

Data Science Capstone Project

Generated by Doxygen 1.9.8

1 Components Module Overview	1
2 Connectors Module Overview	3
3 Core Module Overview	5
3.1 Pattern	5
3.2 Usage	5
4 Source Directory Overview	7
5 Database Example Datasets	9
6 Test Suite Overview	11
7 Namespace Index	13
7.1 Package List	13
8 Hierarchical Index	15
8.1 Class Hierarchy	15
9 Class Index	17
9.1 Class List	17
10 File Index	19
10.1 File List	19
11 Namespace Documentation	21
11.1 BlazorApp Namespace Reference	21
11.2 BlazorApp.Controllers Namespace Reference	21
11.3 BlazorApp.Hubs Namespace Reference	21
11.4 BlazorApp.Models Namespace Reference	21
11.5 conftest Namespace Reference	22
11.5.1 Function Documentation	22
11.5.1.1 docs_db()	22
11.5.1.2 graph_db()	22
11.5.1.3 main_graph()	22
11.5.1.4 pytest_addoption()	22
11.5.1.5 relational_db()	23
11.5.1.6 session()	23
11.6 src Namespace Reference	23
11.7 src.charts Namespace Reference	23
11.8 src.components Namespace Reference	23
11.9 src.components.book_conversion Namespace Reference	24
11.9.1 Variable Documentation	24
11.9.1.1 nlp	24
11.9.1.2 sentencizer	24

11.10 src.components.corpus Namespace Reference	24
11.10.1 Function Documentation	25
11.10.1.1 fuzzy_merge_titles()	25
11.10.1.2 load_booksum()	25
11.10.1.3 load_narrativeqa()	26
11.10.1.4 merge_dataframes()	26
11.10.1.5 normalize_title()	26
11.10.1.6 to_df_booksum()	26
11.10.1.7 to_df_nqa()	26
11.10.2 Variable Documentation	26
11.10.2.1 df	26
11.10.2.2 df_booksum	26
11.10.2.3 df_nqa	26
11.10.2.4 index	27
11.10.2.5 m	27
11.11 src.components.fact_storage Namespace Reference	27
11.12 src.components.metrics Namespace Reference	27
11.12.1 Function Documentation	27
11.12.1.1 chunk_bookscore()	27
11.12.1.2 run_bookscore()	28
11.12.1.3 run_questeval()	29
11.13 src.components.relation_extraction Namespace Reference	29
11.14 src.connectors Namespace Reference	29
11.15 src.connectors.base Namespace Reference	30
11.16 src.connectors.document Namespace Reference	30
11.16.1 Function Documentation	31
11.16.1.1 _docs_to_df()	31
11.16.1.2 _find_compatible_nested_key()	31
11.16.1.3 _flatten_recursive()	32
11.16.1.4 _sanitize_document()	32
11.16.1.5 _sanitize_json()	32
11.16.1.6 mongo_handle()	33
11.16.2 Variable Documentation	33
11.16.2.1 MongoHandle	33
11.17 src.connectors.graph Namespace Reference	33
11.17.1 Function Documentation	34
11.17.1.1 _filter_to_db()	34
11.17.1.2 _normalize_elements()	34
11.17.1.3 _tuples_to_df()	35
11.18 src.connectors.llm Namespace Reference	35
11.19 src.connectors.relational Namespace Reference	35
11.20 src.core Namespace Reference	36

11.21 src.core.boss Namespace Reference	36
11.21.1 Detailed Description	36
11.21.2 Function Documentation	36
11.21.2.1 assign_task_to_worker()	36
11.21.2.2 clear_task_data()	37
11.21.2.3 create_app()	37
11.21.2.4 create_boss_thread()	38
11.21.2.5 load_worker_config()	38
11.21.2.6 post_chunk_status()	38
11.21.2.7 post_process_full_story()	39
11.21.2.8 post_story_status()	39
11.21.3 Variable Documentation	39
11.21.3.1 MongoHandle	39
11.22 src.core.context Namespace Reference	40
11.22.1 Function Documentation	40
11.22.1.1 __getattr__()	40
11.22.1.2 get_session()	41
11.22.2 Variable Documentation	41
11.22.2.1 session	41
11.23 src.core.stages Namespace Reference	41
11.23.1 Function Documentation	42
11.23.1.1 group_21_1_describe_graph()	42
11.23.1.2 group_21_2_send_statistics()	42
11.23.1.3 group_21_3_post_statistics()	42
11.23.1.4 task_01_convert_epub()	42
11.23.1.5 task_02_parse_chapters()	42
11.23.1.6 task_03_chunk_story()	42
11.23.1.7 task_10_random_chunk()	43
11.23.1.8 task_10_sample_chunks()	43
11.23.1.9 task_11_send_chunk()	43
11.23.1.10 task_12_relation_extraction_rebel()	43
11.23.1.11 task_13_concatenate_triples()	43
11.23.1.12 task_14_relation_extraction_llm()	43
11.23.1.13 task_15_sanitize_triples_llm()	43
11.23.1.14 task_20_send_triples()	43
11.23.1.15 task_22_verbalize_triples()	44
11.23.1.16 task_30_summarize_llm()	44
11.23.1.17 task_31_send_summary()	44
11.23.1.18 task_40_post_payload()	44
11.23.1.19 task_40_post_summary()	44
11.24 src.core.worker Namespace Reference	45
11.24.1 Detailed Description	45

11.24.2 Function Documentation	46
11.24.2.1 <code>create_app()</code>	46
11.24.2.2 <code>get_task_info()</code>	46
11.24.2.3 <code>load_boss_config()</code>	46
11.24.2.4 <code>load_imports()</code>	47
11.24.2.5 <code>load_mongo_config()</code>	47
11.24.2.6 <code>mark_task_in_progress()</code>	47
11.24.2.7 <code>notify_boss()</code>	48
11.24.2.8 <code>process_task()</code>	48
11.24.2.9 <code>save_task_result()</code>	49
11.24.3 Variable Documentation	49
11.24.3.1 <code>app</code>	49
11.24.3.2 <code>args</code>	49
11.24.3.3 <code>boss_url</code>	49
11.24.3.4 <code>daemon</code>	50
11.24.3.5 <code>help</code>	50
11.24.3.6 <code>host</code>	50
11.24.3.7 <code>MongoHandle</code>	50
11.24.3.8 <code>parser</code>	50
11.24.3.9 <code>PORT</code>	50
11.24.3.10 <code>port</code>	50
11.24.3.11 <code>required</code>	50
11.24.3.12 <code>target</code>	50
11.24.3.13 <code>task_queue</code>	50
11.24.3.14 <code>task_worker</code>	50
11.24.3.15 <code>True</code>	51
11.24.3.16 <code>use_reloader</code>	51
11.25 <code>src.main</code> Namespace Reference	51
11.25.1 Function Documentation	52
11.25.1.1 <code>full_pipeline()</code>	52
11.25.1.2 <code>old_main()</code>	52
11.25.1.3 <code>pipeline_A()</code>	52
11.25.1.4 <code>pipeline_B()</code>	52
11.25.1.5 <code>pipeline_C()</code>	53
11.25.1.6 <code>pipeline_D()</code>	53
11.25.1.7 <code>pipeline_E()</code>	53
11.25.2 Variable Documentation	53
11.25.2.1 <code>book_id</code>	53
11.25.2.2 <code>book_title</code>	53
11.25.2.3 <code>BOSS_PORT</code>	53
11.25.2.4 <code>checkpoint_path</code>	54
11.25.2.5 <code>chunk</code>	54

11.25.2.6 chunk_id	54
11.25.2.7 chunks	54
11.25.2.8 COLLECTION	54
11.25.2.9 data	54
11.25.2.10 DB_NAME	54
11.25.2.11 exist_ok	54
11.25.2.12 load_from_checkpoint	54
11.25.2.13 response	55
11.25.2.14 story_id	55
11.25.2.15 summary	55
11.25.2.16 triples	55
11.25.2.17 triples_string	55
11.26 src.util Namespace Reference	55
11.26.1 Function Documentation	55
11.26.1.1 check_values()	55
11.26.1.2 df_natural_sorted()	56
11.27 tests Namespace Reference	56
11.28 tests.test_db_basic Namespace Reference	56
11.28.1 Function Documentation	57
11.28.1.1 test_db_docs_comprehensive()	57
11.28.1.2 test_db_docs_minimal()	57
11.28.1.3 test_db_graph_comprehensive()	57
11.28.1.4 test_db_graph_minimal()	57
11.28.1.5 test_db_relational_comprehensive()	57
11.28.1.6 test_db_relational_minimal()	58
11.29 tests.test_db_files Namespace Reference	58
11.29.1 Function Documentation	58
11.29.1.1 _exec_query_file()	58
11.29.1.2 load_examples_relational()	59
11.29.1.3 test_cypher_example_1()	59
11.29.1.4 test_cypher_example_2()	59
11.29.1.5 test_cypher_example_3()	59
11.29.1.6 test_cypher_example_4()	59
11.29.1.7 test_mongo_example_1()	60
11.29.1.8 test_mongo_example_2()	60
11.29.1.9 test_mongo_example_3()	60
11.29.1.10 test_sql_example_1()	60
11.29.1.11 test_sql_example_2()	61
11.30 tests.test_kg_triples Namespace Reference	61
11.30.1 Function Documentation	61
11.30.1.1 nature_scene_graph()	61
11.30.1.2 test_detect_community_clusters_comprehensive()	62

11.30.1.3 test_detect_community_clusters_minimal()	62
11.30.1.4 test_get_neighborhood()	62
11.30.1.5 test_get_neighborhood_comprehensive()	62
11.30.1.6 test_get_random_walk_sample()	63
11.30.1.7 test_get_random_walk_sample_comprehensive()	63
11.30.1.8 test_get_subgraph_by_nodes()	63
11.30.1.9 test_knowledge_graph_triples()	63
11.30.1.10 test_ranked_degree()	64
11.30.1.11 test_ranked_degree_ties()	64
11.31 tests.test_pipeline Namespace Reference	64
11.31.1 Function Documentation	65
11.31.1.1 book_1_data()	65
11.31.1.2 book_2_data()	65
11.31.1.3 book_data()	65
11.31.1.4 test_job_01_convert_epub()	65
11.31.1.5 test_job_02_parse_chapters()	66
11.31.1.6 test_job_03_chunk_story()	66
11.31.1.7 test_job_10_random_chunk()	66
11.31.1.8 test_job_10_sample_chunks()	66
11.31.1.9 test_job_11_send_chunk()	66
11.31.1.10 test_job_13_concatenate_triples()	66
11.31.1.11 test_job_15_sanitize_triples_llm()	66
11.31.1.12 test_job_20_send_triples()	67
11.31.1.13 test_job_21_describe_graph()	67
11.31.1.14 test_job_22_verbalize_triples()	67
11.31.1.15 test_job_31_send_summary()	67
11.31.1.16 test_pipeline_A_from_csv()	67
11.31.1.17 test_pipeline_A_minimal()	67
11.31.1.18 test_pipeline_C_minimal()	68
11.31.1.19 test_pipeline_E_minimal_full_payload()	68
11.31.1.20 test_pipeline_E_minimal_summary_only()	68
12 Class Documentation	69
12.1 Log.BadAddressFailure Class Reference	69
12.1.1 Detailed Description	70
12.1.2 Constructor & Destructor Documentation	71
12.1.2.1 __init__()	71
12.2 Book Class Reference	71
12.2.1 Constructor & Destructor Documentation	71
12.2.1.1 __init__()	71
12.2.2 Member Function Documentation	71
12.2.2.1 stream_chapters()	71

12.3 BookFactory Class Reference	72
12.3.1 Member Function Documentation	72
12.3.1.1 <code>create_book()</code>	72
12.4 BookStream Class Reference	73
12.4.1 Constructor & Destructor Documentation	74
12.4.1.1 <code>__init__()</code>	74
12.4.2 Member Function Documentation	74
12.4.2.1 <code>stream_segments()</code>	74
12.4.3 Member Data Documentation	74
12.4.3.1 <code>book</code>	74
12.5 Chunk Class Reference	74
12.5.1 Detailed Description	75
12.5.2 Constructor & Destructor Documentation	75
12.5.2.1 <code>__init__()</code>	75
12.5.3 Member Function Documentation	76
12.5.3.1 <code>__repr__()</code>	76
12.5.3.2 <code>char_count()</code>	76
12.5.3.3 <code>get_chunk_id()</code>	76
12.5.3.4 <code>to_mongo_dict()</code>	77
12.5.4 Member Data Documentation	77
12.5.4.1 <code>book_id</code>	77
12.5.4.2 <code>chapter_number</code>	77
12.5.4.3 <code>chapter_percent</code>	77
12.5.4.4 <code>line_end</code>	77
12.5.4.5 <code>line_start</code>	77
12.5.4.6 <code>story_id</code>	77
12.5.4.7 <code>story_percent</code>	77
12.5.4.8 <code>text</code>	78
12.6 Connector Class Reference	78
12.6.1 Detailed Description	79
12.6.2 Member Function Documentation	79
12.6.2.1 <code>check_connection()</code>	79
12.6.2.2 <code>execute_file()</code>	80
12.6.2.3 <code>execute_query()</code>	80
12.6.2.4 <code>test_operations()</code>	80
12.7 DatabaseConnector Class Reference	81
12.7.1 Detailed Description	83
12.7.2 Constructor & Destructor Documentation	83
12.7.2.1 <code>__init__()</code>	83
12.7.3 Member Function Documentation	84
12.7.3.1 <code>_is_single_query()</code>	84
12.7.3.2 <code>_parsable_to_df()</code>	84

12.7.3.3 <code>_returns_data()</code>	85
12.7.3.4 <code>_split_combined()</code>	85
12.7.3.5 <code>change_database()</code>	85
12.7.3.6 <code>configure()</code>	86
12.7.3.7 <code>create_database()</code>	86
12.7.3.8 <code>database_exists()</code>	86
12.7.3.9 <code>drop_database()</code>	87
12.7.3.10 <code>execute_combined()</code>	87
12.7.3.11 <code>execute_file()</code>	87
12.7.3.12 <code>execute_query()</code>	88
12.7.3.13 <code>get_dataframe()</code>	88
12.7.3.14 <code>temp_database()</code>	89
12.7.4 Member Data Documentation	89
12.7.4.1 <code>connection_string</code>	89
12.7.4.2 <code>db_engine</code>	89
12.7.4.3 <code>db_type</code>	89
12.7.4.4 <code>host</code>	90
12.7.4.5 <code>password</code>	90
12.7.4.6 <code>port</code>	90
12.7.4.7 <code>username</code>	90
12.7.4.8 <code>verbose</code>	90
12.8 DocumentConnector Class Reference	90
12.8.1 Detailed Description	93
12.8.2 Constructor & Destructor Documentation	93
12.8.2.1 <code>__init__()</code>	93
12.8.3 Member Function Documentation	93
12.8.3.1 <code>_parsable_to_df()</code>	93
12.8.3.2 <code>_returns_data()</code>	94
12.8.3.3 <code>_split_combined()</code>	94
12.8.3.4 <code>change_database()</code>	95
12.8.3.5 <code>check_connection()</code>	95
12.8.3.6 <code>create_database()</code>	95
12.8.3.7 <code>database_exists()</code>	96
12.8.3.8 <code>delete_dummy()</code>	96
12.8.3.9 <code>drop_database()</code>	96
12.8.3.10 <code>execute_query()</code>	97
12.8.3.11 <code>get_dataframe()</code>	97
12.8.3.12 <code>get_unmanaged_handle()</code>	98
12.8.3.13 <code>test_operations()</code>	98
12.8.4 Member Data Documentation	99
12.8.4.1 <code>_auth_suffix</code>	99
12.8.4.2 <code>connection_string</code>	99

12.8.4.3 database_name	99
12.8.4.4 verbose	99
12.9 EPUBToTEI Class Reference	99
12.9.1 Detailed Description	100
12.9.2 Constructor & Destructor Documentation	100
12.9.2.1 __init__()	100
12.9.3 Member Function Documentation	100
12.9.3.1 _prune_bad_tags()	100
12.9.3.2 _sanitize_ids()	101
12.9.3.3 clean_tei()	101
12.9.3.4 convert_to_tei()	101
12.9.4 Member Data Documentation	101
12.9.4.1 clean_tei_content	101
12.9.4.2 encoding	101
12.9.4.3 epub_path	101
12.9.4.4 pandoc_xml_path	102
12.9.4.5 raw_tei_content	102
12.9.4.6 tei_path	102
12.9.4.7 xml_namespace	102
12.10 Log.Failure Class Reference	102
12.10.1 Detailed Description	103
12.10.2 Constructor & Destructor Documentation	103
12.10.2.1 __init__()	103
12.10.3 Member Function Documentation	104
12.10.3.1 __str__()	104
12.10.4 Member Data Documentation	104
12.10.4.1 msg	104
12.10.4.2 prefix	104
12.11 GraphConnector Class Reference	104
12.11.1 Detailed Description	107
12.11.2 Constructor & Destructor Documentation	107
12.11.2.1 __init__()	107
12.11.3 Member Function Documentation	107
12.11.3.1 _execute_retag_db()	107
12.11.3.2 _fetch_latest()	108
12.11.3.3 _parsable_to_df()	109
12.11.3.4 _returns_data()	109
12.11.3.5 _split_combined()	110
12.11.3.6 change_database()	110
12.11.3.7 check_connection()	110
12.11.3.8 create_database()	111
12.11.3.9 database_exists()	111

12.11.3.10 delete_dummy()	112
12.11.3.11 drop_database()	112
12.11.3.12 drop_graph()	112
12.11.3.13 execute_query()	113
12.11.3.14 get_dataframe()	113
12.11.3.15 get_unique()	114
12.11.3.16 graph_exists()	114
12.11.3.17 IS_DUMMY_()	116
12.11.3.18 NOT_DUMMY_()	116
12.11.3.19 SAME_DB_KG_()	116
12.11.3.20 temp_graph()	116
12.11.3.21 test_operations()	117
12.11.4 Member Data Documentation	117
12.11.4.1 __graph_name	117
12.11.4.2 connection_string	117
12.11.4.3 database_name	117
12.11.4.4 verbose	118
12.12 KnowledgeGraph Class Reference	118
12.12.1 Detailed Description	119
12.12.2 Constructor & Destructor Documentation	119
12.12.2.1 __init__()	119
12.12.3 Member Function Documentation	120
12.12.3.1 add_triple()	120
12.12.3.2 add_triples_json()	120
12.12.3.3 detect_community_clusters()	121
12.12.3.4 find_element_names()	121
12.12.3.5 get_all_triples()	122
12.12.3.6 get_by_ranked_degree()	122
12.12.3.7 get_community_subgraph()	123
12.12.3.8 get_degree_range()	123
12.12.3.9 get_edge_counts()	124
12.12.3.10 get_neighborhood()	125
12.12.3.11 get_node_context()	125
12.12.3.12 get_random_walk_sample()	126
12.12.3.13 get_relation_summary()	126
12.12.3.14 get_subgraph_by_nodes()	126
12.12.3.15 get_summary_stats()	127
12.12.3.16 get_triple_properties()	127
12.12.3.17 print_nodes()	128
12.12.3.18 print_triples()	128
12.12.3.19 to_contextualized_string()	128
12.12.3.20 to_narrative()	129

12.12.3.21 <code>to_triples_string()</code>	129
12.12.3.22 <code>triples_to_names()</code>	130
12.12.4 Member Data Documentation	130
12.12.4.1 <code>_first_insert</code>	130
12.12.4.2 <code>database</code>	131
12.12.4.3 <code>graph_name</code>	131
12.12.4.4 <code>verbose</code>	131
12.13 LLMConnector Class Reference	131
12.13.1 Detailed Description	133
12.13.2 Constructor & Destructor Documentation	133
12.13.2.1 <code>__init__()</code>	133
12.13.3 Member Function Documentation	133
12.13.3.1 <code>check_connection()</code>	133
12.13.3.2 <code>configure()</code>	134
12.13.3.3 <code>execute_file()</code>	134
12.13.3.4 <code>execute_full_query()</code>	134
12.13.3.5 <code>execute_query()</code>	134
12.13.3.6 <code>normalize_triples()</code>	135
12.13.3.7 <code>test_operations()</code>	135
12.13.4 Member Data Documentation	136
12.13.4.1 <code>llm</code>	136
12.13.4.2 <code>model_name</code>	136
12.13.4.3 <code>system_prompt</code>	136
12.14 Log Class Reference	136
12.14.1 Detailed Description	140
12.14.2 Member Function Documentation	140
12.14.2.1 <code>chart()</code>	140
12.14.2.2 <code>chart_message()</code>	140
12.14.2.3 <code>clear_timing_data()</code>	140
12.14.2.4 <code>dump_timing_csv()</code>	141
12.14.2.5 <code>elapsed_time()</code>	141
12.14.2.6 <code>fail()</code>	141
12.14.2.7 <code>fail_legacy()</code>	142
12.14.2.8 <code>format_call_chain()</code>	142
12.14.2.9 <code>get_merged_timing()</code>	142
12.14.2.10 <code>get_timing_summary()</code>	143
12.14.2.11 <code>print_timing_summary()</code>	143
12.14.2.12 <code>success()</code>	143
12.14.2.13 <code>success_legacy()</code>	143
12.14.2.14 <code>time()</code>	143
12.14.2.15 <code>time_message()</code>	144
12.14.2.16 <code>timer()</code>	144

12.14.3 your code here	144
12.14.3.1 warn()	144
12.14.4 Member Data Documentation	145
12.14.4.1 _timing_results	145
12.14.4.2 bad_addr	145
12.14.4.3 bad_path	145
12.14.4.4 bad_val	145
12.14.4.5 BRIGHT	145
12.14.4.6 CHART_COLOR	145
12.14.4.7 conn_abc	145
12.14.4.8 create_db	145
12.14.4.9 CYAN	146
12.14.4.10 db_conn_abc	146
12.14.4.11 db_exists	146
12.14.4.12 doc_db	146
12.14.4.13 drop_db	146
12.14.4.14 drop_gr	146
12.14.4.15 FAILURE_COLOR	146
12.14.4.16 FULL_DF	146
12.14.4.17 get_df	146
12.14.4.18 get_unique	147
12.14.4.19 good_val	147
12.14.4.20 gr_db	147
12.14.4.21 gr_rag	147
12.14.4.22 GRAY	147
12.14.4.23 GREEN	147
12.14.4.24 kg	147
12.14.4.25 msg_bad_addr	147
12.14.4.26 msg_bad_coll	147
12.14.4.27 msg_bad_df_parse	148
12.14.4.28 msg_bad_exec_f	148
12.14.4.29 msg_bad_exec_q	148
12.14.4.30 msg_bad_graph	148
12.14.4.31 msg_bad_path	148
12.14.4.32 msg_bad_table	148
12.14.4.33 msg_bad_triples	148
12.14.4.34 msg_chart_saved	148
12.14.4.35 MSG_COLOR	148
12.14.4.36 msg_compare	149
12.14.4.37 msg_db_connect	149
12.14.4.38 msg_db_current	149
12.14.4.39 msg_db_exists	149

12.14.4.40 msg_db_not_found	149
12.14.4.41 msg_elapsed_time	149
12.14.4.42 msg_fail_manage_db	149
12.14.4.43 msg_fail_manage_gr	149
12.14.4.44 msg_fail_parse	150
12.14.4.45 msg_good_coll	150
12.14.4.46 msg_good_df_parse	150
12.14.4.47 msg_good_exec_f	150
12.14.4.48 msg_good_exec_q	150
12.14.4.49 msg_good_exec_qr	150
12.14.4.50 msg_good_graph	150
12.14.4.51 msg_good_path	150
12.14.4.52 msg_good_table	151
12.14.4.53 msg_multiple_query	151
12.14.4.54 msg_none_df	151
12.14.4.55 msg_result	151
12.14.4.56 msg_success_managed_db	151
12.14.4.57 msg_success_managed_gr	151
12.14.4.58 msg_swap_db	151
12.14.4.59 msg_swap_kg	152
12.14.4.60 msg_time_dump	152
12.14.4.61 msg_unknown_error	152
12.14.4.62 pytest_db	152
12.14.4.63 RECORD_TIME	152
12.14.4.64 RED	152
12.14.4.65 rel_db	152
12.14.4.66 run_f	152
12.14.4.67 run_id	152
12.14.4.68 run_q	153
12.14.4.69 sub_gr	153
12.14.4.70 SUCCESS_COLOR	153
12.14.4.71 swap_db	153
12.14.4.72 swap_kg	153
12.14.4.73 t_dump	153
12.14.4.74 test_basic	153
12.14.4.75 test_df	153
12.14.4.76 test_info	153
12.14.4.77 test_ops	153
12.14.4.78 test_tmp_db	154
12.14.4.79 TIME_COLOR	154
12.14.4.80 USE_COLORS	154
12.14.4.81 WARNING_COLOR	154

