

# Documentación del Proyecto: Implementación del Modelo Whisper de OpenAI en Delphi

## Introducción

### ¿Qué es Whisper de OpenAI?

Whisper es un sistema avanzado de reconocimiento automático del habla (ASR, por sus siglas en inglés) desarrollado por OpenAI. Este modelo está diseñado para transcribir audio hablado en texto escrito con alta precisión y eficiencia, incluso en condiciones desafiantes como ruido de fondo, múltiples hablantes y diferentes acentos y dialectos. Whisper es multilingüe y puede manejar varios idiomas, lo que lo hace muy versátil para aplicaciones globales.

### ¿Para qué sirve Whisper?

Whisper se utiliza en una variedad de aplicaciones donde se requiere la transcripción y el procesamiento de audio, incluyendo:

- **Subtitulación automática de videos:** Facilita la generación de subtítulos precisos para contenido de video, mejorando la accesibilidad y la comprensión.
- **Asistentes virtuales y chatbots:** Permite a los asistentes virtuales y chatbots entender y responder a comandos de voz de los usuarios.
- **Transcripción de reuniones y conferencias:** Automatiza la transcripción de discusiones y presentaciones, ayudando en la documentación y el análisis posterior.
- **Mejora de la accesibilidad:** Ayuda a las personas con discapacidades auditivas al proporcionar transcripciones precisas de audio.
- **Análisis de contenido de audio:** Facilita el análisis de contenido hablado para aplicaciones en investigación, marketing y otros campos.

## Funcionalidades de Whisper en este Proyecto

En este proyecto, implementamos las siguientes funcionalidades del modelo Whisper utilizando Delphi:

- **Transcripción de audio:** Convierte audio hablado en texto escrito.
- **Traducción de audio:** Traduce el contenido de audio de un idioma a otro.
- **Síntesis de voz:** Genera audio hablado a partir de texto escrito.

Estas funcionalidades permiten la creación de aplicaciones potentes que pueden interactuar de manera eficiente con los usuarios mediante el uso del habla.

## Descripción del Componente `TAiAudio`

El componente `TAiAudio` es una clase Delphi diseñada para interactuar con el modelo Whisper de OpenAI. Proporciona métodos para la transcripción, traducción y síntesis de voz a partir de archivos de audio. A continuación, se presenta una descripción detallada de su estructura, propiedades y métodos.

### Propiedades del Componente

Las propiedades del componente `TAiAudio` permiten configurar diversos parámetros necesarios para interactuar con la API de OpenAI. Estas propiedades son:

- **ApiKey:** La clave de API utilizada para autenticar las solicitudes a OpenAI.
- **Url:** La URL base de la API de OpenAI.
- **Model:** El modelo de transcripción o síntesis de voz utilizado.
- **Voice:** La voz utilizada para la síntesis de voz.
- **Format:** El formato de salida del archivo de audio (por ejemplo, mp3, opus, aac, flac, pcm).
- **Lenguaje:** El idioma del audio.
- **Speed:** La velocidad del habla en la síntesis de voz.
- **Temperature:** La temperatura utilizada para la generación de texto, influenciando la creatividad del modelo.
- **ResponseFormat:** El formato de la respuesta (por ejemplo, json, text, srt, verbose\_json, vtt).
- **Quality:** La calidad del modelo de síntesis de voz (por ejemplo, tts-1, tts-1-hd).

### Métodos del Componente

El componente `TAiAudio` proporciona varios métodos para interactuar con la API de OpenAI:

- **Speech:** Genera un archivo de audio a partir de un texto.
- **Transcription:** Transcribe el contenido de un archivo de audio a texto.
- **Translation:** Traduce el contenido de un archivo de audio de un idioma a otro.

### Constructor y Destructor

- **Create:** Inicializa una instancia del componente `TAiAudio`, configurando las propiedades por defecto.
- **Destroy:** Libera los recursos utilizados por la instancia del componente.

### Métodos Auxiliares

- **IsValidExtension:** Verifica si una extensión de archivo es válida para procesamiento.

- **ConvertFileFormat:** Convierte el formato de un archivo de audio utilizando ffmpeg.

