|  |  |
| --- | --- |
| Knowledge BAse assistent  7.11.2023 | Abstract  Řešení RestAPI služby, která implementuje funkcionalitu pro potřebu chatbotů založených na externích zdrojích dat. Služba dokáže podporovat více chatbotů založených na různých zdrojích dat. Součástí řešení je i příprava externích dat pro potřeby této služby.  Vladimír Dědourek, Michal Stoklasa |

Obsah

[1 Knowledge Base Assistent 2](#_Toc153213618)

[1.1 Cíle řešení 2](#_Toc153213619)

[2 Browser 2](#_Toc153213620)

[3 WebExtLoader 3](#_Toc153213621)

[4 KBAIndex 4](#_Toc153213622)

[4.1 Model embeddings 5](#_Toc153213623)

[4.1.1 Indexování strukturovaného textu 5](#_Toc153213624)

[4.1.2 Protokol zpracování www.multima.cz 5](#_Toc153213625)

[4.1.3 Protokol zpracování asistenta OpenAI 6](#_Toc153213626)

[5 KBAQnA 7](#_Toc153213627)

[6 Evaluace RAG modelu 7](#_Toc153213628)

[7 RestAPI server 11](#_Toc153213629)

[7.1 Dotaz odpověď – qna 13](#_Toc153213630)

[7.2 Načtení parametrů serveru – get\_srv\_par 13](#_Toc153213631)

[7.3 Načtení parametrů projektu – get\_project\_par 14](#_Toc153213632)

[7.4 Nastavení parametrů serveru – set\_srv\_par 14](#_Toc153213633)

[7.5 Nastavení parametrů projektu – set\_project\_par 15](#_Toc153213634)

[8 Konkrétní řešení 16](#_Toc153213635)

[8.1 www.multima.cz 16](#_Toc153213636)

[8.1.1 Zdrojová data 16](#_Toc153213637)

[8.1.2 Indexování 16](#_Toc153213638)

[8.1.3 Dotazování 17](#_Toc153213639)

[8.2 www.mulouny.cz 18](#_Toc153213640)

[8.2.1 Zdrojová data 18](#_Toc153213641)

[8.2.2 Indexování 19](#_Toc153213642)

[8.2.3 Dotazování 21](#_Toc153213643)

[9 Přílohy 22](#_Toc153213644)

# Knowledge Base Assistent

## Cíle řešení

Cílem řešení je vytvořit framework pro tvorbu webových chatbotů založených na externích zdrojíchy dat. Těmito zdroji mou být jak různé typy textových souborů, tak i webové stránky samotné.

Řešení jse složeno ze 2 částí:

1 – Úoha pro indexování z externích zdrojů dat (KBAIndex)

2 – Rest API služba která vytváří odpovědi na základě položených dotazů. Umožňuje vést konverzaci (KBAQnA)

Migrace

<https://github.com/openai/openai-python/discussions/742>

# Browser

Třída pro zpracování webových stránek prostřednictvím knihovny Selenium. Pro zpracování lze zvolit jeden z následujících browserů:

* Edge
* Chrome
* Firefox

class Browser(object):

"""

Class for work with browser (by Selenium)

-----------------------------------------

browser\_type - browser type (Edge, Chrome, Firefox)

headless - If True, the browser will run in headless mode.

"""

def read\_html(self,

url:str,

parser:str = "html.parser",

selected\_selector:str = "",

remove\_selectors: List[str] = [],

remove\_tags: List[str] = [],

exclude\_link\_text: List[str] = [],

) -> bool:

'''

načtení a vygenerování HTML prostřednictvím Javascriptu

provede naplnění (self.url\_domain, self.url\_scheme, self.html\_original, self.last\_modified, self.html, self.soup )

url - url address

parser - parser (html.parser, xml, lxml, lxml-xml, html5lib, html, html5)

encoding - encoding (utf-8, ...)

