Text-Speech Transformer

Gonzalo Rivas Cortés
Álvaro Mingo Noguerales
Alonso Moros Villalba
Álvaro Pintado Budia

December 15, 2024

Índice

Descripción

El proyecto tiene como objetivo convertir texto en audio utilizando el modelo de inteligencia artificial conocido como Transformers. El texto de entrada puede consistir en palabras aisladas o frases completas. Para procesarlo, cada palabra se convierte en un índice que corresponde a su posición dentro del vocabulario utilizado.

Dependiendo de si el objetivo es generar palabras individuales o frases completas, se emplearán diferentes modelos de Transformers, adaptados según la naturaleza del texto. Estos modelos se entrenan con conjuntos de datos de audio específicos, representados gráficamente mediante espectrogramas. Aunque los modelos varían según el tipo de entrada y salida, comparten una estructura base similar.

Estado del arte

Para realizar este proyecto, hemos utilizado los ejemplos y transparencias aportadas por la asignatura a la hora de entender la estructura, comportamiento básico de un transformer y su entrenamiento. En cuanto al modelo del proyecto, nos hemos basado en documentos y códigos de ejemplos ya subidos y publicados en github para comprender su desarrollo y mejorar la implementación. A su vez, utilizamos dos datasets, uno de palabras sueltas y otro de frases para implementar los distintos modelos, los cuales los hemos descargado de internet.

Metodología y Modelo

- **Primera fase: datasets**

Hay dos datasets principales, uno para palabras sueltas y el otro para las frases, los cuales se han seleccionado de internet. El de palabras sueltas se basa en un archivo con diversas carpetas cuyo nombre es la palabra y su contenido son los audios de la misma. Por otra parte, el dataset de frases consiste en una caperta con audios y un archivo con extensión .csv con las

frases las cuales representan los audios contenidos en la carpeta.

Los audios se pasan a espectrogramas para el tratamiento de los mismo dentro del modelo como si fueran imágenes y, posteriormente se introducen al modelo en pares de texto y su representación gráfica en espectrograma.

- **Segunda fase: Estructura**

La estructura del modelo tiene diversas partes las cuales se estudiarán a continuación:

En primer lugar, el codificador de texto (`TextEncoder`) es el responsable de transformar el texto de entrada en una representación numérica. Este módulo utiliza una capa de incrustación (`Embedding`) que asigna a cada palabra o índice del vocabulario un vector en un espacio de dimensiones específicas. La salida de esta capa es un tensor de forma (`longitud`, `embed_dim`), donde `embed_dim` corresponde a la dimensión del espacio de incrustaciones.

A continuación, el decodificador de espectrogramas (`SpectrogramDecoder`) toma la representación codificada del texto y la transforma en un espectrograma mediante una red convolucional profunda. Esta red está compuesta por múltiples capas `Conv2D` con activación ReLU y normalización por lotes (`BatchNormalization`), lo que permite estabilizar el entrenamiento y mejorar la convergencia. Además, se emplean capas de regularización mediante `Dropout` y reducción dimensional con `MaxPooling2D`. Al final de esta red, una capa densa ajusta las dimensiones de salida para que coincidan con las requeridas por el espectrograma deseado.

Para integrar el procesamiento de texto y la generación de espectrogramas, se introduce el modelo Texto a Espectrograma (`TextToSpectrogram`), que combina el `TextEncoder` y el `SpectrogramDecoder`. Este modelo toma como entrada una secuencia de índices que representa el texto, lo codifica en una representación densa mediante el `TextEncoder` y, posteriormente, la decodifica en un espectrograma a través del `SpectrogramDecoder`. Así, este módulo actúa como un puente entre las representaciones textuales y las gráficas.

En el flujo del modelo, antes de enviar las representaciones del texto al decodificador, se aplica un bloque transformador (`TransformerBlock`) que enriquece la representación generada por el `TextEncoder`. Este bloque incorpora una capa de atención multi-cabezal (`MultiHeadAttention`), que permite capturar relaciones contextuales entre las palabras del texto. Además,

el bloque contiene una red de proyección feed-forward con activación `ReLU`, capas de normalización (`LayerNormalization`) y conexiones residuales, que ayudan a mantener la información original del texto al mismo tiempo que mejora su representación contextual. Las capas de `Dropout` en este bloque también contribuyen a evitar el sobreajuste.

