

Creación de melodías para un estándar de Blues a partir de un archivo MIDI

by Ian Angel León Niño

**Escuela Colombiana de Ingeniería Julio Garavito
2023-2 PROMPT-M**

12 de noviembre de 2023

Abstract

Artificial intelligence has revolutionized musical composition, enabling the creation of original melodies. In this report, we explore how AI is used in music, highlighting its tools, successful applications, and ethical challenges. This technology is transforming music and offering new creative opportunities to composers and musicians.

1. Problemática inicial

La IA aplicada a la composición musical combina técnicas de aprendizaje automático, redes neuronales y procesamiento de datos musicales para generar piezas musicales originales y personalizadas. Estos sistemas pueden aprender de vastas cantidades de música existente, adaptarse a diferentes estilos y géneros, y generar melodías que van desde lo clásico hasta lo vanguardista.

En este informe, exploraremos los diferentes enfoques y herramientas disponibles para utilizar la IA en la composición musical, destacando ejemplos de aplicaciones exitosas y los beneficios que esto conlleva para la industria musical y los compositores individuales. Además, discutiremos los desafíos éticos y creativos asociados con la integración de la IA en el proceso creativo y cómo los compositores pueden aprovechar al máximo esta tecnología para fomentar su propia creatividad y expresión artística.

1.1. Profundización de la problemática

1. Definición del área de la música en la que se desea trabajar:

El área de trabajo en la música es muy amplio por lo que para desarrollar una herramienta exitosa que ayude a las personas a generar un acercamiento en un campo específico, se debe elegir un único género musical en este caso que es de

composición o creación de melodías enfocadas a trabajar.

2. Para este ejercicio vamos a implementar la estructura de un estándar de Blues ya que nos facilita una estructura en la que podremos trabajar y entender como realizar melodías sencillas para generar un acercamiento didáctico a personas que apenas estén empezando a conocer sobre esta temática.
3. Conceptos teóricos a tener en cuenta para el desarrollo del ejercicio:

✓ Tonalidad:

La tonalidad se refiere a la organización de las notas y acordes dentro de una pieza musical en torno a un tono principal, conocido como la tónica. La tonalidad es un elemento esencial para la comprensión de la armonía y la estructura de una composición.

✓ Compás:

En música, un compás, también conocido como medida o tiempo, es una unidad de organización del ritmo que divide una composición musical en segmentos regulares. Los compases son esenciales para mantener un sentido de tiempo y estructura en una pieza musical.

✓ Escala blues:

La escala blues es una escala musical característica que se utiliza ampliamente en el género del blues, así como en otros estilos musicales, como el rock, el jazz y el funk. Esta escala se distingue por su sonido distintivo y su uso de notas "blues." "blue notes," que son notas con una afinación ligeramente disonante o fuera de tono, lo que le da a la escala su característico sabor bluesy. La escala blues se basa en la escala pentatónica, pero agrega una o más de estas notas blue para crear un sonido característico. Aquí está la estructura básica de la escala blues en la tonalidad de Do:

Do (I) Mib (III blue note) Fa (IV) Solb (V blue note) Sol (V) Sib (VII blue note)

La escala blues se puede tocar en cualquier tonalidad, y esta estructura básica se puede trasladar a diferentes tonos comenzando en una nota diferente. El uso de las notas blue (las que tienen el sufijo "blue") indica que son ligeramente más bajas en afinación) es lo que le da a la escala blues su característico sentimiento melódico.

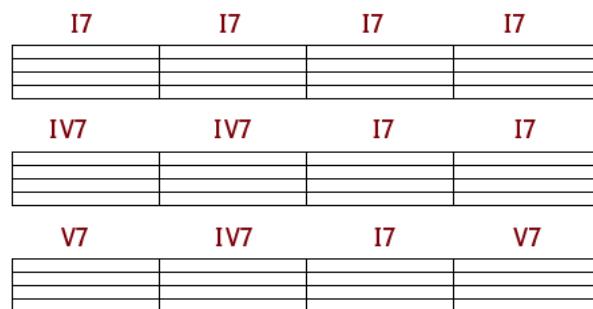
- ✓ MIDI: Un archivo MIDI, acrónimo de "Musical Instrument Digital Interface", es un formato estándar de archivo utilizado para representar información musical en forma digital. A diferencia de otros tipos de archivos de audio, como MP3 o WAV, un archivo MIDI no contiene sonidos reales o grabaciones musicales. En su lugar, almacena datos que describen instrucciones para reproducir música.

4. ¿Qué es un blues y de qué se compone?

El blues es un género musical que se originó en los Estados Unidos, particularmente en el sur, a finales del siglo XIX. Es conocido por su profundo sentimiento, expresión emocional y narrativa lírica, y ha influido en una amplia gama de estilos musicales, incluyendo el rock, el jazz y el rhythm and blues. El blues típicamente se compone de los siguientes elementos:

- ✓ Estructura de 12 compases: El blues generalmente se estructura en ciclos de 12 compases. Cada compás suele consistir en cuatro tiempos, y la progresión de acordes básica es una secuencia de acordes de tres acordes, que a menudo sigue un patrón I-IV-V (primer grado, cuarto grado y quinto grado de la escala).

Figura 1. Estructura de 12 compases



Para el ejercicio en el último compás en vez del quinto grado, se deja el primer grado, por lo que al final serán dos compases de primero.

- ✓ Escala blues: La escala blues es fundamental en este género musical. Es una variación de la escala pentatónica con la adición de una nota "blue" (nota azul) que da al blues su característico sonido melódico. La escala blues se utiliza tanto en la melodía como en la improvisación instrumental.

- ✓ Instrumentación característica: Los instrumentos comunes en el blues incluyen la guitarra, la armónica, el bajo, la batería y el piano. La guitarra eléctrica desempeña un papel destacado en muchas variantes del blues, como el blues eléctrico.

2. Preguntas realizadas a ChatGPT

Figura 2. Pregunta 1

¿Cómo puedo innovar en aplicaciones para ingeniería de PROMT en el ámbito de la música para generar melodías sencillas?

