DatasetAnalyzer

Release 0.1

Samuel Harčár

May 13, 2025

CONTENTS:

Add your content using `reStructuredText` syntax. See the reStructuredText documentation for details.

CHAPTER

ONE

SRC

1.1 constants module

Module where constants are initialized.

1.2 direct_analysis module

Module contains functions for handling main paper analysis.

direct_analysis.**analyze_paper_llm**(*full_text: str*, *prompt_file_name: str*)

Performs main article llm analysis based on provided main analysis prompt config file.

Parameters

- **full_text** (*int*) – Full text of main article
- **prompt_file_name** (*str*) – Main analysis prompt config file name

Returns

tuple: (analysis_dict, debug_dict) analysis_dict: dictionary with LLM responses corresponding to fields analysis_dict: dictionary with debug info corresponding to fields

Type

tuple(dict, dict)

direct_analysis.**get_section**(*text: str*, *keywords: list*, *chars_before: int*, *chars_after: int*)

Cuts out a portion of text around first occurrence of keyword.

Parameters

- **text** (*str*) – Text from which the portion will be extracted
- **keywords** (*list*) – List of regex keywords which will be searched in text
- **chars_before** (*int*) – Number of characters which will be extracted before first keyword match
- **chars_after** (*int*) – Number of characters which will be extracted after first keyword match

Returns

Extracted substring.

Return type

str

1.3 general_utils module

Module contains utility functions for general purpose.

general_utils.**extract_doi_from_url**(*url: str*)

Extracts numerical DOI identifier portion from DOI url

Parameters
url (*str*) – String URL containing DOI

Returns
String DOI identifier

Return type
str

general_utils.**merge_analyses**(*original_analysis: any*, *generated_analysis: dict*, *suffix: str*)

Merges two dict-like analyses into one. If any key is already present in original analysis, then specified suffix will be added to this key in combined output analysis.

Parameters

- **original_analysis** (*dict or TOMLDocument*) – Primary, dict-like analysis. Field names will stay the same in the merged output analysis. Can be a dict or a TOMLDocument.
- **generated_analysis** (*dict*) – Secondary, dict-like analysis. Field names can be modified using suffix in case of key collision with original analysis in the merged output analysis. Must be a dict.
- **suffix** (*str*) – String suffix, which will be added in case of same key names in inputted analyses. Suffix is used to distinguish between fields from primary and secondary analysis. In case of key collision, suffix will be added to the field name from secondary analysis. Key names from the primary analysis will be preserved.

Returns
merged analysis

Return type
Same as original_analysis

general_utils.**substitute_patterns**(*subst_file_name: str*, *text: str*)

Substitutes regex patterns in text according to subst. rules config file

Parameters

- **subst_file_name** (*str*) – Substitution rules config file name
- **text** (*str*) – Text on which the substitution will be performed

Returns
Text with applied substitutions

Return type
str

1.4 interval_blacklist module

Module containing interval blacklist class.

class interval_blacklist.**IntervalBlacklist**

Bases: object

add_interval(*start*, *end*)

Adds interval to blacklist.

Parameters

- **start** (*int*) – Lower boundary of the interval
- **end** (*int*) – Upper boundary of the interval

is_blacklisted(*value*)

Checks if given value is in currently blacklisted intervals.

Parameters
value (*int*) – Value to check

Returns
True if value is blacklisted, False if not.

Return type
bool

1.5 llm_utils module

Module containing functions which handle communication with LLM.

llm_utils.**extract_answer**(*output: str*, *json_field: str = ''*)

Extracts answer from JSON response generated by LLM.

Parameters

- **output** (*str*) – LLM generated JSON response
- **json_field** (*str*) – Name of the analysis field of which value will be extracted

Returns
Response extracted from JSON template.

Return type
str

llm_utils.**llm_query**(*text: str*, *sys_prompt: str*, *prompt: str*, *field_name: str = ''*, *response_type: str = 'string'*)

Sends given query to Ollama server. JSON response format is added for response quality enhancement.

Parameters

- **text** (*str*) – Context which will sent to LLM as part of the prompt
- **sys_prompt** (*str*) – System prompt
- **prompt** (*str*) – Prompt for LLM
- **field_name** (*str*) – Name of the analysis field which the prompt is for
- **response_type** – Data type of the wanted response (string, integer, . . .)

Returns
Extracted response of the LLM

Return type
str

llm_utils.**post_with_retries**(*url: str*, *payload_json: str*, *headers: dict*, *timeout: float*, *max_retries: int*)

Sends HTTP POST request. If timeout is reached without response, function will retry until max_retries is reached.

