# Introduction

This workshop consists of 3 sections.

* LM Studio: Selecting and chatting with a local model  
  This workshop does not require much technical skills. Can be done independently.
* Code completion in VSCode using a local model hosted with Ollama  
  This workshop is mostly interesting to developers. It does not require the LM Studio part of the workshop and can be done independently.
* Implementing sementic search  
  This depends on environment set-up as performed in the code completion part of the workshop and is also mainly focused on developers.

# LM Studio. Selecting and chatting with a local model

Download and install LM Studio: <https://lmstudio.ai/>

It is recommended to install LM Studio not in a virtualized environment. I encountered issues when running in a Windows Sandbox environment.

Download a model. You can choose one from the main screen, for example Llama3 8B Instruct. Once a download is started you can monitor progress:

Afbeelding met tekst, schermopname, software, Multimediasoftware

Automatisch gegenereerde beschrijving

In the search screen you can see how many model downloads there have been for a specific model (more is usually better) and whether a model is likely to run on your computer.

Afbeelding met tekst, schermopname, software, Multimediasoftware

Automatisch gegenereerde beschrijving

Once a model is downloaded, you can first load it and then chat with it using the Chat TAB

Afbeelding met schermopname, tekst, software, Multimediasoftware

Automatisch gegenereerde beschrijving

Before using the model check the preset to the left to fit the model. Mostly i twill automatically be set correctly;

Afbeelding met tekst, schermopname, software, Multimediasoftware

Automatisch gegenereerde beschrijving

In case of errors you can disable the usage of GPU offload as that causes issues sometimes;

Afbeelding met tekst, schermopname, software, Computerpictogram

Automatisch gegenereerde beschrijving

Suggestions for models to run:

* gemma-2-27b-it-IQ4\_XS.gguf for 32Gb RAM  
  https://huggingface.co/bartowski/gemma-2-27b-it-GGUF
* meta-llama-3-8b-instruct.Q4\_K\_M.gguf for 16Gb or 8Gb  
  https://huggingface.co/SanctumAI/Meta-Llama-3-8B-Instruct-GGUF

Please note the following:

* 7B models are small and not ‘smart’. Can only be used for simple tasks. Also they can be used when context is supplied as part of the prompt, for example in RAG solutions.  
  70B models are quite smart and have a lot of knowledge. They require around 64Gb RAM to run in quantized form.
* The model does not have an out of the box way to access the internet such as is usual with ChatGPT and Copilot. The model runs in a sandbox and knowledge is pretrained.

LM Studio can host models and provide an OpenAI compliant API to program against, similar to ollama. LM Studio is a closed source product.

# Code completion in VSCode using a local model hosted with ollama

## Preparation

You can choose to install tools locally or use a Windows Sandbox environment which already has several things prepared.

### Sandbox

Enable Windows Sandbox

More information on: <https://learn.microsoft.com/en-us/windows/security/application-security/application-isolation/windows-sandbox/windows-sandbox-overview>

Start Powershell with Administrative privileges

Execute the following

Enable-WindowsOptionalFeature -FeatureName "Containers-DisposableClientVM" -All -Online

Install Git: https://git-scm.com/downloads

c:

cd \

git clone <https://github.com/AMISConclusion/Workshop_LocalLLM> sandbox

You can choose another path but in that case you will need to update vscode.wsb. This is a simple XML file. You might also need to tweak memory usage when your laptop has less than 32Gb of RAM available.

Execute vscode.wsb

This will create a sandbox environment with VSCode started with the workspace loaded and required files for this workshop in the Downloads folder.

Continue with ‘Install models’

### Locally

Check whether you do not already have the following tools. If not, install them.

Install VSCode: <https://code.visualstudio.com/>

Install Git: https://git-scm.com/downloads

Create a folder for your repositories. Start a command prompt in that folder and do:

git clone <https://github.com/AMISConclusion/Workshop_LocalLLM> sandbox

Open the folder Projects\demo in VSCode and trust the authors

Afbeelding met tekst, schermopname, software, Multimediasoftware

Automatisch gegenereerde beschrijving

Install Python.

Windows:

This can be done in a command prompt window with Administrator privileges by executing the following (this will not work in Powershell since it uses a different curl with other arguments).

curl -L https://www.python.org/ftp/python/3.12.4/python-3.12.4-amd64.exe --output python.exe

python.exe -Wait /PrependPath=1 /quiet InstallAllUsers=1 TargetDir="C:\python" Include\_test=0

Linux RHEL:

sudo yum upgrade python3

Linux DEB (Ubuntu, Debian, Mint)

sudo apt install python3 python3-pip

When on Windows chromadb/onnxruntime requires some libraries tob e installed. Execute the following in a command prompt:

curl -L https://aka.ms/vs/17/release/vs\_BuildTools.exe --output C:\users\WDAGUtilityAccount\Downloads\vs\_BuildTools.exe