12.14.4.82 WHITE	154
12.14.4.83 YELLOW	154
12.15 Metrics Class Reference	154
12.15.1 Detailed Description	155
12.15.2 Constructor & Destructor Documentation	155
12.15.2.1 <code>__init__()</code>	155
12.15.3 Member Function Documentation	156
12.15.3.1 <code>compute_basic_metrics()</code>	156
12.15.3.2 <code>create_summary_payload()</code>	156
12.15.3.3 <code>generate_default_metrics()</code>	156
12.15.3.4 <code>generate_example_metrics()</code>	157
12.15.3.5 <code>post_basic_metrics()</code>	158
12.15.3.6 <code>post_basic_output()</code>	158
12.15.3.7 <code>post_payload()</code>	158
12.15.4 Member Data Documentation	160
12.15.4.1 <code>HOST</code>	160
12.15.4.2 <code>PORT</code>	160
12.15.4.3 <code>timeout_seconds</code>	160
12.15.4.4 <code>url</code>	160
12.16 MetricsController Class Reference	160
12.16.1 Constructor & Destructor Documentation	161
12.16.1.1 <code>MetricsController()</code>	161
12.16.2 Member Function Documentation	161
12.16.2.1 <code>GetAll()</code>	161
12.16.2.2 <code>GetIndex()</code>	162
12.16.2.3 <code>Post()</code>	162
12.16.3 Member Data Documentation	162
12.16.3.1 <code>_hubContext</code>	162
12.16.3.2 <code>_logger</code>	162
12.16.3.3 <code>Summaries</code>	162
12.17 MetricsHub Class Reference	162
12.17.1 Constructor & Destructor Documentation	163
12.17.1.1 <code>MetricsHub()</code>	163
12.17.2 Member Function Documentation	163
12.17.2.1 <code>OnConnectedAsync()</code>	163
12.17.2.2 <code>OnDisconnectedAsync()</code>	163
12.17.3 Member Data Documentation	164
12.17.3.1 <code>_logger</code>	164
12.18 mysqlConnector Class Reference	164
12.18.1 Detailed Description	167
12.18.2 Constructor & Destructor Documentation	167
12.18.2.1 <code>__init__()</code>	167

12.18.3 Member Data Documentation	167
12.18.3.1 specific_queries	167
12.19 ParagraphStreamTEI Class Reference	167
12.19.1 Detailed Description	169
12.19.2 Constructor & Destructor Documentation	169
12.19.2.1 __init__()	169
12.19.3 Member Function Documentation	170
12.19.3.1 pre_compute_segments()	170
12.19.3.2 stream_segments()	170
12.19.4 Member Data Documentation	170
12.19.4.1 allowed_chapters	170
12.19.4.2 book_id	170
12.19.4.3 chunks	170
12.19.4.4 encoding	171
12.19.4.5 end_inclusive	171
12.19.4.6 lines	171
12.19.4.7 root	171
12.19.4.8 start_inclusive	171
12.19.4.9 story_id	171
12.19.4.10 tei_path	171
12.19.4.11 xml_namespace [1/2]	171
12.19.4.12 xml_namespace [2/2]	171
12.20 Plot Class Reference	172
12.20.1 Detailed Description	172
12.20.2 Member Function Documentation	172
12.20.2.1 time_elapsed_by_names()	172
12.21 postgresConnector Class Reference	172
12.21.1 Detailed Description	176
12.21.2 Constructor & Destructor Documentation	176
12.21.2.1 __init__()	176
12.21.3 Member Data Documentation	176
12.21.3.1 specific_queries	176
12.22 PRF1Metric Class Reference	176
12.22.1 Property Documentation	177
12.22.1.1 F1Score	177
12.22.1.2 Name	177
12.22.1.3 Precision	177
12.22.1.4 Recall	177
12.23 QAItem Class Reference	177
12.23.1 Property Documentation	177
12.23.1.1 Accuracy	177
12.23.1.2 GeneratedAnswer	177

12.23.1.3 GoldAnswer	178
12.23.1.4 IsCorrect	178
12.23.1.5 Question	178
12.24 QAMetric Class Reference	178
12.24.1 Property Documentation	178
12.24.1.1 AverageAccuracy	178
12.24.1.2 QAltems	178
12.25 RelationalConnector Class Reference	179
12.25.1 Detailed Description	181
12.25.2 Constructor & Destructor Documentation	182
12.25.2.1 __init__().	182
12.25.3 Member Function Documentation	182
12.25.3.1 _parsable_to_df()	182
12.25.3.2 _returns_data()	182
12.25.3.3 _split_combined()	183
12.25.3.4 change_database()	183
12.25.3.5 check_connection()	184
12.25.3.6 create_database()	184
12.25.3.7 database_exists()	185
12.25.3.8 drop_database()	185
12.25.3.9 execute_query()	185
12.25.3.10 from_env()	186
12.25.3.11 get_dataframe()	186
12.25.3.12 test_operations()	187
12.25.4 Member Data Documentation	187
12.25.4.1 connection_string	187
12.25.4.2 database_name	187
12.25.4.3 db_type	187
12.25.4.4 verbose	188
12.26 RelationExtractor Class Reference	188
12.26.1 Constructor & Destructor Documentation	188
12.26.1.1 __init__().	188
12.26.2 Member Function Documentation	188
12.26.2.1 extract()	188
12.26.3 Member Data Documentation	188
12.26.3.1 max_tokens	188
12.26.3.2 model	189
12.26.3.3 nlp	189
12.26.3.4 sentencizer	189
12.26.3.5 tokenizer	189
12.26.3.6 tuple_delim	189
12.27 ScalarMetric Class Reference	189

12.27.1 Property Documentation	189
12.27.1.1 Name	189
12.27.1.2 Value	189
12.28 Session Class Reference	190
12.28.1 Detailed Description	191
12.28.2 Constructor & Destructor Documentation	191
12.28.2.1 __init__()	191
12.28.3 Member Function Documentation	191
12.28.3.1 __new__()	191
12.28.4 Member Data Documentation	191
12.28.4.1 _created	191
12.28.4.2 _instance	192
12.28.4.3 _session	192
12.28.4.4 docs_db	192
12.28.4.5 graph_db	192
12.28.4.6 main_graph	192
12.28.4.7 metrics	192
12.28.4.8 relational_db	192
12.28.4.9 verbose	192
12.29 Story Class Reference	193
12.29.1 Constructor & Destructor Documentation	193
12.29.1.1 __init__()	193
12.29.2 Member Function Documentation	193
12.29.2.1 _make_single()	193
12.29.2.2 _merge_chunks()	193
12.29.2.3 pre_split_chunks()	194
12.29.2.4 stream_chunks()	194
12.29.3 Member Data Documentation	194
12.29.3.1 reader	194
12.30 StoryStreamAdapter Class Reference	194
12.30.1 Member Function Documentation	195
12.30.1.1 stream_paragraphs()	195
12.30.1.2 stream_segments()	195
12.30.1.3 stream_sentences()	196
12.31 SummaryData Class Reference	196
12.31.1 Property Documentation	196
12.31.1.1 BookID	196
12.31.1.2 BookTitle	196
12.31.1.3 GoldSummaryText	196
12.31.1.4 Metrics	196
12.31.1.5 QAResults	196
12.31.1.6 SummaryText	197

12.32 SummaryMetrics Class Reference	197
12.32.1 Member Function Documentation	197
12.32.1.1 GetDefault()	197
12.32.2 Property Documentation	197
12.32.2.1 PRF1Metrics	197
12.32.2.2 QA	197
12.32.2.3 ScalarMetrics	197
12.33 Triple Class Reference	198
12.33.1 Member Data Documentation	198
12.33.1.1 o	198
12.33.1.2 r	198
12.33.1.3 s	199
13 File Documentation	201
13.1 /home/runner/work/dsci-capstone/dsci-capstone/conftest.py File Reference	201
13.2 /home/runner/work/dsci-capstone/dsci-capstone/src/charts.py File Reference	201
13.3 /home/runner/work/dsci-capstone/dsci-capstone/src/_init__.py File Reference	202
13.4 /home/runner/work/dsci-capstone/dsci-capstone/src/components/_init__.py File Reference	202
13.5 /home/runner/work/dsci-capstone/dsci-capstone/src/connectors/_init__.py File Reference	202
13.6 /home/runner/work/dsci-capstone/dsci-capstone/src/core/_init__.py File Reference	202
13.7 /home/runner/work/dsci-capstone/dsci-capstone/tests/_init__.py File Reference	202
13.8 /home/runner/work/dsci-capstone/dsci-capstone/src/components/book_conversion.py File Reference	203
13.9 /home/runner/work/dsci-capstone/dsci-capstone/src/components/corpus.py File Reference	203
13.10 /home/runner/work/dsci-capstone/dsci-capstone/src/components/fact_storage.py File Reference	204
13.11 /home/runner/work/dsci-capstone/dsci-capstone/src/components/metrics.py File Reference	204
13.12 /home/runner/work/dsci-capstone/dsci-capstone/src/components/README.md File Reference	205
13.13 /home/runner/work/dsci-capstone/dsci-capstone/src/connectors/README.md File Reference	205
13.14 /home/runner/work/dsci-capstone/dsci-capstone/src/core/README.md File Reference	205
13.15 /home/runner/work/dsci-capstone/dsci-capstone/src/README.md File Reference	205
13.16 /home/runner/work/dsci-capstone/dsci-capstone/tests/examples-db/README.md File Reference	205
13.17 /home/runner/work/dsci-capstone/dsci-capstone/tests/README.md File Reference	205
13.18 /home/runner/work/dsci-capstone/dsci-capstone/src/components/relation_extraction.py File Reference	205
13.19 /home/runner/work/dsci-capstone/dsci-capstone/src/connectors/base.py File Reference	205
13.20 /home/runner/work/dsci-capstone/dsci-capstone/src/connectors/document.py File Reference	206
13.21 /home/runner/work/dsci-capstone/dsci-capstone/src/connectors/graph.py File Reference	206
13.22 /home/runner/work/dsci-capstone/dsci-capstone/src/connectors/l1m.py File Reference	207
13.23 /home/runner/work/dsci-capstone/dsci-capstone/src/connectors/relational.py File Reference	207
13.24 /home/runner/work/dsci-capstone/dsci-capstone/src/core/boss.py File Reference	208
13.25 /home/runner/work/dsci-capstone/dsci-capstone/src/core/context.py File Reference	208
13.26 /home/runner/work/dsci-capstone/dsci-capstone/src/core/stages.py File Reference	209
13.27 /home/runner/work/dsci-capstone/dsci-capstone/src/core/worker.py File Reference	210
13.28 /home/runner/work/dsci-capstone/dsci-capstone/src/main.py File Reference	211

13.29 /home/runner/work/dsci-capstone/dsci-capstone/src/util.py File Reference	212
13.30 /home/runner/work/dsci-capstone/dsci-capstone/tests/test_db_basic.py File Reference	212
13.31 /home/runner/work/dsci-capstone/dsci-capstone/tests/test_db_files.py File Reference	213
13.32 /home/runner/work/dsci-capstone/dsci-capstone/tests/test_kg_triples.py File Reference	213
13.33 /home/runner/work/dsci-capstone/dsci-capstone/tests/test_pipeline.py File Reference	214
13.34 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/_Imports.razor File Reference	216
13.35 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/App.razor File Reference	216
13.36 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/Layout/MainLayout.razor File Reference	216
13.37 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/Layout/NavMenu.razor File Reference	216
13.38 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/Pages/Error.razor File Reference	216
13.39 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/Pages/Graph.razor File Reference	216
13.40 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/Pages/Home.razor File Reference	216
13.41 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/Pages/Metrics.razor File Reference	216
13.42 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/Routes.razor File Reference	216
13.43 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Controllers/MetricsController.cs File Reference	216
13.44 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Hubs/MetricsHub.cs File Reference	217
13.45 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/PRF1Metric.cs File Reference	217
13.46 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/QAItem.cs File Reference	217
13.47 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/QAMetric.cs File Reference	217
13.48 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/ScalarMetric.cs File Reference	218
13.49 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/SummaryData.cs File Reference	218
13.50 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/SummaryMetrics.cs File Reference	218
Index	219

Chapter 1

Components Module Overview

This module contains all high-level pipeline functionality:

- **Corpus** — text dataset ingestion (BookSum, NarrativeQA)
- **Relation Extractor** — NLP-based relation extraction
- **Metrics** — evaluation metrics (QuestEval, BookScore, etc.)
- **Fact Storage** — Knowledge graph operations and semantic triple sanitization
- **Book Conversion** — Gutenberg preprocessing helpers (EPUB, HTML, TEI)

These components operate on data retrieved via the connectors and support the full text-processing and knowledge-graph pipeline. Each class remains independent, with dependencies injected through the global Session.

Chapter 2

Connectors Module Overview

This module provides lightweight adapters for each supported database:

- [**relational.py**](#) — MySQL/PostgreSQL connector (Factory Method via `from_env`)
- [**document.py**](#) — MongoDB connector
- [**graph.py**](#) — Neo4j connector
- [**llm.py**](#) — LLM connector via LangChain
- [**base.py**](#) — shared abstract base classes

All connectors expose a consistent operational interface but differ in their underlying database engines. They are instantiated once inside the global Session, ensuring stable configuration and preventing duplicate connections.

Chapter 3

Core Module Overview

This module contains the application's orchestration layer:

- the global **Session** (singleton)
- the **Boss** and **Worker** coordination scripts

3.1 Pattern

- **Separation of Concerns** — Session manages dependency wiring so connectors and components remain focused on their own tasks

3.2 Usage

```
```python
from src.core.context import session

session.relational_db.execute_query("SELECT 1")
session.main_graph.add_triple("Alice", "interactsWith", "Bob")
```



## Chapter 4

# Source Directory Overview

All application code lives inside this directory, organized into clear responsibility-based modules:

- **core/** — orchestration logic (Session, Boss, Worker)
- **connectors/** — adapters for relational, document, and graph databases
- **components/** — pipeline logic (NLP, metrics, corpus processing, book parsing)
- **main.py / util.py** — entry points and shared logging / error-handling utilities

This layout separates low-level adapters, high-level pipeline operations, and global orchestrators, improving maintainability and readability.



## Chapter 5

# Database Example Datasets

This directory contains small, deterministic example files used by the database-related tests. Each subfolder corresponds to a specific connector type implemented in the project:

- **relational/** – SQL queries or fixture datasets for MySQL/PostgreSQL.
- **document/** – JSON or BSON-like structures for MongoDB tests.
- **graph/** – Cypher queries for Neo4j graph tests.

These examples are intentionally minimal so tests:

- run quickly,
- avoid external dependencies,
- stay stable across environments,
- and clearly demonstrate the expected input format for each connector.

All files here are for testing only and do not represent production data.



# Chapter 6

## Test Suite Overview

This project uses pytest with `pytest-order` and `pytest-dependency` to ensure deterministic test sequencing.

Test groups:

- **test\_db\_basic / test\_db\_files** — relational, document, and graph connector tests.
- **test\_kg\_triples** — KnowledgeGraph parsing, ID to name mapping, and graph helpers.
- **examples-db, examples-llm** — minimal deterministic datasets used by the tests.

Run the suite with:

```
pytest -m "not smoke" tests/
```

Or (if Docker is set up):

```
make docker-all-dbs
make docker-test
```



# Chapter 7

## Namespace Index

### 7.1 Package List

Here are the packages with brief descriptions (if available):

BlazorApp . . . . .	21
BlazorApp.Controllers . . . . .	21
BlazorApp.Hubs . . . . .	21
BlazorApp.Models . . . . .	21
conftest . . . . .	22
src . . . . .	23
src.charts . . . . .	23
src.components . . . . .	23
src.components.book_conversion . . . . .	24
src.components.corpus . . . . .	24
src.components.fact_storage . . . . .	27
src.components.metrics . . . . .	27
src.components.relation_extraction . . . . .	29
src.connectors . . . . .	29
src.connectors.base . . . . .	30
src.connectors.document . . . . .	30
src.connectors.graph . . . . .	33
src.connectors.llm . . . . .	35
src.connectors.relational . . . . .	35
src.core . . . . .	36
src.core.boss . . . . .	36
Boss microservice for orchestrating distributed task processing . . . . .	36
src.core.context . . . . .	40
src.core.stages . . . . .	41
src.core.worker . . . . .	45
Generic Flask worker microservice for distributed task processing . . . . .	45
src.main . . . . .	51
src.util . . . . .	55
tests . . . . .	56
tests.test_db_basic . . . . .	56
tests.test_db_files . . . . .	58
tests.test_kg_triples . . . . .	61
tests.test_pipeline . . . . .	64



# Chapter 8

## Hierarchical Index

### 8.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Book . . . . .	71
Chunk . . . . .	74
ControllerBase	
MetricsController . . . . .	160
EPUBToTEI . . . . .	99
Hub	
MetricsHub . . . . .	162
KnowledgeGraph . . . . .	118
Log . . . . .	136
Metrics . . . . .	154
Plot . . . . .	172
PRF1Metric . . . . .	176
QAItem . . . . .	177
QAMetric . . . . .	178
RelationExtractor . . . . .	188
RuntimeError	
Log.Failure . . . . .	102
Log.BadAddressFailure . . . . .	69
ScalarMetric . . . . .	189
Session . . . . .	190
Story . . . . .	193
SummaryData . . . . .	196
SummaryMetrics . . . . .	197
ABC	
BookFactory . . . . .	72
StoryStreamAdapter . . . . .	194
BookStream . . . . .	73
ParagraphStreamTEI . . . . .	167
Connector . . . . .	78
DatabaseConnector . . . . .	81
DocumentConnector . . . . .	90
GraphConnector . . . . .	104
RelationalConnector . . . . .	179
mysqlConnector . . . . .	164
postgresConnector . . . . .	172
LLMConnector . . . . .	131
TypedDict	
Triple . . . . .	198



# Chapter 9

## Class Index

### 9.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">Log.BadAddressFailure</a>	Raised when a database connection string or address is invalid . . . . .	69
<a href="#">Book</a>		71
<a href="#">BookFactory</a>		72
<a href="#">BookStream</a>		73
<a href="#">Chunk</a>	Lightweight container for a span of story text . . . . .	74
<a href="#">Connector</a>	Abstract base class for external connectors . . . . .	78
<a href="#">DatabaseConnector</a>	Abstract base class for database engine connectors . . . . .	81
<a href="#">DocumentConnector</a>	Connector for MongoDB (document database) . . . . .	90
<a href="#">EPUBToTEI</a>	Converts EPUB files to XML format (TEI specification) . . . . .	99
<a href="#">Log.Failure</a>	User-facing base class for custom error handling . . . . .	102
<a href="#">GraphConnector</a>	Connector for Neo4j (graph database) . . . . .	104
<a href="#">KnowledgeGraph</a>	Manages a single graph within Neo4j . . . . .	118
<a href="#">LLMConnector</a>	Connector for prompting and returning LLM output (raw text/JSON) via LangChain . . . . .	131
<a href="#">Log</a>	Standardizes console output . . . . .	136
<a href="#">Metrics</a>	Utility class for computing and posting evaluation metrics . . . . .	154
<a href="#">MetricsController</a>		160
<a href="#">MetricsHub</a>		162
<a href="#">mysqlConnector</a>	A relational database connector configured for MySQL . . . . .	164
<a href="#">ParagraphStreamTEI</a>	Streams paragraphs from a TEI file as Chunk objects . . . . .	167
<a href="#">Plot</a>	Static plotting helpers for visualization . . . . .	172

<a href="#">postgresConnector</a>	A relational database connector configured for PostgreSQL . . . . .	172
<a href="#">PRF1Metric</a>	. . . . .	176
<a href="#">QALtem</a>	. . . . .	177
<a href="#">QAMetric</a>	. . . . .	178
<a href="#">RelationalConnector</a>	Connector for relational databases (MySQL, PostgreSQL) . . . . .	179
<a href="#">RelationExtractor</a>	. . . . .	188
<a href="#">ScalarMetric</a>	. . . . .	189
<a href="#">Session</a>	Stores active database connections and configuration settings . . . . .	190
<a href="#">Story</a>	. . . . .	193
<a href="#">StoryStreamAdapter</a>	. . . . .	194
<a href="#">SummaryData</a>	. . . . .	196
<a href="#">SummaryMetrics</a>	. . . . .	197
<a href="#">Triple</a>	. . . . .	198

# Chapter 10

## File Index

### 10.1 File List

Here is a list of all files with brief descriptions:

/home/runner/work/dsci-capstone/dsci-capstone/ <a href="#">conftest.py</a> . . . . .	201
/home/runner/work/dsci-capstone/dsci-capstone/src/ <a href="#">__init__.py</a> . . . . .	202
/home/runner/work/dsci-capstone/dsci-capstone/src/charts.py . . . . .	201
/home/runner/work/dsci-capstone/dsci-capstone/src/main.py . . . . .	211
/home/runner/work/dsci-capstone/dsci-capstone/src/util.py . . . . .	212
/home/runner/work/dsci-capstone/dsci-capstone/src/components/ <a href="#">__init__.py</a> . . . . .	202
/home/runner/work/dsci-capstone/dsci-capstone/src/components/ <a href="#">book_conversion.py</a> . . . . .	203
/home/runner/work/dsci-capstone/dsci-capstone/src/components/ <a href="#">corpus.py</a> . . . . .	203
/home/runner/work/dsci-capstone/dsci-capstone/src/components/ <a href="#">fact_storage.py</a> . . . . .	204
/home/runner/work/dsci-capstone/dsci-capstone/src/components/ <a href="#">metrics.py</a> . . . . .	204
/home/runner/work/dsci-capstone/dsci-capstone/src/components/ <a href="#">relation_extraction.py</a> . . . . .	205
/home/runner/work/dsci-capstone/dsci-capstone/src/connectors/ <a href="#">__init__.py</a> . . . . .	202
/home/runner/work/dsci-capstone/dsci-capstone/src/connectors/ <a href="#">base.py</a> . . . . .	205
/home/runner/work/dsci-capstone/dsci-capstone/src/connectors/ <a href="#">document.py</a> . . . . .	206
/home/runner/work/dsci-capstone/dsci-capstone/src/connectors/ <a href="#">graph.py</a> . . . . .	206
/home/runner/work/dsci-capstone/dsci-capstone/src/connectors/ <a href="#">llm.py</a> . . . . .	207
/home/runner/work/dsci-capstone/dsci-capstone/src/connectors/ <a href="#">relational.py</a> . . . . .	207
/home/runner/work/dsci-capstone/dsci-capstone/src/core/ <a href="#">__init__.py</a> . . . . .	202
/home/runner/work/dsci-capstone/dsci-capstone/src/core/ <a href="#">boss.py</a> . . . . .	208
/home/runner/work/dsci-capstone/dsci-capstone/src/core/ <a href="#">context.py</a> . . . . .	208
/home/runner/work/dsci-capstone/dsci-capstone/src/core/ <a href="#">stages.py</a> . . . . .	209
/home/runner/work/dsci-capstone/dsci-capstone/src/core/ <a href="#">worker.py</a> . . . . .	210
/home/runner/work/dsci-capstone/dsci-capstone/tests/ <a href="#">__init__.py</a> . . . . .	202
/home/runner/work/dsci-capstone/dsci-capstone/tests/ <a href="#">test_db_basic.py</a> . . . . .	212
/home/runner/work/dsci-capstone/dsci-capstone/tests/ <a href="#">test_db_files.py</a> . . . . .	213
/home/runner/work/dsci-capstone/dsci-capstone/tests/ <a href="#">test_kg_triples.py</a> . . . . .	213
/home/runner/work/dsci-capstone/dsci-capstone/tests/ <a href="#">test_pipeline.py</a> . . . . .	214
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/ <a href="#">_Imports.razor</a> . . . . .	216
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/ <a href="#">App.razor</a> . . . . .	216
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/ <a href="#">Routes.razor</a> . . . . .	216
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/Layout/ <a href="#">MainLayout.razor</a> . . . . .	216
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/Layout/ <a href="#">NavMenu.razor</a> . . . . .	216
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/Pages/ <a href="#">Error.razor</a> . . . . .	216

/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/Pages/ <a href="#">Graph.razor</a>	216
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/Pages/ <a href="#">Home.razor</a>	216
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/Pages/ <a href="#">Metrics.razor</a>	216
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Controllers/ <a href="#">MetricsController.cs</a>	216
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Hubs/ <a href="#">MetricsHub.cs</a>	217
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/ <a href="#">PRF1Metric.cs</a>	217
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/ <a href="#">QAItem.cs</a>	217
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/ <a href="#">QAMetric.cs</a>	217
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/ <a href="#">ScalarMetric.cs</a>	218
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/ <a href="#">SummaryData.cs</a>	218
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/ <a href="#">SummaryMetrics.cs</a>	218

# Chapter 11

## Namespace Documentation

### 11.1 BlazorApp Namespace Reference

#### Namespaces

- namespace [Controllers](#)
- namespace [Hubs](#)
- namespace [Models](#)

### 11.2 BlazorApp.Controllers Namespace Reference

#### Classes

- class [MetricsController](#)

### 11.3 BlazorApp.Hubs Namespace Reference

#### Classes

- class [MetricsHub](#)

### 11.4 BlazorApp.Models Namespace Reference

#### Classes

- class [PRF1Metric](#)
- class [QALtem](#)
- class [QAMetric](#)
- class [ScalarMetric](#)
- class [SummaryData](#)
- class [SummaryMetrics](#)

## 11.5 conftest Namespace Reference

### Functions

- None `pytest_adoption` (Any parser)
- Generator[`Session`, None, None] `session` (pytest.FixtureRequest request)
 

*Fixture to create session.*
- Generator[`RelationalConnector`, None, None] `relational_db` (`Session session`)
 

*Fixture to get relational database connection.*
- Generator[`DocumentConnector`, None, None] `docs_db` (`Session session`)
 

*Fixture to get document database connection.*
- Generator[`GraphConnector`, None, None] `graph_db` (`Session session`)
 

*Fixture to get document database connection.*
- Generator[`KnowledgeGraph`, None, None] `main_graph` (pytest.FixtureRequest request, `GraphConnector graph_db`, `Session session`)
 

*Fixture to get document database connection.*

### 11.5.1 Function Documentation

#### 11.5.1.1 `docs_db()`

