4: Unit Note Length Count in original dataset. [Own Compilation]

6.2.3 Song Body Visualisation

The only feature extracted from the body was the chord progression since it would give us a good grasp of what chords are inside each piece. One can see in Figure 6.5 that there a lot of chords whose occurrence in the dataset is not very high.

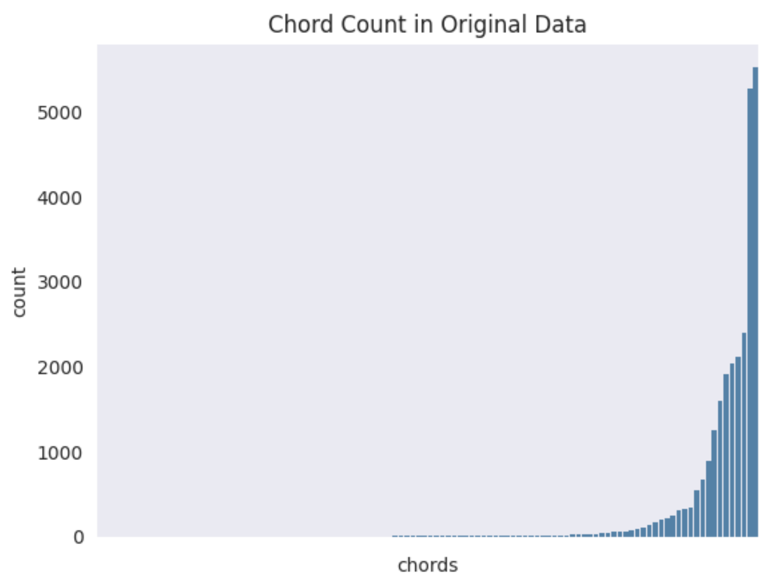

Figure 6.5: Chord Count in original dataset. [Own Compilation]

Additionally one can tell that there are some chords that have a higher count than the rest, this is due to the fact that most of the songs are in two keys separated by a fifth, therefore the two keys share the at least one of the I, IV or V degree chords of their keys which generally have a high occurrence within a piece with tonal western harmony.

6.2.4 Cleaning the Dataset

While exploring the dataset, different characteristics and anomalies where encountered and are summarised as follows:

- Repetition markings were found.
- Header fields were found in the body of the song.
- Chords with special format were present in the data.
- Chords with low occurrence were found.
- Nonexistent chord progression or to small length.
- Songs with more than one voice.
- Unit note length empty in a large amount of data.
- Additional strange symbols were found.

Details on how the aforementioned irregularities were treated are described in their corresponding subsection accompanied by a comment on the possible repercussions of their treatment.

Repetition Markings

- **Proposed Solution:** According to the ABC standard the symbols "`|: :| |1 |2`" represent various types of repetition within the piece. Said tokens were simply erased from the songs since they are of no use for our task given that we want to generate small form exercises with no repetition.
- **Possible Repercussions:** Given the possibility of having multiple endings by erasing the repetitions the number of measures may not align with

the time signature leading to a poor performance when finding a pattern in the structure. Moreover, this could also affect the quality of the melody given that some harmony concepts are tied to the structure such as finishing a phrase in a V degree chord and finishing the song in the I degree. This is just mainly to give a sense of tension and release to the listener.

Body with Header Fields

- **Proposed Solution:** Erase every encountered field and concatenate the bodies.
- **Possible Repercussions:** Headers fields indicating the time signature and and parts were found. Erasing the first would cause a wrong prediction of the resolution of the notes, while erasing the second could introduce noise to the chord progression by concatenating parts that are not consecutive.

Special Formatted Chords

- **Proposed Solution:** Given their low occurrence in the dataset extended chords (augmented; diminished; added 7th, 8th, etc) or chords with bass indication (”Gmaj/d”) can be dropped.
- **Possible Repercussions:** None.