La innovación en aplicaciones de Procesamiento de Lenguaje Natural (NLP) como PROMT en el ámbito de la música para generar melodías sencillas es un desafío interesante que combina aspectos técnicos y creativos. Aquí hay algunas ideas y enfoques para explorar:

1. **Combinación de Letras y Melodía:** Puedes desarrollar una aplicación que tome letras de canciones o texto escrito y genere una melodía que se ajuste a esas letras. Esto implicaría analizar el texto para determinar el ritmo, el tono y la estructura de la melodía.
2. **Melodías Generativas Basadas en Reglas:** Crea algoritmos que generen melodías simples basadas en reglas predefinidas. Por ejemplo, podrías establecer reglas para que ciertas palabras generen patrones rítmicos o melódicos específicos.
3. **Inteligencia Artificial y Aprendizaje Automático:** Utiliza modelos de aprendizaje automático y técnicas de inteligencia artificial para entrenar un sistema en la generación de melodías basadas en texto o en emociones asociadas al texto. Estos modelos pueden aprender a mapear palabras o frases a melodías de manera más sofisticada.
4. **Interacción del Usuario:** Permite que los usuarios introduzcan texto, como una frase o una emoción, y generen melodías personalizadas en función de su entrada. Esto puede hacer que la aplicación sea más interactiva y creativa.
5. **Bases de Datos Musicales:** Utiliza bases de datos musicales y catálogos de melodías preexistentes para inspirarte y combinar elementos de diferentes melodías de manera novedosa.
6. **Integración con Plataformas de Música:** Si estás interesado en la música generativa, puedes explorar cómo integrar tu aplicación con plataformas de música como SoundCloud o Spotify para permitir a los usuarios crear y compartir sus composiciones.

Figura 3

7. **Colaboración con Músicos y Compositores:** Trabaja con músicos y compositores para combinar la creatividad humana con la generación automática de melodías. La colaboración puede dar lugar a resultados únicos.
8. **Evaluación y Retroalimentación de Usuarios:** Recopila comentarios y opiniones de los usuarios para ajustar y mejorar continuamente la generación de melodías en función de su experiencia y preferencias.

Recuerda que la generación de música es una tarea desafiante, ya que implica aspectos artísticos y subjetivos. La innovación en este campo requerirá una combinación de conocimientos en música, procesamiento de lenguaje natural y programación, así como la capacidad de experimentar y ser creativo. Además, es importante tener en cuenta los derechos de autor al utilizar o distribuir música generada automáticamente.

Figura 4. Pregunta 2



¿Que código me podrías proporcionar para generar melodias sencillas en base a una tonalidad que yo elija para hacer una canción?



Generar melodías sencillas en base a una tonalidad específica es un proceso creativo, y el código necesario puede ser bastante complejo. Sin embargo, puedo proporcionarte un ejemplo simple de cómo generar una melodía utilizando Python y la biblioteca `music21`. En este ejemplo, generaré una melodía simple en la tonalidad de Do mayor (C Major), pero puedes personalizar la tonalidad según tus preferencias.



Primero, asegúrate de instalar la biblioteca `music21` si aún no lo has hecho. Ejecuta esta celda de código en Google Colab o en tu entorno de Python:

```
python
```

Copy code

```
!pip install music21
```

Figura 5

A continuación, puedes utilizar el siguiente código para generar una melodía simple en la tonalidad de Do mayor (C Major). La melodía se genera automáticamente y se imprime en una partitura:

```
python Copy code

from music21 import stream, note, scale

# Define la tonalidad (C Major)
tonalidad = scale.MajorScale('C')

# Crea un objeto Stream para la melodía
melodia = stream.Stream()

# Define una secuencia de grados de la tonalidad
grados = [1, 2, 3, 4, 5, 6, 7, 8]

# Agrega notas a la melodía en la tonalidad de Do mayor
for grado in grados:
    nota = tonalidad.getChord(grado, 4)[0]
    nueva_nota = note.Note(nota)
    melodía.append(nueva_nota)

# Muestra la partitura de la melodía
melodia.show()
```

Este código utiliza la biblioteca `music21` para crear una melodía en la tonalidad de Do mayor (C Major). Genera una secuencia de notas basada en la tonalidad y muestra la partitura resultante.

Puedes personalizar la tonalidad y la duración de las notas, así como experimentar con diferentes patrones de melodía. La biblioteca `music21` ofrece muchas opciones para trabajar con la música de manera más avanzada, por lo que puedes explorar sus funciones y características adicionales para crear melodías más elaboradas.

Figura 6. Pregunta 3

The screenshot shows a Q&A platform interface. A user asks: "Explícame que es \"music21\"". The answer provided is:

music21 es una biblioteca de Python que se utiliza para trabajar con y analizar música. Esta biblioteca proporciona herramientas y funcionalidades para manipular, analizar, componer y visualizar música de manera programática. Es especialmente útil para músicos, compositores, investigadores de música y programadores interesados en el análisis y la generación de música utilizando Python.

Algunas de las características clave de **music21** incluyen:

- Representación de Notación Musical:** **music21** permite crear, editar y representar partituras musicales en notación estándar, como partituras en pentagrama, acordes, escalas y ritmos.
- Análisis Musical:** Puedes realizar análisis musical utilizando **music21**, como identificar escalas, acordes, progresiones armónicas, contrapuntos y más. Esto es útil tanto para estudiantes de música como para investigadores.
- Generación de Música:** Puedes generar música de forma programática con **music21**. Esto es útil para compositores y músicos que desean crear nuevas composiciones musicales.
- MIDI y MusicXML:** La biblioteca es compatible con formatos comunes de música, como archivos MIDI y MusicXML, lo que facilita la importación y exportación de datos musicales.

Figura 7. Se ven las diferentes funciones que nos ofrece la librería

5. **Visualización de Música:** `music21` permite la creación de gráficos y visualizaciones de partituras musicales. Puedes mostrar partituras en notación musical o representar información musical de diversas formas.
6. **Manipulación de Datos Musicales:** Puedes manipular datos musicales, como transposición, inversión, transcripción y más, utilizando las herramientas de la biblioteca.
7. **Interfaz de Programación de Aplicaciones (API):** `music21` proporciona una API rica y flexible que te permite acceder a todas sus funciones y realizar tareas avanzadas de análisis y generación de música.

`music21` es una biblioteca versátil que puede ser utilizada en una variedad de aplicaciones, desde la educación musical y la investigación hasta la creación de música asistida por ordenador. Si estás interesado en trabajar con música y programación en Python, `music21` es una excelente opción para explorar y experimentar con la música de manera programática.