Parameters

- **url** (*str*) – URL, to which the request will be sent
- **payload_json** (*str*) – JSON payload which will be sent in the request
- **headers** (*dict*) – HTTP headers which will be sent in request
- **timeout** (*float*) – Request timeout in seconds
- **max_retries** (*int*) – Maximum number of retries to perform.

Returns
HTTP response or None if max_retries is reached without success

Return type
str in case of success, None in case of failure

1.6 logging_utils module

Handles logging operations.

logging_utils.**setup_logger**(*name: str*, *log_file: str*, *level: int = 20*)

Setups specialized logger object.

Parameters

- **name** (*str*) – File or module name
- **log_file** – Output file, where the logs will be written
- **level** (*int*) – Level of logging

Returns
Logger object

Return type
logging.Logger

1.7 main module

1.8 pdf_utils module

Module contains utility functions for PDF processing.

pdf_utils.**read_pdf**(*filename: str*)

Reads PDF using unstructured OCR.

Parameters
filename (*str*) – File name of the PDF.

Returns
tuple(main_text, full_text) main_text: Main text excluding reference section full_text: Text of the reference section

Return type
tuple(str, str)

pdf_utils.**read_pdf_pdfminer**(*filename: str*)

Reads PDF using pdfminer.

Parameters
filename (*str*) – File name of the PDF.

Returns
Text of the pdf file.

Return type
str

pdf_utils.**read_pdf_pypdf**(*file_name*)

Reads PDF using pypdf.

Parameters
filename – File name of the PDF.

Returns
Text of the pdf file.

Return type
str

1.9 reference_analysis module

Module handles analysis of referencing papers.

reference_analysis.**analyze_reference**(*file_name: str*, *main_text: str*, *full_text: str*, *input_analysis: dict*, *prompt_file_name: str*)

Extracts information from single referencing paper using LLM.

Parameters

- **file_name** (*str*) – File name of referencing paper
- **main_text** (*str*) – Main text excluding reference section
- **full_text** (*str*) – Text including the reference section
- **input_analysis** (*dict*) – Dictionary containing at least paper title and doi of the main paper
- **prompt_file_name** (*str*) – File name of reference prompts config file

Returns
tuple: (analysis_dict, debug_dict) analysis_dict: dictionary with LLM responses corresponding to fields analysis_dict: dictionary with debug info corresponding to fields

Return type
tuple(dict, dict)

reference_analysis.**analyze_references**(*directory_name: str*, *input_analysis: dict*)

Extracts information from all referencing papers using LLM.

Parameters

- **directory_name** (*str*) – Name of the directory where referencing papers are located
- **input_analysis** (*dict*) – Dictionary containing at least paper title and doi of the main paper

Returns
tuple: (output_analysis, debug_dict) output_analysis: dictionary with per paper analysis containing LLM responses corresponding to fields analysis_dict: dictionary with debug info corresponding to papers and fields

Return type
tuple(dict, dict)

reference_analysis.**extract_number_from_reference_identifier**(*identifier: str*)

Extracts reference number from reference identifier.

Parameters
identifier (*str*) – String identifier of reference (containing only single reference number e.g. [5])

Returns
Extracted reference number as int

Return type
int

reference_analysis.**find_citation_number**(*input_analysis: dict*, *text: str*)

Finds citation number of the main paper in references section of a reference paper.

Parameters

- **input_analysis** (*dict*) – Dictionary containing at least paper title and doi of the main paper
- **text** (*str*) – Text of the paper including reference section

Returns
Extracted reference number as int

Return type
int

reference_analysis.**get_sections_by_reference_number_or_title**(*text: str*, *reference_number: int*, *chars_before: int*, *chars_after: int*, *dataset_title: str*)

Finds all occurrences of reference to main paper or dataset title, extracts contexts around them and concatenates them.

Parameters

- **text** (*str*) – Text of the referencing paper without references section
- **reference_number** (*int*) – Reference number of main paper
- **chars_before** (*int*) – Number of characters to include in context before individual references
- **chars_after** (*int*) – Number of characters to include in context after individual references
- **dataset_title** (*int*) – Title of the dataset which is being analyzed

Returns
tuple(aggregation, num_occurrences) aggregation - Extracted context in which the analyzed dataset is referenced num_occurrences - Number of citations in the paper

Return type
tuple(str, int)

reference_analysis.**identifier_to_number_set**(*identifier: str*)

Converts citation identifier to set of all included numbers. For example: [8 - 10] -> set{8, 9, 10}

[3, 7] -> set{3, 7} [5] -> set{5}.