Chord Low Occurrence

- **Proposed Solution:** Drop songs which have chords that have a lower count than a threshold.
- **Possible Repercussions:** If the threshold is to high it may cause to drop a quite large amount of songs which would reduce the variance of the dataset and therefore reducing the predicting capabilities of the model.

Chord Progression Length

- **Proposed Solution:** Drop songs that have chord progressions that are too small to be relevant. Empty chord progressions would mean that the song has no chord labels and therefore is safe to erase them.
- **Possible Repercussions:** None, since songs that have these characteristics have a low occurrence.

Multiple Voices

- **Proposed Solution:** Erase the songs that have more than one voice (multiple pitches at the same time are played).
- **Possible Repercussions:** None, since most of the songs have only one voice.

Unit Note Length Missing

- **Proposed Solution:** Input the empty values according to the ABC Standard since it can be calculated from the Meter field.
- **Possible Repercussions:** None.

Additional Symbols in Body

- **Proposed Solution:** Symbols that indicate dynamics such as `"!trill! !lowermordent! !uppermordent!"` can be safely erased since we are focusing on the structure and harmony of the song rather the way it is played.
- **Possible Repercussions:** None

6.2.5 Visualising Clean vs Original Dataset

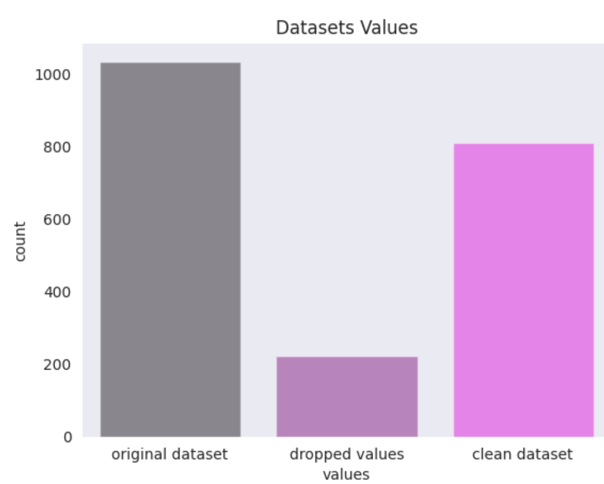

Figure 6.6: Dropped values vs Clean and Original Data. [Own Compilation]

In figure 6.6 we can observe that around 200 values were dropped after applying the aforementioned cleaning procedures, leaving a clean dataset with 800 values approximately

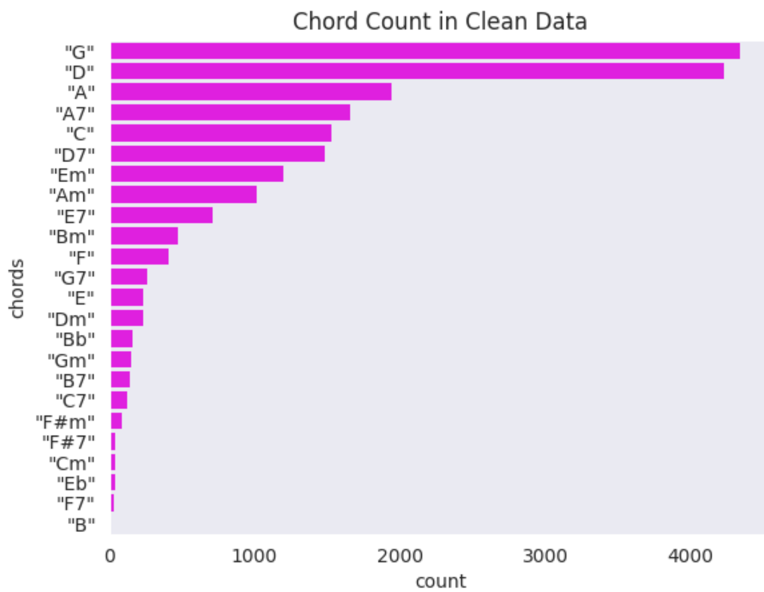

Figure 6.7: Chord Count in Clean Dataset. [Own Compilation]

Looking at the Chord count of the clean dataset (Fig. 6.7) and contrasting it with the one of the original data (Fig. 6.5) one can see that a quite large amount of chords were dropped from the dataset, and now we can clearly see that the most frequent chords match exactly the pitch of the most frequent keys.