selected\_selector - if is used, then is selected only this selector (for example "situaces")

remove\_tags - list of removed tags with texts preserved

remove\_selectors - list of removed selectors including texts

exclude\_link\_text - list of excluded link texts

return True, False

'''

def get\_html(self,

formatted:bool=False,

) -> str:

'''

Getting HTML data after read\_html

------------------------------------------------

formatted - True - html is formatted, False - html is wihout formating

return HTML string

'''

def get\_text(self) -> str:

'''

získání textu z HTML po načtení metodou read\_html

'''

def get\_last\_modification(self) -> str:

'''

získání data poslední modifikace stránky po načtení metodou read\_html.

datum je ve tvaru yyyy-mm-dd HH:MM:SS (jestliže existuje)

'''

def get\_links(self) -> list[str]:

'''

Získání všech linků z HTML po načtení metodou read\_html

Scrape all links from the page

'''

# WebExtLoader

Vlastní Langchain loader pro načtení obsahu webových stránek.

class WebExtLoader(BaseLoader):

"""

Extended Web page loader

-----------------------------------------

web\_path - url

parser - parser (html.parser, xml, lxml, lxml-xml, html5lib, html, html5)

selected\_selector - if is used, then is selected only this selector (for example "situaces")

remove\_selectors - list of removed selectors including texts

remove\_tags - list of removed tags with texts preserved

array\_item - array item under selected\_selector (for example "situace")

content\_func - method content\_func(item) for extract text of the item

metadata\_func - method metadata\_func(item) for extract metadata of the item

"""

def load\_json(self) -> str:

"""Load text from the url in web\_path to JSON."""

def load(self) -> List[Document]:

"""Load text from the url(s) in web\_path."""

# KBAIndex

Úloha pro indexování externích zdrojů dat a uložení do vektorové databáze pro potřeby Rest API služby. Funkcionalita je implementovaná ve třídě Pythonu KBAIndex. Funkcionalita je založena na modelu vytvoření embeddings, které jsou uloženy v Qdrant databázi.

## Model embeddings



### Indexování strukturovaného textu

Pro indexování se berou texty z externích zdrojůt. Tyto texty jsou rozděleny na menší segmenty (chunks) tak, aby maximální velikost nepřekročila maximální počet tokenu (max\_tokens).

### Protokol zpracování [www.multima.cz](http://www.multima.cz)

\*\*\* Website Crawling

https://www.multima.cz

https://www.multima.cz/o-nas

[.](https://www.multima.cz/vyvoj-software) . .

Total count: 96 Elapsed time: 111.594 s

\*\*\* Data Loading

./Data/ChatbotDescription.txt

Fetching pages: 0%| | 0/89 [00:00<?, ?it/s]

Fetching pages: 1%|1 | 1/89 [00:02<03:30, 2.39s/it]

. . .

Total number of documents: 144

Elapsed time 91.415 seconds

Total length 1 of texts: 768955

\*\*\* Cleaning whitespace

\*\*\* Cleaning blocks

\*\*\* Cleaning blocks

\*\*\* Cleaning texts

\*\*\* Cleaning whitespace

\*\*\* Cleaning phone number

\*\*\* Cleaning email

Total length 2 of texts: 506452

Elapsed time: 0.552 s

\*\*\* Creating chunks

Created a chunk of size 575, which is longer than the specified 500

Created a chunk of size 501, which is longer than the specified 500

. . .

Total number of chunks: 653

Elapsed time 0.311 seconds

\*\*\* Create embeddings

Elapsed time 23.174 seconds

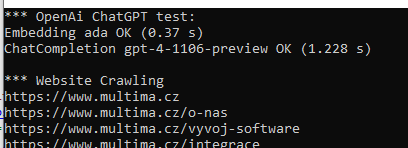
Chroma local collections: ['www.multima.cz']:

Count of chunks: 653

Total length of chunks: 505434

Average length of chunk: 774

### Protokol zpracování asistenta OpenAI



A screen shot of a computer

Description automatically generated

A computer screen shot of a computer program

Description automatically generated

# KBAQnA

Třída Pythonu, která zpřístupňuje služby pro generování odpovědi na základě dotazů.