```
Generator[DocumentConnector, None, None] docs_db (
 Session session)
```

Fixture to get document database connection.

#### 11.5.1.2 `graph_db()`

```
Generator[GraphConnector, None, None] graph_db (
 Session session)
```

Fixture to get document database connection.

#### 11.5.1.3 `main_graph()`

```
Generator[KnowledgeGraph, None, None] main_graph (
 pytest.FixtureRequest request,
 GraphConnector graph_db,
 Session session)
```

Fixture to get document database connection.

#### 11.5.1.4 `pytest_adoption()`

```
None pytest_adoption (
 Any parser)
```

### 11.5.1.5 relational\_db()

```
Generator[RelationalConnector, None, None] relational_db (
 Session session)
```

Fixture to get relational database connection.

### 11.5.1.6 session()

```
Generator[Session, None, None] session (
 pytest.FixtureRequest request)
```

Fixture to create session.

## 11.6 src Namespace Reference

### Namespaces

- namespace [charts](#)
- namespace [components](#)
- namespace [connectors](#)
- namespace [core](#)
- namespace [main](#)
- namespace [util](#)

## 11.7 src.charts Namespace Reference

### Classes

- class [Plot](#)

*Static plotting helpers for visualization.*

## 11.8 src.components Namespace Reference

### Namespaces

- namespace [book\\_conversion](#)
- namespace [corpus](#)
- namespace [fact\\_storage](#)
- namespace [metrics](#)
- namespace [relation\\_extraction](#)

## 11.9 src.components.book\_conversion Namespace Reference

### Classes

- class [Book](#)
- class [BookFactory](#)
- class [BookStream](#)
- class [Chunk](#)  
*Lightweight container for a span of story text.*
- class [EPUBToTEI](#)  
*Converts EPUB files to XML format (TEI specification).*
- class [ParagraphStreamTEI](#)  
*Streams paragraphs from a TEI file as Chunk objects.*
- class [Story](#)
- class [StoryStreamAdapter](#)

### Variables

- `nlp` = spacy.blank("en")
- `sentencizer` = nlp.add\_pipe("sentencizer")

### 11.9.1 Variable Documentation

#### 11.9.1.1 `nlp`

```
nlp = spacy.blank("en")
```

#### 11.9.1.2 `sentencizer`

```
sentencizer = nlp.add_pipe("sentencizer")
```

## 11.10 src.components.corpus Namespace Reference

### Functions

- [load\\_booksum \(\)](#)
- [to\\_df\\_booksum \(ds\)](#)
- [load\\_narrativeqa \(\)](#)
- [to\\_df\\_nqa \(ds\)](#)
- [normalize\\_title \(t\)](#)
- [merge\\_dataframes \(df1, df2, suffix1, suffix2, key\\_columns\)](#)
- [fuzzy\\_merge\\_titles \(df1, df2, suffix1, suffix2, key="title", threshold=90, scorer=fuzz.token\\_sort\\_ratio\)](#)

*Perform a two-way fuzzy merge between two DataFrames on a text column (e.g., book titles).*

## Variables

- `df_booksum = load_booksum()`
- `df_nqa = load_narrativeqa()`
- `df = fuzzy_merge_titles(df_booksum, df_nqa, "_booksum", "_nqa", key="title", threshold=70)`
- `index`
- `m = Metrics()`

## 11.10.1 Function Documentation

### 11.10.1.1 `fuzzy_merge_titles()`

```
fuzzy_merge_titles (
 df1,
 df2,
 suffix1,
 suffix2,
 key = "title",
 threshold = 90,
 scorer = fuzz.token_sort_ratio)
```

Perform a two-way fuzzy merge between two DataFrames on a text column (e.g., book titles).

For each row in the left DataFrame, the function searches the right DataFrame for the most similar string in the specified key column using RapidFuzz. It returns a merged DataFrame containing the best matches above a similarity threshold.

#### Parameters

<code>df1</code>	The left-hand DataFrame containing a text column to match on.
<code>df2</code>	The right-hand DataFrame containing a text column to match against.
<code>suffix1</code>	The name of the left-hand column.
<code>suffix2</code>	The name of the right-hand column.
<code>key</code>	The name of the column containing the strings to compare (default: "title").
<code>threshold</code>	Minimum similarity score (0–100) required to consider a match valid. Defaults to 90.
<code>scorer</code>	A RapidFuzz scoring function such as <code>fuzz.token_sort_ratio</code> or <code>fuzz.token_set_ratio</code> .

#### Returns

A new pandas DataFrame containing the compared strings, score, and all other columns.

#### Note

This function performs a one-to-one best match per left row. To ensure only confident matches are kept, adjust the `threshold` parameter.

### 11.10.1.2 `load_booksum()`

```
load_booksum ()
```

#### 11.10.1.3 `load_narrativeqa()`

```
load_narrativeqa ()
```

#### 11.10.1.4 `merge_dataframes()`

```
merge_dataframes (
 df1,
 df2,
 suffix1,
 suffix2,
 key_columns)
```

#### 11.10.1.5 `normalize_title()`

```
normalize_title (
 t)
```

#### 11.10.1.6 `to_df_booksum()`

```
to_df_booksum (
 ds)
```

#### 11.10.1.7 `to_df_nqa()`

```
to_df_nqa (
 ds)
```

### 11.10.2 Variable Documentation

#### 11.10.2.1 `df`

```
df = fuzzy_merge_titles\(df_booksum, df_nqa, "_booksum", "_nqa", key="title", threshold=70\)
```

#### 11.10.2.2 `df_booksum`

```
df_booksum = load_booksum\(\)
```

#### 11.10.2.3 `df_nqa`

```
df_nqa = load_narrativeqa\(\)
```

### 11.10.2.4 index

```
index
```

### 11.10.2.5 m

```
m = Metrics()
```

## 11.11 src.components.fact\_storage Namespace Reference

### Classes

- class [KnowledgeGraph](#)  
*Manages a single graph within Neo4j.*
- class [Triple](#)

## 11.12 src.components.metrics Namespace Reference

### Classes

- class [Metrics](#)  
*Utility class for computing and posting evaluation metrics.*

### Functions

- Dict[str, Any] [run\\_questeval](#) (Dict[str, Any] chunk, \*str qeval\_task="summarization", bool use\_cuda=False, bool use\_question\_weighter=True)  
*Run QuestEval metric calculation.*
- Dict[str, Any] [run\\_bookscore](#) (Dict[str, Any] chunk, \*str model="gpt-3.5-turbo", int batch\_size=10, bool use\_v2=True)  
*Run BoookScore metric for long-form summarization.*
- str [chunk\\_bookscore](#) (str book\_text, str book\_title='book', int chunk\_size=2048)  
*Chunk a book into BoookScore segments.*

### 11.12.1 Function Documentation

#### 11.12.1.1 [chunk\\_bookscore\(\)](#)

```
str chunk_bookscore (
 str book_text,
 str book_title = 'book',
 int chunk_size = 2048)
```

Chunk a book into BoookScore segments.

Standardizes long-form input into chunks BoookScore can process. Creates a temporary directory and writes chunked pickle for later scoring. This step can be reused independently for multiple summaries.

**Parameters**

<i>book_text</i>	Full book text to be chunked.
<i>book_title</i>	Name or identifier for the book (default 'book').
<i>chunk_size</i>	Maximum chunk size for book text (default 2048).

**Returns**

Path to chunked pickle file containing BoookScore-ready segments.

**Exceptions**

<i>RuntimeError</i>	If BoookScore chunking fails.
---------------------	-------------------------------

**11.12.1.2 run\_bookscore()**

```
Dict[str, Any] run_bookscore (
 Dict[str, Any] chunk,
 *str model = "gpt-3.5-turbo",
 int batch_size = 10,
 bool use_v2 = True)
```

Run BoookScore metric for long-form summarization.

LLM-based coherence evaluation using BoookScore. Runs in CLI via subprocess. Handles full workflow: scoring summary, postprocessing. Can be run on a single chunk or entire book (if already chunked).

**Parameters**

<i>chunk</i>	MongoDB document containing: <ul style="list-style-type: none"> <li>summary: Generated summary (required)</li> <li>text: Full or partial book text (required)</li> <li>book_title: Book title for identification (optional, for pickling)</li> </ul>
<i>model</i>	Model name (optional, default 'gpt-4')
<i>batch_size</i>	Sentences per batch for v2 (optional, default 10)
<i>use_v2</i>	Use batched evaluation (optional, default True)

**Returns**

Dict containing a score (range 0-1) and metadata for the provided summary. bookscore: Coherence score for one summary. annotations: True if a gold reference summary was provided. model\_used: String describing the LLM model and API used.

**Exceptions**

<i>KeyError</i>	If required fields are missing from chunk.
<i>RuntimeError</i>	If subprocess execution fails.

### 11.12.1.3 run\_questeval()

```
Dict[str, Any] run_questeval (
 Dict[str, Any] chunk,
 *str qeval_task = "summarization",
 bool use_cuda = False,
 bool use_question_weighter = True)
```

Run QuestEval metric calculation.

Question-answering based evaluation. Generates questions from source/reference, and checks if answers can be found in the summary. For more parameters, see: [https://github.com/ThomasScialom/QuestEval/blob/main/questeval/questeval\\_metric.py](https://github.com/ThomasScialom/QuestEval/blob/main/questeval/questeval_metric.py)

#### Parameters

<i>chunk</i>	MongoDB document containing keys: <ul style="list-style-type: none"> <li>summary: Generated summary (required)</li> <li>text: Source document text (required)</li> <li>gold_summary: Reference summary (optional, filters for better questions)</li> </ul>
<i>qeval_task</i>	Task performed by QuestEval (optional, default is summarization). Must be one of the following: generation / nlg, qa, dialogue, data2text, translation.
<i>use_cuda</i>	Run transformers with GPU enabled.
<i>use_question_weighter</i>	Make some questions more important based on relevancy.

#### Returns

Dict containing a score (range 0-1) and metadata for the provided summary. *questeval\_score*: Overall semantic precision-recall score for one example (a Summary to evaluate, Source text, and Reference summary). *has\_reference*: True if a gold reference summary was provided.

#### Exceptions

<i>ImportError</i>	If questeval package not installed.
<i>KeyError</i>	If required fields are missing from chunk.

## 11.13 src.components.relation\_extraction Namespace Reference

### Classes

- class [RelationExtractor](#)

## 11.14 src.connectors Namespace Reference

### Namespaces

- namespace [base](#)

- namespace [document](#)
- namespace [graph](#)
- namespace [llm](#)
- namespace [relational](#)

## 11.15 src.connectors.base Namespace Reference

### Classes

- class [Connector](#)  
*Abstract base class for external connectors.*
- class [DatabaseConnector](#)  
*Abstract base class for database engine connectors.*

## 11.16 src.connectors.document Namespace Reference

### Classes

- class [DocumentConnector](#)  
*Connector for MongoDB (document database)*

### Functions

- [MongoHandle mongo\\_handle](#) (str host, str alias)  
*Establish a temporary connection to MongoDB.*
- [DataFrame \\_flatten\\_recursive](#) (DataFrame df)  
*Explode all list columns and flatten dict columns until only scalars remain.*
- [str \\_sanitize\\_json](#) (str text)  
*Remove comments and other non-JSON content from a MongoDB query string.*
- [Dict\[str, Any\] \\_sanitize\\_document](#) (Dict[str, Any] doc, Dict[str, Set[Type[Any]]] type\_registry)  
*Normalize document fields to consistent types for DataFrame construction.*
- [DataFrame \\_docs\\_to\\_df](#) (List[Dict[str, Any]] docs, bool merge\_unspecified=True)  
*Convert raw MongoDB documents to a Pandas DataFrame.*
- [str \\_find\\_compatible\\_nested\\_key](#) (Type[Any] value\_type, Dict[str, Set[Type[Any]]] nested\_schema, bool merge\_unspecified)  
*Find a nested column compatible with the given primitive type.*

### Variables

- [MongoHandle](#) = Generator["Database[Any]", None, None]

## 11.16.1 Function Documentation

### 11.16.1.1 `_docs_to_df()`

```
DataFrame _docs_to_df (
 List[Dict[str, Any]] docs,
 bool merge_unspecified = True) [protected]
```

Convert raw MongoDB documents to a Pandas DataFrame.

Handles schema inconsistencies by:

1. First pass: identify all nested column names and their types
2. Second pass: sanitize and wrap primitives using type-compatible nested columns
3. Flatten structures into final DataFrame

#### Parameters

<code>docs</code>	List of MongoDB documents to convert.
<code>merge_unspecified</code>	If True, merge primitives into type-compatible nested columns using aggressive type casting (int→float, bool→int→float). If False, keep as <code>_unspecified_type</code> columns.

#### Exceptions

<code>Log.Failure</code>	If parsing query results to JSON fails.
--------------------------	-----------------------------------------

### 11.16.1.2 `_find_compatible_nested_key()`

```
str _find_compatible_nested_key (
 Type[Any] value_type,
 Dict[str, Set[Type[Any]]] nested_schema,
 bool merge_unspecified) [protected]
```

Find a nested column compatible with the given primitive type.

Uses type compatibility hierarchy for aggressive merging: bool → int → float (numeric types) str (isolated, only matches str) Searches for exact match first, then compatible types.

#### Parameters

<code>value_type</code>	The type of the primitive value to map (e.g., str, int, float).
<code>nested_schema</code>	Dict mapping nested keys to sets of observed types.
<code>merge_unspecified</code>	Whether to attempt type-compatible merging.

#### Returns

The nested key name to use for wrapping the primitive.

### 11.16.1.3 `_flatten_recursive()`

```
DataFrame _flatten_recursive (
 DataFrame df) [protected]
```

Explode all list columns and flatten dict columns until only scalars remain.

Recursive Process:

1. Find columns containing lists → explode to create new rows
2. Find columns containing dicts → normalize to create new columns
3. Repeat until no lists or dicts remain

Parameters

<code>df</code>	DataFrame with potentially nested structures.
-----------------	-----------------------------------------------

Returns

Fully flattened DataFrame with only scalar values.

### 11.16.1.4 `_sanitize_document()`

```
Dict[str, Any] _sanitize_document (
 Dict[str, Any] doc,
 Dict[str, Set[Type[Any]]] type_registry) [protected]
```

Normalize document fields to consistent types for DataFrame construction.

Converts all field values to lists and tracks type patterns.

- ObjectId → string
- Single value → [value]
- Mixed types tracked in type\_registry for conflict resolution

Parameters

<code>doc</code>	MongoDB document to sanitize.
<code>type_registry</code>	Tracks observed types per field path (e.g., {"effects": {str, list}}).

Returns

Document with all fields as lists.

### 11.16.1.5 `_sanitize_json()`

```
str _sanitize_json (
 str text) [protected]
```

Remove comments and other non-JSON content from a MongoDB query string.

Removes the following elements:

- Block comments /\* ... \*/
- Single-line comments //
- Half-line comments ... //
- Trailing commas before closing braces
- Newlines and whitespace Preserves bad text inside JSON string values.

#### Parameters

<i>text</i>	Raw text that may contain comments.
-------------	-------------------------------------

#### Returns

Cleaned text suitable for JSON parsing.

### 11.16.1.6 mongo\_handle()

```
MongoHandle mongo_handle (
 str host,
 str alias)
```

Establish a temporary connection to MongoDB.

#### Parameters

<i>host</i>	A valid MongoDB connection string.
<i>alias</i>	A unique name for the usage of this connection.

Allows scoped access to the low-level PyMongo handle from MongoEngine. Usage: with mongo\_handle(host=self.connection\_string, alias="create\_db") as db: (your code here...) This will disconnect all connections on the alias once finished. Helpful when test\_operations wants to call execute\_query, but continue using its existing db handle after execute\_query disconnects.

## 11.16.2 Variable Documentation

### 11.16.2.1 MongoHandle

```
MongoHandle = Generator["Database[Any]", None, None]
```

## 11.17 src.connectors.graph Namespace Reference

#### Classes

- class [GraphConnector](#)  
*Connector for Neo4j (graph database).*

## Functions

- DataFrame [\\_filter\\_to\\_db](#) (DataFrame df, str database\_name)  
*Filter a DataFrame by database context.*
- DataFrame [\\_tuples\\_to\\_df](#) (List[Tuple[Any,...]] tuples, List[str] meta)  
*Convert Neo4j query results (nodes and relationships) into a Pandas DataFrame.*
- DataFrame [\\_normalize\\_elements](#) (DataFrame df)  
*Convert Neo4j query results (nodes and relationships) into a Pandas DataFrame.*

### 11.17.1 Function Documentation

#### 11.17.1.1 [\\_filter\\_to\\_db\(\)](#)

```
DataFrame _filter_to_db (
 DataFrame df,
 str database_name) [protected]
```

Filter a DataFrame by database context.

- Keeps nodes where 'db' matches the current database name and `_init` is False/absent.
- Keeps relationships if either endpoint node (by `elementId`) matches the same db.
- Works on unflattened frames where cells are dicts (node/rel maps).
- Safely ignores missing fields; drops a top-level '`_init`' column if present.

#### Parameters

<code>df</code>	DataFrame containing node and relationship rows.
<code>database_name</code>	The name of the current pseudo-database.

#### Returns

Filtered DataFrame restricted to the active database.

#### 11.17.1.2 [\\_normalize\\_elements\(\)](#)

```
DataFrame _normalize_elements (
 DataFrame df) [protected]
```

Convert Neo4j query results (nodes and relationships) into a Pandas DataFrame.

- Accepts the DataFrame output of [src.connectors.graph.GraphConnector.execute\\_query](#).
- Explodes dict-cast elements from columns into rows, resulting in 1 node or relation per row.
- Normalizes node and relation properties as columns. `element_id`, `element_type` are shared.
- Node-only properties (e.g. labels) are None for relationships, and likewise for relations (e.g. `start_node`).
- Returns an empty DataFrame for no results.

**Parameters**

<code>df</code>	DataFrame containing dict-cast nodes and relationships.
-----------------	---------------------------------------------------------

**Returns**

DataFrame suitable for downstream filtering and analysis.

**11.17.1.3 `_tuples_to_df()`**

```
DataFrame _tuples_to_df (
 List[Tuple[Any, ...]] tuples,
 List[str] meta) [protected]
```

Convert Neo4j query results (nodes and relationships) into a Pandas DataFrame.

- Accepts the `tuples` output of `db.cypher_query()`.
- Automatically unwraps NeoModel Node/Relationship objects into plain dicts via `.__properties__`.
- Adds an 'element\_type' property distinguishing nodes vs relationships. Example Query: MATCH (a)-[r]->(b) RETURN a AS node\_1, r AS edge, b AS node\_2; Result: DataFrame with `node_1` and `node_2` columns containing Nodes converted to dicts, and an `edge` column containing a dict-cast relationship (`element_type: relation`).

**Parameters**

<code>tuples</code>	List of Neo4j query result tuples.
<code>meta</code>	List of element aliases returned by the query, and used here as column names.

**Returns**

DataFrame with requested columns.

**11.18 src.connectors.llm Namespace Reference****Classes**

- class [LLMConnector](#)  
*Connector for prompting and returning LLM output (raw text/JSON) via LangChain.*

**11.19 src.connectors.relational Namespace Reference****Classes**

- class [mysqlConnector](#)  
*A relational database connector configured for MySQL.*
- class [postgresConnector](#)  
*A relational database connector configured for PostgreSQL.*
- class [RelationalConnector](#)  
*Connector for relational databases (MySQL, PostgreSQL).*

## 11.20 src.core Namespace Reference

### Namespaces

- namespace `boss`  
*Boss microservice for orchestrating distributed task processing.*
- namespace `context`
- namespace `stages`
- namespace `worker`  
*Generic Flask worker microservice for distributed task processing.*

## 11.21 src.core.boss Namespace Reference

Boss microservice for orchestrating distributed task processing.

### Functions

- Dict[str, str] `load_worker_config` (List[str] `task_types`)  
*Load worker service URLs from environment variables.*
- None `clear_task_data` (MongoHandle `mongo_db`, str `collection_name`, str `chunk_id`, str `task_name`)  
*Clear any existing task data before assigning new task to worker.*
- bool `assign_task_to_worker` (str `worker_url`, str `database_name`, str `collection_name`, str `chunk_id`)  
*Assign a task to a worker microservice.*
- Flask `create_app` (DocumentConnector `docs_db`, str `database_name`, str `collection_name`, Dict[str, str] `worker_urls`)  
*Create and configure Flask application for boss service.*
- None `create_boss_thread` (str `DB_NAME`, int `BOSS_PORT`, str `COLLECTION`)
- requests.models.Response `post_story_status` (int `boss_port`, int `story_id`, str `task`, str `status`)  
*Helpers to interact with the Flask boss thread.*
- requests.models.Response `post_chunk_status` (int `boss_port`, str `chunk_id`, int `story_id`, str `task`, str `status`)  
*Send a chunk-level update to the boss Flask app.*
- requests.models.Response `post_process_full_story` (int `boss_port`, int `story_id`, str `task_type`)  
*Process all chunks in MongoDB matching the provided story ID.*

### Variables

- MongoHandle = Generator["Database[Any]", None, None]

### 11.21.1 Detailed Description

Boss microservice for orchestrating distributed task processing.

Manages task distribution to workers and tracks completion order.

### 11.21.2 Function Documentation

#### 11.21.2.1 assign\_task\_to\_worker()

```
bool assign_task_to_worker (
 str worker_url,
 str database_name,
 str collection_name,
 str chunk_id)
```

Assign a task to a worker microservice.

**Parameters**

<i>worker_url</i>	Full URL of the worker's /start endpoint.
<i>database_name</i>	Name of the MongoDB database to use.
<i>collection_name</i>	The name of our primary chunk storage collection in Mongo.
<i>chunk_id</i>	Unique identifier for the chunk within the story.

**Returns**

True if task was successfully assigned, False otherwise.

**11.21.2.2 clear\_task\_data()**

```
None clear_task_data (
 MongoHandle mongo_db,
 str collection_name,
 str chunk_id,
 str task_name)
```

Clear any existing task data before assigning new task to worker.

**Parameters**

<i>mongo_db</i>	MongoDB database handle.
<i>collection_name</i>	The name of our primary chunk storage collection in Mongo.
<i>chunk_id</i>	Unique identifier for the chunk within the story.
<i>task_name</i>	Name of the task to clear.

**11.21.2.3 create\_app()**

```
Flask create_app (
 DocumentConnector docs_db,
 str database_name,
 str collection_name,
 Dict[str, str] worker_urls)
```

Create and configure Flask application for boss service.

**Parameters**

<i>docs_db</i>	MongoDB connector class.
<i>database_name</i>	Name of the MongoDB database to use.
<i>collection_name</i>	The name of our primary chunk storage collection in Mongo.
<i>worker_urls</i>	Dictionary mapping task names to worker URLs.

**Returns**

Configured Flask application instance.

**11.21.2.4 create\_boss\_thread()**

```
None create_boss_thread (
 str DB_NAME,
 int BOSS_PORT,
 str COLLECTION)
```

**11.21.2.5 load\_worker\_config()**

```
Dict[str, str] load_worker_config (
 List[str] task_types)
```

Load worker service URLs from environment variables.

**Parameters**

<i>task_types</i>	List of valid task keys to use when searching the .env
-------------------	--------------------------------------------------------

**Returns**

Dictionary mapping task names to worker URLs.

**11.21.2.6 post\_chunk\_status()**

```
requests.models.Response post_chunk_status (
 int boss_port,
 str chunk_id,
 int story_id,
 str task,
 str status)
```

Send a chunk-level update to the boss Flask app.

**Parameters**

<i>boss_port</i>	Port the boss microservice is running on.
<i>chunk_id</i>	Unique identifier for the chunk.
<i>story_id</i>	Unique identifier for the story.
<i>task</i>	Task name (extraction, load_to_mongo, etc.).
<i>status</i>	Status (pending, assigned, in-progress, completed, failed).

**Returns**

JSON response indicating success or failure.

### 11.21.2.7 post\_process\_full\_story()

```
requests.models.Response post_process_full_story (
 int boss_port,
 int story_id,
 str task_type)
```

Process all chunks in MongoDB matching the provided story ID.

#### Parameters

<i>boss_port</i>	Port the boss microservice is running on.
<i>story_id</i>	Unique identifier for the story.
<i>task_type</i>	Worker name (questeval, bookscore).

#### Returns

JSON response indicating success or failure.

### 11.21.2.8 post\_story\_status()

```
requests.models.Response post_story_status (
 int boss_port,
 int story_id,
 str task,
 str status)
```

Helpers to interact with the Flask boss thread.

Used to process our set of example books on pipeline start.

Send a story-level update to the boss Flask app.

#### Parameters

<i>boss_port</i>	Port the boss microservice is running on.
<i>story_id</i>	Unique identifier for the story.
<i>task</i>	Task name (extraction, load_to_mongo, etc.).
<i>status</i>	Status (pending, assigned, in-progress, completed, failed).

#### Returns

JSON response indicating success or failure.

## 11.21.3 Variable Documentation

### 11.21.3.1 MongoHandle