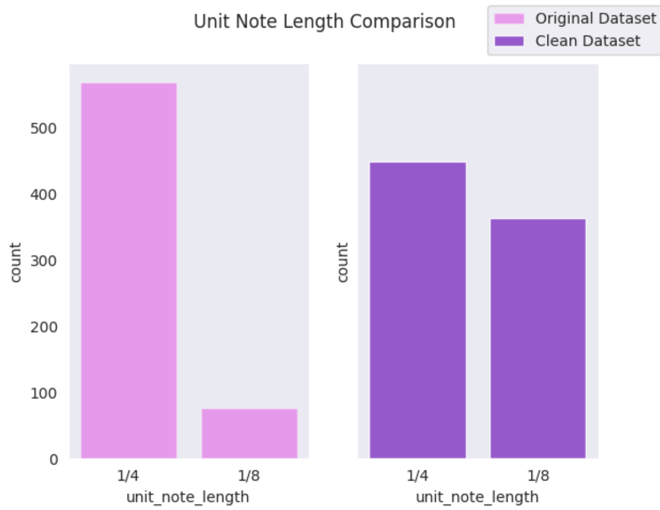

Figure 6.8: Note Length Count Clean vs Original Data. [Own Compilation]

Contrasting the note lengths from both datasets (Fig, 6.8) we now have a more balanced feature after inputting the necessary values in the empty header fields of the original dataset.

Looking at figure 6.9, one can tell that it has the same distribution after discarding 200 songs but low occurrences were not dropped in this case due to the fact that the dataset was dwindling rapidly and if we put in place more measures dropping measures we could be left with almost half of the data

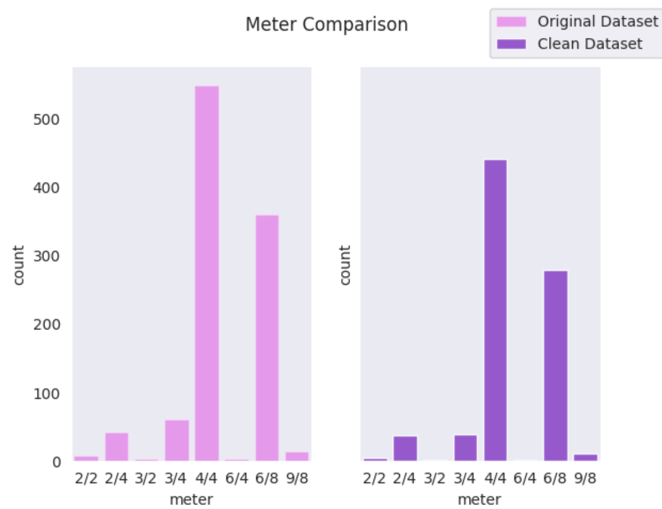

Figure 6.9: Meter Count Clean vs Original Data. [Own Compilation]

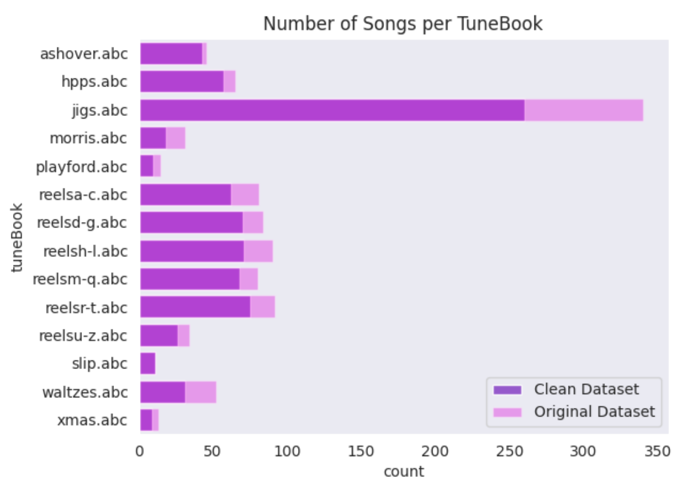

Figure 6.10: Tunes Per Book Clean vs Original Data. [Own Compilation]