# Evaluace RAG modelu

Retriever s embeddings (chunk = 500, overlap=50, k = 8)

A black screen with white text

Description automatically generated

Retriever s embeddings (chunk = 667, overlap=67, k = 6)

\*\*\* Creating chunks

Total number of chunks: 1138

Elapsed time 0.672 seconds

\*\*\* Create embeddings

Elapsed time 62.782 seconds

\*\*\* Create BM25

Elapsed time write 0.038 seconds

Chroma local collections: ['www.multima.cz']:

Count of chunks: 1138

Total length of chunks: 541495

Average length of chunk: 476

Retriever s embeddings (chunk = 800, overlap=80, k = 5)

A black background with white text

Description automatically generated

Retriever s embeddings (chunk = 1000, overlap=100, k = 4)

A screenshot of a computer program

Description automatically generated

Postup:

1 – Vytvoření indexů kroky 1 – 3, 6 – 10.

|  |  |  |
| --- | --- | --- |
| Krok | Název | Poznámka |
| 1 | Website crawling | 95 sec |
| 2 | Data loading | 65 sec |
| 3 | Cleaning data | 0.4 sec |
| 6 | Extracting metadata from the text |  |
| 7 | Creating chunks | 0.3 sec |
| 8 | Creating embeddings and save to database | 49 sec |
| 9 | Creating BM25 indexing | 0.03 sec |
| 10 | get statistics | 0 sec |

2 – Vytvořit QnA pro RAGAS (“Query”, „Ground thruth“) v kroku 4. Toto lze vytvořit pouze jednou. Otázky jsou generovány automaticky z vytvořených segmentů (chunks). Výsledek uložen jako JSON soubor (qna.json).

3 – vytvoření testovacího souboru pro RAGAS v kroku 13. Výsledek uložen jako JSON soubor (test\_dataset.json)

500 \* 8



4700 s

667 \* 6

A screen shot of a computer

Description automatically generated

800 \* 5

A screen shot of a computer

Description automatically generated

1000 \* 4

A black screen with white text

Description automatically generated

7 – evaluace testovacího souboru některou z metod v kroku 14

500 \* 8

A black screen with white text

Description automatically generated

667 \* 6

A black background with white text

Description automatically generated

800 \* 5

A black background with white text

Description automatically generated

1000 \* 4

A black background with white text

Description automatically generated

Vyhodnocení délky segmentu

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Délka segmentu | Overlap | K | Semantic similarity | Počet chybných odpovědí |
| 500 | 50 | 8 | 0.971588 | 3 |
| 667 | 67 | 6 | 0.978469 | 2 |
| 800 | 80 | 5 | 0.978636 | 2 |
| 1000 | 100 | 4 | 0.976911 | 3 |
|  |  |  |  |  |

# RestAPI server

Rest API služba ve formě serveru, která zpřístupňuje služby pro generování odpovědi na základě dotazů. Využívá třídu KBAQnA



Programový kód aplikace je veden v projektu PythonFlaskRestApi. Je synchronizován do Git úložiště na GitHubu. Při změně v Git repository jsou změny automaticky sestaveny a publikovány do webové služby v Azure.

A screenshot of a chatbot

Description automatically generated

Jednotlivé služby jsou popsány v následující kapitole.

Služby jsou zabezpečeny Basic autorizací (username, password).

## Dotaz odpověď – qna

/qna - Question / Answer service. It cooperates with class KBAQnA. Use embeddings in vector database Qdrant, where are prepared embeddings data for the project.

POST method.