Finalmente, el modelo completo se implementa mediante la función `create_transformer_model`, que integra todos los módulos descritos. En este proceso, el texto de entrada, representado como una secuencia de índices, pasa primero por el `TextEncoder` para obtener una representación inicial. Luego, esta representación se procesa mediante el `TransformerBlock`, lo que permite añadir contexto a las palabras o frases codificadas. Posteriormente, el resultado se expande en una dimensión adicional y se pasa al `SpectrogramDecoder`, que genera un espectrograma plano. Para garantizar que la salida tenga la forma deseada, se incluye una capa de ajuste (`Reshape`), que organiza los datos en la estructura tridimensional requerida.

En resumen, este modelo combina lo mejor de los transformadores y las redes convolucionales profundas. El uso del TextEncoder asegura una representación eficaz del texto, mientras que el TransformerBlock añade un nivel de sofisticación contextual. Por su parte, el SpectrogramDecoder convierte esta información enriquecida en un espectrograma. Ambos modelos siguen esta misma estructura, pero se diferencian en las dimensiones de entrada y salida, las cuales son diferentes dependiendo de si son frases o palabras individuales.

- **Ultima fase: Entrenamiento**

El entrenamiento comienza con la definición de parametros esenciales, por ejemplo el tamaño del texto, las dimensiones de las capas. Posteriormente se compila el modelo con un optimizador y una función de pérdida y el modelo procesa los datos de entrada (texto y espectrograma). Con cada iteración, el modelo realiza una propagación hacia delante pasando el texto a espectograma con capas de codificación, bloques transformadores y decodificadores. Se calcula la pérdida comparando el espectro generado con el real utilizando la función `MAE` y se realiza una propagación hacia atrás para ajustar los pesos del modelo. El modelo se evalúa periódicamente utilizando datos de validación para detectar posibles problemas que puedan surgir como por ejemplo el sobreajuste. Este proceso se repite constantemente hasta que se completen las `epochs` o vueltas. Finalmente, los resultados son mostrados través de diversos gráficos donde se pueden observar cómo ha ido evolucionando el modelo.

Resultados

Después del entrenamiento, en ambos modelos obtenemos los espectogramas del texto que le pasamos. En las frases, podemos observar que los espectrogramas, aunque se aprecia que empiezan a asemejarse a los reales, todavía les falta por mejorar. Por otro lado, en las palabras individuales, una vez obtenidos los espectrogramas, obtenemos el audio correspondiente de dicha representación y, vemos que aunque tienen bastante ruido, podemos apreciar lo que dicen, reconociendo las palabras.

Estos audios pueden ser obtenidos en la carpeta AudiosModelo, dentro de la carpeta Código en el repositorio de GitHub de la práctica.

Conclusiones

En este proyecto, hemos implementado con éxito un modelo basado en Transformers capaz de convertir texto en espectrogramas, los cuales pueden ser reconstruidos en audio.

Los resultados obtenidos reflejan un desempeño aceptable al trabajar con palabras individuales, mientras que en frases completas se observan aún ciertas limitaciones en la calidad de los espectrogramas generados. Estas restricciones se deben principalmente a los recursos computacionales disponibles, que no permiten entrenar modelos más complejos debido a la memoria limitada con la que trabajamos.

Con acceso a equipos más potentes y dedicados al entrenamiento de modelos, sería posible implementar arquitecturas más sofisticadas, capaces de manejar la complejidad de los espectrogramas y capturar con mayor precisión las características del audio. A pesar de estas limitaciones, hemos demostrado la viabilidad de nuestra metodología, logrando que los audios generados sean reconocibles, lo que valida la eficacia de la arquitectura propuesta.

Bibliografía

- Tacotron 2 (NVIDIA GitHub Repository)
- WaveGlow (NVIDIA GitHub Repository)
- WaveGlow: A Flow-based Generative Network for Speech Synthesis (arXiv Paper)
- Building Large Language Models from the Ground Up (GitHub Repository)
- WaveTerm (GitHub Repository)
- ViT Image Retrieval (GitHub Repository)
- The Annotated Transformer (Harvard NLP)
- LLMs from the Ground Up Workshop (Lightning AI)
- Text-to-Image Generation (PapersWithCode)
- Generative Adversarial Text-to-Image Synthesis (arXiv 1605.05396)
- Text-to-Image Using GAN (GitHub Repository)
- GigaGAN: Large-scale GAN for Text-to-Image Synthesis
- GigaGAN: Scaling up GANs for Text-to-Image Synthesis (arXiv 2303.07909)
- Hugging Face Diffusers: Stable Diffusion Text-to-Image Pipeline Documentation