```
MongoHandle = Generator["Database[Any]", None, None]
```

## 11.22 src.core.context Namespace Reference

### Classes

- class [Session](#)  
*Stores active database connections and configuration settings.*

### Functions

- [Session get\\_session](#) (\*Any args, \*\*Any kwargs)  
*Lazily creates a session on first call, otherwise returns the existing session.*
- Any [\\_\\_getattr\\_\\_](#) (str name)  
*Lazy attribute resolution for module-level imports.*

### Variables

- [Session session](#)  
*The global instance of the singleton Session class.*

### 11.22.1 Function Documentation

#### 11.22.1.1 [\\_\\_getattr\\_\\_\(\)](#)

```
Any __getattr__ (
 str name)
```

Lazy attribute resolution for module-level imports.

- Only called when normal attribute lookup fails (i.e., name not in module globals).
- Enables lazy session creation: `from src.core.context import session`
- Regular imports (Session, get\_session, etc.) bypass this entirely.

#### Parameters

<code>name</code>	The attribute name being accessed.
-------------------	------------------------------------

#### Returns

The session singleton if 'session' is requested.

#### Exceptions

<code>AttributeError</code>	If an unknown/undefined attribute is requested.
-----------------------------	-------------------------------------------------

### 11.22.1.2 get\_session()

```
Session get_session (
 *Any args,
 **Any kwargs)
```

Lazily creates a session on first call, otherwise returns the existing session.

#### Note

Will ignore any arguments passed after creation.

#### Parameters

<code>*args</code>	Positional arguments forwarded to Session().
<code>**kwargs</code>	Keyword arguments forwarded to Session().

#### Returns

The global instance of the Session class.

## 11.22.2 Variable Documentation

### 11.22.2.1 session

```
Session session
```

The global instance of the singleton Session class.

## 11.23 src.core.stages Namespace Reference

### Functions

- str [task\\_01\\_convert\\_epub](#) (str epub\_path, Optional[[EPUBToTEI](#)] converter=None)  
*Will revisit later - Book classes need refactoring #####.*
- [task\\_02\\_parse\\_chapters](#) (tei\_path, book\_chapters, book\_id, story\_id, start\_str, end\_str)
- [task\\_03\\_chunk\\_story](#) (story, max\_chunk\_length=1500)
- [task\\_10\\_random\\_chunk](#) (chunks)
- [task\\_10\\_sample\\_chunks](#) (chunks, n\_sample)
- [task\\_11\\_send\\_chunk](#) (c, collection\_name, book\_title)
- [task\\_12\\_relation\\_extraction\\_rebel](#) (text, max\_tokens=1024, parse\_tuples=True)
- [task\\_13\\_concatenate\\_triples](#) (extracted)
- [task\\_14\\_relation\\_extraction\\_llm](#) (triples\_string, text)
- str [task\\_15\\_sanitize\\_triples\\_llm](#) (str llm\_output)
- [task\\_20\\_send\\_triples](#) (triples)
- [group\\_21\\_1\\_describe\\_graph](#) (top\_n=3)
- [group\\_21\\_2\\_send\\_statistics](#) ()
- [group\\_21\\_3\\_post\\_statistics](#) ()
- [task\\_22\\_verbalize\\_triples](#) (mode="triple")

- `task_30_summarize_llm` (`triples_string`)  
*Prompt LLM to generate summary.*
- `task_31_send_summary` (`summary, collection_name, chunk_id`)
- `task_40_post_summary` (`book_id, book_title, summary`)  
*Send book info to Blazor.*
- `task_40_post_payload` (`book_id, book_title, summary, gold_summary, chunk, bookscore, questeval`)  
*Send metrics to Blazor.*

## 11.23.1 Function Documentation

### 11.23.1.1 `group_21_1_describe_graph()`

```
group_21_1_describe_graph (
 top_n = 3)
```

### 11.23.1.2 `group_21_2_send_statistics()`

```
group_21_2_send_statistics ()
```

### 11.23.1.3 `group_21_3_post_statistics()`

```
group_21_3_post_statistics ()
```

### 11.23.1.4 `task_01_convert_epub()`

```
str task_01_convert_epub (
 str epub_path,
 Optional[EPUBToTEI] converter = None)
```

Will revisit later - Book classes need refactoring ###.

### 11.23.1.5 `task_02_parse_chapters()`

```
task_02_parse_chapters (
 tei_path,
 book_chapters,
 book_id,
 story_id,
 start_str,
 end_str)
```

### 11.23.1.6 `task_03_chunk_story()`

```
task_03_chunk_story (
 story,
 max_chunk_length = 1500)
```

**11.23.1.7 task\_10\_random\_chunk()**

```
task_10_random_chunk (
 chunks)
```

**11.23.1.8 task\_10\_sample\_chunks()**

```
task_10_sample_chunks (
 chunks,
 n_sample)
```

**11.23.1.9 task\_11\_send\_chunk()**

```
task_11_send_chunk (
 c,
 collection_name,
 book_title)
```

**11.23.1.10 task\_12\_relation\_extraction\_rebel()**

```
task_12_relation_extraction_rebel (
 text,
 max_tokens = 1024,
 parse_tuples = True)
```

**11.23.1.11 task\_13\_concatenate\_triples()**

```
task_13_concatenate_triples (
 extracted)
```

**11.23.1.12 task\_14\_relation\_extraction\_llm()**

```
task_14_relation_extraction_llm (
 triples_string,
 text)
```

**11.23.1.13 task\_15\_sanitize\_triples\_llm()**

```
str task_15_sanitize_triples_llm (
 str llm_output)
```

**11.23.1.14 task\_20\_send\_triples()**

```
task_20_send_triples (
 triples)
```

#### 11.23.1.15 task\_22\_verbalize\_triples()

```
task_22_verbalize_triples (
 mode = "triple")
```

#### 11.23.1.16 task\_30\_summarize\_llm()

```
task_30_summarize_llm (
 triples_string)
```

Prompt LLM to generate summary.

#### 11.23.1.17 task\_31\_send\_summary()

```
task_31_send_summary (
 summary,
 collection_name,
 chunk_id)
```

#### 11.23.1.18 task\_40\_post\_payload()

```
task_40_post_payload (
 book_id,
 book_title,
 summary,
 gold_summary,
 chunk,
 bookscore,
 questeval)
```

Send metrics to Blazor.

- Compute basic metrics (ROUGE, BERTScore)
- Wait for advanced metrics (QuestEval, BoookScore)
- Post to Blazor metrics page

#### 11.23.1.19 task\_40\_post\_summary()

```
task_40_post_summary (
 book_id,
 book_title,
 summary)
```

Send book info to Blazor.

- Post to Blazor metrics page

## 11.24 src.core.worker Namespace Reference

Generic Flask worker microservice for distributed task processing.

### Functions

- None `process_task` (`MongoHandle mongo_db, str collection_name, str chunk_id, str task_name, Dict[str, Any] chunk_doc, str boss_url, Callable[[Dict[str, Any]], Dict[str, Any]] task_handler, Any task_kwargs=None)`  
*Perform the assigned task in a background thread.*
- str `load_mongo_config` (`str database`)  
*Load MongoDB configuration from environment variables.*
- str `load_boss_config` ()  
*Load boss service callback URL from environment variables.*
- Tuple[Callable[[Dict[str, Any]], Dict[str, Any]], Dict[str, Any]] `get_task_info` (`str task_name`)  
*Dynamically import and return the appropriate task handler function.*
- `load_imports` (`func`)  
*Pre-warm the task by importing requirements.*
- None `mark_task_in_progress` (`MongoHandle mongo_db, str collection_name, str chunk_id, str task_name`)  
*Mark a task as in-progress in MongoDB before processing begins.*
- None `save_task_result` (`MongoHandle mongo_db, str collection_name, str chunk_id, str task_name, Dict[str, Any] result`)  
*Save completed task results to MongoDB.*
- None `notify_boss` (`str boss_url, str chunk_id, str task_name, str status`)  
*Send completion notification to boss service.*
- Flask `create_app` (`str task_name, str boss_url`)  
*Create and configure Flask application for task processing.*

### Variables

- `MongoHandle` = Generator["Database[Any]", None, None]
- `parser` = `argparse.ArgumentParser(description="Flask worker microservice")`
- `required`
- `True`
- `help`
- `args` = `parser.parse_args()`
- str `task_queue` = `Queue()`
- `target`
- `task_worker` ()  
*Background threading system for non-blocking task handling.*
- `daemon`
- str `boss_url` = `load_boss_config()`
- `PORT` = `int(os.environ[f'{args.task.upper()}_PORT'])`
- Flask `app` = `create_app(args.task, boss_url)`
- `host`
- `port`
- `use_reloader`

### 11.24.1 Detailed Description

Generic Flask worker microservice for distributed task processing.

Supports multiple task types via command-line arguments and dynamic imports.

## 11.24.2 Function Documentation

### 11.24.2.1 create\_app()

```
Flask create_app (
 str task_name,
 str boss_url)
```

Create and configure Flask application for task processing.

#### Parameters

<i>task_name</i>	Type of task this worker will process.
<i>boss_url</i>	Callback URL for the boss service.

#### Returns

Configured Flask application instance.

### 11.24.2.2 get\_task\_info()

```
Tuple[Callable[[Dict[str, Any]], Dict[str, Any]], Dict[str, Any]] get_task_info (
 str task_name)
```

Dynamically import and return the appropriate task handler function.

#### Parameters

<i>task_name</i>	Name of the task type to execute.
------------------	-----------------------------------

#### Returns

Callable that processes the task data and returns results.

#### Exceptions

<i>ImportError</i>	If the task module cannot be imported.
<i>AttributeError</i>	If the task function is not found in the module.

### 11.24.2.3 load\_boss\_config()

```
str load_boss_config ()
```

Load boss service callback URL from environment variables.

#### Returns

Full callback URL for the boss service.

**Exceptions**

<i>KeyError</i>	If PYTHON_HOST environment variable is missing.
-----------------	-------------------------------------------------

**11.24.2.4 load\_imports()**

```
load_imports (
 func)
```

Pre-warm the task by importing requirements.

**Parameters**

<i>func</i>	The function to perform a dummy call on.
-------------	------------------------------------------

**11.24.2.5 load\_mongo\_config()**

```
str load_mongo_config (
 str database)
```

Load MongoDB configuration from environment variables.

**Parameters**

<i>database</i>	Name of the MongoDB database to connect to.
-----------------	---------------------------------------------

**Returns**

MongoDB connection string.

**Exceptions**

<i>KeyError</i>	If required environment variables are missing.
-----------------	------------------------------------------------

**11.24.2.6 mark\_task\_in\_progress()**

```
None mark_task_in_progress (
 MongoHandle mongo_db,
 str collection_name,
 str chunk_id,
 str task_name)
```

Mark a task as in-progress in MongoDB before processing begins.

**Parameters**

<i>mongo_db</i>	MongoDB database instance.
<i>collection_name</i>	The name of our primary chunk storage collection in Mongo.
<i>chunk_id</i>	Unique identifier for the chunk within the story.
<i>task_name</i>	Name of the task being executed.

**Exceptions**

<i>RuntimeError</i>	If task data already exists (preventing overwrites).
---------------------	------------------------------------------------------

**11.24.2.7 notify\_boss()**

```
None notify_boss (
 str boss_url,
 str chunk_id,
 str task_name,
 str status)
```

Send completion notification to boss service.

**Parameters**

<i>boss_url</i>	Callback URL for the boss service.
<i>chunk_id</i>	Unique identifier for the chunk within the story.
<i>task_name</i>	Name of the completed task.
<i>status</i>	Task completion status ('completed' or 'failed').

**11.24.2.8 process\_task()**

```
None process_task (
 MongoHandle mongo_db,
 str collection_name,
 str chunk_id,
 str task_name,
 Dict[str, Any] chunk_doc,
 str boss_url,
 Callable[[Dict[str, Any]], Dict[str, Any]] task_handler,
 Any task_kwargs = None)
```

Perform the assigned task in a background thread.

This includes updating task status, running the handler, saving results, and notifying the boss service when complete.

**Parameters**

<i>mongo_db</i>	MongoDB database instance.
<i>collection_name</i>	The name of the target MongoDB collection.

**Parameters**

<i>chunk_id</i>	Unique identifier for the chunk within the story.
<i>task_name</i>	Name of the task being executed.
<i>chunk_doc</i>	Document data for the current chunk.
<i>boss_url</i>	Callback URL for the boss service.
<i>task_handler</i>	Function that performs the actual task computation.
<i>task_kwargs</i>	Dict of configuration settings for each task.

**Exceptions**

<i>Exception</i>	Logs and reports failures to the boss service.
------------------	------------------------------------------------

**11.24.2.9 save\_task\_result()**

```
None save_task_result (
 MongoHandle mongo_db,
 str collection_name,
 str chunk_id,
 str task_name,
 Dict[str, Any] result)
```

Save completed task results to MongoDB.

**Parameters**

<i>mongo_db</i>	MongoDB database instance.
<i>collection_name</i>	The name of our primary chunk storage collection in Mongo.
<i>chunk_id</i>	Unique identifier for the chunk within the story.
<i>task_name</i>	Name of the task that was executed.
<i>result</i>	Dictionary containing task results to be stored.

**11.24.3 Variable Documentation****11.24.3.1 app**

```
Flask app = create_app(args.task, boss_url)
```

**11.24.3.2 args**

```
args = parser.parse_args()
```

**11.24.3.3 boss\_url**

```
str boss_url = load_boss_config()
```

**11.24.3.4 daemon**

```
daemon
```

**11.24.3.5 help**

```
help
```

**11.24.3.6 host**

```
host
```

**11.24.3.7 MongoHandle**

```
MongoHandle = Generator["Database[Any]", None, None]
```

**11.24.3.8 parser**

```
parser = argparse.ArgumentParser(description="Flask worker microservice")
```

**11.24.3.9 PORT**

```
PORT = int(os.environ[f"{args.task.upper()}__PORT"])
```

**11.24.3.10 port**

```
port
```

**11.24.3.11 required**

```
required
```

**11.24.3.12 target**

```
target
```

**11.24.3.13 task\_queue**

```
str task_queue = Queue()
```

**11.24.3.14 task\_worker**

```
task_worker ()
```

Background threading system for non-blocking task handling.

Allows Flask to immediately respond to the boss service (202: accepted) while processing continues asynchronously in a separate thread.

Continuously process tasks from the global queue in the background.

Each task runs sequentially (or with limited concurrency if multiple workers are started).

## Exceptions

<i>Exception</i>	Logs any runtime errors that occur during task execution.
------------------	-----------------------------------------------------------

### 11.24.3.15 True

True

### 11.24.3.16 use\_reloader

use\_reloader

## 11.25 src.main Namespace Reference

### Functions

- **pipeline\_A** (epub\_path, book\_chapters, start\_str, end\_str, *book\_id*, *story\_id*)  
*Connects all components to convert an EPUB file to a book summary.*
- **pipeline\_B** (collection\_name, *chunks*, *book\_title*)  
*Extracts triples from a random chunk.*
- **pipeline\_C** (json\_triples)  
*Generates a LLM summary using Neo4j triples.*
- **pipeline\_D** (collection\_name, triples\_string, chunk\_id)  
*Generate chunk summary.*
- None **pipeline\_E** (str *summary*, str *book\_title*, str *book\_id*, str *chunk*='', str *gold\_summary*='', float *bookscore*=None, float *questeval*=None)  
*Compute metrics and send available data to Blazor.*
- **full\_pipeline** (collection\_name, epub\_path, book\_chapters, start\_str, end\_str, *book\_id*, *story\_id*, *book\_title*)
- **old\_main** (collection\_name)

### Variables

- **DB\_NAME** = os.environ["DB\_NAME"]
- **BOSS\_PORT** = int(os.environ["PYTHON\_PORT"])
- **COLLECTION** = os.environ["COLLECTION\_NAME"]
- bool **load\_from\_checkpoint** = False
- str **checkpoint\_path** = "./datasets/checkpoint.pkl"
- **exist\_ok**
- int **story\_id** = 1
- int **book\_id** = 2
- str **book\_title** = "The Phoenix and the Carpet"
- **data** = pickle.load(f\_read)
- **triples** = **data**["triples"]
- **chunk** = **data**["chunk"]
- **chunks**
- **chunk\_id** = **chunk**.get\_chunk\_id()
- **triples\_string** = **pipeline\_C**(**triples**)
- **summary** = **pipeline\_D**(**COLLECTION**, **triples\_string**, **chunk.get\_chunk\_id()**)
- **response** = post\_process\_full\_story(**BOSS\_PORT**, **story\_id**, **task\_type**)

## 11.25.1 Function Documentation

### 11.25.1.1 full\_pipeline()

```
full_pipeline (
 collection_name,
 epub_path,
 book_chapters,
 start_str,
 end_str,
 book_id,
 story_id,
 book_title)
```

### 11.25.1.2 old\_main()

```
old_main (
 collection_name)
```

### 11.25.1.3 pipeline\_A()

```
pipeline_A (
 epub_path,
 book_chapters,
 start_str,
 end_str,
 book_id,
 story_id)
```

Connects all components to convert an EPUB file to a book summary.

Data conversions:

- EPUB file
- XML (TEI)

### 11.25.1.4 pipeline\_B()

```
pipeline_B (
 collection_name,
 chunks,
 book_title)
```

Extracts triples from a random chunk.

- JSON triples (NLP & LLM)

### 11.25.1.5 pipeline\_C()

```
pipeline_C (
 json_triples)
```

Generates a LLM summary using Neo4j triples.

- Neo4j graph database
- Blazor graph page

### 11.25.1.6 pipeline\_D()

```
pipeline_D (
 collection_name,
 triples_string,
 chunk_id)
```

Generate chunk summary.

### 11.25.1.7 pipeline\_E()

```
None pipeline_E (
 str summary,
 str book_title,
 str book_id,
 str chunk = "",
 str gold_summary = "",
 float bookscore = None,
 float questeval = None)
```

Compute metrics and send available data to Blazor.

## 11.25.2 Variable Documentation

### 11.25.2.1 book\_id

```
int book_id = 2
```

### 11.25.2.2 book\_title

```
str book_title = "The Phoenix and the Carpet"
```

### 11.25.2.3 BOSS\_PORT

```
BOSS_PORT = int(os.environ["PYTHON_PORT"])
```

#### 11.25.2.4 checkpoint\_path

```
str checkpoint_path = "./datasets/checkpoint.pkl"
```

#### 11.25.2.5 chunk

```
chunk = data["chunk"]
```

#### 11.25.2.6 chunk\_id

```
chunk_id = chunk.get_chunk_id()
```

#### 11.25.2.7 chunks

```
chunks
```

##### Initial value:

```
00001 = pipeline_A(
00002 epub_path="./tests/examples-pipeline/epub/trilogy-wishes-2.epub",
00003 book_chapters=,
00004 start_str="",
00005 end_str="end of the Phoenix and the Carpet.",
00006 book_id=book_id,
00007 story_id=story_id,
00008)
```

#### 11.25.2.8 COLLECTION

```
COLLECTION = os.environ["COLLECTION_NAME"]
```

#### 11.25.2.9 data

```
data = pickle.load(f_read)
```

#### 11.25.2.10 DB\_NAME

```
DB_NAME = os.environ["DB_NAME"]
```

#### 11.25.2.11 exist\_ok

```
exist_ok
```

#### 11.25.2.12 load\_from\_checkpoint

```
bool load_from_checkpoint = False
```

### 11.25.2.13 response

```
response = post_process_full_story(BOSS_PORT, story_id, task_type)
```

### 11.25.2.14 story\_id

```
int story_id = 1
```

### 11.25.2.15 summary

```
summary = pipeline_D(COLLECTION, triples_string, chunk.get_chunk_id())
```

### 11.25.2.16 triples

```
triples = data["triples"]
```

### 11.25.2.17 triples\_string

```
triples_string = pipeline_C(triples)
```

## 11.26 src.util Namespace Reference

### Classes

- class [Log](#)

*The Log class standardizes console output.*

### Functions

- DataFrame [df\\_natural\\_sorted](#) (DataFrame df, List[str] ignored\_columns=[], List[str] sort\_columns=[])  
*Sort a DataFrame in natural order using only certain columns.*
- bool [check\\_values](#) (List[Any] results, List[Any] expected, bool verbose, str log\_source, bool raise\_error)  
*Safely compare two lists of values.*

### 11.26.1 Function Documentation

#### 11.26.1.1 check\_values()

```
bool check_values (
 List[Any] results,
 List[Any] expected,
 bool verbose,
 str log_source,
 bool raise_error)
```

Safely compare two lists of values.

Helper for [src.connectors.relational.RelationalConnector.test\\_operations](#)

**Parameters**

<code>results</code>	A list of observed values from the database.
<code>expected</code>	A list of correct values to compare against.
<code>verbose</code>	Whether to print success messages.
<code>log_source</code>	The Log class prefix indicating which method is performing the check.
<code>raise_error</code>	Whether to raise an error on connection failure.

**Exceptions**

<code>Log.Failure</code>	If any result does not match what was expected.
--------------------------	-------------------------------------------------

**11.26.1.2 df\_natural\_sorted()**

```
DataFrame df_natural_sorted (
 DataFrame df,
 List[str] ignored_columns = [],
 List[str] sort_columns = [])
```

Sort a DataFrame in natural order using only certain columns.

- Column order is alphabetic too, for completely predictable behavior.
- The provided DataFrame will not be modified, since `inplace=False` by default.
- Existing row numbers will be deleted and regenerated to match the sorted order.

**Parameters**

<code>df</code>	The DataFrame containing unsorted rows.
<code>ignored_columns</code>	A list of column names to NOT sort by.
<code>sort_columns</code>	A list of column names to sort by FIRST.

**11.27 tests Namespace Reference****Namespaces**

- namespace `test_db_basic`
- namespace `test_db_files`
- namespace `test_kg_triples`
- namespace `test_pipeline`

**11.28 tests.test\_db\_basic Namespace Reference****Functions**

- None `test_db_relational_minimal (RelationalConnector relational_db)`

- Tests if the RelationalConnector has a valid connection string.*
- None `test_db_docs_minimal (DocumentConnector docs_db)`  
*Tests if the DocumentConnector has a valid connection string.*
  - None `test_db_graph_minimal (GraphConnector graph_db)`  
*Tests if the GraphConnector has a valid connection string.*
  - None `test_db_relational_comprehensive (RelationalConnector relational_db)`  
*Tests if the GraphConnector is working as intended.*
  - None `test_db_docs_comprehensive (DocumentConnector docs_db)`  
*Tests if the GraphConnector is working as intended.*
  - None `test_db_graph_comprehensive (GraphConnector graph_db)`  
*Tests if the GraphConnector is working as intended.*

## 11.28.1 Function Documentation

### 11.28.1.1 test\_db\_docs\_comprehensive()

```
None test_db_docs_comprehensive (
 DocumentConnector docs_db)
```

Tests if the GraphConnector is working as intended.

### 11.28.1.2 test\_db\_docs\_minimal()

```
None test_db_docs_minimal (
 DocumentConnector docs_db)
```

Tests if the DocumentConnector has a valid connection string.

### 11.28.1.3 test\_db\_graph\_comprehensive()

```
None test_db_graph_comprehensive (
 GraphConnector graph_db)
```

Tests if the GraphConnector is working as intended.

### 11.28.1.4 test\_db\_graph\_minimal()

```
None test_db_graph_minimal (
 GraphConnector graph_db)
```

Tests if the GraphConnector has a valid connection string.

### 11.28.1.5 test\_db\_relational\_comprehensive()

```
None test_db_relational_comprehensive (
 RelationalConnector relational_db)
```

Tests if the GraphConnector is working as intended.

### 11.28.1.6 test\_db\_relational\_minimal()

```
None test_db_relational_minimal (
 RelationalConnector relational_db)
```

Tests if the RelationalConnector has a valid connection string.

## 11.29 tests.test\_db\_files Namespace Reference

### Functions

- Generator[None, None, None] [load\\_examples\\_relational](#) (RelationalConnector relational\_db)
   
*Fixture to create relational tables using engine-specific syntax.*
- None [test\\_sql\\_example\\_1](#) (RelationalConnector relational\_db, Generator[None, None, None] [load\\_examples\\_relational](#))
   
*Run queries contained within test files.*
- None [test\\_sql\\_example\\_2](#) (RelationalConnector relational\_db, Generator[None, None, None] [load\\_examples\\_relational](#))
   
*Run queries contained within test files.*
- None [test\\_mongo\\_example\\_1](#) (DocumentConnector docs\_db)
   
*Run queries contained within test files.*
- None [test\\_mongo\\_example\\_2](#) (DocumentConnector docs\_db)
   
*Run queries contained within test files.*
- None [test\\_mongo\\_example\\_3](#) (DocumentConnector docs\_db)
   
*Run queries contained within test files.*
- None [test\\_cypher\\_example\\_1](#) (GraphConnector graph\_db)
   
*Run queries contained within test files.*
- None [test\\_cypher\\_example\\_2](#) (GraphConnector graph\_db)
   
*Test social network graph with relationships and mixed query patterns.*
- None [test\\_cypher\\_example\\_3](#) (GraphConnector graph\_db)
   
*Test scene and dialogue graphs with proper isolation.*
- None [test\\_cypher\\_example\\_4](#) (GraphConnector graph\_db)
   
*Test event graph with property mutations and multi-hop traversal.*
- None [\\_exec\\_query\\_file](#) (DatabaseConnector db\_fixture, str filename, List[str] valid\_files)
   
*Run queries from a local file through the database.*

### 11.29.1 Function Documentation

#### 11.29.1.1 \_exec\_query\_file()

