My Project

Generated by Doxygen 1.14.0

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Namespace Documentation

3.1 config Namespace Reference

Variables

- str MAIN_FOLDER = "Main_term_sheet"
- str GEMINI_MODEL = "gemini-2.5-flash"
- str GROQ_MODEL = "llama-3.3-70b-versatile"
- int TOP_K = 6
- int CHUNK_SIZE = 2000
- int OVERLAP = 200
- str PROMPTS_FILE = "Prompts/prompts_term_sheet.json"
- str EXCEL_FILE = "TermSheet Output.xlsx"

3.1.1 Variable Documentation

3.1.1.1 CHUNK_SIZE

```
int config.CHUNK_SIZE = 2000
```

3.1.1.2 EXCEL_FILE

```
str config.EXCEL_FILE = "TermSheet Output.xlsx"
```

3.1.1.3 GEMINI_MODEL

```
str config.GEMINI_MODEL = "gemini-2.5-flash"
```

3.1.1.4 GROQ_MODEL

```
str config.GROQ_MODEL = "llama-3.3-70b-versatile"
```

3.1.1.5 MAIN_FOLDER

```
str config.MAIN_FOLDER = "Main_term_sheet"
```

3.1.1.6 OVERLAP

```
int config.OVERLAP = 200
```

3.1.1.7 PROMPTS_FILE

```
str config.PROMPTS_FILE = "Prompts/prompts_term_sheet.json"
```

3.1.1.8 TOP_K

```
int config.TOP_K = 6
```

3.2 extractor Namespace Reference

Functions

- List[str] extract_text_from_pdf (str path)
- List[str] chunk_text (str text, int chunk_size=2000, int overlap=200)
- List[Dict] extract_chunks_from_termsheet (str termsheet_pdf, int chunk_size=500, int overlap=200, str folder_name=None)

3.2.1 Detailed Description

```
extractor.py
- extract_text_from_pdf(pdf_path): returns list of pages' text
- extract_chunks_from_termsheet(termsheet_pdf, chunk_size=2000, overlap=200)
  returns list of dicts: { "chunk": str, "source": "termsheet", "page": int, "folder": folder_name }
```

3.2.2 Function Documentation

3.2.2.1 chunk_text()

```
List[str] extractor.chunk_text (
            str text,
            int  chunk_size = 2000,
            int  overlap = 200)
```

```
Naive chunking by characters with overlap.
```

3.2.2.2 extract_chunks_from_termsheet()

```
List[Dict] extractor.extract_chunks_from_termsheet (
            str termsheet_pdf,
            int  chunk_size = 500,
            int  overlap = 200,
            str  folder_name = None)
```

```
Extracts text from a single Term Sheet PDF, chunks per page,
and returns list of chunk dicts.
Each dict: {chunk, source="termsheet", page, chunk_index, folder}
```

3.2.2.3 extract_text_from_pdf()

```
List[str] extractor.extract_text_from_pdf (
            str path)
```

```
Return list of page texts (index 0 == page 1). Uses pdfplumber.
```

3.3 main Namespace Reference

Functions

- find_all_pdfs (str folder_path)
- main ()

3.3.1 Detailed Description

```
Main pipeline for Term Sheet extraction
- Walk MAIN_FOLDER, find all PDFs
- Extract chunks -> parse -> write to Excel
- Each PDF corresponds to one row in EXPORT sheet
```

3.3.2 Function Documentation

3.3.2.1 find_all_pdfs()

```
main.find_all_pdfs (
            str folder_path)
```

```
Return full paths of all PDFs in folder (non-recursive).
```

3.3.2.2 main()

```
main.main ()
```

3.4 parser Namespace Reference

Functions

- Dict build_tfidf_index (List[Dict] chunks)
- List[int] retrieve_top_k (str query, Dict index, int k=5)
- str assemble_context (List[Dict] chunks, List[int] top_indices)
- Dict call_groq (str model, List[Dict] messages, float temperature=0.0, int max_retries=3)
- List[Dict] parse_with_llm (List[Dict] chunks, str prompts_path, str groq_model, int top_k=5)
- List[Dict] parse_with_llm_gemini (List[Dict] chunks, str prompts_path, str gemini_model, int top_k=5)
- Dict call_gemini (str model_gemini, List[Dict] messages, float temperature=0.0, int max_retries=3)

3.4.1 Detailed Description

```
parser.py
- Loads prompts from Prompts/prompts_spo_framework.json
- Builds a TF-IDF index over extracted chunks
- For each prompt, filter chunks by "run_for" (framework/spo/both),
  retrieve top_k chunks, create system/user message,
  and call Groq LLM to produce the output JSON.
- Exports a list of parsed JSONs (one per prompt).
```