Figura 8. Pregunta 4



```
Ahora se me presenta este error "SubConverterException
Traceback (most recent call last)
<ipython-input-28-e682f158ca28> in <cell line: 19>()
    17
    18 # Muestra la partitura de la melodía
---> 19 melodía.show()

5 frames
/usr/local/lib/python3.10/dist-
packages/music21/converter/museScore.py in
runThroughMuseScore(fp, subformats, dpi, trimEdges, leaveMargin,
**keywords)
    53     + 'put a link to it in your .music21rc via Environment.')
    54 if not museScorePath.exists():
---> 55     raise SubConverterException(
    56     "Cannot find a path to the 'mscore' file at "
    57     + f'{museScorePath} -- download MuseScore')

SubConverterException: Cannot find a path to the 'mscore' file at
/usr/bin/mscore3 -- download MuseScore "
```

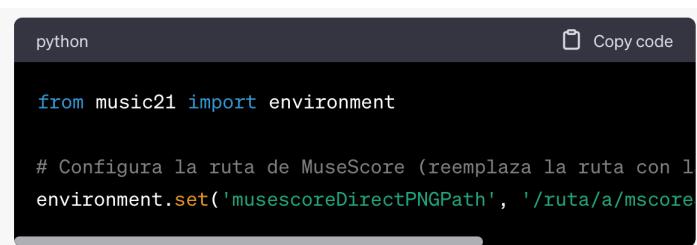


El error que estás viendo se debe a que `music21` está tratando de usar MuseScore para mostrar la partitura, pero no puede encontrar el archivo ejecutable de MuseScore en la ubicación predeterminada. Para solucionar este problema, puedes seguir estos pasos:



1. **Instalar MuseScore:** Asegúrate de que MuseScore esté instalado en tu sistema. Puedes descargar e instalar MuseScore desde su sitio web oficial (<https://musescore.org/>). Asegúrate de que MuseScore esté instalado y funcionando correctamente en tu computadora.
2. **Configurar la Ruta de MuseScore:** Despues de instalar MuseScore, debes configurar la ruta en `music21` para que sepa dónde encontrar MuseScore. Puedes hacerlo en tu código de la siguiente manera:

Figura 9. Se procede a intalar el programa MuseScore para poder visualizar la exportacion del archivo MIDI generado por el codigo.



```
python
from music21 import environment

# Configura la ruta de MuseScore (reemplaza la ruta con la
# correcta de tu sistema)
environment.set('musescoreDirectPNGPath', '/ruta/a/mscore3')
```

Asegúrate de reemplazar `'/ruta/a/mscore3'` con la ubicación correcta del archivo ejecutable de MuseScore en tu sistema.

1. **Ejecutar el Código:** Despues de configurar la ruta de MuseScore, ejecuta tu código nuevamente. Con la ruta de MuseScore configurada correctamente, `music21` debería poder utilizar MuseScore para mostrar la partitura de la melodía.

Siguiendo estos pasos, deberías poder resolver el problema y visualizar la partitura de la melodía correctamente.

Figura 10. Pregunta 5

 ¿Puedes brindarme ese mismo código pero que la melodía sea una secuencia básica de blues?

 Claro, aquí tienes un código que genera una secuencia básica de blues en la tonalidad de Do mayor (C Major) y la exporta como un archivo MIDI:

python Copy code

```
from music21 import stream, note, scale

# Define la tonalidad (C Major)
tonalidad = scale.MajorScale('C')

# Crea un objeto Stream para la secuencia de blues
secuencia_blues = stream.Stream()

# Define una secuencia de grados de la tonalidad para una
grados = [1, 4, 1, 1, 4, 4, 1, 1, 5, 4, 1, 5]

# Duración de las notas (en fracciones)
duraciones = [1/4, 1/4, 1/4, 1/4, 1/4, 1/4, 1/4, 1/4, 1/4, 1/4]

# Agrega notas a la secuencia de blues en la tonalidad de
for grado, duracion in zip(grados, duraciones):
    nota = tonalidad.pitchFromDegree(grado)
    nueva_nota = note.Note(nota)
    nueva_nota.duration.quarterLength = duracion
    secuencia_blues.append(nueva_nota)

# Exporta la secuencia de blues a un archivo MIDI
secuencia_blues.write('midi', 'secuencia_blues.mid')
```

Este código crea una secuencia de blues básica en la tonalidad de Do mayor (C Major) y la exporta como un archivo MIDI llamado "secuencia_blues.mid". Puedes ajustar los grados y las duraciones de las notas según tus preferencias para crear diferentes secuencias de blues.