```
None _exec_query_file (
 DatabaseConnector db_fixture,
 str filename,
 List[str] valid_files) [protected]
```

Run queries from a local file through the database.

### Parameters

<code>db_fixture</code>	Fixture corresponding to the current session's database.
<code>filename</code>	The name of a query file (for example ./tests/example1.sql).
<code>valid_files</code>	A list of file extensions valid for this database type.

### 11.29.1.2 load\_examples\_relational()

```
Generator[None, None, None] load_examples_relational (
 RelationalConnector relational_db)
```

Fixture to create relational tables using engine-specific syntax.

### 11.29.1.3 test\_cypher\_example\_1()

```
None test_cypher_example_1 (
 GraphConnector graph_db)
```

Run queries contained within test files.

Internal errors are handled by the class itself, and ruled out earlier. Here we just assert that the received results DataFrame matches what we expected.

### 11.29.1.4 test\_cypher\_example\_2()

```
None test_cypher_example_2 (
 GraphConnector graph_db)
```

Test social network graph with relationships and mixed query patterns.

Validates comment parsing, semicolon splitting, CREATE/MERGE/MATCH, relationships with properties, and TAG\_NODES\_ with/without RETURN.

### 11.29.1.5 test\_cypher\_example\_3()

```
None test_cypher_example_3 (
 GraphConnector graph_db)
```

Test scene and dialogue graphs with proper isolation.

Validates kg property isolation using a scene graph (spatial relationships) and dialogue graph (conversation flow with object references). Tests temp\_graph context manager and filter\_valid correctness across different graph contexts.

### 11.29.1.6 test\_cypher\_example\_4()

```
None test_cypher_example_4 (
 GraphConnector graph_db)
```

Test event graph with property mutations and multi-hop traversal.

Validates MERGE property updates (2-wave assignment), relationship chains in DAG structure, consistent rel\_type with varied properties, and multi-hop path queries. Tests that properties added via SET after initial CREATE are properly stored.

### 11.29.1.7 test\_mongo\_example\_1()

```
None test_mongo_example_1 (
 DocumentConnector docs_db)
```

Run queries contained within test files.

Internal errors are handled by the class itself, and ruled out earlier. Here we just assert that the received results DataFrame matches what we expected.

### 11.29.1.8 test\_mongo\_example\_2()

```
None test_mongo_example_2 (
 DocumentConnector docs_db)
```

Run queries contained within test files.

Internal errors are handled by the class itself, and ruled out earlier. Here we just assert that the received results DataFrame matches what we expected.

### 11.29.1.9 test\_mongo\_example\_3()

```
None test_mongo_example_3 (
 DocumentConnector docs_db)
```

Run queries contained within test files.

Internal errors are handled by the class itself, and ruled out earlier. Here we just assert that the received results DataFrame matches what we expected.

### 11.29.1.10 test\_sql\_example\_1()

```
None test_sql_example_1 (
 RelationalConnector relational_db,
 Generator[None, None, None] load_examples_relational)
```

Run queries contained within test files.

Internal errors are handled by the class itself, and ruled out earlier. Here we just assert that the received results DataFrame matches what we expected.

#### Note

Uses a table-creation fixture to load / unload schema.

### 11.29.1.11 test\_sql\_example\_2()

```
None test_sql_example_2 (
 RelationalConnector relational_db,
 Generator[None, None, None] load_examples_relational)
```

Run queries contained within test files.

Internal errors are handled by the class itself, and ruled out earlier. Here we just assert that the received results DataFrame matches what we expected.

#### Note

Uses a table-creation fixture to load / unload schema.

## 11.30 tests.test\_kg\_triples Namespace Reference

### Functions

- None `test_knowledge_graph_triples` (`KnowledgeGraph main_graph`)
 

*Test KnowledgeGraph triple operations using add\_triple and get\_all\_triples.*
- Generator[`KnowledgeGraph`, `None`, `None`] `nature_scene_graph` (`KnowledgeGraph main_graph`)
 

*Create a scene graph with multiple location-based communities for testing.*
- None `test_get_subgraph_by_nodes` (`KnowledgeGraph nature_scene_graph`)
 

*Test filtering triples by specific node IDs.*
- None `test_get_neighborhood` (`KnowledgeGraph nature_scene_graph`)
 

*Test k-hop neighborhood expansion around a central node.*
- None `test_get_random_walk_sample` (`KnowledgeGraph nature_scene_graph`)
 

*Test random walk sampling starting from specified nodes.*
- None `test_get_neighborhood_comprehensive` (`KnowledgeGraph nature_scene_graph`)
 

*Comprehensive test for k-hop neighborhood expansion.*
- None `test_get_random_walk_sample_comprehensive` (`KnowledgeGraph nature_scene_graph`)
 

*Comprehensive test for random walk sampling.*
- None `test_detect_community_clusters_minimal` (`KnowledgeGraph nature_scene_graph`)
 

*Test basic community detection functionality.*
- None `test_detect_community_clusters_comprehensive` (`KnowledgeGraph nature_scene_graph`)
 

*Comprehensive test for community detection with various parameters.*
- None `test_ranked_degree` (`KnowledgeGraph nature_scene_graph`)
 

*Test filtering triples by ranked node degree.*
- None `test_ranked_degree_ties` (`KnowledgeGraph main_graph`)
 

*Test that degree ranking correctly handles ties with minimal data.*

### 11.30.1 Function Documentation

#### 11.30.1.1 nature\_scene\_graph()

```
Generator[KnowledgeGraph, None, None] nature_scene_graph (
 KnowledgeGraph main_graph)
```

Create a scene graph with multiple location-based communities for testing.

Graph structure represents a park with distinct areas:

- Playground: swings, slide, kids
- Bench area: bench, parents
- Forest: trees, rock, path
- School building: doors, windows, classroom Each area forms a natural community for GraphRAG testing.

### 11.30.1.2 test\_detect\_community\_clusters\_comprehensive()

```
None test_detect_community_clusters_comprehensive (
 KnowledgeGraph nature_scene_graph)
```

Comprehensive test for community detection with various parameters.

Tests:

- Multi-level hierarchical detection
- community\_list structure for hierarchical summarization
- Invalid method handling
- Community stability and coverage

### 11.30.1.3 test\_detect\_community\_clusters\_minimal()

```
None test_detect_community_clusters_minimal (
 KnowledgeGraph nature_scene_graph)
```

Test basic community detection functionality.

Validates that detect\_community\_clusters assigns community\_id properties to nodes and that get\_community\_←  
subgraph retrieves triples within a community. Tests both Leiden and Louvain methods.

### 11.30.1.4 test\_get\_neighborhood()

```
None test_get_neighborhood (
 KnowledgeGraph nature_scene_graph)
```

Test k-hop neighborhood expansion around a central node.

Validates that get\_neighborhood correctly finds all triples within k hops of a starting node.

### 11.30.1.5 test\_get\_neighborhood\_comprehensive()

```
None test_get_neighborhood_comprehensive (
 KnowledgeGraph nature_scene_graph)
```

Comprehensive test for k-hop neighborhood expansion.

Tests edge cases and advanced features:

- depth=0 (no expansion)
- Disconnected nodes
- Maximum depth reaching entire connected component
- Cycle handling (no infinite loops)
- Consistent results across multiple calls

### 11.30.1.6 test\_get\_random\_walk\_sample()

```
None test_get_random_walk_sample (
 KnowledgeGraph nature_scene_graph)
```

Test random walk sampling starting from specified nodes.

Validates that get\_random\_walk\_sample generates a representative subgraph by following random paths through the graph.

### 11.30.1.7 test\_get\_random\_walk\_sample\_comprehensive()

```
None test_get_random_walk_sample_comprehensive (
 KnowledgeGraph nature_scene_graph)
```

Comprehensive test for random walk sampling.

Tests edge cases and advanced features:

- Empty start\_nodes list (should sample from any node)
- Dead-end nodes (leaf nodes with no outgoing edges)
- Walk length limits are respected
- Deterministic subset property
- Stochasticity verification

### 11.30.1.8 test\_get\_subgraph\_by\_nodes()

```
None test_get_subgraph_by_nodes (
 KnowledgeGraph nature_scene_graph)
```

Test filtering triples by specific node IDs.

Validates that get\_subgraph\_by\_nodes correctly filters triples where either subject or object matches the provided node list.

### 11.30.1.9 test\_knowledge\_graph\_triples()

```
None test_knowledge_graph_triples (
 KnowledgeGraph main_graph)
```

Test KnowledgeGraph triple operations using add\_triple and get\_all\_triples.

Validates the KnowledgeGraph wrapper for semantic triple management:

- add\_triple() creates nodes and relationships
- get\_all\_triples() retrieves triples as element IDs
- get\_triple\_properties() constructs a DataFrame with element properties as columns
- triples\_to\_names() maps IDs to human-readable names

### 11.30.1.10 test\_ranked\_degree()

```
None test_ranked_degree (
 KnowledgeGraph nature_scene_graph)
```

Test filtering triples by ranked node degree.

Validates that get\_by\_ranked\_degree correctly returns triples whose endpoints belong to nodes within the specified degree rank range.

### 11.30.1.11 test\_ranked\_degree\_ties()

```
None test_ranked_degree_ties (
 KnowledgeGraph main_graph)
```

Test that degree ranking correctly handles ties with minimal data.

Verifies that nodes with equal degrees receive the same rank and querying for non-existent ranks returns empty DataFrame.

## 11.31 tests.test\_pipeline Namespace Reference

### Functions

- [book\\_data](#) (request)
 

*Fixtures.*
- [book\\_1\\_data](#) ()
 

*Example data for Book 1: Five Children and It.*
- [book\\_2\\_data](#) ()
 

*Example data for Book 2: The Phoenix and the Carpet - realistic pipeline data.*
- [test\\_job\\_01\\_convert\\_epub](#) (book\_data)
 

*Tests.*
- [test\\_job\\_02\\_parse\\_chapters](#) (book\_data)
 

*Test TEI -> Story parsing for multiple books.*
- [test\\_job\\_03\\_chunk\\_story](#) (book\_data)
 

*Test Story -> chunks splitting for multiple books.*
- [test\\_job\\_10\\_sample\\_chunks](#) (book\_data)
 

*Test sampling multiple chunks from a list.*
- [test\\_job\\_10\\_random\\_chunk](#) (book\_data)
 

*Test selecting a single random chunk.*
- [test\\_job\\_11\\_send\\_chunk](#) (docs\_db, book\_data)
 

*Test inserting chunk into MongoDB collection.*
- [test\\_job\\_13\\_concatenate\\_triples](#) (book\_data)
 

*Test converting extracted triples to newline-delimited string.*
- [test\\_job\\_15\\_sanitize\\_triples\\_llm](#) (book\_data)
 

*Test parsing LLM output JSON into triples list.*
- [test\\_job\\_20\\_send\\_triples](#) (main\_graph, book\_data)
 

*Test inserting triples into knowledge graph.*
- [test\\_job\\_21\\_describe\\_graph](#) (main\_graph, book\_data)
 

*Test generating edge count summary of knowledge graph.*

- [test\\_job\\_22\\_verbalize\\_triples](#) (*main\_graph, book\_data*)  
*Test converting high-degree triples to string format.*
- [test\\_job\\_31\\_send\\_summary](#) (*docs\_db, book\_data*)  
*Test updating chunk with summary in MongoDB.*
- [test\\_pipeline\\_A\\_minimal](#) (*book\_data*)  
*Minimal aggregate test.*
- [test\\_pipeline\\_A\\_from\\_csv](#) ()  
*Read example CSV and run pipeline\_A for each row.*
- [test\\_pipeline\\_C\\_minimal](#) (*main\_graph, book\_data*)  
*Test running pipeline\_C with smoke test data.*
- [test\\_pipeline\\_E\\_minimal\\_summary\\_only](#) (*book\_data*)  
*Test running pipeline\_E with summary-only mode.*
- [test\\_pipeline\\_E\\_minimal\\_full\\_payload](#) (*book\_data*)  
*Test running pipeline\_E with full payload including metrics.*

### 11.31.1 Function Documentation

#### 11.31.1.1 `book_1_data()`

```
book_1_data ()
```

Example data for Book 1: Five Children and It.

#### 11.31.1.2 `book_2_data()`

```
book_2_data ()
```

Example data for Book 2: The Phoenix and the Carpet - realistic pipeline data.

#### 11.31.1.3 `book_data()`

```
book_data (request)
```

Fixtures.

#### 11.31.1.4 `test_job_01_convert_epub()`

```
test_job_01_convert_epub (book_data)
```

Tests.

Test EPUB -> TEI conversion for multiple books.

### 11.31.1.5 test\_job\_02\_parse\_chapters()

```
test_job_02_parse_chapters (
 book_data)
```

Test TEI -> Story parsing for multiple books.

### 11.31.1.6 test\_job\_03\_chunk\_story()

```
test_job_03_chunk_story (
 book_data)
```

Test Story -> chunks splitting for multiple books.

### 11.31.1.7 test\_job\_10\_random\_chunk()

```
test_job_10_random_chunk (
 book_data)
```

Test selecting a single random chunk.

### 11.31.1.8 test\_job\_10\_sample\_chunks()

```
test_job_10_sample_chunks (
 book_data)
```

Test sampling multiple chunks from a list.

### 11.31.1.9 test\_job\_11\_send\_chunk()

```
test_job_11_send_chunk (
 docs_db,
 book_data)
```

Test inserting chunk into MongoDB collection.

### 11.31.1.10 test\_job\_13\_concatenate\_triples()

```
test_job_13_concatenate_triples (
 book_data)
```

Test converting extracted triples to newline-delimited string.

### 11.31.1.11 test\_job\_15\_sanitize\_triples\_llm()

```
test_job_15_sanitize_triples_llm (
 book_data)
```

Test parsing LLM output JSON into triples list.

**11.31.1.12 test\_job\_20\_send\_triples()**

```
test_job_20_send_triples (
 main_graph,
 book_data)
```

Test inserting triples into knowledge graph.

**11.31.1.13 test\_job\_21\_describe\_graph()**

```
test_job_21_describe_graph (
 main_graph,
 book_data)
```

Test generating edge count summary of knowledge graph.

**11.31.1.14 test\_job\_22\_verbalize\_triples()**

```
test_job_22_verbalize_triples (
 main_graph,
 book_data)
```

Test converting high-degree triples to string format.

**11.31.1.15 test\_job\_31\_send\_summary()**

```
test_job_31_send_summary (
 docs_db,
 book_data)
```

Test updating chunk with summary in MongoDB.

**11.31.1.16 test\_pipeline\_A\_from\_csv()**

```
test_pipeline_A_from_csv ()
```

Read example CSV and run pipeline\_A for each row.

- Excel -> Save As -> CSV (UTF-8)
- Pandas will convert all blanks to None, so we must undo using fillna.

**11.31.1.17 test\_pipeline\_A\_minimal()**

```
test_pipeline_A_minimal (
 book_data)
```

Minimal aggregate test.

Test running the aggregate pipeline\_A on a single book.

### 11.31.1.18 `test_pipeline_C_minimal()`

```
test_pipeline_C_minimal (
 main_graph,
 book_data)
```

Test running pipeline\_C with smoke test data.

### 11.31.1.19 `test_pipeline_E_minimal_full_payload()`

```
test_pipeline_E_minimal_full_payload (
 book_data)
```

Test running pipeline\_E with full payload including metrics.

### 11.31.1.20 `test_pipeline_E_minimal_summary_only()`

```
test_pipeline_E_minimal_summary_only (
 book_data)
```

Test running pipeline\_E with summary-only mode.

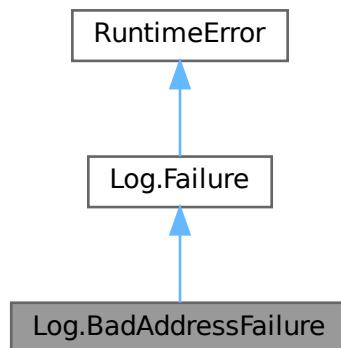
# Chapter 12

## Class Documentation

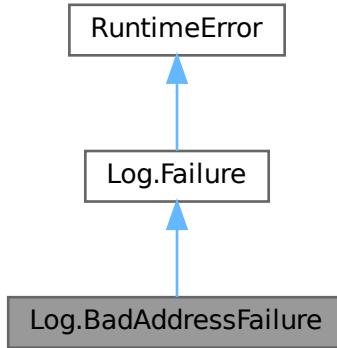
### 12.1 Log.BadAddressFailure Class Reference

Raised when a database connection string or address is invalid.

Inheritance diagram for Log.BadAddressFailure:



Collaboration diagram for Log.BadAddressFailure:



## Public Member Functions

- `__init__` (self, str source\_prefix, str connection\_string)

## Public Member Functions inherited from [Log.Failure](#)

- `__str__` (self)

## Additional Inherited Members

## Public Attributes inherited from [Log.Failure](#)

- `prefix`
- `msg`

### 12.1.1 Detailed Description

Raised when a database connection string or address is invalid.

- We support 4+ database engines and 2 endpoint frameworks (Blazor & Flask), each of which has a different error type when unable to connect.
- To avoid flooding the console with these, this error should not be chained.
- Usage: raise BadAddressFailure(source\_prefix, connection\_string) from None

### 12.1.2 Constructor & Destructor Documentation

#### 12.1.2.1 `__init__()`

```
__init__ (
 self,
 str source_prefix,
 str connection_string)
```

Reimplemented from [Log.Failure](#).

The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/src/[util.py](#)

## 12.2 Book Class Reference

### Public Member Functions

- None `__init__` (self, str title\_key="Title:", str author\_key="Author:", str language\_key="Language:", str date\_key="Release date:")
- Iterator[Tuple[str, Dict[str, Any]]] [stream\\_chapters](#) (self)

### 12.2.1 Constructor & Destructor Documentation

#### 12.2.1.1 `__init__()`

```
None __init__ (
 self,
 str title_key = "Title:",
 str author_key = "Author:",
 str language_key = "Language:",
 str date_key = "Release date:")
```

### 12.2.2 Member Function Documentation

#### 12.2.2.1 `stream_chapters()`

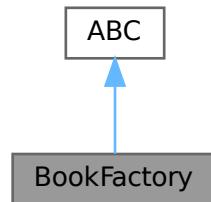
```
Iterator[Tuple[str, Dict[str, Any]]] stream_chapters (
 self)
```

The documentation for this class was generated from the following file:

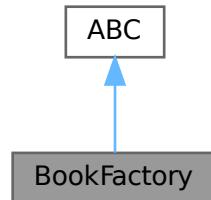
- /home/runner/work/dsci-capstone/dsci-capstone/src/components/[book\\_conversion.py](#)

## 12.3 BookFactory Class Reference

Inheritance diagram for BookFactory:



Collaboration diagram for BookFactory:



### Public Member Functions

- [Book create\\_book \(self\)](#)

#### 12.3.1 Member Function Documentation

##### 12.3.1.1 `create_book()`

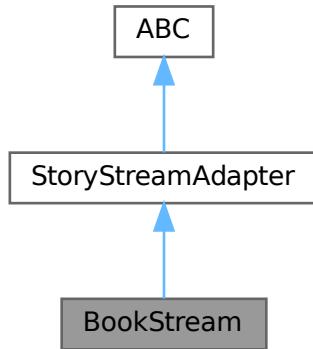
```
Book create_book (
 self)
```

The documentation for this class was generated from the following file:

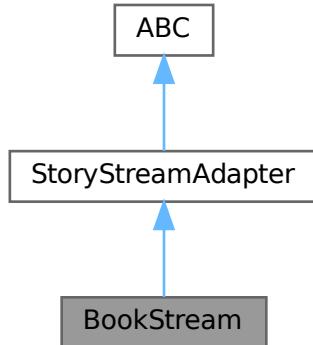
- [/home/runner/work/dsci-capstone/dsci-capstone/src/components/book\\_conversion.py](#)

## 12.4 BookStream Class Reference

Inheritance diagram for BookStream:



Collaboration diagram for BookStream:



### Public Member Functions

- None [\\_\\_init\\_\\_](#) (self, Book book)
- Iterator[Chunk] [stream\\_segments](#) (self)  
*Yields sanitized parts of a book.*

### Public Member Functions inherited from [StoryStreamAdapter](#)

- Iterator[Chunk] [stream\\_paragraphs](#) (self)  
*Concrete helper method to split segments into paragraphs.*
- Iterator[str] [stream\\_sentences](#) (self)  
*Concrete helper method to split paragraphs into sentences.*

## Public Attributes

- `book`

### 12.4.1 Constructor & Destructor Documentation

#### 12.4.1.1 `__init__()`

```
None __init__ (
 self,
 Book book)
```

### 12.4.2 Member Function Documentation

#### 12.4.2.1 `stream_segments()`

```
Iterator[Chunk] stream_segments (
 self)
```

Yields sanitized parts of a book.

- Story segments usually correspond to chapters.
- They serve as borders between chunking operations, ensuring chunks do not span multiple chapters. Implementation is handled by child classes BookStream, etc.
- Segments should be pre-cleaned and must contain 1 paragraph per line with all other newlines removed.

Reimplemented from [StoryStreamAdapter](#).

### 12.4.3 Member Data Documentation

#### 12.4.3.1 `book`

`book`

The documentation for this class was generated from the following file:

- [/home/runner/work/dsci-capstone/dsci-capstone/src/components/book\\_conversion.py](#)

## 12.5 Chunk Class Reference

Lightweight container for a span of story text.

## Public Member Functions

- None `__init__` (self, str `text`, int `book_id`, int `chapter_number`, int `line_start`, int `line_end`, int `story_id`, float `story_percent`, float `chapter_percent`, int `max_chunk_length=-1`)  
*Construct a Chunk.*
- int `char_count` (self, bool `prune_newlines=False`)  
*Computes the character count.*
- str `get_chunk_id` (self)  
*Use story ID, book ID, chapter, and chapter percentage to generate a chunk ID.*
- Dict[str, Any] `to_mongo_dict` (self)  
*Convert Chunk to Mongo document format.*
- str `__repr__` (self)

## Public Attributes

- `text`
- `book_id`
- `chapter_number`
- `line_start`
- `line_end`
- `story_id`
- `story_percent`
- `chapter_percent`

### 12.5.1 Detailed Description

Lightweight container for a span of story text.

- Carries positional metadata so downstream consumers can reconstruct context.
- Filter by `story_id` to fetch all chunks for a particular story.
- Use `story_percent` and `chapter_percent` to quickly sort chunks by intended order.
- Use `book_id`, `chapter_number`, `line_start`, and `line_end` to locate this chunk within source material.

### 12.5.2 Constructor & Destructor Documentation

#### 12.5.2.1 `__init__()`

```
None __init__ (
 self,
 str text,
 int book_id,
 int chapter_number,
 int line_start,
 int line_end,
 int story_id,
 float story_percent,
 float chapter_percent,
 int max_chunk_length = -1)
```

Construct a Chunk.

**Parameters**

<i>text</i>	The text content for this span.
<i>book_id</i>	Corresponds to a single book file in the dataset.
<i>chapter_number</i>	The chapter containing this chunk in the book file, 1-based.
<i>line_start</i>	The starting line within the TEI file, 1-based.
<i>line_end</i>	The inclusive ending line index within the TEI file ( $\geq$ <i>line_start</i> ).
<i>story_id</i>	A stable id for the overall story. May be identical to <i>book_id</i>
<i>story_percent</i>	Approximate progress through the whole story [0.0, 100.0].
<i>chapter_percent</i>	Approximate progress through the current segment [0.0, 100.0].
<i>max_chunk_length</i>	Max allowed characters (<= 0 means "no limit").

**Exceptions**

<i>ValueError</i>	if text exceeds <i>max_chunk_length</i> when <i>max_chunk_length</i> > 0.
-------------------	---------------------------------------------------------------------------

**12.5.3 Member Function Documentation****12.5.3.1 \_\_repr\_\_()**

```
str __repr__ (
 self)
```

**12.5.3.2 char\_count()**

```
int char_count (
 self,
 bool prune_newlines = False)
```

Computes the character count.

**Parameters**

<i>prune_newlines</i>	Whether to remove newlines for the count.
-----------------------	-------------------------------------------

**Returns**

The number of characters in the chunk text.

**12.5.3.3 get\_chunk\_id()**

```
str get_chunk_id (
 self)
```

Use story ID, book ID, chapter, and chapter percentage to generate a chunk ID.

**Returns**

A string uniquely identifying a chunk.

**12.5.3.4 `to_mongo_dict()`**

```
Dict[str, Any] to_mongo_dict (
 self)
```

Convert Chunk to Mongo document format.

**Returns**

A dictionary which can be easily loaded into MongoDB.

**12.5.4 Member Data Documentation****12.5.4.1 `book_id`**