---

## Ejemplo de Uso

A continuación, se presenta un ejemplo básico de cómo utilizar el componente `TAiAudio` para transcribir un archivo de audio:

```
var
  AiAudio: TAiAudio;
  TranscriptionText: String;
  AudioStream: TMemoryStream;
begin
  AiAudio := TAiAudio.Create(nil);
  try
    AiAudio.ApiKey := 'tu_clave_de_api';
    AudioStream := TMemoryStream.Create;
    try
      AudioStream.LoadFromFile('ruta/a/tu/audio/file.mp3');
      TranscriptionText := AiAudio.Transcription(AudioStream, 'file.mp3', 'Este es un
prompt de ejemplo');
      ShowMessage(TranscriptionText);
    finally
      AudioStream.Free;
    end;
  finally
    AiAudio.Free;
  end;
end;
```

## Detalle de Propiedades del Componente `TAiAudio`

### Propiedades

#### 1. **ApiKey:**

- **Descripción:** Clave de API utilizada para autenticar las solicitudes a OpenAI.
- **Tipo:** String
- **Uso:**

```
AiAudio.ApiKey := 'tu_clave_de_api';
```

#### 2. **Url:**

- **Descripción:** URL base de la API de OpenAI.
- **Tipo:** String
- **Uso:**

```
AiAudio.Url := 'https://api.openai.com/v1/';
```

### 3. Model:

- **Descripción:** Modelo de transcripción o síntesis de voz utilizado.
- **Tipo:** String
- **Valores:** whisper-1 para transcripciones y tts para texto a voz

'tts-1', 'tts-1-hd' si desea velocidad es tts-1 y para mayor calidad tts-2-hd

- **Uso:**

```
AiAudio.Model := 'tts-1';
```

### 4. Voice:

- **Descripción:** Voz utilizada para la síntesis de voz.
- **Tipo:** String
- **Valores:** 'alloy', 'echo', 'fable', 'onyx', 'nova', 'shimmer'
- **Uso:**

```
AiAudio.Voice := 'nova';
```

### 5. Format:

- **Descripción:** Formato de salida del archivo de audio.
- **Tipo:** String
- **Valores:** 'mp3', 'opus', 'aac', 'flac', 'pcm'
- **Uso:**

```
AiAudio.Format := 'mp3';
```

### 6. Lenguaje:

- **Descripción:** Idioma del audio.
- **Tipo:** String
- **Uso:**

```
AiAudio.Lenguaje := 'es';
```

### 7. Speed:

- **Descripción:** Velocidad del habla en la síntesis de voz.
- **Tipo:** Single entre 0.25 a 4.0 por defecto es 1
- **Uso:**

```
AiAudio.Speed := 1.0;
```

### 8. Temperature:

- **Descripción:** Temperatura utilizada para la generación de texto, influenciando la creatividad del modelo.
- **Tipo:** Single entre 0 y 1, 0 es más estricto y 1 más random

- **Uso:**

```
AiAudio.Temperature := 0.7;
```

#### 9. **ResponseFormat:**

- **Descripción:** Formato de la respuesta.
- **Tipo:** String
- **Valores:** 'json', 'text', 'srt', 'verbose\_json', 'vtt'
- **Uso:**

```
AiAudio.ResponseFormat := 'text';
```

#### 10. **Quality:**

- **Descripción:** Calidad del modelo de síntesis de voz.
- **Tipo:** String
- **Valores:** 'tts-1', 'tts-1-hd'
- **Uso:**

```
AiAudio.Quality := 'tts-1';
```

## Detalle de Métodos del Componente **TAiAudio**

### Speech

- **Descripción:** Genera un archivo de audio a partir de un texto.
- **Parámetros:**
  - **aText:** El texto que se desea convertir en audio.
  - **aVoice** (opcional): La voz utilizada para la síntesis de voz.
- **Retorno:** TMemoryStream con el archivo de audio generado.
- **Ejemplo:**

```
var
  AudioStream: TMemoryStream;
begin
  AudioStream := AiAudio.Speech('Hola, este es un ejemplo de síntesis de voz.');
```

```
  try
    AudioStream.SaveToFile('output.mp3');
```

```
  finally
    AudioStream.Free;
  end;
end;
```

### Transcription

- **Descripción:** Transcribe el contenido de un archivo de audio a texto.
- **Parámetros:**
  - **aStream:** El flujo de memoria que contiene el archivo de audio.
  - **aFileName:** El nombre del archivo de audio.
  - **aPrompt:** El prompt para el modelo de transcripción.