Figura 11. Pregunta 6



Modifica este código para que yo pueda subir cualquier posición de los grados que yo quiera una octava arriba " from music21 import stream, note, scale



```
# Define la tonalidad con un séptimo grado bemol (Bb) en la tonalidad de Do mayor (C Major)
tonalidad = scale.MajorScale('C')
tonalidad.getPitches()[6] = tonalidad.getPitches()[6].transpose(-1) # Cambiar el séptimo grado a Bb

# Crea un objeto Stream para la secuencia de blues
secuencia_blues = stream.Stream()

# Define una secuencia de grados para la secuencia de blues
grados = [1, 1, 1, 4, 4, 1, 1, 5, 4, 1, 5]

# Duración de las notas (en fracciones)
duraciones = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

# Grado que deseas poner una octava arriba (por ejemplo, el primer grado)
grado_octava_arriba = 1

# Agrega notas a la secuencia de blues en la tonalidad elegida
for i, grado in enumerate(grados):
    if i == grado_octava_arriba: # Si es el grado que deseas cambiar
        # Cambia la altura del grado a una octava arriba
        nota = tonalidad.pitchFromDegree(grado).transpose(12)
    else:
        nota = tonalidad.pitchFromDegree(grado)
    nueva_nota = note.Note(nota)
    nueva_nota.duration.quarterLength = duraciones[i]
    secuencia_blues.append(nueva_nota)

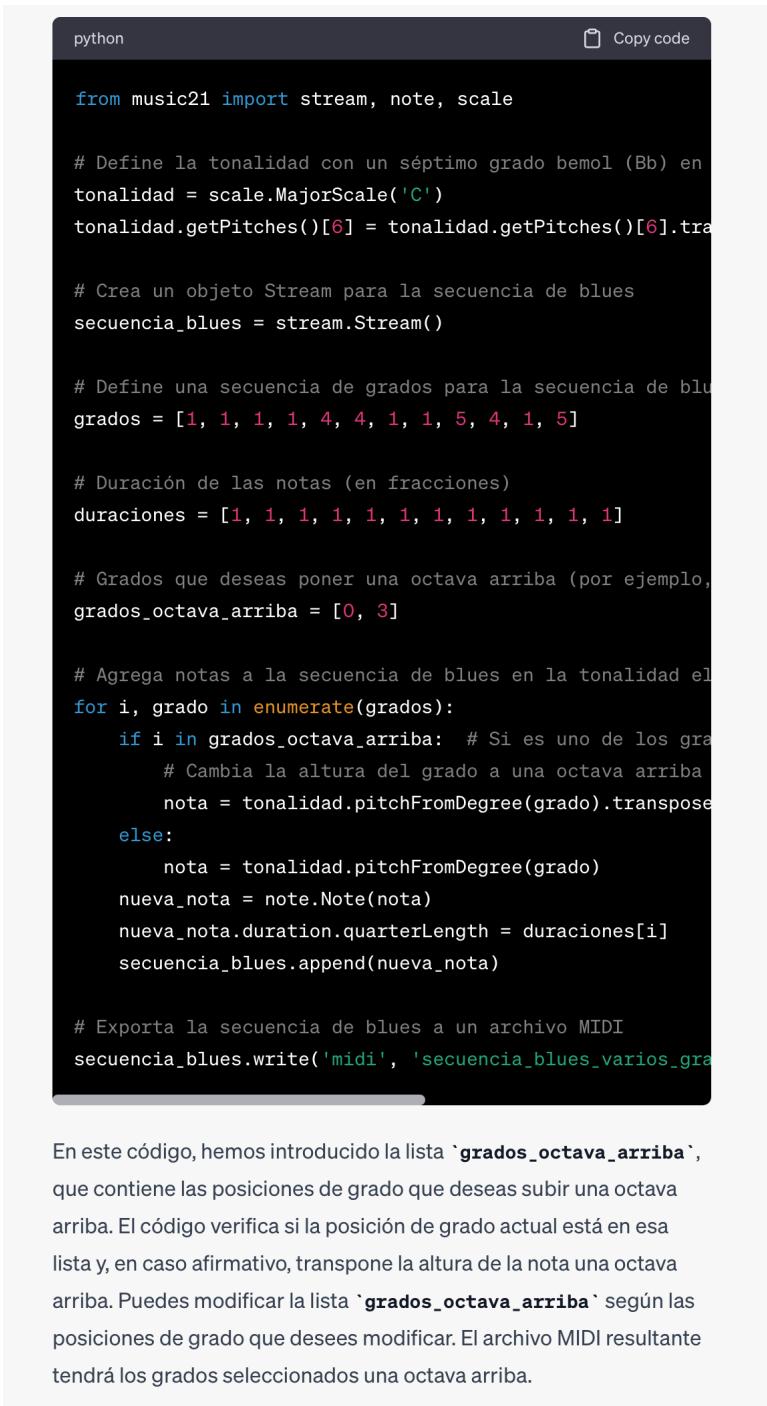
# Exporta la secuencia de blues a un archivo MIDI
secuencia_blues.write('midi', 'secuencia_blues_octava_arriba.mid') "
```



Para permitir que subas cualquier posición de los grados una octava arriba, puedes modificar el código de la siguiente manera. En lugar de especificar un grado fijo, puedes proporcionar una lista de posiciones de grado que deseas subir una octava arriba:



Figura 12



```
python
Copy code

from music21 import stream, note, scale

# Define la tonalidad con un séptimo grado bemol (Bb) en
tonalidad = scale.MajorScale('C')
tonalidad.getPitches()[6] = tonalidad.getPitches()[6].transpo
# Crea un objeto Stream para la secuencia de blues
secuencia_blues = stream.Stream()

# Define una secuencia de grados para la secuencia de blues
grados = [1, 1, 1, 1, 4, 4, 1, 1, 5, 4, 1, 5]

# Duración de las notas (en fracciones)
duraciones = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

# Grados que deseas poner una octava arriba (por ejemplo,
grados_octava_arriba = [0, 3]

# Agrega notas a la secuencia de blues en la tonalidad elegida
for i, grado in enumerate(grados):
    if i in grados_octava_arriba: # Si es uno de los grados
        # Cambia la altura del grado a una octava arriba
        nota = tonalidad.pitchFromDegree(grado).transpose(12)
    else:
        nota = tonalidad.pitchFromDegree(grado)
    nueva_nota = note.Note(nota)
    nueva_nota.duration.quarterLength = duraciones[i]
    secuencia_blues.append(nueva_nota)

# Exporta la secuencia de blues a un archivo MIDI
secuencia_blues.write('midi', 'secuencia_blues_varios_grados.mid')

En este código, hemos introducido la lista `grados_octava_arriba`, que contiene las posiciones de grado que deseas subir una octava arriba. El código verifica si la posición de grado actual está en esa lista y, en caso afirmativo, transpone la altura de la nota una octava arriba. Puedes modificar la lista `grados_octava_arriba` según las posiciones de grado que deseas modificar. El archivo MIDI resultante tendrá los grados seleccionados una octava arriba.
```

Figura 13. Pregunta 7

 Dame ese mismo código pero ahora con una función parecida que me permita bajarle medio todo a cualquier grado que yo quiera modificar

 Para permitir que puedas bajar medio tono a cualquier grado que deseas modificar en la secuencia de blues, puedes crear una función que tome como argumento la posición del grado que deseas modificar y aplique la transposición de medio tono. Aquí tienes el código con una función para bajar medio tono un grado específico:

```
python
from music21 import stream, note, scale

# Define la tonalidad con un séptimo grado bemol (Bb) en la tonalidad de Do
tonalidad = scale.MajorScale('C')
tonalidad.getPitches()[6] = tonalidad.getPitches()[6].transpose(-1) # Cambia el séptimo grado a Bb

# Función para bajar medio tono un grado específico
def bajar_medio_tono(grado):
    return grado - 0.5

# Crea un objeto Stream para la secuencia de blues
secuencia_blues = stream.Stream()

# Define una secuencia de grados para la secuencia de blues
grados = [1, 1, 1, 1, 4, 4, 1, 1, 5, 4, 1, 5]

# Duración de las notas (en fracciones)
duraciones = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

# Grado que deseas bajar medio tono (por ejemplo, el primer grado)
grado_bajar_medio_tono = 1

# Agrega notas a la secuencia de blues en la tonalidad elegida
for i, grado in enumerate(grados):
    if i == grado_bajar_medio_tono: # Si es el grado que deseas cambiar
        # Baja medio tono el grado
        grado = bajar_medio_tono(grado)
        nota = tonalidad.pitchFromDegree(grado)
    else:
        nota = tonalidad.pitchFromDegree(grado)
    nueva_nota = note.Note(nota)
    nueva_nota.duration.quarterLength = duraciones[i]
    secuencia_blues.append(nueva_nota)

# Exporta la secuencia de blues a un archivo MIDI
secuencia_blues.write('midi', 'secuencia_blues_medio_tono_abajo.mid')
```