Seeing the comparison of the distribution of the songs across the different tuneBooks (Fig. 6.10) we can see that the songs were dropped evenly across the tuneBooks which is desirable since the songs are grouped by style this would mean that no style will predominate in the training.

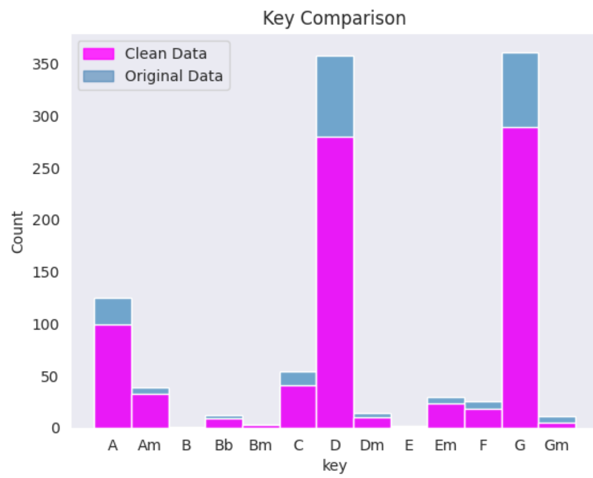

Figure 6.11: Key Comparison Clean vs Original Data. [Own Compilation]

After comparing the Keys in both datasets in figure 6.11 we can see that songs that did not met the requirements were proportionally distributed across the different pitches. Additionally we can see that the samples of each category is not balanced at all but we will solve this issue in the next section.

6.2.6 Data Augmentation

Given that we are dealing with a music dataset a common data augmentation technique is to transpose each song to every other key by semitones which would yield 12 songs for each tune.

In order perform data augmentation the **abcjs**[27] library was used to transpose each of the songs by moving six semitones upwards and 5 semitones backwards in order to avoid too many ledger lines in the music sheets of the predicted exercises.