Parameters
identifier (*str*) – String representation of identifier (e.g. "[8 - 10]")

Returns
Set of included citation numbers

Return type
set

reference_analysis.**normalize_text**(*text: str*)

Converts string to lower case and substitutes all white character and new line sequences with single space.

Parameters
text (*str*) – String to be normalized

Returns
Normalized string

Return type
str

1.10 reference_analysis_aggregation module

Module handles aggregation of analyses from referencing papers.

reference_analysis_aggregation.**aggregate_reference_analysis**(*per_paper_analysis: dict*, *reference_prompts_file_name: str*)

Aggregates responses from corresponding fields from all referencing papers.

Parameters

- **per_paper_analysis** (*dict*) – Dictionary with structured analyses of referencing papers
- **reference_prompts_file_name** (*str*) – File name of reference analysis prompts config file

Returns
Aggregated structured analysis of referencing papers as dictionary

Return type
dict

reference_analysis_aggregation.**concatenate_field**(*analyses_list: list*, *field_name: str*, *default_value: str*, *skip_default: bool*, *concatenation_format: str*)

Concatenates responses from given field from all referencing papers while adding paper headers to each record.

Parameters

- **analyses_list** (*dict*) – List of referencing paper analyses
- **field_name** (*str*) – Name of the field which will be concatenated
- **default_value** (*str*) – Default value of the field which is present if analyzed dataset was not referenced
- **skip_default** (*int*) – Whether to skip papers with default values

- **concatenation_format** (*str*) – Format of each paper record including header and response placement

Returns
Aggregated responses as string

Return type
str

reference_analysis_aggregation.**value_count_aggregation**(*analyses_list: list*, *field_name: str*)

Counts responses from given field from all referencing papers and lists all unique values and their counts.

Parameters

- **analyses_list** (*list*) – List of referencing paper analyses
- **field_name** (*str*) – Name of the field which will be aggregated

Returns
Aggregated response as string

Return type
str

1.11 toml_report_utils module

Module handles reading and writing toml files.

toml_report_utils.**postprocess_report**(*file_name: str*)

Substitutes starting and ending quotation marks for three quotation marks, making strings multiline.

Parameters
file_name (*int*) – Name of TOML file

Returns
None

Return type
None

toml_report_utils.**read_report**(*filename: str*)

Reads TOML file using toml library.

Parameters
filename (*str*) – File name of the TOML file

Returns
Content of the file as dictionary

Return type
tuple(dict, dict)

toml_report_utils.**read_report_tomlkit**(*filename: str*)

Reads TOML file using tomlkit library.

Parameters
filename (*str*) – File name of the TOML file

Returns
Content of the file as TOMLDocument

Return type
TOMLDocument

toml_report_utils.**write_report_tomlkit**(*doc: TOMLDocument*, *filename: str*)

Writes TOML file using tomlkit library.

Parameters

- **doc** (*TOMLDocument*) – TOML content which will be saved
- **filename** (*str*) – File name of the TOML file

Returns
None

Return type
None

1.12 vector_database module

class vector_database.**VectorDatabase**(*full_text: str*, *chunk_size: int = 500*, *chunk_overlap: int = 50*)

Bases: object

get_expanded_chunks(*query: str*, *top_k: int = 5*, *chunks_before: int = 1*, *chunks_after: int = 1*)

Finds the most relevant chunks according to search query and expands them with more chunks before and after.

Parameters

- **query** (*str*) – Query for target text
- **top_k** (*int*) – Number of the best results to return
- **chunks_before** (*int*) – Number of chunks to add before to the found chunks
- **chunks_after** (*int*) – Number of chunks to add after to the found chunks

Returns
List of *k* most relevant chunks (expanded)

Return type
list

get_prepared_context(*query: str*, *top_k: int*, *chunks_before: int*, *chunks_after: int*)

Finds the most relevant chunks according to search query and expands them with more chunks before and after and concatenates them into single context.

Parameters

- **query** (*str*) – Query for target text
- **top_k** (*int*) – Number of the best results to return
- **chunks_before** (*int*) – Number of chunks to add before to the found chunks
- **chunks_after** (*int*) – Number of chunks to add after to the found chunks

Returns
String of most relevant chunks (expanded and concatenated)

Return type
str

get_relevant_chunks(*query: str*, *top_k: int = 5*)

Finds the most relevant chunks according to search query.

Parameters

- **query** (*str*) – Query for target text
- **top_k** (*int*) – Number of the best results to return

Returns

List of *k* most relevant chunks

Return type

list

vector_database.**add_prefixes_to_chunks**(*chunks: list*, *prefix: str*)

PYTHON MODULE INDEX

INDEX