C:\users\WDAGUtilityAccount\Downloads\vs\_BuildTools.exe --wait --includeRecommended --quiet --add Microsoft.Component.MSBuild --add Microsoft.VisualStudio.Component.CoreBuildTools --add Microsoft.VisualStudio.Component.VC.CoreBuildTools --add Microsoft.VisualStudio.Component.VC.Tools.x86.x64 --add Microsoft.VisualStudio.Component.VC.Redist.14.Latest --add Microsoft.VisualStudio.Component.VC.CoreIde --add Microsoft.VisualStudio.Component.Windows11SDK.22621 --add Microsoft.VisualStudio.ComponentGroup.NativeDesktop.Core --add Microsoft.VisualStudio.Workload.MSBuildTools --add Microsoft.VisualStudio.Workload.VCTools

Install required Python packages

pip install llama\_index llama-index-embeddings-huggingface chromadb onnxruntime openai

Download and install ollama

Windows:

curl -L "https://ollama.com/download/OllamaSetup.exe" --output OllamaSetup.exe

OllamaSetup.exe /verysilent

Linux:

curl -fsSL https://ollama.com/install.sh | sh

### Install models

Start a command prompt and download local models

For 16Gb RAM

ollama pull deepseek-coder:6.7b-base

Afbeelding met tekst, software, Lettertype, Multimediasoftware

Automatisch gegenereerde beschrijving

### Install required extensions

When using the Windows Sandbox, you can wait until VSCode pops up. Otherwise start VSCode and open the checked-out Projects/demo directory.

A prompt in VSCode will pop up asking to install recommended extensions. Click Install.

Afbeelding met tekst, schermopname, software, Multimediasoftware

Automatisch gegenereerde beschrijving

### Configure VSCode extentions

Configure the Continue plugin

Afbeelding met tekst, schermopname, software

Automatisch gegenereerde beschrijving

Choose Local model

Afbeelding met tekst, schermopname, software, Lettertype

Automatisch gegenereerde beschrijving

Select the downloaded model: Deepseek Coder 6.7B. Click continue a couple of times to exit the wizard and update the Continue extension configuration.

Afbeelding met tekst, software, Multimediasoftware, Grafische software

Automatisch gegenereerde beschrijving

{

"models": [

{

"title": "Ollama",

"provider": "ollama",

"model": "deepseek-coder:6.7b-base"

}

],

"customCommands": [

{

"name": "test",

"prompt": "{{{ input }}}\n\nWrite a comprehensive set of unit tests for the selected code. It should setup, run tests that check for correctness including important edge cases, and teardown. Ensure that the tests are complete and sophisticated. Give the tests just as chat output, don't edit any file.",

"description": "Write unit tests for highlighted code"

}

],

"tabAutocompleteModel": {

"title": "Ollama",

"provider": "ollama",

"model": "deepseek-coder:6.7b-base"

},

"allowAnonymousTelemetry": true,

"embeddingsProvider": {

"provider": "transformers.js"

}

}

## Code completion and content generation

Open the test.py Python file

Afbeelding met schermopname, tekst, software, Multimediasoftware

Automatisch gegenereerde beschrijving

Press CTRL-I and ask the question: create python code to open a webbrowser and go to www.google.com

Afbeelding met schermopname, Multimediasoftware, tekst, software

Automatisch gegenereerde beschrijving

This generated:

import webbrowser

webbrowser.open('https://www.google.com')

Accept the generated code and run it

Afbeelding met schermopname, tekst, software, Computerpictogram

Automatisch gegenereerde beschrijving

Try out the TAB completion by typing on a second line: webbrowser. A second line could be something like:

webbrowser.open\_new\_tab('https://www.nu.nl')

Try out the new code.

Select some tekst and press CTRL-I. Ask to generate documentation.

Afbeelding met tekst, schermopname, software, Multimediasoftware

Automatisch gegenereerde beschrijving

Accept the suggested change

# Implementing simple semantic search using a local model

When using the sandbox, the used ollama model should already be available. Otherwise, first do the following in a command prompt:

ollama pull dolphin-mistral:7b

Open local\_rag\_example.py (https://github.com/AMISConclusion/Workshop\_LocalLLM/blob/main/Projects/demo/local\_rag\_example.py) and execute it.Afbeelding met tekst, software, schermopname, Multimediasoftware

Automatisch gegenereerde beschrijving

Check the code to understand what it does.

* Local JSON documents are vectorized using the embedding model mixedbread-ai/mxbai-embed-large-v1 (which is first downloaded)
* The Documents and vectors are stored in a vector database: ChromaDB
* A semantic query is performed on the vectorDB using the initial question. The top 5 documents which semantically match the query will be returned
* The documents as context in addition to the initial question are sent to the dolphin-mistral:7b model which is hosted locally by ollama. The script uses the OpenAI compliant API offered by the ollama instance. The answer is returned and displayed in the console.

You can improve the previous code to;

* Generate additional metadata using the LLM to improve the semantic search.   
  For example generate summaries or titles based in the content
* Evaluate retrieved documents on relevance. Semantic search results based on a question can still mean the retrieved document does not help in ansering the original question.
* Use a semantic splitter to chop up documents into smaller parts. A semantic splitter chops documents up in a way they do not loose to much of their meaning (such as for example by chopping up mid sentence)

Look at the following sample code: <https://github.com/MaartenSmeets/autogen_play/blob/main/llamaindex_rag_local_embedding.py>

Consider the impact which growing contexts of LLMs have on RAG implementations.