SGCWEBSOCKETS · TECHNICAL DOCUMENT

AI Embeddings

Embeddings generator and consumer for vector search — text-embedding-ada-002, text-embedding-3-large and compatible providers.

eSeGeCe

WWW.ESEGECE.COM

Overview

Embeddings are a way to represent words, phrases, or even other types of data, like images or audio, in a numerical form.

At a glance

COMPONENT CLASS `TsgcAIOpenAIEmbeddings`	STANDARDS / SPEC **OpenAI Embeddings API**
TRANSPORTS **TCP, TLS**	PLATFORMS **Windows, macOS, Linux, iOS, Android**
FRAMEWORKS **VCL, FireMonkey, Lazarus / FPC**	EDITION **Standard / Professional / Enterprise**

Features

- Native Delphi implementation with full ANSI/Unicode support.

Technical specification

Standards & specs	OpenAI Embeddings API · OpenAI Embeddings guide
Component class	`TsgcAIOpenAIEmbeddings` (unit `sgcAI_Embeddings`)
Frameworks	VCL, FireMonkey, Lazarus / FPC
Platforms	Windows, macOS, Linux, iOS, Android

Main properties

The principal published / public properties used to configure and drive the component. Consult the online help for the full list.

`OpenAIOptions`	Connection settings for the OpenAI API (API key, organization, logging and retry).
`EmbeddingsOptions`	Parameters used when generating embeddings (model, chunk size, user and throttling).
`Database`	Vector database component used to persist the generated embeddings.
`Version`	Read-only string with the current version of the sgcWebSockets library.

Main methods

The principal public methods exposed by the component.

`CreateEmbeddings()`	Splits the input text into chunks and generates an embedding for each chunk.
`CreateEmbeddingsFromFile()`	Loads a text file and generates embeddings from its contents.

Public events

The component exposes the following published events; consult the online help for full event-handler signatures.

`OnAfterCreateEmbedding`	Fired after each chunk is processed, exposing the OpenAI request and response.

`OnBeforeCreateEmbedding`	TsgcAIOpenAIEmbeddings › Events › OnBeforeCreateEmbedding
`OnCreateEmbeddingsProgress`	TsgcAIOpenAIEmbeddings › Events › OnCreateEmbeddingsProgress
`OnCreateEmbeddingsStart`	TsgcAIOpenAIEmbeddings › Events › OnCreateEmbeddingsStart
`OnCreateEmbeddingsStop`	property OnCreateEmbeddingsStop: TsgcOpenAICreateEmbeddingsStop; // TsgcOpenAICreateEmbeddingsStop = procedure(Sender: TObject) of object __property TsgcOpenAICreateEmbeddingsStop OnCreateEmbeddingsSt...

Quick Start

Drop the component on a form, configure the properties below and activate it. The snippet that follows shows the typical **Embeddings | ChatBot** configuration sourced from the online help.

> **About this scenario.** Once we've converted all our data to vectors, we can start to build our own model. The idea behind it is very simple: every time we ask the bot, first we convert the question to a vector, then we search our database for which vector is most similar to the question, and finally we use the most similar data and add it as context.

Delphi (VCL / FireMonkey)

```
procedure AskToChatGPT(const aQuestion: string);
var
 oChatBot: TsgcAIOpenAIChatBot;
 oEmbeddings: TsgcAIOpenAIEmbeddings;
 oFile: TsgcAIDatabaseVectorFile;
 vContext: string;
begin
 oChatBot := TsgcAIOpenAIChatBot.Create(nil);
 Try
  oChatBot.OpenAIOptions.ApiKey := '<your api key>';
  oEmbeddings := TsgcAIOpenAIEmbeddings.Create(nil);
  Try
   oChatBot.Embeddings := oEmbeddings;
   oFile := TsgcAIDatabaseVectorFile.Create(nil);
   Try
    oEmbeddings.Database := oFile;
    vContext := oChatBot.GetEmbedding(aQuestion);
    oChatBot.ChatAsUser('Answer the question based on the context below.\n\nContext:\n' +
     vContext + '\nQuestion:' + aQuestion + '\nAnswer:');
   Finally
    oFile.Free;
   End;
  Finally
   oEmbeddings.Free;
  End;
 Finally
  FreeAndNil(oDialog);
 End;
end;</code>
<code class="delphi">
```

C++ Builder

```
void AskToChatGPT(const std::string& aQuestion)
{
  TsgcAIOpenAIChatBot* oChatBot = new TsgcAIOpenAIChatBot(NULL);
  try
  {
    oChatBot->OpenAIOptions->ApiKey = "<your api key>";
    TsgcAIOpenAIEmbeddings* oEmbeddings = new TsgcAIOpenAIEmbeddings(NULL);
    try
    {
      oChatBot->Embeddings = oEmbeddings;
      TsgcAIDatabaseVectorFile* oFile = new TsgcAIDatabaseVectorFile(NULL);
      try
      {
        oEmbeddings->Database = oFile;
        std::string vContext = oChatBot->GetEmbedding(aQuestion);
        std::string message = "Answer the question based on the context below.\n\nContext:\n" +
                  vContext + "\nQuestion:" + aQuestion + "\nAnswer:";
        oChatBot->ChatAsUser(message.c_str());
      }
      __finally
      {
        delete oFile;
      }
    }
    __finally
    {
      delete oEmbeddings;
    }
  }
  __finally
  {
    delete oChatBot;
  }
}
```

