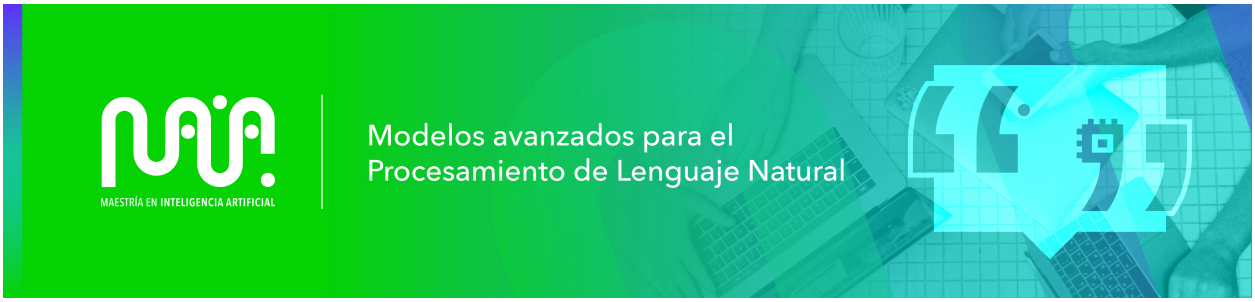

Micro proyecto 1

Rubén Manrique

Universidad de Los Andes, Colombia

1 Introducción

En este documento se describe de manera completa el micro proyecto 1 del curso, el cual comprende el contenido de la semana 1 y 2. Además de conocer el propósito del proyecto, también se presenta la rúbrica con la que se realizará la evaluación del mismo. El documento está estructurado de la siguiente forma:

Contents

2 Resultados de Aprendizaje

Con esta actividad se busca que el estudiante pueda poner en práctica el uso de modelos de redes neuronales para trabajar con datos secuenciales. Tras realizar esta actividad se espera que el estudiante esté en capacidad

de:

1. Estructurar datos secuenciales de forma que pueden ser usados por modelos de redes neuronales.
2. Desarrollar soluciones de machine learning para datos secuenciales en distintos dominios.
3. Entrenar modelos de machine learning con múltiples salidas.
4. Entrenar modelos de machine learning con una función de costo compuesta.
5. Utilizar el modelo entrenado para generar nuevas secuencias.

3 Descripción del micro proyecto

Para este micro proyecto, usted dispone de 292 producciones musicales de reconocidos compositores como Ludwig Van Beethoven. El objetivo del proyecto es que usted entrene un modelo con la producción musical de alguno de los compositores que tenga la capacidad de generar música en el estilo del compositor seleccionado.

Elija alguno de los instrumentos disponibles en las composiciones, y utilice toda la producción de un único compositor como datos de entrenamiento y validación. Los datos que debe utilizar para entrenar el modelo puede obtenerlos de la siguiente forma mediante la librería *pretty midi*:

```
import pretty_midi
pm = pretty_midi.PrettyMIDI("path_to_file")
instrument = pm.instruments[0]
instrument_name = pretty_midi.program_to_instrument_name(instrument.program)
print('Instrument name:', instrument_name)
print(instrument.notes)
```

Para el entrenamiento del modelo use las siguientes características que puede calcularlas o extraerlas directamente del objeto *Note* para cada nota en el arreglo `instrument.notes`:

- pitch: Esta característica es una variable categórica que representa la nota musical.
- step: Esta característica corresponde a una variable numérica que se debe calcular como el *start* de la nota actual menos el *start* de la nota anterior, y representa el espacio de tiempo que hay entre la nota anterior y la nota actual.
- duration: Esta característica también es numérica, y se calcula como la diferencia entre el tiempo en el que termina la nota actual (*end*) menos el tiempo en el que inició la nota (*start*), lo que refleja la duración en tiempo de la nota.
- velocity (opcional): Esta característica se refiere a la velocidad de la nota (numérica), pero incluirla al modelo es opcional. En caso de que se decida ignorarla, al momento de generación todas las notas deberán tener un valor por defecto que usted seleccione.