Input:

{

"question": question,

"user\_id": user\_id,

"project": project,

[system\_msg": system\_msg,]

["api\_type": api\_type,]

["api\_base": api\_base,]

["api\_key": api\_key,]

["api\_version": api\_version,]

["api\_model": api\_model,]

}

Output:

{

"answer": answer

"error": error

}

Parameters:

question - question (is mandatory)

user\_id - unique user id (is mandatory)

project - project name (is collection name in vector db). Is mandatory.

system\_msg - partial text which will be added at the begin of the system message (can be empty)

api\_type - OpenAI type - open\_ai, azure (if empty then environment variable "OPENAI\_API\_TYPE" is used )

api\_base - URL base of the ChatGPT API (if empty then environment variable "OPENAI\_API\_BASE" is used

api\_key - API key of the ChatGPT (if empty then environment variable "OPENAI\_API\_KEY" is used)

api\_version - version of the ChatGPT API (if empty then environment variable "OPENAI\_API\_VERSION" is used)

api\_model - model of the ChatGPT API. (if empty then environment variable "OPENAI\_API\_MODEL\_GPT" is used)

For open\_ai: gpt-3.5-turbo, gpt-3.5-turbo-0613, gpt-3.5-turbo-16k, gpt-3.5-turbo-16k-0613

gpt-4, gpt-4-0613, gpt-4-32k, gpt-4-32k-0613

For azure: deployment name

answer - answer

error - normally it is empty. It contains a text error if there is a problem

## Načtení parametrů serveru – get\_srv\_par

/get\_srv\_par - Get server parameters

GET method.

Output:

{

"db\_type": db\_type,

"db\_dir": db\_dir,

"system\_msg": system\_msg,

"k\_history": k\_history,

"time\_limit\_history": time\_limit\_history,

"verbose": verbose,

"answer\_time": answer\_time,

}

Parameters:

db\_type - vector database type (if empty then unchanged):

local - local Chroma DB in db directory,

qdrant - Qdrant database.

db\_dir - directory, where is saved local vector Chroma db (only for db = local)

system\_msg - partial text which will be added at the begin of the system message (can be empty)

k\_history - the maximum length of history that is used for the conversation

time\_limit\_history - the time interval in seconds after which the history is erased

verbose - True - logging process question/answer to system output, False - without logging

answer\_time - True - the answer contains the time spent in seconds, False - answer is without spent time

## Načtení parametrů projektu – get\_project\_par

/get\_project\_par - Get project parameters

GET method.

Output:

{

"project": project,

"system\_msg": system\_msg,

"api\_model": api\_model,

"answer\_time": answer\_time,

"citation": citation,

}

Parameters:

project - project name. Is mandatory.

system\_msg - partial text which will be added at the begin of the system message

api\_model - model of the ChatGPT API.

For open\_ai: gpt-3.5-turbo, gpt-3.5-turbo-0613, gpt-3.5-turbo-16k, gpt-3.5-turbo-16k-0613

gpt-4, gpt-4-0613, gpt-4-32k, gpt-4-32k-0613

For azure: deployment name

answer\_time - True - answer is with elapsed time, False - answer is without elapsed time

citation - True - at the end of answer add web page references, False - without web page references

## Nastavení parametrů serveru – set\_srv\_par

/set\_srv\_par - Set server parameters

POST method.

{

["db\_type": db\_type,]

["db\_dir": db\_dir,]

["system\_msg": system\_msg,]

["k\_history": k\_history,]

["time\_limit\_history": time\_limit\_history,]

["verbose": verbose,]

["answer\_time": answer\_time,]

["erase\_history": erase\_history]

}

Parameters (if parameter isn't used then is unchanged):

db\_type - vector database type (if empty then unchanged):

local - local Chroma DB in db directory,

qdrant - Qdrant database. Needs environment variables: QDRANT\_URL, QDRANT\_API\_KEY

db\_dir - directory, where is saved local vector Chroma db (only for db = local)

system\_msg - partial text which will be added at the begin of the system message ((if empty then it is unchanged))

k\_history - the maximum length of history that is used for the conversation

time\_limit\_history - the time interval in seconds after which the history is cleared

verbose - True - logging process question/answer to system output, False - without logging

answer\_time - True - the answer contains the time spent in seconds, False - answer is without spent time

Others:

erase\_history - True - question/answer history will be erased, False - question/answer history will not be erased

Default False.