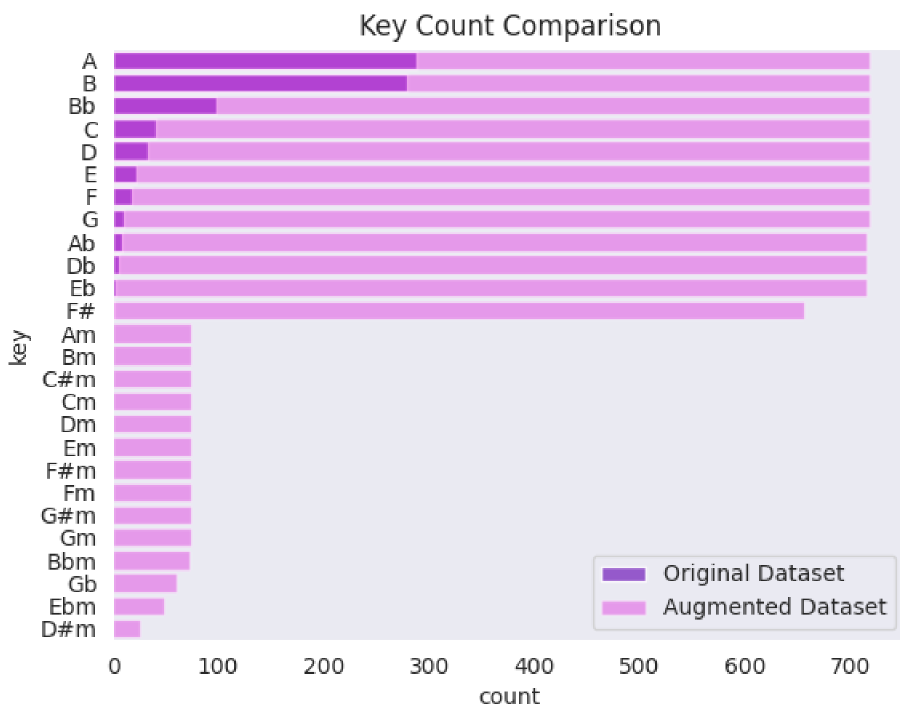

Figure 6.12: Key Comparison Clean vs Augmented Data. [Own Compilation]

As we can see in figure 6.12, now the keys are evenly distributed in most of the categories. It is worth noticing that there is not the same number of songs for each Major and Minor modes since the library may have injected some artefacts in the songs and the preprocessing step discarded them.

CHAPTER 7

Appendix

7.1 Music Notation

Following there is a description of some terms used in music notation:

- Staff: The staff is a set of five horizontal lines and four spaces on which musical notes are written. Each line and space represents a different pitch.
- Clef: The clef is a symbol placed at the beginning of the staff to indicate the pitch of the notes written on it. The two most common clefs are the treble clef and the bass clef.
- Note: A note (♩) is a symbol used to represent the duration and pitch of a sound. The shape of the notehead (the round or oval part of the note) and the presence or absence of a stem (the vertical line attached to the notehead) indicate the duration of the note . The position of the note on the staff indicates its pitch.
- Rest: A rest is a symbol used to represent a period of silence in the music. Like notes, rests have different shapes to indicate their duration.
- Time Signature: The time signature is a symbol placed at the beginning of a piece of music, after the clef, to indicate how many beats are in each

measure and which note value receives one beat.

- Key Signature: The key signature is a set of sharps or flats placed at the beginning of a piece of music, after the time signature, to indicate which pitches should be played sharp or flat throughout the piece.
- Bar Line: A bar line is a vertical line drawn across the staff to divide it into measures. Each measure contains a specific number of beats, as indicated by the time signature.
- Repeat Sign: A repeat sign is a pair of dots placed on either side of a bar line to indicate that the music within the repeat should be played again.
- Accidental: An accidental is a symbol placed before a note to alter its pitch. The most common accidentals are the sharp ($\sharp$), which raises the pitch by a half step, and the flat ($\flat$), which lowers the pitch by a half step.
- Beam: A beam is a horizontal line used to connect multiple eighth notes (quavers) or shorter notes, indicating that they should be performed as a rhythmic group.
- Breath Mark: A breath mark is a symbol that looks like an apostrophe, placed above the staff to indicate where the performer should take a breath.
- Chord: A chord is a group of two or more notes played simultaneously.
- Dynamics: Dynamics are symbols used to indicate the volume of the music. The most common dynamic symbols are p (piano, meaning soft), f (forte, meaning loud), and m (mezzo, meaning medium).
- Fermata: A fermata is a symbol placed above a note or rest to indicate that it should be held for longer than its written duration.
- Grace Note: A grace note is a small note written before a main note, indicating that it should be played quickly and lightly as an ornament.
- Ledger Line: A ledger line is a short horizontal line added above or below the staff to extend its range and allow for the notation of pitches above or below the staff.

- Slur: A slur is a curved line placed over or under two or more notes to indicate that they should be played smoothly and connectedly, without any separation between them.
- Tie: A tie is a curved line connecting two notes of the same pitch, indicating that they should be played as a single, sustained note with the combined duration of both notes.