3.4.2 Function Documentation

3.4.2.1 assemble_context()

```
str parser.assemble_context (
            List[Dict] chunks,
            List[int] top_indices)
```

```
Create a human-readable context block with source and page metadata.
```

3.4.2.2 build_tfidf_index()

```
Dict parser.build_tfidf_index (
            List[Dict] chunks)
```

```
Return vectorizer and matrix for search, plus the chunk texts.
```

3.4.2.3 call_gemini()

```
Dict parser.call_gemini (
            str model_gemini,
            List[Dict] messages,
            float  temperature = 0.0,
            int  max_retries = 3)
```

```
Call Gemini chat model with retries.
messages: list of {"role": "system"|"user"|"assistant", "content": str}
```

3.4.2.4 call_groq()

```
Dict parser.call_groq (
            str model,
            List[Dict] messages,
            float  temperature = 0.0,
            int  max_retries = 3)
```

3.4.2.5 parse_with_llm()

```
List[Dict] parser.parse_with_llm (
            List[Dict] chunks,
            str prompts_path,
            str  groq_model,
            int  top_k = 5)
```

```
chunks: list of dicts from extractor.py
prompts_path: path to prompts_spo_frameworks.json
groq_model: Groq model name
returns: list of dicts { "prompt_id": ..., "result": <parsed json> }
```

3.4.2.6 parse_with_llm_gemini()

```
List[Dict] parser.parse_with_llm_gemini (
            List[Dict] chunks,
            str prompts_path,
            str gemini_model,
            int  top_k = 5)
```

```
chunks: list of dicts from extractor.py
prompts_path: path to prompts.json
gemini_model: Gemini model name (e.g., "gemini-1.5-flash")
returns: list of dicts { "prompt_id": ..., "result": <parsed json> }
```

3.4.2.7 retrieve_top_k()

```
List[int] parser.retrieve_top_k (
            str query,
            Dict index,
            int  k = 5)
```

```
Return top-k indices (into index['texts']) most similar to query.
```

3.5 writer Namespace Reference

Functions

- Workbook _init_workbook ()
- None write_to_excel (Dict json_data)

3.5.1 Function Documentation

3.5.1.1 _init_workbook()

```
Workbook writer._init_workbook ()  [protected]
```

Initialize the workbook with a single sheet 'EXPORT' and required headers.

3.5.1.2 write_to_excel()

```
None writer.write_to_excel (
            Dict json_data)
```

Write parsed Term Sheet JSON data into the EXPORT sheet.
json_data: dictionary output from parser.py

Chapter 4

File Documentation

4.1 config.py File Reference

Namespaces

- namespace config

Variables

- str config.MAIN_FOLDER = "Main_term_sheet"
- str config.GEMINI_MODEL = "gemini-2.5-flash"
- str config.GROQ_MODEL = "llama-3.3-70b-versatile"
- int config.TOP_K = 6
- int config.CHUNK_SIZE = 2000
- int config.OVERLAP = 200
- str config.PROMPTS_FILE = "Prompts/prompts_term_sheet.json"
- str config.EXCEL_FILE = "TermSheet Output.xlsx"

4.2 extractor.py File Reference

Namespaces

- namespace extractor

Functions

- List[str] extractor.extract_text_from_pdf (str path)
- List[str] extractor.chunk_text (str text, int chunk_size=2000, int overlap=200)
- List[Dict] extractor.extract_chunks_from_termsheet (str termsheet_pdf, int chunk_size=500, int overlap=200, str folder_name=None)

4.3 main.py File Reference

Namespaces

- namespace main

Functions

- main.find_all_pdfs (str folder_path)
- main.main ()

4.4 parser.py File Reference

Namespaces

- namespace parser

Functions

- Dict parser.build_tfidf_index (List[Dict] chunks)
- List[int] parser.retrieve_top_k (str query, Dict index, int k=5)
- str parser.assemble_context (List[Dict] chunks, List[int] top_indices)
- Dict parser.call_groq (str model, List[Dict] messages, float temperature=0.0, int max_retries=3)
- List[Dict] parser.parse_with_llm (List[Dict] chunks, str prompts_path, str groq_model, int top_k=5)
- List[Dict] parser.parse_with_llm_gemini (List[Dict] chunks, str prompts_path, str gemini_model, int top_k=5)
- Dict parser.call_gemini (str model_gemini, List[Dict] messages, float temperature=0.0, int max_retries=3)

4.5 writer.py File Reference

Namespaces

- namespace writer

Functions

- Workbook writer._init_workbook ()
- None writer.write_to_excel (Dict json_data)