## Nastavení parametrů projektu – set\_project\_par

/set\_project\_par - Set project parameters

POST method.

{

"project": project,

["system\_msg": system\_msg,]

["api\_model": api\_model,]

["answer\_time": answer\_time,]

[ "citation": citation,]

["erase\_history": erase\_history]

}

Parameters: (when is empty or None then are unchanged)

project - project name (is collection name in vector db). Is mandatory.

system\_msg - partial text which will be added at the begin of the system message (if is empty then is unchanged)

api\_model - model of the ChatGPT API. (if empty then environment variable "OPENAI\_API\_MODEL\_GPT" is used)

For open\_ai: gpt-3.5-turbo, gpt-3.5-turbo-0613, gpt-3.5-turbo-16k, gpt-3.5-turbo-16k-0613

gpt-4, gpt-4-0613, gpt-4-32k, gpt-4-32k-0613

For azure: deployment name

answer\_time - True - answer is with elapsed time, False - answer is without elapsed time (if is None or isn't presented then is unchanged)

citation - True - at the end of answer add web page references, False - without web page references time (if is None or isn't presented then is unchanged)

erase\_history - True - question/answer history will be erased, False - question/answer history will not be erased (default False)

# Konkrétní řešení

## [www.multima.cz](http://www.multima.cz)

Chatbot pro webové stránky Multima a.s.

### Zdrojová data

Zdrojovými daty je obsah webových stránek [www.multima.cz](http://www.multima.cz). Tyto qwebové stránky mají odkazy i do dalších samostatných webů (Keymate, Nathan, Dokladovna)

Zpracovány jsou pouze webové stránky, které obsahují v doménách 1. – 3. Řádu:

'www.multima.cz', 'www.keymate.cz', 'www.nathan-ai.cz', 'www.dokladovna.cz'

Na stránkách jsou i anglické varianty, které jsou vyjmuty ze zpracování (duplicitní informace):

www.multima.cz/en/

### Indexování

Webové stránky obsahují patičky, hlavičky a menu, které se objevují na většině stránek. Tyto informace jsou nadbytečné , a proto jsou ze zdrojových dat odstraněny.

V případě některých kontaktních informací (telefonní čísla, emailové adresy) chybí u těchto informací popisná anotace, která byž mohla činit obtíže při vyhledání. Tato anotace je automaticky doplněna.

Např.

Údaje bez anotace:

+420 606 792 604

vdedoure@multima.cz

Údaje s anotací:

Telefon: +420 606 792 604

Email: vdedoure@multima.cz

Při vyhledání klíčových informací při zadýání dotazů je problém s vyhledáním správného kontextu po zadání dotazu. Proto jsou automaticky generovány následující anotace ze zdrojových textů dle popisu.

schema\_def = {

"type": "object",

"properties": {

"subject": {

"type": "string",

"enum": ["Produkty", "Služby", "Kontaktní informace", "Kariéra", "Informace o firmě", "Jiné",],

"description": "Jaké jsou převažující informace v textu."

},

"price\_list": {

"type": "boolean",

"description": "Zda je v textu obsažen ceník.",

},

"product": {

"type": "string",

"enum": ["Nathan AI", "Dokladovna", "Keymate", "Odtahovka", "Mentor", "Řízená dokumentace", "Multiskills",],

"description": "Název produktu nabízeného Multimou. Pouze v případě, že informace v textu popisují produkt."