Cree un modelo que haga uso de las características mencionadas de notas anteriores para generar la siguiente nota, la cual está compuesta por las mismas características, por lo que su modelo debe tener la capacidad de generar varias salidas. Vea la Figura 1 para ver el detalle de las entradas y salidas del modelo.

Finalmente, para poder apreciar la secuencia de notas generada por su modelo, puede crear un archivo *MIDI*. En primera instancia, se crea un objeto *PrettyMIDI*, se selecciona el instrumento, y posteriormente se agregan las notas y se guarda el archivo. Tenga en cuenta que la clase *Note* se incializa con los argumentos *velocity*, *pitch*, *start*, y *end*, por lo que antes debe calcular el valor de *start* y *end* por medio de las características *step* y *duration* predichas para cada una de las notas. A continuación vea un fragmento de código guía para generar un archivo con las notas generadas:

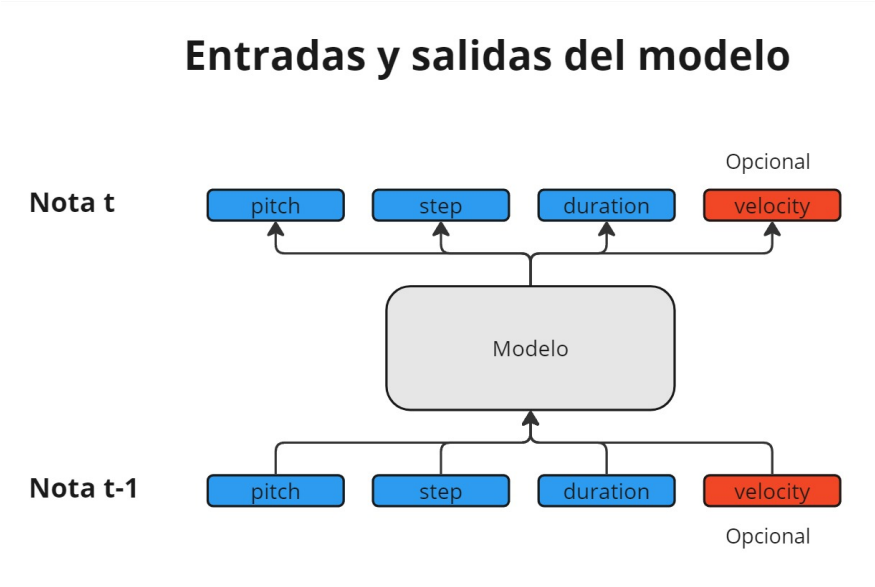

Figure 1: Diagrama con los datos de entrada y de salida del modelo que debe implementar. Su modelo recibirá como entrada las 4 características (velocity es opcional) de las notas que su modelo recibe como contexto ($Nota_{t-1}, Nota_{t-2}, ..., Nota_{t-c}$ donde c es el tamaño del contexto), y predice las mismas 4 características para la nota en el tiempo t ($Nota_t$)

.

```
pm = pretty_midi.PrettyMIDI() #Se crea un objeto PrettyMIDI
instrument = pretty_midi.Instrument(
  program=pretty_midi.instrument_name_to_program(
      instrument_name)) # Se crea el instrumento para el cual se generaron notas

note = pretty_midi.Note(velocity=velocity, pitch=pitch,
                                start=start, end=end) # se crea una nota
instrument.notes.append(note) # Se agrega una nota al instrumento
pm.instruments.append(instrument) # Se agrega el instrumento al archivo
pm.write('path_to_save_the_file') # Se guarda el archivo
```

Algunos reproductores de música pueden reproducir directamente los archivos *MIDI*, pero también puede convertirlos a formatos más comúnes como *WAV* mediante la siguiente página web `https://audio.online-convert.com/es/convertir/midi-a-wav`.