Bibliography

1. Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning* http://www.deeplearningbook.org (MIT Press, 2016).
2. Fitch, F. B. McCulloch Warren S. and Pitts Walter. A logical calculus of the ideas immanent in nervous activity. Bulletin of mathematical biophysics, vol. 5, pp. 115–133 (1944).
3. Neutelings, I. *Neural networks with tiks* https://tikz.net/neural_networks/.
4. Bird, S., Klein, E. & Loper, E. *Natural Language Processing with Python* ISBN: 9780596516499. https://books.google.es/books?id=ScL3wAEACAAJ (O'Reilly Media, 2009).
5. Neubig, G. Neural machine translation and sequence-to-sequence models: A tutorial. *arXiv preprint arXiv:1703.01619* (2017).
6. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural computation* **9,** 1735–1780 (1997).
7. Cho, K., Van Merriënboer, B., Bahdanau, D. & Bengio, Y. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* (2014).
8. Clay-Atlas. *Graphical Introduction Note About GRU* https://clay-atlas.com/us/blog/2021/07/27/gru-en-introduction-note/.

9. Chung, J., Gulcehre, C., Cho, K. & Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

10. Vaswani, A. *et al.* Attention is all you need. *Advances in neural information processing systems* **30** (2017).

11. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

12. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., *et al.* Improving language understanding by generative pre-training (2018).

13. Huang, C.-Z. A. *et al.* Music Transformer: Generating Music with Long-Term Structure. *arXiv preprint arXiv:1809.04281* (2018).

14. Agostinelli, A. *et al.* MusicLM: Generating Music From Text. *arXiv preprint arXiv:2301.11325* (2023).

15. Alvira, J. R. *teoria.com is a website dedicated to the study and practice of music theory.* https://www.teoria.com/.

16. Geerlings, C. & Meroño-Peñuela, A. *Interacting with GPT-2 to Generate Controlled and Believable Musical Sequences in ABC Notation* in *NLP4MUSA* (2020).

17. Wu, S. & Sun, M. *TunesFormer: Forming Tunes with Control Codes* 2023. https://arxiv.org/abs/2301.02884.

18. Schwaber, K. & Sutherland, J. *The 2020 Scrum Guide* https://scrumguides.org/download.html.

19. Of Labour, M. & of Labour, S. E. D.-G. *Working hours, leave and holidays* https://administracion.gob.es/pag_Home/en/Tu-espacio-europeo/derechos-obligaciones/ciudadanos/trabajo-jubilacion/condiciones-trabajo/jornada-permisos.html.

20. PayScale. *Salary Comparison, Salary Survey, Search Wages* https://www.payscale.com/research/ES/.

21. Blanchar, C. *Mean Rent of a flat in Barcelona and Catalonia* https://elpais.com/espana/catalunya/2023-03-07/1077-euros-al-mes-por-un-piso-el-precio-del-alquiler-en-barcelona-alcanza-el-salario-minimo.html.

22. Agency, E. U. S. E. P. *Greenhouse Gas Equivalencies Calculator* https://www.epa.gov/energy/greenhouse-gas-equivalencies-calculator.

23. Dredge, S. *UK government rethinks plans for AI-training copyright exception* https://musically.com/2023/02/02/uk-government-rethinks-plans-for-ai-training-copyright-exception/.

24. Dong, H.-W., Chen, K., McAuley, J. & Berg-Kirkpatrick, T. Muspy: A toolkit for symbolic music generation. *arXiv preprint arXiv:2008.01951* (2020).

25. Boulanger-Lewandowski. *JSB Chorales* https://paperswithcode.com/dataset/jsb-chorales.

26. Bayron, C. J. *Autochord - Automatic Chord Recognition Library and Chord Visualization App* https://github.com/cjbayron/autochord.

27. Rosen, P. & Dyke., G. *abcjs - Javascript library for rendering standard music notation in a browser.* https://www.abcjs.net/.

28. Walshaw, C. *The abc music standard 2.1* https://abcnotation.com/wiki/abc:standard:v2.1.

29. Kerlinger, C. *The Most Popular Key In Music: A Comparison Of Classical And Popular Songs* https://www.benvaughn.com/the-most-popular-key-in-music-a-comparison-of-classical-and-popular-songs/.