},

"service": {

"type": "string",

"enum": ["Vývoj software", "Integrace", "AI - umělá inteligence", "Cloudifikace", "Powerapps", "Sharepoint", "Správa obsahu v Microsoft 365",],

"description": "Název služby nabízené Multimou. Pouze v případě, že informace v textu popisují službu."

},

"case\_study": {

"type": "string",

"enum": ["Pojišťovny", "Farmacie",],

"description": "Název případové studie nabízené Multimou. Pouze v případě, že informace v textu popisují případovou studii."

},

},

"required": ["subject", "price\_list"],

}

kba\_index.create\_tagging(

schema\_def = schema\_def,

field\_list=["subject", "price\_list", "product", "service", "case\_study",],

)

Segmentace textu (chunks):

Metoda: RecursiveCharacterTextSplitter

Chunk = 800, overlap = 80 (optimální hodnoty při evaluaci)

### Dotazování

Retriever: SelfQueryRetriever

Nastavení:

system\_msg = "You are AI Assistant named Vanda."

self\_doc\_descr = "Informace o produktech, službách a aktivitách společnosti Multima."

self\_metadata = [

AttributeInfo(

name="subject",

description="Jaké jsou převažující informace v textu. Jeden z ['Produkty', 'Služby', 'Kontaktní informace', 'Kariéra', 'Informace o firmě', 'Jiné']",

type="string",

),

AttributeInfo(

name="price\_list",

description="Zda je v textu obsažen ceník.",

type="boolean",

),

AttributeInfo(

name="product",

description="Název produktu nabízeného Multimou. Jeden z ['Nathan AI', 'Dokladovna', 'Keymate', 'Odtahovka', 'Mentor', 'Řízená dokumentace', 'Multiskills']",

type="string",

),

AttributeInfo(

name="service",

description="Název služby nabízené Multimou. Jeden z ['Vývoj software', 'Integrace', 'AI - umělá inteligence', 'Cloudifikace', 'Powerapps', 'Sharepoint', 'Správa obsahu v Microsoft 365']",

type="string",

),

AttributeInfo(

name="case\_study",

description="Název případové studie nabízené Multimou. Jeden z ['Pojišťovny', 'Farmacie']",

type="string",

),

]

## [www.mulouny.cz](http://www.mulouny.cz)

Chatbot pro webové stránky městského úřadu Louny.

Data zde ve formátu xml:

<https://www.mulouny.cz/mobile/xml.php?akce=openai_export_zivotnisituace&appID=7>

Webové stránky:

<https://www.mulouny.cz/cs/mestsky-urad/jak-kde-a-co-vyridite/>

### Zdrojová data

Zdrojovými daty je formát xml. Celé webové stránky jsou uloženy v jediné xml struktuře viz následující obrázek.

Legenda:

\* - element je array

? – element je nepovinný



### Indexování

Vzhledem k tomu, že zdrojová data jsou přísně strukturovaná, potom je indexování provedeno následující transformací:

html -> xml -> JSON

Z dat uvedených na webu ve formátu html je extrahována XML část. jelikož jsou v textech obsaženy formátovací html tagy, tak ty jsoui odstraněny. Xml struktura je převedena do JSON formátu. Tato data jsou extrahována do textů a příslušných metadat.

Extrakce textů z elementů: #text, text @href, #text

Extrakce medat viz tabulka

|  |  |  |
| --- | --- | --- |
| Element | Metadata | popis |
| @sekce | Sekce | Skupina životních situací ve struktuře identifikována číslem. To je konvertováno do popisu. |
| keywords | keywords | Klíčové výrazy životní situace |
| #text | situace | Název životmní situace. Je součástí textu pro situaci s @id = 3. N8zev situace v popis7u kon49 uvozovkami. |

Pro extrakci dat z webové stránky zvoleno následující nastavení WebExtLoader.

web\_path = "https://www.mulouny.cz/mobile/xml.php?akce=openai\_export\_zivotnisituace&appID=7"