```
book_id
```

**12.5.4.2 `chapter_number`**

```
chapter_number
```

**12.5.4.3 `chapter_percent`**

```
chapter_percent
```

**12.5.4.4 `line_end`**

```
line_end
```

**12.5.4.5 `line_start`**

```
line_start
```

**12.5.4.6 `story_id`**

```
story_id
```

**12.5.4.7 `story_percent`**

```
story_percent
```

#### 12.5.4.8 text

text

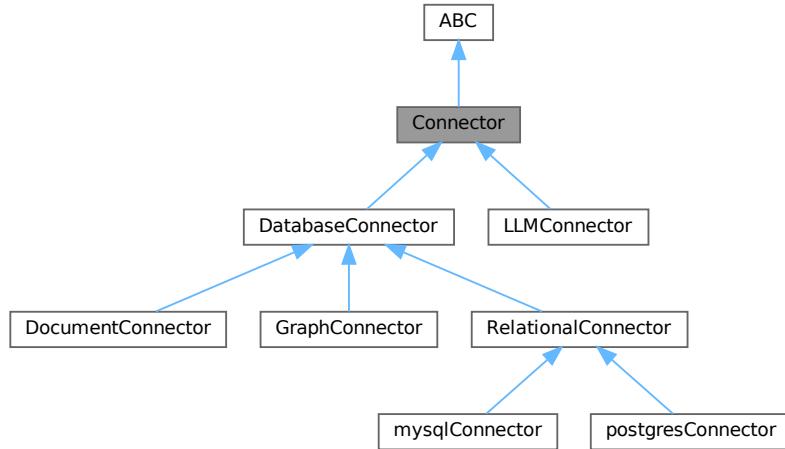
The documentation for this class was generated from the following file:

- [/home/runner/work/dsci-capstone/dsci-capstone/src/components/book\\_conversion.py](#)

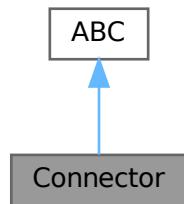
## 12.6 Connector Class Reference

Abstract base class for external connectors.

Inheritance diagram for Connector:



Collaboration diagram for Connector:



## Public Member Functions

- bool `test_operations` (self, bool `raise_error=True`)  
*Establish a basic connection to the database, and test full functionality.*
- bool `check_connection` (self, str `log_source`, bool `raise_error`)  
*Minimal connection test to determine if our connection string is valid.*
- Optional[DataFrame] `execute_query` (self, str `query`)  
*Send a single command through the connection.*
- List[Optional[DataFrame]] `execute_file` (self, str `filename`)  
*Run several commands from a file.*

### 12.6.1 Detailed Description

Abstract base class for external connectors.

#### Note

Credentials are specified in the .env file.

Derived classes should implement:

- `init`
- `src.connectors.base.Connector.test_operations`
- `src.connectors.base.Connector.execute_query`
- `src.connectors.base.Connector.execute_file`

### 12.6.2 Member Function Documentation

#### 12.6.2.1 `check_connection()`

```
bool check_connection (
 self,
 str log_source,
 bool raise_error)
```

Minimal connection test to determine if our connection string is valid.

#### Parameters

<code>log_source</code>	The Log class prefix indicating which method is performing the check.
<code>raise_error</code>	Whether to raise an error on connection failure.

#### Returns

Whether the connection test was successful.

**Exceptions**

<i>Log.Failure</i>	If <code>raise_error</code> is <code>True</code> and the connection test fails to complete.
--------------------	---------------------------------------------------------------------------------------------

Reimplemented in [LLMConnector](#), [DocumentConnector](#), [GraphConnector](#), and [RelationalConnector](#).

**12.6.2.2 execute\_file()**

```
List[Optional[DataFrame]] execute_file (
 self,
 str filename)
```

Run several commands from a file.

**Parameters**

<i>filename</i>	The path to a specified query or prompt file (.sql, .txt).
-----------------	------------------------------------------------------------

**Returns**

Whether the query was performed successfully.

Reimplemented in [DatabaseConnector](#), and [LLMConnector](#).

**12.6.2.3 execute\_query()**

```
Optional[DataFrame] execute_query (
 self,
 str query)
```

Send a single command through the connection.

**Parameters**

<i>query</i>	A single query to perform on the database.
--------------	--------------------------------------------

**Returns**

The result of the query, or `None`

Reimplemented in [DatabaseConnector](#), [DocumentConnector](#), [LLMConnector](#), [RelationalConnector](#), and [GraphConnector](#).

**12.6.2.4 test\_operations()**

```
bool test_operations (
 self,
 bool raise_error = True)
```

Establish a basic connection to the database, and test full functionality.

Can be configured to fail silently, which enables retries or external handling.

#### Parameters

<i>raise_error</i>	Whether to raise an error on connection failure.
--------------------	--------------------------------------------------

#### Returns

Whether the connection test was successful.

#### Exceptions

<i>LogFailure</i>	If <i>raise_error</i> is True and the connection test fails to complete.
-------------------	--------------------------------------------------------------------------

Reimplemented in [DocumentConnector](#), [GraphConnector](#), [LLMConnector](#), and [RelationalConnector](#).

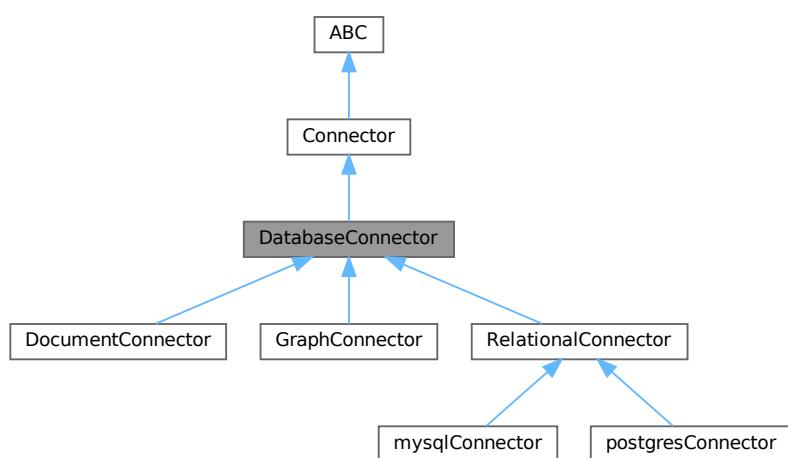
The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/src/connectors/[base.py](#)

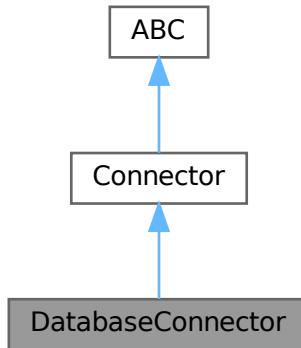
## 12.7 DatabaseConnector Class Reference

Abstract base class for database engine connectors.

Inheritance diagram for DatabaseConnector:



Collaboration diagram for DatabaseConnector:



## Public Member Functions

- None `__init__` (self, bool `verbose=False`)
 

*Initialize the connector.*
- None `configure` (self, str DB, str database\_name)
 

*Read connection settings from the .env file.*
- None `change_database` (self, str new\_database)
 

*Update the connection URI to reference a different database in the same engine.*
- Generator[None, None, None] `temp_database` (self, str database\_name)
 

*Temporarily switch to a pseudo-database, creating and dropping it if needed.*
- Optional[DataFrame] `execute_query` (self, str query)
 

*Send a single command through the connection.*
- List[Optional[DataFrame]] `execute_combined` (self, str multi\_query)
 

*Run several database commands in sequence.*
- List[Optional[DataFrame]] `execute_file` (self, str filename)
 

*Run several database commands from a file.*
- DataFrame `get_dataframe` (self, str name, List[str] columns=[])
 

*Automatically generate and run a query for the specified resource.*
- None `create_database` (self, str database\_name)
 

*Use the current database connection to create a sibling database in this engine.*
- None `drop_database` (self, str database\_name)
 

*Delete all data stored in a particular database.*
- bool `database_exists` (self, str database\_name)
 

*Search for an existing database using the provided name.*

## Public Member Functions inherited from [Connector](#)

- bool `test_operations` (self, bool raise\_error=True)
 

*Establish a basic connection to the database, and test full functionality.*
- bool `check_connection` (self, str log\_source, bool raise\_error)
 

*Minimal connection test to determine if our connection string is valid.*

## Public Attributes

- `verbose`  
*Whether to print debug messages.*
- `db_type`
- `db_engine`
- `username`
- `password`
- `host`
- `port`
- `connection_string`

## Protected Member Functions

- `bool _is_single_query (self, str query)`  
*Checks if a string contains multiple queries.*
- `List[str] _split_combined (self, str multi_query)`  
*Checks if a string contains multiple queries.*
- `bool _returns_data (self, str query)`  
*Checks if a query is structured in a way that returns real data, and not status messages.*
- `bool _parsable_to_df (self, Any result)`  
*Checks if the result of a query is valid (i.e.*

### 12.7.1 Detailed Description

Abstract base class for database engine connectors.

Derived classes should implement:

- `src.connectors.base.DatabaseConnector.__init__`
- `src.connectors.base.DatabaseConnector.test_operations`
- `src.connectors.base.DatabaseConnector.execute_query`
- `src.connectors.base.DatabaseConnector._split_combined`
- `src.connectors.base.DatabaseConnector.get_dataframe`
- `src.connectors.base.DatabaseConnector.create_database`
- `src.connectors.base.DatabaseConnector.drop_database`
- `src.connectors.base.DatabaseConnector.change_database`
- `src.connectors.base.DatabaseConnector.database_exists`

### 12.7.2 Constructor & Destructor Documentation

#### 12.7.2.1 \_\_init\_\_()

```
None __init__ (
 self,
 bool verbose = False)
```

Initialize the connector.

**Parameters**

<i>verbose</i>	Whether to print debug messages.
----------------	----------------------------------

**Note**

Attributes will be set to None until [src.connectors.base.DatabaseConnector.configure\(\)](#) is called.

Reimplemented in [RelationalConnector](#), [DocumentConnector](#), [GraphConnector](#), [mysqlConnector](#), and [postgresConnector](#).

### 12.7.3 Member Function Documentation

#### 12.7.3.1 [\\_is\\_single\\_query\(\)](#)

```
bool _is_single_query (
 self,
 str query) [protected]
```

Checks if a string contains multiple queries.

**Parameters**

<i>query</i>	A single or combined query string.
--------------	------------------------------------

**Returns**

Whether the query is single (true) or combined (false).

#### 12.7.3.2 [\\_parsable\\_to\\_df\(\)](#)

```
bool _parsable_to_df (
 self,
 Any result) [protected]
```

Checks if the result of a query is valid (i.e.

can be converted to a Pandas DataFrame).

**Parameters**

<i>result</i>	The result of a SQL, Cypher, or JSON query.
---------------	---------------------------------------------

**Returns**

Whether the object is parsable to DataFrame.

Reimplemented in [DocumentConnector](#), [GraphConnector](#), and [RelationalConnector](#).

### 12.7.3.3 `_returns_data()`

```
bool _returns_data (
 self,
 str query) [protected]
```

Checks if a query is structured in a way that returns real data, and not status messages.

#### Parameters

<i>query</i>	A single query string.
--------------	------------------------

#### Returns

Whether the query is intended to fetch data (true) or might return a status message (false).

Reimplemented in [DocumentConnector](#), [GraphConnector](#), and [RelationalConnector](#).

### 12.7.3.4 `_split_combined()`

```
List[str] _split_combined (
 self,
 str multi_query) [protected]
```

Checks if a string contains multiple queries.

#### Parameters

<i>multi_query</i>	A string containing multiple queries.
--------------------	---------------------------------------

#### Returns

A list of single-query strings.

Reimplemented in [DocumentConnector](#), [GraphConnector](#), and [RelationalConnector](#).

### 12.7.3.5 `change_database()`

```
None change_database (
 self,
 str new_database)
```

Update the connection URI to reference a different database in the same engine.

#### Parameters

<i>new_database</i>	The name of the database to connect to.
---------------------	-----------------------------------------

Reimplemented in [DocumentConnector](#), [GraphConnector](#), and [RelationalConnector](#).

#### 12.7.3.6 configure()

```
None configure (
 self,
 str DB,
 str database_name)
```

Read connection settings from the .env file.

##### Parameters

<i>DB</i>	The prefix of fetched database credentials.
<i>database_name</i>	The name of the database to connect to.

#### 12.7.3.7 create\_database()

```
None create_database (
 self,
 str database_name)
```

Use the current database connection to create a sibling database in this engine.

##### Parameters

<i>database_name</i>	The name of the new database to create.
----------------------	-----------------------------------------

##### Exceptions

<i>Log.Failure</i>	If the database already exists.
--------------------	---------------------------------

Reimplemented in [DocumentConnector](#), [GraphConnector](#), and [RelationalConnector](#).

#### 12.7.3.8 database\_exists()

```
bool database_exists (
 self,
 str database_name)
```

Search for an existing database using the provided name.

##### Parameters

<i>database_name</i>	The name of a database to search for.
----------------------	---------------------------------------

**Returns**

Whether the database is visible to this connector.

Reimplemented in [DocumentConnector](#), [GraphConnector](#), and [RelationalConnector](#).

### 12.7.3.9 drop\_database()

```
None drop_database (
 self,
 str database_name)
```

Delete all data stored in a particular database.

**Parameters**

<i>database_name</i>	The name of an existing database.
----------------------	-----------------------------------

**Exceptions**

<i>Log.Failure</i>	If the database does not exist.
--------------------	---------------------------------

Reimplemented in [DocumentConnector](#), [GraphConnector](#), and [RelationalConnector](#).

### 12.7.3.10 execute\_combined()

```
List[Optional[DataFrame]] execute_combined (
 self,
 str multi_query)
```

Run several database commands in sequence.

**Parameters**

<i>multi_query</i>	A string containing multiple queries.
--------------------	---------------------------------------

**Returns**

A list of query results converted to DataFrames.

### 12.7.3.11 execute\_file()

```
List[Optional[DataFrame]] execute_file (
 self,
 str filename)
```

Run several database commands from a file.

**Note**

Loads the entire file into memory at once.

**Parameters**

<i>filename</i>	The path to a specified query file (.sql, .cql, .json).
-----------------	---------------------------------------------------------

**Returns**

Whether the query was performed successfully.

**Exceptions**

<i>Log.Failure</i>	If any query in the file fails to execute.
--------------------	--------------------------------------------

Reimplemented from [Connector](#).

**12.7.3.12 execute\_query()**

```
Optional[DataFrame] execute_query (
 self,
 str query)
```

Send a single command through the connection.

**Note**

If a result is returned, it will be converted to a DataFrame.

**Parameters**

<i>query</i>	A single query to perform on the database.
--------------	--------------------------------------------

**Returns**

DataFrame containing the result of the query, or None

**Exceptions**

<i>Log.Failure</i>	If the query fails to execute.
--------------------	--------------------------------

Reimplemented from [Connector](#).

Reimplemented in [DocumentConnector](#), [RelationalConnector](#), and [GraphConnector](#).

**12.7.3.13 get\_dataframe()**

```
DataFrame get_dataframe (
 self,
```

```
 str name,
 List[str] columns = [])
```

Automatically generate and run a query for the specified resource.

#### Parameters

<i>name</i>	The name of an existing table or collection in the database.
<i>columns</i>	A list of column names to keep.

#### Returns

DataFrame containing the requested data

Reimplemented in [DocumentConnector](#), [GraphConnector](#), and [RelationalConnector](#).

#### 12.7.3.14 temp\_database()

```
Generator[None, None, None] temp_database (
 self,
 str database_name)
```

Temporarily switch to a pseudo-database, creating and dropping it if needed.

- If the target database does not exist, it will be created before yielding and dropped automatically afterward.
- If it already exists, it will be left intact.

#### Parameters

<i>database_name</i>	The name of the pseudo-database to use temporarily.
----------------------	-----------------------------------------------------

### 12.7.4 Member Data Documentation

#### 12.7.4.1 connection\_string

```
connection_string
```

#### 12.7.4.2 db\_engine

```
db_engine
```

#### 12.7.4.3 db\_type

```
db_type
```

#### 12.7.4.4 host

host

#### 12.7.4.5 password

password

#### 12.7.4.6 port

port

#### 12.7.4.7 username

username

#### 12.7.4.8 verbose

verbose

Whether to print debug messages.

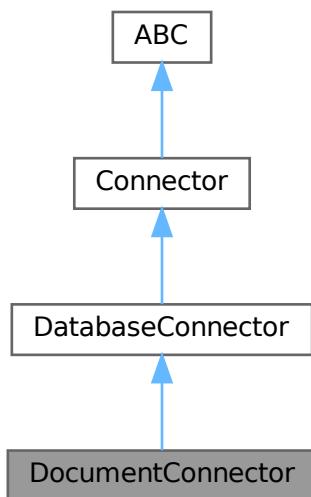
The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/src/connectors/[base.py](#)

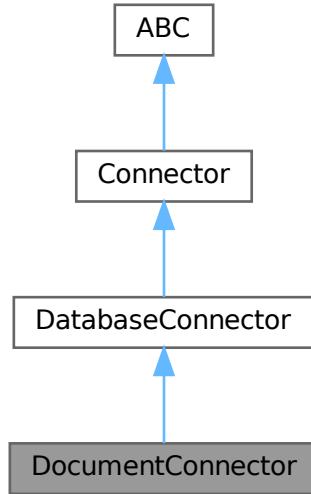
## 12.8 DocumentConnector Class Reference

Connector for MongoDB (document database)

Inheritance diagram for DocumentConnector:



Collaboration diagram for DocumentConnector:



## Public Member Functions

- None `__init__` (self, bool `verbose=False`)  
*Creates a new MongoDB connector.*
- None `change_database` (self, str `new_database`)  
*Update the connection URI to reference a different database in the same engine.*
- bool `test_operations` (self, bool `raise_error=True`)  
*Establish a basic connection to the MongoDB database, and test full functionality.*
- bool `check_connection` (self, str `log_source`, bool `raise_error=True`)  
*Minimal connection test to determine if our connection string is valid.*
- `get_unmanaged_handle` (self)  
*Expose the low-level PyMongo handle for external use.*
- Optional[DataFrame] `execute_query` (self, str `query`)  
*Send a single MongoDB command using PyMongo.*
- DataFrame `get_dataframe` (self, str `name`, List[str] `columns=[]`)  
*Automatically generate and run a query for the specified collection.*
- None `create_database` (self, str `database_name`)  
*Use the current database connection to create a sibling database in this engine.*
- None `drop_database` (self, str `database_name`)  
*Delete all data stored in a particular database.*
- bool `database_exists` (self, str `database_name`)  
*Search for an existing database using the provided name.*
- None `delete_dummy` (self)  
*Delete the initial dummy collection from the database.*

## Public Member Functions inherited from DatabaseConnector

- None `configure` (self, str DB, str database\_name)  
*Read connection settings from the .env file.*
- Generator[None, None, None] `temp_database` (self, str database\_name)  
*Temporarily switch to a pseudo-database, creating and dropping it if needed.*
- List[Optional[DataFrame]] `execute_combined` (self, str multi\_query)  
*Run several database commands in sequence.*
- List[Optional[DataFrame]] `execute_file` (self, str filename)  
*Run several database commands from a file.*

## Public Attributes

- `database_name`
- `verbose`
- `connection_string`

## Public Attributes inherited from DatabaseConnector

- `verbose`  
*Whether to print debug messages.*
- `db_type`
- `db_engine`
- `username`
- `password`
- `host`
- `port`
- `connection_string`

## Protected Member Functions

- list[str] `_split_combined` (self, str multi\_query)  
*Divides a string into non-divisible MongoDB commands by splitting on semicolons at depth 0.*
- bool `_returns_data` (self, str query)  
*Checks if a query is structured in a way that returns real data, and not status messages.*
- bool `_parsable_to_df` (self, Any result)  
*Checks if the result of a query is valid (i.e.*

## Protected Member Functions inherited from DatabaseConnector

- bool `_is_single_query` (self, str query)  
*Checks if a string contains multiple queries.*

## Protected Attributes

- `_auth_suffix`

### 12.8.1 Detailed Description

Connector for MongoDB (document database)

- Uses mongoengine.connect(...) on-demand for connections.
- Low-level operations use pymongo via mongoengine.get\_db().
- create\_database uses an init collection insertion (MongoDB is lazy).

### 12.8.2 Constructor & Destructor Documentation

#### 12.8.2.1 \_\_init\_\_()

```
None __init__ (
 self,
 bool verbose = False)
```

Creates a new MongoDB connector.

##### Parameters

<code>verbose</code>	Whether to print debug messages.
----------------------	----------------------------------

Reimplemented from [DatabaseConnector](#).

### 12.8.3 Member Function Documentation

#### 12.8.3.1 \_parsable\_to\_df()

```
bool _parsable_to_df (
 self,
 Any result) [protected]
```

Checks if the result of a query is valid (i.e.

can be converted to a Pandas DataFrame).

- Handles cursor-style dicts (with 'cursor' or 'firstBatch'), list-of-dict results, and single-document results.
- Excludes pure status/meta responses like {'ok': 1}.

##### Parameters

<code>result</code>	The result of a JSON query.
---------------------	-----------------------------

**Returns**

Whether the object is parsable to DataFrame.

Reimplemented from [DatabaseConnector](#).

### 12.8.3.2 `_returns_data()`

```
bool _returns_data (
 self,
 str query) [protected]
```

Checks if a query is structured in a way that returns real data, and not status messages.

Determines whether a MongoDB command should yield cursor data.

- Uses an exclusion list - commands that definitely return a status.
- Everything else falls through to execution for validation.

**Parameters**

<code>query</code>	A single pre-validated JSON query string.
--------------------	-------------------------------------------

**Returns**

Whether the query is intended to fetch data (true) or might return a status message (false).

Reimplemented from [DatabaseConnector](#).

### 12.8.3.3 `_split_combined()`

```
list[str] _split_combined (
 self,
 str multi_query) [protected]
```

Divides a string into non-divisible MongoDB commands by splitting on semicolons at depth 0.

Handles nested brackets and semicolons inside JSON strings.

**Parameters**

<code>multi_query</code>	A string containing multiple queries with possible comments.
--------------------------	--------------------------------------------------------------

**Returns**

A list of single-query strings (cleaned, ready for JSON parsing).

Reimplemented from [DatabaseConnector](#).

### 12.8.3.4 change\_database()

```
None change_database (
 self,
 str new_database)
```

Update the connection URI to reference a different database in the same engine.

#### Note

Additional settings are appended as a suffix to the MongoDB connection string.

#### Parameters

<code>new_database</code>	The name of the database to connect to.
---------------------------	-----------------------------------------

Reimplemented from [DatabaseConnector](#).

### 12.8.3.5 check\_connection()

```
bool check_connection (
 self,
 str log_source,
 bool raise_error = True)
```

Minimal connection test to determine if our connection string is valid.

Connect to MongoDB using `MongoEngine.connect()`

#### Parameters

<code>log_source</code>	The Log class prefix indicating which method is performing the check.
<code>raise_error</code>	Whether to raise an error on connection failure.

#### Returns

Whether the connection test was successful.

#### Exceptions

<code>Log.Failure</code>	If <code>raise_error</code> is <code>True</code> and the connection test fails to complete.
--------------------------	---------------------------------------------------------------------------------------------

Reimplemented from [Connector](#).

### 12.8.3.6 create\_database()

```
None create_database (
 self,
 str database_name)
```

Use the current database connection to create a sibling database in this engine.

#### Note

Forces MongoDB to actually create it by inserting a small init document.

#### Parameters

<code>database_name</code>	The name of the new database to create.
----------------------------	-----------------------------------------

#### Exceptions

<code>Log.Failure</code>	If we fail to create the requested database for any reason.
--------------------------	-------------------------------------------------------------

Reimplemented from [DatabaseConnector](#).

### 12.8.3.7 `database_exists()`

```
bool database_exists (
 self,
 str database_name)
```

Search for an existing database using the provided name.

#### Parameters

<code>database_name</code>	The name of a database to search for.
----------------------------	---------------------------------------

#### Returns

Whether the database is visible to this connector.

Reimplemented from [DatabaseConnector](#).

### 12.8.3.8 `delete_dummy()`

```
None delete_dummy (
 self)
```

Delete the initial dummy collection from the database.

#### Note

Call this method whenever real data is being added to avoid pollution.

### 12.8.3.9 `drop_database()`

```
None drop_database (
 self,
 str database_name)
```

Delete all data stored in a particular database.

**Parameters**

<i>database_name</i>	The name of an existing database.
----------------------	-----------------------------------

**Exceptions**

<i>Log.Failure</i>	If we fail to drop the target database for any reason.
--------------------	--------------------------------------------------------

Reimplemented from [DatabaseConnector](#).

**12.8.3.10 execute\_query()**

```
Optional[DataFrame] execute_query (
 self,
 str query)
```

Send a single MongoDB command using PyMongo.

- The query must be a valid JSON command object (e.g. {"find": "users", "filter": {...}}).
- Mongo shell syntax such as db.users.find({ ... }) or .js files will NOT work.
- If a result is returned, it will be converted to a DataFrame.

**Exceptions**

<i>Log.Failure</i>	If the query fails to execute.
--------------------	--------------------------------

Reimplemented from [DatabaseConnector](#).

**12.8.3.11 get\_dataframe()**