.NET (C#)

```
public void AskToChatGPT(string aQuestion)
{
  using (TsgcAIOpenAIChatBot oChatBot = new TsgcAIOpenAIChatBot())
  {
    oChatBot.OpenAIOptions.ApiKey = "<your api key>";
    using (TsgcAIOpenAIEmbeddings oEmbeddings = new TsgcAIOpenAIEmbeddings())
    {
      oChatBot.Embeddings = oEmbeddings;
      using (TsgcAIDatabaseVectorFile oFile = new TsgcAIDatabaseVectorFile())
      {
        oEmbeddings.Database = oFile;
        string vContext = oChatBot.GetEmbedding(aQuestion);
        string message = "Answer the question based on the context below.\n\nContext:\n" +
                vContext + "\nQuestion:" + aQuestion + "\nAnswer:";
        oChatBot.ChatAsUser(message);
      }
    }
  }
}
```

Common scenarios

The following scenarios are lifted verbatim from the online help. Each shows the configuration and method calls needed to drive the component through a specific real-world flow.

1 · Embeddings | Create Vectors

To use the embeddings, first we must convert our data to vectors.

Delphi (VCL / FireMonkey)

```
procedure ConvertFileToVector;
var
 oDialog: TOpenDialog;
 oEmbeddings: TsgcAIOpenAIEmbeddings;
 oFile: TsgcAIDatabaseVectorFile;
begin
 oDialog := TOpenDialog.Create(nil);
 Try
  oDialog.Filter := 'TXT Files|*.txt';
  if oDialog.Execute then
  begin
   oEmbeddings := TsgcAIOpenAIEmbeddings.Create(nil);
   Try
    oFile := TsgcAIDatabaseVectorFile.Create(nil);
    Try
     oEmbeddings.Database := oFile;
     oEmbeddings.OpenAIOptions.ApiKey := '<your api key>';
     oEmbeddings.CreateEmbeddingsFromFile(oDialog.FileName);
    Finally
     oFile.Free;
    End;
   Finally
    oEmbeddings.Free;
   End;
  end;
 Finally
  FreeAndNil(oDialog);
 End;
end;
```

C++ Builder

```
void ConvertFileToVector()
{
  TOpenDialog* oDialog = new TOpenDialog(NULL);
  try
  {
    oDialog->Filter = "TXT Files|*.txt";
    if (oDialog->Execute())
    {
      TsgcAIOpenAIEmbeddings* oEmbeddings = new TsgcAIOpenAIEmbeddings(NULL);
      try
      {
        TsgcAIDatabaseVectorFile* oFile = new TsgcAIDatabaseVectorFile(NULL);
        try
        {
          oEmbeddings->Database = oFile;
          oEmbeddings->OpenAIOptions->ApiKey = "<your api key>";
          oEmbeddings->CreateEmbeddingsFromFile(oDialog->FileName);
        }
        __finally
        {
          delete oFile;
        }
      }
      __finally
      {
        delete oEmbeddings;
      }
    }
  }
  __finally
  {
    delete oDialog;
  }
}
```

.NET (C#)

```
public void ConvertFileToVector()
{
  using (OpenFileDialog oDialog = new OpenFileDialog())
  {
    oDialog.Filter = "TXT Files|*.txt";
    if (oDialog.ShowDialog() == DialogResult.OK)
    {
      using (TsgcAIOpenAIEmbeddings oEmbeddings = new TsgcAIOpenAIEmbeddings())
      {
        using (TsgcAIDatabaseVectorFile oFile = new TsgcAIDatabaseVectorFile())
        {
          oEmbeddings.Database = oFile;
          oEmbeddings.OpenAIOptions.ApiKey = "<your api key>";
          oEmbeddings.CreateEmbeddingsFromFile(oDialog.FileName);
        }
      }
    }
  }
}
```

Sources used to build this document

Every external claim links back to a primary source. The online-help references decode the canonical deep-link the company maintains for this component.

Primary standard / spec — OpenAI Embeddings API	platform.openai.com/docs/api-reference/embeddings
Primary standard / spec — OpenAI Embeddings guide	platform.openai.com/docs/guides/embeddings
Online help — component page	www.esegece.com/help/sgcWebSockets/Components/AI/Applications/Embeddings/TsgcAIOpenAIEmbeddings.htm
Delphi demo project (in the sgcWebSockets package)	`Demos\15.AI\10.Vector_Database\01.Pinecone`
Component page	www.esegece.com/products/websockets/ai/embeddings/
Product page	www.esegece.com/products/websockets/

> **Document scope.** This document covers the publicly-documented surface of the AI Embeddings component shipped with sgcWebSockets. For full property, method and event reference consult the online help linked above.