# 1

loader = WebExtLoader(

web\_path= web\_path,

selected\_selector = "situaces",

remove\_tags = ['h3', 'ul', 'li', 'br'],

array\_item = "situace",

content\_func=content\_func,

metadata\_func = metadata\_func,

)

data = loader.load()

def content\_func(item:dict) -> str:

content = ""

for bod in item["bod"]:

# points 1 and >= 25 are rejected

bod\_id = int(bod["@id"])

if bod\_id == 1 or bod\_id >= 25: continue

if bod\_id == 3:

parts = bod["#text"].split(":")

content += parts[1].strip() + ".\n"

continue

content += "\n" + bod["@id"] + ". "

if "#text" in bod:

content += bod["#text"] + "\n"

if "radek" in bod:

if type(bod["radek"]) is list:

for radek in bod["radek"]:

if "text" in radek:

content += radek["text"] + "\n"

if "odkaz" in radek:

if "@href" in radek["odkaz"]:

content += radek["odkaz"]["@href"] + "\n"

if "#text" in radek["odkaz"]:

content += radek["odkaz"]["#text"] + "\n"

else:

radek = bod["radek"]

if "text" in radek:

content += radek["text"] + "\n"

if "odkaz" in radek:

if "@href" in radek["odkaz"]:

content += radek["odkaz"]["@href"] + "\n"

if "#text" in radek["odkaz"]:

content += radek["odkaz"]["#text"] + "\n"

return content.strip()

def metadata\_func(item:dict) -> dict:

metadata = {}

metadata["sekce"] = item["@sekce"]

match int(item["@sekce"]):

case 1:

metadata["sekce"] = "Osobní doklady"

case 2:

metadata["sekce"] = "Živnosti"

case 3:

metadata["sekce"] = "Finance"

case 4:

metadata["sekce"] = "ŽS různé"

case 6:

metadata["sekce"] = "Majetek města"

case 7:

metadata["sekce"] = "Stavební činnost"

case 8:

metadata["sekce"] = "Životní prostředí"

case 9:

metadata["sekce"] = "Památková péče"

case 10:

metadata["sekce"] = "Matrika"

case 11:

metadata["sekce"] = "Doprava ostatní"

case 12:

metadata["sekce"] = "Doprava a komunikace"

case 13:

metadata["sekce"] = "Registr vozidel"

case 10:

metadata["sekce"] = "Registr řidičů"

case \_:

metadata["sekce"] = "Jiné"

if "keywords" in item:

metadata["keywords"] = item["keywords"]

for bod in item["bod"]:

if bod["@id"] == "3":

parts = bod["#text"].split(":")

metadata["situace"] = parts[1].strip()

return metadata

Segmentace textu (chunks):

Metoda: RecursiveCharacterTextSplitter

Chunk = 800, overlap = 80

A screenshot of a computer

Description automatically generated

### Dotazování

Retriever: SelfQueryRetriever

Nastavení:

system\_msg = "You are AI Assistant named Vanda."

self\_doc\_descr = "Informace o obsluhovaných situacích městského úřadu Louny."

sekce = "'Osobní doklady', 'Živnosti', 'Finance', 'ŽS různé', 'Majetek města', 'Stavební činnost', 'Životní prostředí', 'Památková péče', 'Matrika'," +\

" 'Doprava ostatní', 'Doprava a komunikace', 'Registr vozidel', 'Registr řidičů', 'Jiné'"

self\_metadata = [

AttributeInfo(

name="situace",

description="Název obsluhované situace",

type="string",

),

AttributeInfo(

name="sekce",

description="Název sekce, do které spadá obsluhovaná situace. Jedna z [" + sekce + "]",

type="string",

),

AttributeInfo(

name="keywords",

description="Seznam klíčovách výrazů oddělených čárkou, které charakterizují obsluhovanou situaci",

type="string",

),

]

# Přílohy