Figura 14. Pregunta 8



A este código agrégale una función que le baje medio tono a los grado que yo quiera modificar "from music21 import stream, note, scale

```
# Define la tonalidad con un séptimo grado bemol (Bb) en la tonalidad de Do mayor (C Major)
tonalidad = scale.MajorScale('C')
tonalidad.getPitches()[6] = tonalidad.getPitches()[6].transpose(-1) # Cambiar el séptimo grado a Bb

# Crea un objeto Stream para la secuencia de blues
secuencia_blues = stream.Stream()

# Define una secuencia de grados para la secuencia de blues
grados = [1, 1, 1, 1, 4, 4, 1, 1, 5, 4, 1, 5]

# Duración de las notas (en fracciones)
duraciones = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

# Grados que deseas poner una octava arriba (por ejemplo, el primer y el cuarto grado)
grados_octava_arriba = [0, 3]

# Agrega notas a la secuencia de blues en la tonalidad elegida
for i, grado in enumerate(grados):
    if i in grados_octava_arriba: # Si es uno de los grados que deseas cambiar
        # Cambia la altura del grado a una octava arriba
        nota = tonalidad.pitchFromDegree(grado).transpose(12)
    else:
        nota = tonalidad.pitchFromDegree(grado)
    nueva_nota = note.Note(nota)
    nueva_nota.duration.quarterLength = duraciones[i]
    secuencia_blues.append(nueva_nota)

# Exporta la secuencia de blues a un archivo MIDI
secuencia_blues.write('midi', 'secuencia_blues_varios_grados_octava_arriba.mid')
```

Figura 15



Para agregar una función que permita bajar medio tono a los grados que deseas modificar en la secuencia de blues, puedes seguir el siguiente código:

```
python
from music21 import stream, note, scale

# Define la tonalidad con un séptimo grado bemol (Bb) en la tonalidad de Do
tonalidad = scale.MajorScale('C')
tonalidad.getPitches()[6] = tonalidad.getPitches()[6].transpose(-1) # Cambia el séptimo grado a Bb

# Crea un objeto Stream para la secuencia de blues
secuencia_blues = stream.Stream()

# Define una secuencia de grados para la secuencia de blues
grados = [1, 1, 1, 1, 4, 4, 1, 1, 5, 4, 1, 5]

# Duración de las notas (en fracciones)
duraciones = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

# Función para bajar medio tono un grado específico
def bajar_medio_tono(grado):
    return grado - 0.5

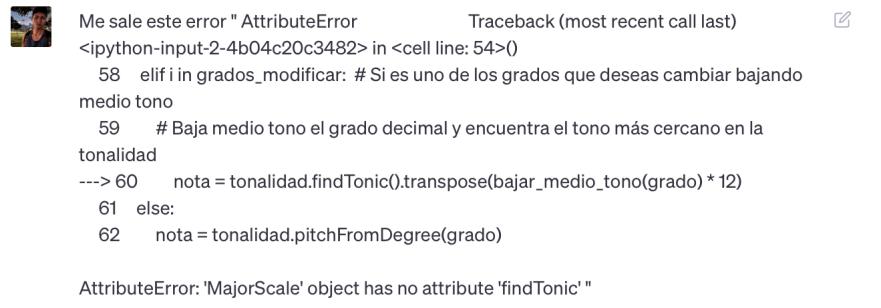
# Grados que deseas modificar (por ejemplo, el primer y el cuarto grado)
grados_modificar = [0, 3]

# Agrega notas a la secuencia de blues en la tonalidad elegida
for i, grado in enumerate(grados):
    if i in grados_modificar: # Si es uno de los grados que deseas cambiar
        # Baja medio tono el grado
        grado = bajar_medio_tono(grado)
        nota = tonalidad.pitchFromDegree(grado)
    else:
        nota = tonalidad.pitchFromDegree(grado)
    nueva_nota = note.Note(nota)
    nueva_nota.duration.quarterLength = duraciones[i]
    secuencia_blues.append(nueva_nota)

# Exporta la secuencia de blues a un archivo MIDI
secuencia_blues.write('midi', 'secuencia_blues_grados_modificados.mid')
```

En este código, hemos creado una función llamada `bajar_medio_tono` que toma un grado como argumento y lo baja medio tono. Luego, hemos definido una lista llamada `grados_modificar` que contiene las posiciones de los grados que deseas modificar. La función verifica si la posición de grado actual está en la lista `grados_modificar` y, en caso afirmativo, baja medio tono el grado. El archivo MIDI resultante tendrá los grados seleccionados bajados medio tono. Puedes modificar la lista `grados_modificar` para especificar qué grados deseas modificar en la secuencia de blues.

Figura 16. Despues de varios ajustes en los grados y corrección de errores obtenemos:

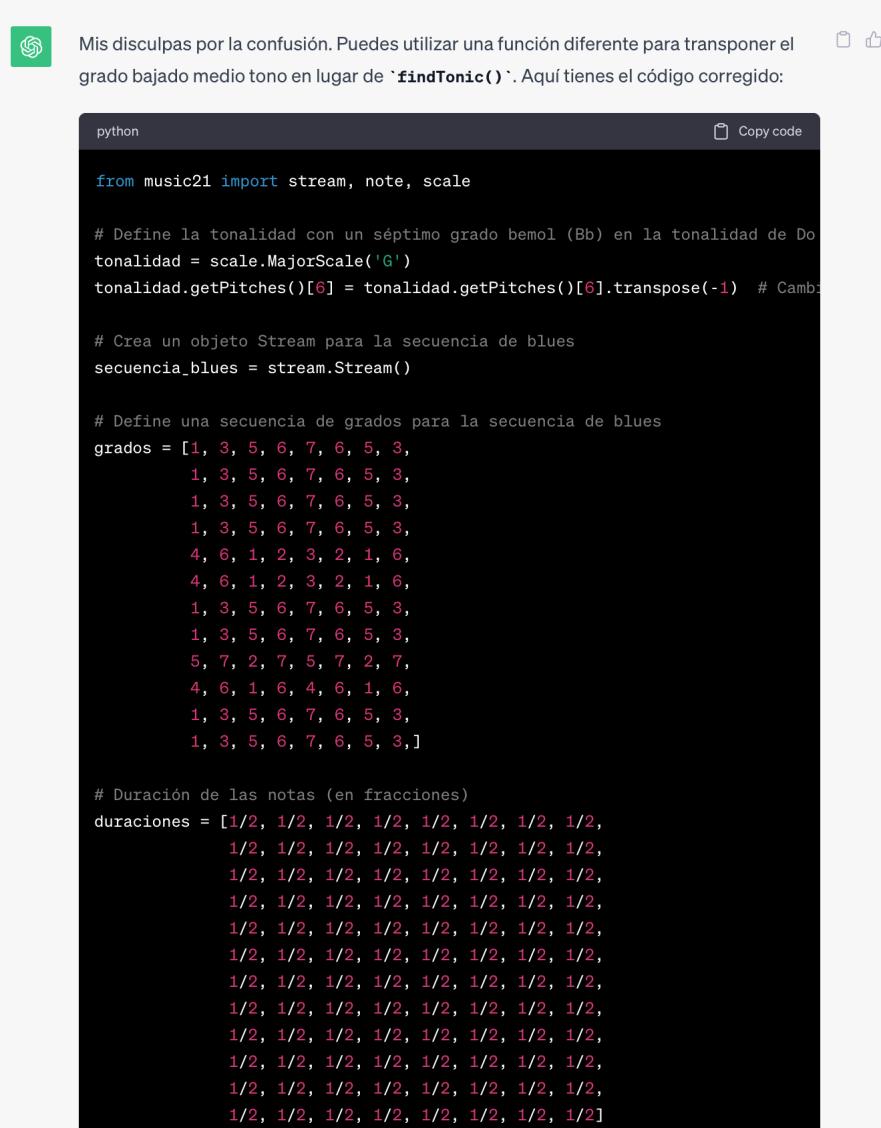


```

Me sale este error " AttributeError Traceback (most recent call last)
<ipython-input-2-4b04c20c3482> in <cell line: 54>()
      58     elif i in grados_modificar: # Si es uno de los grados que deseas cambiar bajando
      59         # Baja medio tono el grado decimal y encuentra el tono más cercano en la
tonalidad
--> 60     nota = tonalidad.findTonic().transpose(bajar_modo_tono(grado) * 12)
      61 else:
      62     nota = tonalidad.pitchFromDegree(grado)

AttributeError: 'MajorScale' object has no attribute 'findTonic' "

```



```

Mis disculpas por la confusión. Puedes utilizar una función diferente para transponer el
grado bajado medio tono en lugar de `findTonic()`. Aquí tienes el código corregido:

python
Copy code

from music21 import stream, note, scale

# Define la tonalidad con un séptimo grado bemol (Bb) en la tonalidad de Do
tonalidad = scale.MajorScale('G')
tonalidad.getPitches()[6] = tonalidad.getPitches()[6].transpose(-1) # Cambia el séptimo grado a Bb

# Crea un objeto Stream para la secuencia de blues
secuencia_blues = stream.Stream()

# Define una secuencia de grados para la secuencia de blues
grados = [1, 3, 5, 6, 7, 6, 5, 3,
          1, 3, 5, 6, 7, 6, 5, 3,
          1, 3, 5, 6, 7, 6, 5, 3,
          1, 3, 5, 6, 7, 6, 5, 3,
          4, 6, 1, 2, 3, 2, 1, 6,
          4, 6, 1, 2, 3, 2, 1, 6,
          1, 3, 5, 6, 7, 6, 5, 3,
          1, 3, 5, 6, 7, 6, 5, 3,
          5, 7, 2, 7, 5, 7, 2, 7,
          4, 6, 1, 6, 4, 6, 1, 6,
          1, 3, 5, 6, 7, 6, 5, 3,
          1, 3, 5, 6, 7, 6, 5, 3]

# Duración de las notas (en fracciones)
duraciones = [1/2, 1/2, 1/2, 1/2, 1/2, 1/2, 1/2, 1/2,
               1/2, 1/2, 1/2, 1/2, 1/2, 1/2, 1/2, 1/2,
               1/2, 1/2, 1/2, 1/2, 1/2, 1/2, 1/2, 1/2,
               1/2, 1/2, 1/2, 1/2, 1/2, 1/2, 1/2, 1/2,
               1/2, 1/2, 1/2, 1/2, 1/2, 1/2, 1/2, 1/2,
               1/2, 1/2, 1/2, 1/2, 1/2, 1/2, 1/2, 1/2,
               1/2, 1/2, 1/2, 1/2, 1/2, 1/2, 1/2, 1/2,
               1/2, 1/2, 1/2, 1/2, 1/2, 1/2, 1/2, 1/2,
               1/2, 1/2, 1/2, 1/2, 1/2, 1/2, 1/2, 1/2,
               1/2, 1/2, 1/2, 1/2, 1/2, 1/2, 1/2, 1/2]

```

Figura 17. Se optiene el primer resultado del ejercicio:

```
# Función para bajar medio tono un grado específico
def bajar_medio_tono(grado):
    return grado - 0.5

# Grados que deseas modificar bajando medio tono (por ejemplo, el segundo y
grados_modificar = [4, 12, 20, 28, 36, 44, 52, 60, 68, 76]

# Grados que deseas poner una octava arriba (por ejemplo, el primer y el cuarto)
grados_octava_arriba = [34, 35, 36, 37, 38, 42, 43, 44, 45, 46, 66, 70, 74, 75]

# Agrega notas a la secuencia de blues en la tonalidad elegida
for i, grado in enumerate(grados):
    if i in grados_octava_arriba: # Si es uno de los grados que deseas cambiar
        # Cambia la altura del grado a una octava arriba
        nota = tonalidad.pitchFromDegree(grado).transpose(12)
    elif i in grados_modificar: # Si es uno de los grados que deseas cambiar
        # Baja medio tono el grado
        nota = tonalidad.pitchFromDegree(bajar_medio_tono(grado))
    else:
        nota = tonalidad.pitchFromDegree(grado)
    nueva_nota = note.Note(nota)
    nueva_nota.duration.quarterLength = duraciones[i]
    secuencia_blues.append(nueva_nota)

# Exporta la secuencia de blues a un archivo MIDI
secuencia_blues.write('midi', 'secuencia_blues_varios_grados_modificados.mid')
```

Este código debería funcionar correctamente y bajar medio tono los grados especificados en la lista `grados_modificar`.