3.1 Detalles del dataset

El dataset se compone de la producción musical de 19 reconocidos compositores:

- Isaac Albéniz
- Johann Sebastian Bach
- Mily Alexeyevich Balakirev
- Ludwig van Beethoven
- Aleksandr Borodín
- Johannes Brahms

- Friedrich Burgmüller
- Frédéric Chopin
- Claude Debussy
- Enrique Granados
- Edvard Grieg
- Joseph Haydn
- Franz Liszt
- Felix Mendelssohn
- Wolfgang Amadeus Mozart
- Modest Mussorgsky
- Franz Schubert
- Robert Schumann
- Pyotr Ilyich Tchaikovsky

4 Rúbrica de la nota del micro proyecto

La nota del micro proyecto se compone de la forma descrita en la siguiente tabla:

Item	**Porcentaje de la nota**
Arquitectura seleccionada	30%
Implementación del modelo	20%
Entrenamiento del modelo	30%
Resultados obtenidos	20%

Table 1: Composición de la nota del micro proyecto

4.1 Arquitectura seleccionada

El estudiante debe crear un diagrama que detalle la arquitectura implementada. El diagrama no tiene una forma específica definida, pero si debe reflejar los datos de entrada que acepta, la forma en que se procesan los datos por el modelo, y sus salidas en tiempo de entrenamiento. Se evluará adicionalmente la idoneidad de la arquitectura planteada para la solución del problema. El diagrama debe estar acompañado por una explicación detallada de la arquitectura, y la justificación de su selección.

4.2 Implementación del modelo

Se evaluará que el modelo corresponda a la arquitectura planteada, y que la implementación sea técnicamente correcta. Adicionalmente, se evaluará el grado en el que se demuestra el concocimiento sobre el uso de *Pytorch*, y de redes neuronales en general.

4.3 Entrenamiento del modelo

Se evaluará la implementación del bucle de entrenamiento construido, la función de pérdida utilizada (la cual se espera sea una composición de distintos objetivos de entrenamiento), y la selección razonable de los hiper parámetros del modelo. Adicionalmente, se evaluará la forma en la que se usaron los datos para realizar el entrenamiento del modelo.

4.4 Resultados obtenidos

El estudiante debe reservar al menos una pequeña porción de los datos para evaluar la capacidad de generalización del modelo. Se evaluará el grado en el que el modelo demuestra tener la capacidad de ajustarse a los datos, y las acciones realizadas para evitar el sobre ajuste de los datos.

El estudiante debe crear por los menos menos 3 archivos tipo *WAV* que contengan 200 notas generadas por el modelo cada uno.

4.5 Cumplimiento de requisitos de la actividad

Recuerde que su modelo no tendrá validez si no cumple los requisitos que se mencionan a continuación. Por lo tanto, obtendrá una calificación de 0 en la actividad correspondiente al micro proyecto.

Los requisitos son los siguientes:

- Para el entrenamiento del modelo solo se debe usar los datos proporcionados, no podrá usar datos de otras fuentes.
- Debe implementar y entrenar modelos desde cero haciendo uso de *Pytorch*, y si lo desea de las capas disponibles en la librería como *Linear*, *LSTM*, *Embedding*, entre otras.
- El código enviado debe poder ser ejecutado por el equipo evaluador, y el código debe ser completamente de su autoría.

5 Entregables en Coursera

En la semana 2 del curso se habilitará una actividad en donde usted debe subir los siguientes entregables:

1. Jupyter Notebook que se utilizó para realizar la implementación y el entrenamiento del modelo. La imagen del diagrama junto con cualquier texto debe estar embebido dentro del Jupyter notebook.
2. La imagen con el diagrama también debe adjuntarla por aparte para asegurarse de que el equipo evaluador pueda visualizarla correctamente.
3. Tres archivos tipo *WAV* en donde cada archivo debe contener 200 notas generadas por su modelo.