- **Retorno:** String con el texto transcrito.
- **Ejemplo:**

```
var
  TranscriptionText: String;
begin
  TranscriptionText := AiAudio.Transcription(AudioStream, 'file.mp3', 'Este es un prompt
de ejemplo');
  ShowMessage(TranscriptionText);
end;
```

### Translation

- **Descripción:** Traduce el contenido de un archivo de audio de un idioma a otro.
- **Parámetros:**
  - aStream: El flujo de memoria que contiene el archivo de audio.
  - aFileName: El nombre del archivo de audio.
  - aPrompt: El prompt para el modelo de traducción.
- **Retorno:** String con el texto traducido.
- **Ejemplo:**

```
var
  TranslationText: String;
begin
  TranslationText := AiAudio.Translation(AudioStream, 'file.mp3', 'Este es un prompt de
ejemplo');
  ShowMessage(TranslationText);
end;
```

### Falta por Implementar

1. La funcionalidad de Streaming en tts falta por implementar.

# RAG y Bases de Datos Vectoriales

---

## Retrieval-Augmented Generation (RAG)

RAG es una técnica que combina la recuperación de información con la generación de texto:

1. Recuperación: Busca información relevante en una base de conocimientos.
2. Generación: Utiliza la información recuperada para generar respuestas más precisas y contextualizadas.

Beneficios:

- Mejora la precisión de las respuestas
- Reduce las alucinaciones en modelos de lenguaje
- Permite actualizar el conocimiento sin reentrenar el modelo

## Bases de Datos Vectoriales

Son sistemas de almacenamiento optimizados para datos en formato de vectores:

- Almacenan embeddings (representaciones numéricas) de datos
- Permiten búsquedas eficientes basadas en similitud

Ejemplos: Pinecone, Faiss, Milvus

## Construcción de Vectores

Los vectores (embeddings) se construyen mediante:

1. Tokenización: Divide el texto en unidades más pequeñas (tokens)
2. Embedding: Convierte los tokens en vectores numéricos
3. Modelos de embedding: Utilizan redes neuronales (ej. BERT, Word2Vec)

Proceso:

Texto -> Tokenización -> Modelo de Embedding -> Vector Numérico

## Consulta de Vectores

La consulta en bases de datos vectoriales implica:

1. Vectorización de la consulta: Convierte la consulta en un vector
2. Búsqueda de similitud: Encuentra vectores cercanos en el espacio vectorial
3. Ranking: Ordena los resultados por similitud

Métodos comunes:

- Coseno de similitud
- Distancia euclidiana
- Approximate Nearest Neighbors (ANN)

Proceso típico:

Consulta -> Vectorización -> Búsqueda de Similitud -> Ranking -> Resultados

## Configurar demo

Se lee los parámetros de conexión al modelo desde un archivo config.json que ponemos en la carpeta de la aplicación. Se asignan a los componentes TAIOpenChat y TAIEmbeddings.

El componente TAIragChat conecta TAIOpenChat y TAIDataVec.

El componente TAIDataVec conecta con TAIEmbeddings.

## Funcionamiento

Con el RAG buscamos ofrecer a la IA un contexto con datos propios de forma que obtengamos respuestas más precisas. Si al abrir el programa vamos a la segunda pestaña de Consultas RAG y escribimos una consulta sobre unos datos específicos nuestros, lo más normal es que la IA no sepa qué responder

o de resultados inexactos. El contexto de RAG lo podemos proporcionar cargando un texto plano o un JSON, al cargarlo se hace una petición a OpenAI que genera los vectores de estos archivos. Estos vectores los podemos guardar en memoria o en una base de datos preparada para hacerlo, por ejemplo PostgreSQL con la extensión PgVector.

Cuando hacemos una consulta a OpenAI, realizamos una consulta previa a la base de datos de vectores

para obtener los vectores más próximos a nuestro prompt y que añadimos a la consulta como contexto.