```
DataFrame get_dataframe (
 self,
 str name,
 List[str] columns = [])
```

Automatically generate and run a query for the specified collection.

**Parameters**

<i>name</i>	The name of an existing table or collection in the database.
<i>columns</i>	A list of column names to keep.

**Returns**

Sorted DataFrame containing the requested data

**Exceptions**

<i>Log.Failure</i>	If we fail to create the requested DataFrame for any reason.
--------------------	--------------------------------------------------------------

Reimplemented from [DatabaseConnector](#).

**12.8.3.12 get\_unmanaged\_handle()**

```
get_unmanaged_handle (
 self)
```

Expose the low-level PyMongo handle for external use.

**Warning**

Connection remains open - use for long-lived services only.

**Returns**

PyMongo database instance.

**12.8.3.13 test\_operations()**

```
bool test_operations (
 self,
 bool raise_error = True)
```

Establish a basic connection to the MongoDB database, and test full functionality.

Can be configured to fail silently, which enables retries or external handling.

**Parameters**

<i>raise_error</i>	Whether to raise an error on connection failure.
--------------------	--------------------------------------------------

**Returns**

Whether the connection test was successful.

**Exceptions**

<i>Log.Failure</i>	If <i>raise_error</i> is True and the connection test fails to complete.
--------------------	--------------------------------------------------------------------------

Reimplemented from [Connector](#).

## 12.8.4 Member Data Documentation

### 12.8.4.1 `_auth_suffix`

```
_auth_suffix [protected]
```

### 12.8.4.2 `connection_string`

```
connection_string
```

### 12.8.4.3 `database_name`

```
database_name
```

### 12.8.4.4 `verbose`

```
verbose
```

The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/src/connectors/[document.py](#)

## 12.9 EPUBToTEI Class Reference

Converts EPUB files to XML format (TEI specification).

### Public Member Functions

- None `__init__` (self, str [epub\\_path](#), bool `save_pandoc=False`, bool `save_tei=True`)  
*Initialize the converter.*
- None `convert_to_tei` (self)  
*Uses Pandoc to draft a TEI string from EPUB.*
- None `clean_tei` (self)  
*Wrap root if missing, sanitize ids, and save cleaned TEI.*

### Public Attributes

- `epub_path`
- `pandoc_xml_path`
- `raw_tei_content`
- `clean_tei_content`
- `tei_path`

## Static Public Attributes

- dict `xml_namespace` = {"tei": "http://www.tei-c.org/ns/1.0"}
- str `encoding` = "utf-8"

## Protected Member Functions

- str `_sanitize_ids` (self, str content)  
*Sanitize XML IDs in the TEI content to ensure they are valid and consistent.*
- str `_prune_bad_tags` (self, str content)  
*Replace all `<lb>` tags with newline characters in TEI.*

### 12.9.1 Detailed Description

Converts EPUB files to XML format (TEI specification).

Takes an EPUB book file and converts it to TEI in order to represent its chapter hierarchy.

### 12.9.2 Constructor & Destructor Documentation

#### 12.9.2.1 `__init__()`

```
None __init__ (
 self,
 str epub_path,
 bool save_pandoc = False,
 bool save_teis = True)
```

Initialize the converter.

##### Parameters

<code>epub_path</code>	String containing the relative path to an EPUB file.
<code>save_pandoc</code>	Flag to save the intermediate Pandoc output to .tei.xml
<code>save_teis</code>	Flag to save the final TEI file as .tei

### 12.9.3 Member Function Documentation

#### 12.9.3.1 `_prune_bad_tags()`

```
str _prune_bad_tags (
 self,
 str content) [protected]
```

Replace all `<lb>` tags with newline characters in TEI.

### 12.9.3.2 `_sanitize_ids()`

```
str _sanitize_ids (
 self,
 str content) [protected]
```

Sanitize XML IDs in the TEI content to ensure they are valid and consistent.

Pandoc sometimes generates invalid or non-unique `xml:id` attributes (e.g., containing spaces, punctuation, or mixed casing). Since we rely on these IDs as dictionary keys / anchors, we sanitize them using a regex to enforce alphanumeric/underscore/dash format.

#### Parameters

<code>content</code>	The raw TEI XML string possibly containing invalid <code>xml:id</code> attributes.
----------------------	------------------------------------------------------------------------------------

#### Returns

A TEI XML string with valid NCNames, prefixed with 'id\_'.

### 12.9.3.3 `clean_teis()`

```
None clean_teis (
 self)
```

Wrap root if missing, sanitize ids, and save cleaned TEI.

### 12.9.3.4 `convert_to_teis()`

```
None convert_to_teis (
 self)
```

Uses Pandoc to draft a TEI string from EPUB.

## 12.9.4 Member Data Documentation

### 12.9.4.1 `clean_teis_content`

```
clean_teis_content
```

### 12.9.4.2 `encoding`

```
str encoding = "utf-8" [static]
```

### 12.9.4.3 `epub_path`

```
epub_path
```

#### 12.9.4.4 pandoc\_xml\_path

pandoc\_xml\_path

#### 12.9.4.5 raw\_tei\_content

raw\_tei\_content

#### 12.9.4.6 tei\_path

tei\_path

#### 12.9.4.7 xml\_namespace

```
dict xml_namespace = {"tei": "http://www.tei-c.org/ns/1.0"} [static]
```

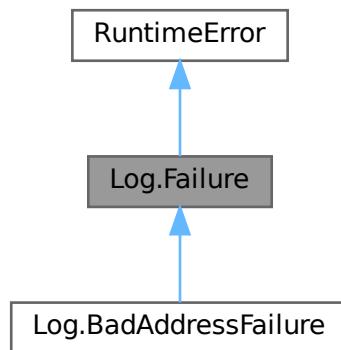
The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/src/components/[book\\_conversion.py](#)

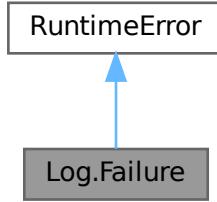
## 12.10 Log.Failure Class Reference

User-facing base class for custom error handling.

Inheritance diagram for Log.Failure:



Collaboration diagram for Log.Failure:



### Public Member Functions

- `__init__` (self, str `prefix`="ERROR: ", str `msg`=")
- `__str__` (self)

### Public Attributes

- `prefix`
- `msg`

## 12.10.1 Detailed Description

User-facing base class for custom error handling.

- Builder Pattern - User can combine and chain standard message strings from the Log class.
- Prefixes (e.g., "GRAPH DB:", "FILE IO:") are redundant with tracebacks but improve readability by highlighting the semantic source of the error - not just a line number.
- Enforces a consistent color scheme across all raised errors for quick scanning.

## 12.10.2 Constructor & Destructor Documentation

### 12.10.2.1 `__init__()`

```
__init__ (
 self,
 str prefix = "ERROR: ",
 str msg = "")
```

Reimplemented in [Log.BadAddressFailure](#).

### 12.10.3 Member Function Documentation

#### 12.10.3.1 `__str__()`

```
__str__ (
 self)
```

### 12.10.4 Member Data Documentation

#### 12.10.4.1 `msg`

`msg`

#### 12.10.4.2 `prefix`

`prefix`

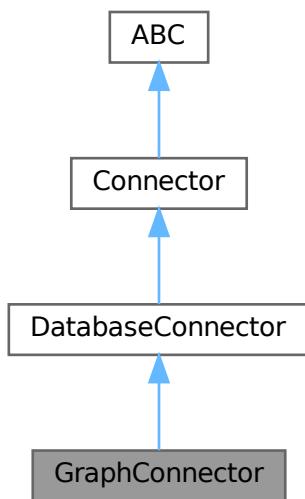
The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/src/[util.py](#)

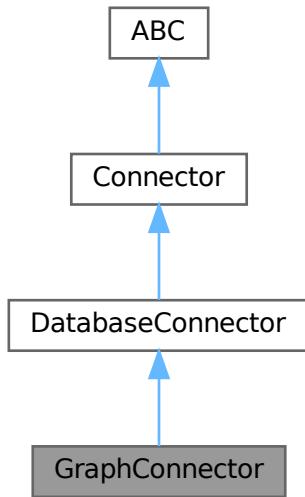
## 12.11 GraphConnector Class Reference

Connector for Neo4j (graph database).

Inheritance diagram for GraphConnector:



Collaboration diagram for GraphConnector:



## Public Member Functions

- None `__init__(self, bool verbose=False)`  
*Creates a new Neo4j connector.*
- None `change_database(self, str new_database)`  
*Update the connection URI to reference a different database in the same engine.*
- Generator[None, None, None] `temp_graph(self, str graph_name)`  
*Temporarily inspect the specified graph, then swap back when finished.*
- bool `test_operations(self, bool raise_error=True)`  
*Establish a basic connection to the Neo4j database, and test full functionality.*
- bool `check_connection(self, str log_source, bool raise_error=True)`  
*Minimal connection test to determine if our connection string is valid.*
- Optional[DataFrame] `execute_query(self, str query, bool _filter_results=True)`  
*Send a single Cypher query to Neo4j.*
- DataFrame `get_dataframe(self, str name, List[str] columns=[ ])`  
*Automatically generate and run a query for the specified Knowledge Graph collection.*
- List[str] `get_unique(self, str key)`  
*Retrieve all unique values for a specified node property.*
- None `create_database(self, str database_name)`  
*Create a fresh pseudo-database if it does not already exist.*
- None `drop_database(self, str database_name)`  
*Delete all nodes stored under a particular database name.*
- None `drop_graph(self, str graph_name)`  
*Delete all nodes stored under a particular graph name.*
- bool `database_exists(self, str database_name)`  
*Search for an existing database using the provided name.*
- bool `graph_exists(self, str graph_name)`

- None `delete_dummy` (self)
 

*Search for an existing graph using the provided name.*
- str `IS_DUMMY_` (self, str alias='n')
 

*Delete the initial dummy node from the database.*
- str `NOT_DUMMY_` (self, str alias='n')
 

*Generates Cypher code to select dummy nodes inside a WHERE clause.*
- str `SAME_DB_KG_` (self)
 

*Generates Cypher code to select non-dummy nodes inside a WHERE clause.*
- str `SAME_DB_GN_` (self)
 

*Generates a Cypher pattern dictionary to match nodes by current database and graph name.*

## Public Member Functions inherited from [DatabaseConnector](#)

- None `configure` (self, str DB, str database\_name)
 

*Read connection settings from the .env file.*
- Generator[None, None, None] `temp_database` (self, str database\_name)
 

*Temporarily switch to a pseudo-database, creating and dropping it if needed.*
- List[Optional[DataFrame]] `execute_combined` (self, str multi\_query)
 

*Run several database commands in sequence.*
- List[Optional[DataFrame]] `execute_file` (self, str filename)
 

*Run several database commands from a file.*

## Public Attributes

- `database_name`
- `verbose`
- `connection_string`

## Public Attributes inherited from [DatabaseConnector](#)

- `verbose`

*Whether to print debug messages.*
- `db_type`
- `db_engine`
- `username`
- `password`
- `host`
- `port`
- `connection_string`

## Protected Member Functions

- List[str] `_split_combined` (self, str multi\_query)
 

*Divides a string into non-divisible CQL queries, ignoring comments.*
- bool `_returns_data` (self, str query)
 

*Checks if a query is structured in a way that returns real data, and not status messages.*
- bool `_parsable_to_df` (self, Tuple[Any, Any] result)
 

*Checks if the result of a Neo4j query is valid (i.e.*
- None `_execute_retag_db` (self)
 

*Sweeps the database for untagged nodes and relationships, and adds a 'db' attribute.*
- Tuple[Optional[List[Tuple[Any,...]]], Optional[List[str]]] `_fetch_latest` (self, List[Tuple[Any,...]] results)
 

*Re-fetch nodes and edges after changing the remote copy in Neo4j.*

## Protected Member Functions inherited from [DatabaseConnector](#)

- bool [\\_is\\_single\\_query](#) (self, str query)  
*Checks if a string contains multiple queries.*

## Protected Attributes

- [\\_graph\\_name](#)

### 12.11.1 Detailed Description

Connector for Neo4j (graph database).

- Uses neomodel to abstract some operations, but raw CQL is required for many tasks.
- Neo4j does not support multiple logical databases in community edition, so we emulate them.
- This is achieved by using a 'db' property (database name) and 'kg' property (graph name) on nodes.

### 12.11.2 Constructor & Destructor Documentation

#### 12.11.2.1 [\\_\\_init\\_\\_\(\)](#)

```
None __init__ (
 self,
 bool verbose = False)
```

Creates a new Neo4j connector.

#### Parameters

<i>verbose</i>	Whether to print success and failure messages.
----------------	------------------------------------------------

Reimplemented from [DatabaseConnector](#).

### 12.11.3 Member Function Documentation

#### 12.11.3.1 [\\_execute\\_retag\\_db\(\)](#)

```
None _execute_retag_db (
 self) [protected]
```

Sweeps the database for untagged nodes and relationships, and adds a 'db' attribute.

### 12.11.3.2 `_fetch_latest()`

```
Tuple[Optional[List[Tuple[Any, ...]]], Optional[List[str]]] _fetch_latest (
 self,
 List[Tuple[Any, ...]] results) [protected]
```

Re-fetch nodes and edges after changing the remote copy in Neo4j.

**Parameters**

<i>results</i>	Original list of untagged tuples from db.cypher_query().
----------------	----------------------------------------------------------

**Returns**

Latest version of results fetched from the database.

**12.11.3.3 `_parsable_to_df()`**

```
bool _parsable_to_df (
 self,
 Tuple[Any, Any] result) [protected]
```

Checks if the result of a Neo4j query is valid (i.e.

can be converted to a Pandas DataFrame).

- Validates shape: (records, meta)
- Validates content: rows are iterable, elements are dict-like or have `.__properties__` (NeoModel Node/Rel)

**Parameters**

<i>result</i>	The result of a Cypher query.
---------------	-------------------------------

**Returns**

Whether the object is parsable to DataFrame.

Reimplemented from [DatabaseConnector](#).

**12.11.3.4 `_returns_data()`**

```
bool _returns_data (
 self,
 str query) [protected]
```

Checks if a query is structured in a way that returns real data, and not status messages.

Determines whether a Cypher query should yield records.

- RETURN must be present as a keyword (not in a string value) to return data.
- YIELD is used for stored procedures, and might return data.

**Parameters**

<i>query</i>	A single pre-validated CQL query string.
--------------	------------------------------------------

**Returns**

Whether the query is intended to fetch data (true) or might return a status message (false).

Reimplemented from [DatabaseConnector](#).

**12.11.3.5 `_split_combined()`**

```
List[str] _split_combined (
 self,
 str multi_query) [protected]
```

Divides a string into non-divisible CQL queries, ignoring comments.

**Parameters**

<i>multi_query</i>	A string containing multiple queries.
--------------------	---------------------------------------

**Returns**

A list of single-query strings.

Reimplemented from [DatabaseConnector](#).

**12.11.3.6 `change_database()`**

```
None change_database (
 self,
 str new_database)
```

Update the connection URI to reference a different database in the same engine.

**Note**

Neo4j does not accept database names routed through the connection string.

**Parameters**

<i>new_database</i>	The name of the database to connect to.
---------------------	-----------------------------------------

Reimplemented from [DatabaseConnector](#).

**12.11.3.7 `check_connection()`**

```
bool check_connection (
 self,
 str log_source,
 bool raise_error = True)
```

Minimal connection test to determine if our connection string is valid.

Connect to Neo4j executing a query: db.cypher\_query()

#### Parameters

<i>log_source</i>	The Log class prefix indicating which method is performing the check.
<i>raise_error</i>	Whether to raise an error on connection failure.

#### Returns

Whether the connection test was successful.

#### Exceptions

<i>Log.Failure</i>	If <i>raise_error</i> is True and the connection test fails to complete.
--------------------	--------------------------------------------------------------------------

Reimplemented from [Connector](#).

### 12.11.3.8 `create_database()`

```
None create_database (
 self,
 str database_name)
```

Create a fresh pseudo-database if it does not already exist.

#### Note

This change will apply to any new nodes created after [src.connectors.base.DatabaseConnector.change\\_database](#) is called.

#### Parameters

<i>database_name</i>	A database ID specifying the pseudo-database.
----------------------	-----------------------------------------------

#### Exceptions

<i>Log.Failure</i>	If we fail to create the requested database for any reason.
--------------------	-------------------------------------------------------------

Reimplemented from [DatabaseConnector](#).

### 12.11.3.9 `database_exists()`

```
bool database_exists (
 self,
 str database_name)
```

Search for an existing database using the provided name.

**Parameters**

<i>database_name</i>	The name of a database to search for.
----------------------	---------------------------------------

**Returns**

Whether the database is visible to this connector.

Reimplemented from [DatabaseConnector](#).

**12.11.3.10 delete\_dummy()**

```
None delete_dummy (
 self)
```

Delete the initial dummy node from the database.

**Note**

Never use this. Enables the existence of an "empty" database.

**12.11.3.11 drop\_database()**

```
None drop_database (
 self,
 str database_name)
```

Delete all nodes stored under a particular database name.

**Parameters**

<i>database_name</i>	A database ID specifying the pseudo-database.
----------------------	-----------------------------------------------

**Exceptions**

<i>Log.Failure</i>	If we fail to drop the target database for any reason.
--------------------	--------------------------------------------------------

Reimplemented from [DatabaseConnector](#).

**12.11.3.12 drop\_graph()**

```
None drop_graph (
 self,
 str graph_name)
```

Delete all nodes stored under a particular graph name.

**Parameters**

<i>graph_name</i>	The name of a graph in the current database.
-------------------	----------------------------------------------

**Exceptions**

<i>Log.Failure</i>	If we fail to drop the target graph for any reason.
--------------------	-----------------------------------------------------

**12.11.3.13 execute\_query()**

```
Optional[DataFrame] execute_query (
 self,
 str query,
 bool _filter_results = True)
```

Send a single Cypher query to Neo4j.

**Note**

If a result is returned, it will be converted to a DataFrame.

**Parameters**

<i>query</i>	A single query to perform on the database.
<i>_filter_results</i>	If True, limit results to the current database. Internal helper functions need unfiltered results.

**Returns**

DataFrame containing the result of the query, or None

**Exceptions**

<i>Log.Failure</i>	If the query fails to execute.
--------------------	--------------------------------

Reimplemented from [DatabaseConnector](#).

**12.11.3.14 get\_dataframe()**

```
DataFrame get_dataframe (
 self,
 str name,
 List[str] columns = [])
```

Automatically generate and run a query for the specified Knowledge Graph collection.

- Fetches all nodes and relationships belonging to the active database + graph name.

- Includes public attributes, element\_id, labels, and element\_type.
- Uses execute\_query() for DataFrame conversion and filtering.
- Does not explode lists or nested values.

**Parameters**

<i>name</i>	The name of an existing graph or subgraph.
<i>columns</i>	A list of column names to keep.

**Returns**

Sorted DataFrame containing the requested data.

**Exceptions**

<i>Log.Failure</i>	If we fail to create the requested DataFrame for any reason.
--------------------	--------------------------------------------------------------

Reimplemented from [DatabaseConnector](#).

**12.11.3.15 get\_unique()**

```
List[str] get_unique (
 self,
 str key)
```

Retrieve all unique values for a specified node property.

Queries all nodes in the database and extracts distinct values for the given key.

**Parameters**

<i>key</i>	The node property name to extract unique values from (e.g. 'db' or 'kg').
------------	---------------------------------------------------------------------------

**Returns**

A list of unique values for the specified key, or an empty list if none exist.

**Exceptions**

<i>Log.Failure</i>	If the query fails to execute.
--------------------	--------------------------------

**12.11.3.16 graph\_exists()**

```
bool graph_exists (
 self,
 str graph_name)
```

Search for an existing graph using the provided name.

**Parameters**

<code>graph_name</code>	The name of a graph to search for.
-------------------------	------------------------------------

**Returns**

Whether the graph is visible to this connector.

**12.11.3.17 IS\_DUMMY\_()**

```
str IS_DUMMY_ (
 self,
 str alias = 'n')
```

Generates Cypher code to select dummy nodes inside a WHERE clause.

Usage: MATCH (n) WHERE {self.IS\_DUMMY\_('n')};

**Returns**

A string containing Cypher code.

**12.11.3.18 NOT\_DUMMY\_()**

```
str NOT_DUMMY_ (
 self,
 str alias = 'n')
```

Generates Cypher code to select non-dummy nodes inside a WHERE clause.

Usage: MATCH (n) WHERE {self.NOT\_DUMMY\_('n')};

**Returns**

A string containing Cypher code.

**12.11.3.19 SAME\_DB\_KG\_()**

```
str SAME_DB_KG_ (
 self)
```

Generates a Cypher pattern dictionary to match nodes by current database and graph name.

Usage: MATCH (n {self.SAME\_DB\_KG\_()})

**Returns**

A string containing Cypher code.

**12.11.3.20 temp\_graph()**

```
Generator[None, None, None] temp_graph (
 self,
 str graph_name)
```

Temporarily inspect the specified graph, then swap back when finished.

**Parameters**

<i>graph_name</i>	The name of a graph in the current database.
-------------------	----------------------------------------------

**12.11.3.21 test\_operations()**

```
bool test_operations (
 self,
 bool raise_error = True)
```

Establish a basic connection to the Neo4j database, and test full functionality.

Can be configured to fail silently, which enables retries or external handling.

**Parameters**

<i>raise_error</i>	Whether to raise an error on connection failure.
--------------------	--------------------------------------------------

**Returns**

Whether the connection test was successful.

**Exceptions**

<i>LogFailure</i>	If <i>raise_error</i> is True and the connection test fails to complete.
-------------------	--------------------------------------------------------------------------

Reimplemented from [Connector](#).

**12.11.4 Member Data Documentation****12.11.4.1 \_graph\_name**

\_graph\_name [protected]

**12.11.4.2 connection\_string**

connection\_string

**12.11.4.3 database\_name**

database\_name

#### 12.11.4.4 verbose

verbose

The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/src/connectors/graph.py

## 12.12 KnowledgeGraph Class Reference

Manages a single graph within Neo4j.

### Public Member Functions

- None `__init__` (self, str name, GraphConnector database, bool verbose=False)
  - None `add_triple` (self, str subject, str relation, str object\_)
 

*Add a semantic triple to the graph using raw Cypher.*
  - None `add_triples_json` (self, List[Triple] triples\_json)
 

*Add several semantic triples to the graph from pre-verified JSON.*
  - DataFrame `get_all_triples` (self)
 

*Return all triples in the specified graph as a pandas DataFrame.*
  - Optional[DataFrame] `get_triple_properties` (self)
 

*Pivot the graph elements DataFrame to expose node and relationship properties as columns.*
  - DataFrame `triples_to_names` (self, DataFrame df\_ids, bool drop\_ids=False, Optional[DataFrame] df\_lookup=None)
 

*Maps a DataFrame containing element ID columns to human-readable names.*
  - DataFrame `find_element_names` (self, DataFrame df\_ids, List[str] name\_columns, List[str] id\_columns, str element\_type, str name\_property, bool drop\_ids=False, Optional[DataFrame] df\_lookup=None)
 

*Helper function which maps element IDs to human-readable names.*
  - DataFrame `get_subgraph_by_nodes` (self, List[str] node\_ids, List[str] id\_columns=["subject\_id", "object\_id"])
 

*Return all triples where subject or object is in the specified node list.*
  - DataFrame `get_neighborhood` (self, str node\_id, int depth=1)
 

*Get k-hop neighborhood around a central node.*
  - DataFrame `get_degree_range` (self, int min\_degree=1, int max\_degree=-1, List[str] id\_columns=["subject\_id", "object\_id"])
 

*Return triples associated with nodes whose degree lies within the specified bounds.*
  - DataFrame `get_by_ranked_degree` (self, int best\_rank=1, int worst\_rank=-1, bool enforce\_count=False, List[str] id\_columns=["subject\_id", "object\_id"])
 

*Return triples associated with nodes whose degree rank lies in the specified range.*
  - DataFrame `get_random_walk_sample` (self, List[str] start\_nodes, int walk\_length, int num\_walks=1)
 

*Sample subgraph using directed random walk traversal starting from specified nodes.*
  - DataFrame `get_community_subgraph` (self, int community\_id)
 

*Return all triples belonging to a specific GraphRAG community.*
  - None `detect_community_clusters` (self, str method="leiden", bool multi\_level=False, int max\_levels=10)
 

*Run community detection on the graph as described by the GraphRAG paper.*
  - str `to_triples_string` (self, Optional[DataFrame] triple\_names\_df=None, str mode="triple")
 

*Convert triples to string representation in various formats.*
  - str `to_contextualized_string` (self, Optional[List[str]] focus\_nodes=None, int top\_n=5)
 

*Convert triples to contextualized string grouped by focus nodes.*

- str [to\\_narrative](#) (self, str strategy="path", Optional[str] start\_node=None, int max\_triples=50)  
*Convert graph to narrative text using specified strategy.*
- Dict[str, Any] [get\\_summary\\_stats](#) (self)  
*Return summary statistics about the graph structure.*
- DataFrame [get\\_edge\\_counts](#) (self, int top\_n=-1)  
*Return node names and their edge counts, ordered by edge count descending.*
- str [get\\_node\\_context](#) (self, str node\_id, bool include\_neighbors=True)  
*Return natural language description of a node and its relationships.*
- DataFrame [get\\_relation\\_summary](#) (self)  
*Return summary of relationship types and their frequencies.*
- None [print\\_nodes](#) (self, int max\_rows=20, int max\_col\_width=50)  
*Print all nodes and edges in the current pseudo-database with row/column formatting.*
- None [print\\_triples](#) (self, int max\_rows=20, int max\_col\_width=50)  
*Print all nodes and edges in the current pseudo-database with row/column formatting.*

## Public Attributes

- [graph\\_name](#)  
*The name of this graph.*
- [database](#)  
*Reference to a pre-configured graph database wrapper.*
- [verbose](#)  
*Whether to print debug messages.*

## Protected Attributes

- [\\_first\\_insert](#)  
*Flag to drop any existing graph when the first triple is added.*

### 12.12.1 Detailed Description

Manages a single graph within Neo4j.

- Handles safe conversion of LLM output to structured triples.
- Provides helper functions to add and retrieve triples.

### 12.12.2 Constructor & Destructor Documentation

#### 12.12.2.1 [\\_\\_init\\_\\_\(\)](#)