Figura 18. Ahora planteando la solución a ChatGPT de una manera totalmente específica se obtiene lo siguiente:

 Me podrías proporcionar un código para google colab que implemente la librería "music21" en el cual genere secuencia de blues que se exporte en un archivo MIDI en la tonalidad mayor que yo elija y que contenga doce compases. Cuatro compases del primer grado, dos compases del cuarto grado, dos compases del primer grado, un compas del quinto grado, un compas del cuarto grado, un compas del primer grado y un compas del quinto grado, en ese mismo orden y en cada compas quiero una secuencia de bajo caminante.

 Claro, aquí tienes un código para Google Colab que genera una secuencia de blues en la tonalidad mayor que elijas y exporta la secuencia como un archivo MIDI. Cada compás contiene una secuencia de bajo caminante (walking bass) que sigue la progresión de acordes típica del blues que has descrito:

```
python
from music21 import stream, chord, scale, note

# Define la tonalidad (ingresa la tonalidad que deseas, por ejemplo, 'C Major')
tonalidad = scale.MajorScale('C')

# Define una secuencia de acordes para los 12 compases del blues
acordes_blues = [chord.Chord(['C4', 'E4', 'G4']), # C Major
                  chord.Chord(['C4', 'E4', 'G4']), # C Major
                  chord.Chord(['C4', 'E4', 'G4']), # C Major
                  chord.Chord(['C4', 'E4', 'G4']), # C Major
                  chord.Chord(['F4', 'A4', 'C5']), # F Major
                  chord.Chord(['F4', 'A4', 'C5']), # F Major
                  chord.Chord(['C4', 'E4', 'G4']), # C Major
                  chord.Chord(['C4', 'E4', 'G4']), # C Major
                  chord.Chord(['G4', 'B4', 'D5']), # G Major
                  chord.Chord(['F4', 'A4', 'C5']), # F Major
                  chord.Chord(['C4', 'E4', 'G4']), # C Major
                  chord.Chord(['G4', 'B4', 'D5'])] # G Major

# Define las notas del bajo caminante
bajo_caminante = [note.Note('C2'), note.Note('E2'), note.Note('D2'), note.Note('G2'), note.Note('B2'), note.Note('A2'), note.Note('C3'), note.Note('E3'), note.Note('D3'), note.Note('G3'), note.Note('B3'), note.Note('A3')]

# Crea un objeto Stream para la secuencia de blues
secuencia_blues = stream.Stream()

# Agrega los compases de blues con bajo caminante
for acorde in acordes_blues:
    compas = stream.Measure()
    compas.append(acorde)
    compas.append(bajo_caminante)
    secuencia_blues.append(compas)

# Exporta la secuencia de blues a un archivo MIDI
secuencia_blues.write('midi', 'secuencia_blues.mid')
```

Puedes ingresar la tonalidad deseada en la variable `tonalidad`, como 'C Major' para la tonalidad de Do mayor. Este código generará una secuencia de blues de 12 compases con bajo caminante que sigue la progresión de acordes que has descrito. El archivo MIDI resultante se guardará en el entorno de Google Colab y podrás descargarlo desde allí.

Asegúrate de haber instalado la biblioteca `music21` en tu entorno de Google Colab para que este código funcione correctamente.

 Regenerate

3. Resultados

Se muestran el resultado de los archivos MIDI ejecutados en MuseScore:

Figura 19. Resultado de archivo MIDI varios grados modificados

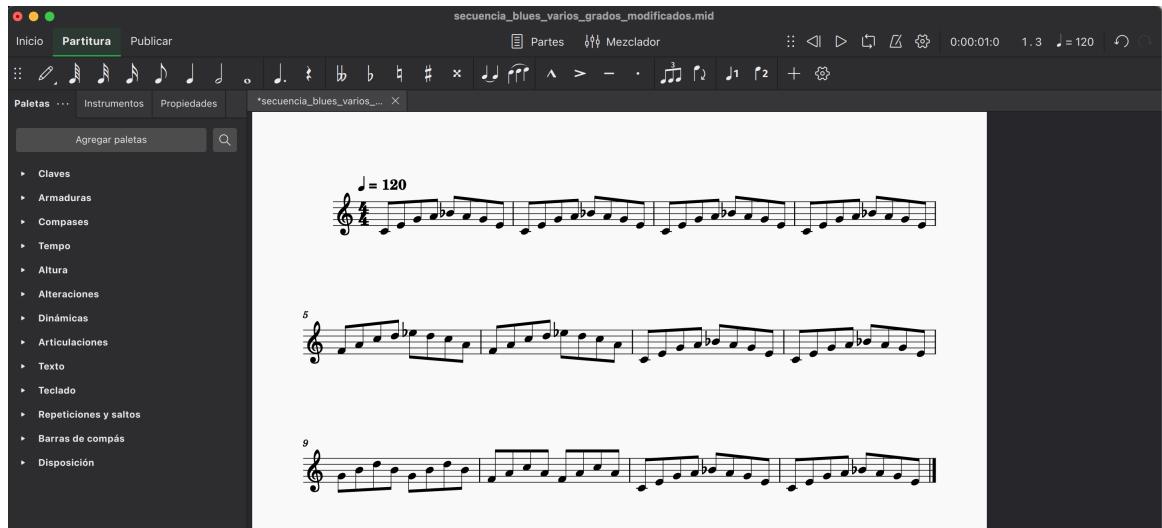


Figura 20. Resultado de archivo MIDI secuencia básica de blues

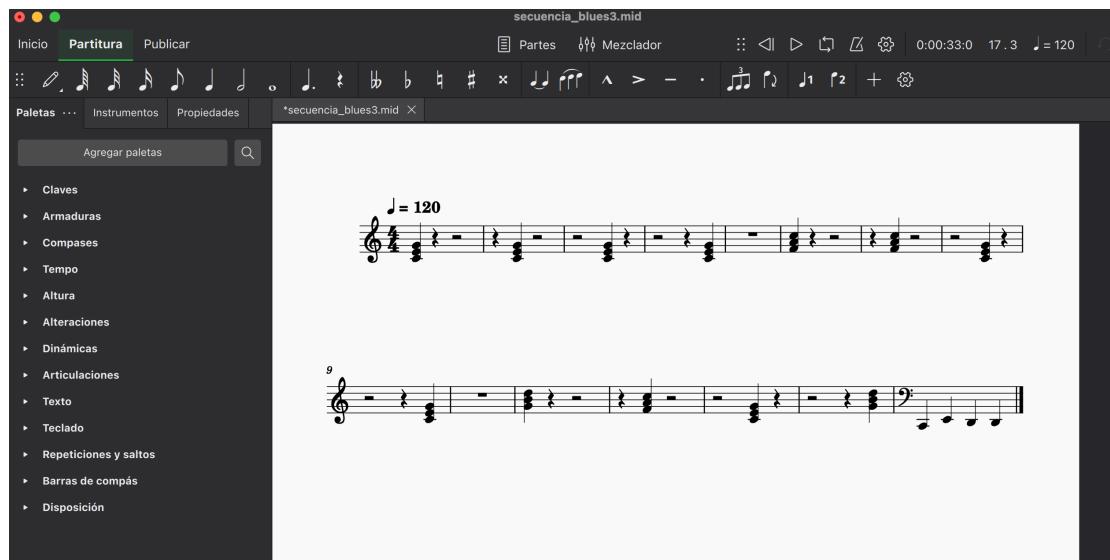
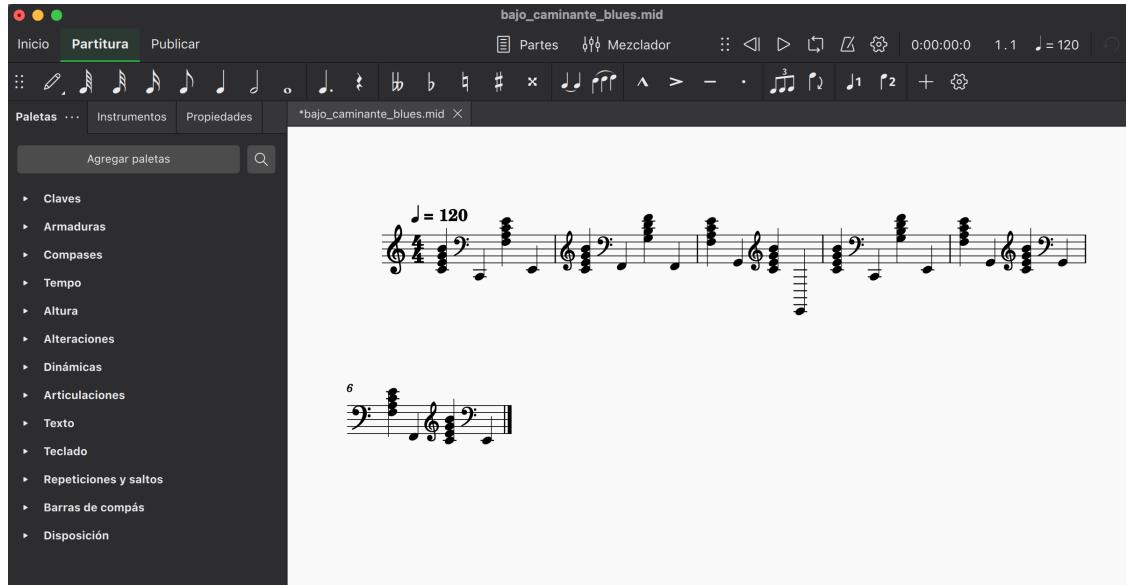


Figura 21. Resultado de archivo MIDI secuencia básica bajo caminante de blues



4. Debate ético

¿Es ético y creativo que una inteligencia artificial participe en el proceso de creaciones musicales?

La participación de la inteligencia artificial en el proceso creativo musical plantea interrogantes éticas y creativas relevantes. Desde la perspectiva ética, se cuestiona el rol de la creatividad genuina y la autoría en la música generada por sistemas computacionales. A su vez, desde el ángulo creativo, se debate cómo la colaboración con la inteligencia artificial puede ampliar los límites de la expresión artística.

En términos éticos, la preocupación principal radica en la atribución de la obra. ¿Quién es el autor cuando una pieza musical es generada por una IA? Esta pregunta desafía conceptos arraigados de autoría y originalidad en la creación artística. Además, la posible manipulación de la audiencia a través de la producción masiva de música generada por IA plantea cuestiones sobre la autenticidad y la conexión emocional con la obra.

En el aspecto creativo, la inteligencia artificial se percibe como una herramienta innovadora que desafía y expande los límites de la creatividad humana. La colaboración con sistemas de IA puede inspirar a los compositores, presentando nuevas ideas y perspectivas que quizás no hubieran explorado de otra manera. Este enfoque puede fomentar la experimentación y el descubrimiento de nuevos estilos, fusiones y formas musicales.

La clave radica en encontrar un equilibrio entre el uso de la inteligencia artificial como una herramienta que potencia la creatividad humana y el respeto por los principios éticos y la integridad artística. La combinación de la capacidad creativa humana con las capacidades analíticas y predictivas de la inteligencia artificial puede dar lugar a obras musicales emocionantes y auténticas, siempre y cuando se maneje de manera ética y se preserve la esencia de la expresión artística.

5. Análisis de los resultados

- ✓ El número de compases solicitado cuando se fue muy específico no cumplió con el requerimiento, pero si tuvo un resultado acercado a lo que en principio se pidió.
- ✓ El nivel de entrenamiento de las IA's llega a ser un factor fundamental en la dependencia de lograr alcanzar el propósito en la realización de las melodías o en general la creación de música.
- ✓ El código que ChatGPT proporcionó permitió realizar cambios en la métrica de las notas para lograr el resultado de los doce compases.

6. Conclusiones

- ✓ El trabajo conjunto con las herramientas de inteligencia artificial se acerca de formas más efectivas hacia lo que uno desea realizar específicamente.
- ✓ La comprensión que hoy en día tienen las IA's frente a las necesidades específicas no siempre es acertada dependiendo del tema a trabajar y la información que posea.
- ✓ La composición musical es una tarea que requiere de factores como la creatividad y libertad de expresión, por lo que este ejercicio solo proporcionará resultados sencillos.