```
None __init__ (
 self,
 str name,
 GraphConnector database,
 bool verbose = False)
```

### 12.12.3 Member Function Documentation

#### 12.12.3.1 add\_triple()

```
None add_triple (
 self,
 str subject,
 str relation,
 str object_)
```

Add a semantic triple to the graph using raw Cypher.

##### Parameters

<i>subject</i>	A string representing the entity performing an action.
<i>relation</i>	A string describing the action.
<i>object_</i>	A string representing the entity being acted upon.
_	

##### Note

LLM output should be pre-normalized using [src.connectors.llm.LLMConnector.normalize\\_triples](#).

##### Exceptions

<i>Log.Failure</i>	If the triple cannot be added to our graph database.
--------------------	------------------------------------------------------

#### 12.12.3.2 add\_triples\_json()

```
None add_triples_json (
 self,
 List[Triple] triples_json)
```

Add several semantic triples to the graph from pre-verified JSON.

##### Note

JSONshould be pre-normalized using [src.connectors.llm.LLMConnector.normalize\\_triples](#).

##### Parameters

<i>triples_json</i>	A list of Triple dictionaries containing keys: 's', 'r', and 'o'.
---------------------	-------------------------------------------------------------------

##### Exceptions

<i>Log.Failure</i>	If any triple cannot be added to the graph database.
--------------------	------------------------------------------------------

### 12.12.3.3 detect\_community\_clusters()

```
None detect_community_clusters (
 self,
 str method = "leiden",
 bool multi_level = False,
 int max_levels = 10)
```

Run community detection on the graph as described by the GraphRAG paper.

- Assigns a `community_id` property to all nodes, and optionally `level_id`.
- Partitions the graph's nodes into topic-coherent communities.
- Afterwards, you can call `get_community_subgraph()` to extract each community's triples for summarization. Clustering Methods
- Leiden (recommended) - improvement of Louvain ensuring well-connected, stable communities; supports multi-level hierarchy.
- Louvain - quickly groups nodes but may yield fragmented subcommunities.

#### Parameters

<code>method</code>	The community detection algorithm to run. Options: "leiden" (default) or "louvain".
<code>multi_level</code>	Whether to record hierarchical levels ( <code>level_id</code> ) for multi-scale summarization.
<code>max_levels</code>	Maximum hierarchy depth to compute (default: 10).

#### Exceptions

<code>Log.Failure</code>	If GDS is unavailable or any query fails.
--------------------------	-------------------------------------------

### 12.12.3.4 find\_element\_names()

```
DataFrame find_element_names (
 self,
 DataFrame df_ids,
 List[str] name_columns,
 List[str] id_columns,
 str element_type,
 str name_property,
 bool drop_ids = False,
 Optional[DataFrame] df_lookup = None)
```

Helper function which maps element IDs to human-readable names.

#### Note

- Requires the provided nodes or edges to still exist in the graph database; otherwise must specify `df_← lookup`.

**Parameters**

<i>df_ids</i>	DataFrame with required columns: <i>id_columns</i> .
<i>name_columns</i>	Required list of column names to create.
<i>id_columns</i>	Required list of columns containing element IDs.
<i>element_type</i>	Whether to match nodes or relationships. Value must be "node" or "relationship".
<i>name_property</i>	Required element property from <i>df_lookup</i> to use as the display name.
<i>drop_ids</i>	Whether to remove <i>id_columns</i> from results.
<i>df_lookup</i>	Optional DataFrame fetched from <a href="#">src.connectors.graph.GraphConnector.get_dataframe</a> with required columns: <i>element_id</i> , <i>element_type</i> , and <i>name_property</i> .

**Returns**

DataFrame with added columns: *name\_columns*.

**Exceptions**

<i>Log.Failure</i>	If mapping fails or required IDs are missing.
--------------------	-----------------------------------------------

**12.12.3.5 get\_all\_triples()**

```
DataFrame get_all_triples (
 self)
```

Return all triples in the specified graph as a pandas DataFrame.

**Returns**

Returns (subject, relation, object) columns only.

**Exceptions**

<i>Log.Failure</i>	If the query fails to retrieve or process the DataFrame.
--------------------	----------------------------------------------------------

**12.12.3.6 get\_by\_ranked\_degree()**

```
DataFrame get_by_ranked_degree (
 self,
 int best_rank = 1,
 int worst_rank = -1,
 bool enforce_count = False,
 List[str] id_columns = ["subject_id", "object_id"])
```

Return triples associated with nodes whose degree rank lies in the specified range.

- Computes degree (edge count) for all nodes.
- Sorts nodes by degree descending, assigns ranks, and selects those with *best\_rank*  $\leq$  rank  $\leq$  *worst\_rank*.
- Returns all triples where subject\_id or object\_id matches a selected node.

**Parameters**

<i>best_rank</i>	Minimum degree rank. Inclusive.
<i>worst_rank</i>	Maximum degree rank (-1 = maximum degree) to include. Inclusive.
<i>enforce_count</i>	Always return ( <i>worst_rank</i> - <i>best_rank</i> + 1) rows (fallback to node_id order).
<i>id_columns</i>	List of columns to compare against. Can be 'subject_id', 'object_id', or both.

**Returns**

DataFrame containing the triples for ranked nodes; columns: subject\_id, relation\_id, object\_id.

**Exceptions**

<i>Log.Failure</i>	If the graph cannot be queried.
<i>ValueError</i>	If best_rank or worst_rank values are invalid.

**12.12.3.7 get\_community\_subgraph()**

```
DataFrame get_community_subgraph (
 self,
 int community_id)
```

Return all triples belonging to a specific GraphRAG community.

- Communities are densely connected subgraphs detected via clustering algorithms.
- This enables GraphRAG-style hierarchical summarization where each community can be summarized independently. Requires nodes to have a 'community\_id' property assigned.
- Afterwards, you may run a summary step which generates community summaries for each cluster (as described in the paper).

**Parameters**

<i>community_id</i>	The identifier of the community to retrieve.
---------------------	----------------------------------------------

**Returns**

DataFrame with columns: subject\_id, relation\_id, object\_id

**Exceptions**

<i>Log.Failure</i>	If the query fails to retrieve the requested DataFrame or community detection has not been run.
--------------------	-------------------------------------------------------------------------------------------------

**12.12.3.8 get\_degree\_range()**

```
DataFrame get_degree_range (
```

```

 self,
 int min_degree = 1,
 int max_degree = -1,
 List[str] id_columns = ["subject_id", "object_id"])

```

Return triples associated with nodes whose degree lies within the specified bounds.

- Degree is defined as the number of relationships where a node appears as start\_node\_id or end\_node\_id.
- Selects all nodes satisfying min\_degree <= degree <= max\_degree and returns triples incident to those nodes.

#### Parameters

<i>max_degree</i>	Maximum number of edges allowed for a node to be included (-1 = infer highest edge count).
<i>min_degree</i>	Minimum number of edges required for a node to be included.
<i>id_columns</i>	List of columns to compare against. Can be 'subject_id', 'object_id', or both.

#### Returns

DataFrame containing an arbitrary number of triples for nodes in the specified degree range.

#### Exceptions

<i>Log.Failure</i>	If the graph fails to load or degree computation fails.
<i>ValueError</i>	If min_degree or max_degree values are invalid.

### 12.12.3.9 get\_edge\_counts()

```

DataFrame get_edge_counts (
 self,
 int top_n = -1)

```

Return node names and their edge counts, ordered by edge count descending.

#### Parameters

<i>top_n</i>	Number of top nodes to return (by edge count). Default is -1 (all nodes).
--------------	---------------------------------------------------------------------------

#### Returns

DataFrame with columns: node\_id, edge\_count

#### Exceptions

<i>Log.Failure</i>	If the query fails to retrieve the requested DataFrame.
--------------------	---------------------------------------------------------

### 12.12.3.10 get\_neighborhood()

```
DataFrame get_neighborhood (
 self,
 str node_id,
 int depth = 1)
```

Get k-hop neighborhood around a central node.

Returns all triples within k hops of the specified node. A 1-hop neighborhood includes all direct neighbors, 2-hop includes neighbors-of-neighbors, etc.

#### Parameters

<i>node_id</i>	The element ID of the central node.
<i>depth</i>	Number of hops to traverse (default: 1).

#### Returns

DataFrame with columns: subject\_id, relation\_id, object\_id

#### Exceptions

<i>Log.Failure</i>	If the query fails to retrieve the requested DataFrame.
--------------------	---------------------------------------------------------

### 12.12.3.11 get\_node\_context()

```
str get_node_context (
 self,
 str node_id,
 bool include_neighbors = True)
```

Return natural language description of a node and its relationships.

Generates a human-readable summary of a single node suitable for LLM context. Example: "Alice is connected to 5 entities. She knows Bob and Charlie, works at Company, lives in City, and follows Dave."

#### Parameters

<i>node_id</i>	The element ID of the node to describe.
<i>include_neighbors</i>	Whether to list neighbor node IDs (default: True).

#### Returns

Natural language description of the node.

#### Exceptions

<i>Log.Failure</i>	If the node does not exist in the graph.
--------------------	------------------------------------------

### 12.12.3.12 get\_random\_walk\_sample()

```
DataFrame get_random_walk_sample (
 self,
 List[str] start_nodes,
 int walk_length,
 int num_walks = 1)
```

Sample subgraph using directed random walk traversal starting from specified nodes.

- More diverse than degree-based filtering (nodes with many edges) and better preserves graph structure.
- Each walk starts from a random node in start\_nodes and continues for walk\_length steps.

#### Parameters

<code>start_nodes</code>	List of node IDs to use as starting points.
<code>walk_length</code>	Number of steps in each random walk.
<code>num_walks</code>	Number of random walks to perform (default: 1).

#### Returns

DataFrame with columns: subject\_id, relation\_id, object\_id

#### Exceptions

<code>Log.Failure</code>	If the query fails to retrieve the requested DataFrame.
--------------------------	---------------------------------------------------------

### 12.12.3.13 get\_relation\_summary()

```
DataFrame get_relation_summary (
 self)
```

Return summary of relationship types and their frequencies.

Provides an overview of what types of relationships exist in the graph and how common each type is. Useful for understanding graph schema.

#### Returns

DataFrame with columns: relation\_type, count, example\_triple

### 12.12.3.14 get\_subgraph\_by\_nodes()

```
DataFrame get_subgraph_by_nodes (
 self,
 List[str] node_ids,
 List[str] id_columns = ["subject_id", "object_id"])
```

Return all triples where subject or object is in the specified node list.

**Parameters**

<i>node_ids</i>	List of node element IDs to filter by.
<i>id_columns</i>	List of columns to compare against. Can be 'subject_id', 'object_id', or both.

**Returns**

DataFrame with columns: subject\_id, relation\_id, object\_id

**Exceptions**

<i>Log.Failure</i>	If the query fails to retrieve the requested DataFrame.
<i>KeyError</i>	If the provided column names are invalid.

**12.12.3.15 get\_summary\_stats()**

```
Dict[str, Any] get_summary_stats (
 self)
```

Return summary statistics about the graph structure.

Provides metadata useful for LLM context, including:

- *node\_count*: Total number of nodes
- *edge\_count*: Total number of relationships
- *relation\_types*: List of unique relationship types
- *avg\_degree*: Average node degree (edges per node)
- *top\_nodes*: List of most connected nodes (top 5 by degree)
- *density*: Graph density (actual edges / possible edges)

**Returns**

Dictionary containing graph statistics.

**12.12.3.16 get\_triple\_properties()**

```
Optional[DataFrame] get_triple_properties (
 self)
```

Pivot the graph elements DataFrame to expose node and relationship properties as columns.

- Builds a joined view of properties from both nodes (n1, n2) and the relationship (r).
- Removes redundant fields such as: db, kg, element\_type, start\_node\_id, and end\_node\_id.
- Usage: n1.element\_id, r.rel\_type, n2.name, etc.

**Returns**

DataFrame where each row represents one triple (n1, r, n2).

**Exceptions**

<i>Log.Failure</i>	If the elements DataFrame cannot be loaded or pivoting fails.
--------------------	---------------------------------------------------------------

**12.12.3.17 print\_nodes()**

```
None print_nodes (
 self,
 int max_rows = 20,
 int max_col_width = 50)
```

Print all nodes and edges in the current pseudo-database with row/column formatting.

**12.12.3.18 print\_triples()**

```
None print_triples (
 self,
 int max_rows = 20,
 int max_col_width = 50)
```

Print all nodes and edges in the current pseudo-database with row/column formatting.

**12.12.3.19 to\_contextualized\_string()**

```
str to_contextualized_string (
 self,
 Optional[List[str]] focus_nodes = None,
 int top_n = 5)
```

Convert triples to contextualized string grouped by focus nodes.

Groups triples by subject nodes and formats them with context headers. This provides better structure for LLM comprehension compared to flat triple lists. If `focus_nodes` is `None`, uses the `top_n` most connected nodes. Example output: Facts about Alice:

- knows Bob
- works\_at Company
- lives\_in City

**Parameters**

<code>focus_nodes</code>	List of node names to group by. If <code>None</code> , uses <code>top_n</code> by degree.
<code>top_n</code>	Number of top nodes to use if <code>focus_nodes</code> is <code>None</code> (default: 5).

**Returns**

Formatted string with contextualized triple groups.

**12.12.3.20 to\_narrative()**

```
str to_narrative (
 self,
 str strategy = "path",
 Optional[str] start_node = None,
 int max_triples = 50)
```

Convert graph to narrative text using specified strategy.

Transforms structured triples into natural language narrative:

- "path": Follow edges sequentially from start\_node, creating a story-like flow
- "cluster": Group related entities and describe them thematically
- "summary": High-level overview of graph contents and structure

**Parameters**

<i>strategy</i>	Narrative generation strategy: "path", "cluster", or "summary" (default: "path").
<i>start_node</i>	Starting node for "path" strategy. If None, uses highest-degree node.
<i>max_triples</i>	Maximum number of triples to include (default: 50).

**Returns**

Natural language narrative describing the graph.

**Exceptions**

<i>ValueError</i>	If strategy is not recognized.
-------------------	--------------------------------

**12.12.3.21 to\_triples\_string()**

```
str to_triples_string (
 self,
 Optional[DataFrame] triple_names_df = None,
 str mode = "triple")
```

Convert triples to string representation in various formats.

Supports multiple output formats for LLM consumption:

- "natural": Human-readable sentences (e.g., "Alice employed by Bob.")
- "triple": Raw triple format (e.g., "Alice employedBy Bob")
- "json": JSON array of objects with s/r/o keys

**Parameters**

<code>triple_names</code> <code>_df</code>	DataFrame with subject, relation, object columns. If None, uses all triples from this graph.
<code>mode</code>	Output format: "natural", "triple", or "json" (default: "triple").

**Returns**

String representation of triples in the specified format.

**Exceptions**

<code>ValueError</code>	If format is not recognized.
-------------------------	------------------------------

**12.12.3.22 `triples_to_names()`**

```
DataFrame triples_to_names (
 self,
 DataFrame df_ids,
 bool drop_ids = False,
 Optional[DataFrame] df_lookup = None)
```

Maps a DataFrame containing element ID columns to human-readable names.

**Note**

- Requires the provided nodes to still exist in the graph database; otherwise must specify `df_lookup`.

**Parameters**

<code>df_ids</code>	DataFrame with added columns: <code>subject_id</code> , <code>relation_id</code> , <code>object_id</code> .
<code>drop_ids</code>	Whether to remove columns from results: <code>subject_id</code> , <code>relation_id</code> , <code>object_id</code> .
<code>df_lookup</code>	Optional DataFrame fetched from <a href="#">src.connectors.graph.GraphConnector.get_dataframe</a> with required columns: <code>element_id</code> , <code>element_type</code> , <code>name</code> , and <code>rel_type</code> .

**Returns**

DataFrame with added columns: `subject`, `relation`, `object`.

**Exceptions**

<code>Log.Failure</code>	If mapping fails or required IDs are missing.
--------------------------	-----------------------------------------------

**12.12.4 Member Data Documentation****12.12.4.1 `_first_insert`**

```
_first_insert [protected]
```

Flag to drop any existing graph when the first triple is added.

#### 12.12.4.2 database

database

Reference to a pre-configured graph database wrapper.

#### 12.12.4.3 graph\_name

graph\_name

The name of this graph.

Matches node.kg for all nodes in the graph database.

#### 12.12.4.4 verbose

verbose

Whether to print debug messages.

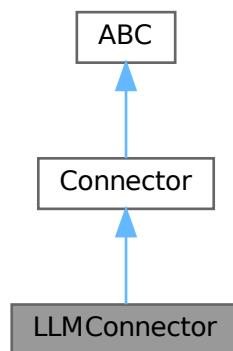
The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/src/components/fact\_storage.py

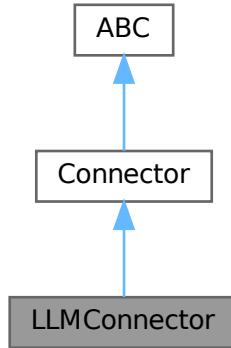
## 12.13 LLMConnector Class Reference

Connector for prompting and returning LLM output (raw text/JSON) via LangChain.

Inheritance diagram for LLMConnector:



Collaboration diagram for LLMConnector:



## Public Member Functions

- None `__init__` (self, float temperature=0, str `system_prompt`="You are a helpful assistant.")  
*Initialize the connector.*
- None `configure` (self)  
*Initialize the LangChain LLM using environment credentials.*
- bool `test_operations` (self, bool `raise_error`=True)  
*Establish a basic connection to the database, and test full functionality.*
- bool `check_connection` (self, str `log_source`, bool `raise_error`)  
*Send a trivial prompt to verify LLM connectivity.*
- str `execute_full_query` (self, str `system_prompt`, str `human_prompt`)  
*Send a single prompt to the LLM with separate system and human instructions.*
- str `execute_query` (self, str `query`)  
*Send a single prompt through the connection and return raw LLM output.*
- str `execute_file` (self, str `filename`)  
*Run a single prompt from a file.*

## Static Public Member Functions

- List[Tuple[str, str, str]] `normalize_triples` (Any data)  
*Normalize flexible LLM output into a list of clean (subject, relation, object) triples.*

## Public Attributes

- `model_name`
- `llm`
- `system_prompt`

### 12.13.1 Detailed Description

Connector for prompting and returning LLM output (raw text/JSON) via LangChain.

#### Note

The method `src.connectors.llm.LLMConnector.execute_query` simplifies the prompt process.

### 12.13.2 Constructor & Destructor Documentation

#### 12.13.2.1 `__init__()`

```
None __init__ (
 self,
 float temperature = 0,
 str system_prompt = "You are a helpful assistant.")
```

Initialize the connector.

#### Note

Model name is specified in the .env file.

### 12.13.3 Member Function Documentation

#### 12.13.3.1 `check_connection()`

```
bool check_connection (
 self,
 str log_source,
 bool raise_error)
```

Send a trivial prompt to verify LLM connectivity.

#### Parameters

<code>log_source</code>	The Log class prefix indicating which method is performing the check.
<code>raise_error</code>	Whether to raise an error on connection failure.

#### Returns

Whether the prompt executed successfully.

#### Exceptions

<code>Log.Failure</code>	If <code>raise_error</code> is True and the connection test fails to complete.
--------------------------	--------------------------------------------------------------------------------

Reimplemented from [Connector](#).

### 12.13.3.2 configure()

```
None configure (
 self)
```

Initialize the LangChain LLM using environment credentials.

Reads:

- OPENAI\_API\_KEY from .env for authentication
- LLM\_MODEL and LLM\_TEMPERATURE to override defaults

### 12.13.3.3 execute\_file()

```
str execute_file (
 self,
 str filename)
```

Run a single prompt from a file.

Reads the entire file as a single string and sends it to execute\_query.

Parameters

<code>filename</code>	Path to the prompt file (.txt)
-----------------------	--------------------------------

Returns

Raw LLM response as a string.

Reimplemented from [Connector](#).

### 12.13.3.4 execute\_full\_query()

```
str execute_full_query (
 self,
 str system_prompt,
 str human_prompt)
```

Send a single prompt to the LLM with separate system and human instructions.

### 12.13.3.5 execute\_query()

```
str execute_query (
 self,
 str query)
```

Send a single prompt through the connection and return raw LLM output.

**Parameters**

<i>query</i>	A single string prompt to send to the LLM.
--------------	--------------------------------------------

**Returns**

Raw LLM response as a string.

Reimplemented from [Connector](#).

**12.13.3.6 normalize\_triples()**

```
List[Tuple[str, str, str]] normalize_triples (
 Any data) [static]
```

Normalize flexible LLM output into a list of clean (subject, relation, object) triples.

- Accepts dicts, lists of dicts, tuples, or dicts-of-lists.
- Joins list values, trims, and sanitizes for Cypher safety.
- Enforces uppercase underscore-safe relation labels.

**Parameters**

<i>data</i>	Raw LLM output to normalize.
-------------	------------------------------

**Returns**

List of sanitized (s, r, o) triples ready for insertion.

**Exceptions**

<i>ValueError</i>	If input format cannot be parsed.
-------------------	-----------------------------------

**12.13.3.7 test\_operations()**

```
bool test_operations (
 self,
 bool raise_error = True)
```

Establish a basic connection to the database, and test full functionality.

Can be configured to fail silently, which enables retries or external handling.

**Parameters**

<i>raise_error</i>	Whether to raise an error on connection failure.
--------------------	--------------------------------------------------

**Returns**

Whether the prompt executed successfully.

**Exceptions**

<i>LogFailure</i>	If <code>raise_error</code> is <code>True</code> and the connection test fails to complete.
-------------------	---------------------------------------------------------------------------------------------

Reimplemented from [Connector](#).

## 12.13.4 Member Data Documentation

### 12.13.4.1 llm

`llm`

### 12.13.4.2 model\_name

`model_name`

### 12.13.4.3 system\_prompt

`system_prompt`

The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/src/connectors/[llm.py](#)

## 12.14 Log Class Reference

The `Log` class standardizes console output.

Collaboration diagram for `Log`:



## Classes

- class [BadAddressFailure](#)  
*Raised when a database connection string or address is invalid.*
- class [Failure](#)  
*User-facing base class for custom error handling.*

## Static Public Member Functions

- None [success](#) (str prefix="PASS: ", str msg="", bool verbose=True)  
*A success message begins with a green prefix.*
- None [warn](#) (str prefix="WARN: ", str msg="", bool verbose=True)  
*A warning message begins with a yellow prefix.*
- None [fail](#) (str prefix="ERROR: ", str msg="", bool raise\_error=True, Optional[Exception] other\_error=None)  
*A failure message begins with a red prefix.*
- None [success\\_legacy](#) (str msg="")  
*A legacy success message begins with a Green Plus.*
- None [fail\\_legacy](#) (str msg="")  
*A legacy failure message begins with a Red X.*
- None [time\\_message](#) (str prefix="[TIME] ", str msg="", bool verbose=True)  
*A time message begins with a light blue prefix.*
- None [chart\\_message](#) (str prefix="[CHART] ", str msg="", bool verbose=True)  
*A chart message begins with a gray prefix.*
- None [chart](#) (str title, str filename, bool verbose=True)  
*Print the time taken to complete a function.*
- None [elapsed\\_time](#) (str name, float seconds, str call\_chain, bool verbose=True)  
*Print the time taken to complete a function.*
- str [format\\_call\\_chain](#) (List[FrameInfo] stack, str name)  
*Sanitize and concatenate the full call stack for console output.*
- Callable[..., Any] [time](#) (Callable[..., Any] func)  
*Logs the time elapsed for a function call.*
- Generator[None, None, None] [timer](#) (str name=None)  
*Context manager for recording the execution time of code blocks.*
- DataFrame [get\\_timing\\_summary](#) ()  
*Returns timing results as a pandas DataFrame.*
- DataFrame [get\\_merged\\_timing](#) (str file\_path="../logs/elapsed\_time.csv")  
*Reads the existing file, deletes rows matching this run\_id, and adds current data.*
- None [dump\\_timing\\_csv](#) (str file\_path="../logs/elapsed\_time.csv")  
*Save timing results to a CSV file, appending if it already exists.*
- [clear\\_timing\\_data](#) ()  
*Clears all recorded timing data.*
- [print\\_timing\\_summary](#) ()  
*Prints a formatted timing summary grouped by function.*

## Static Public Attributes

- bool `USE_COLORS` = True  
*Enable ANSI colors in output.*
- bool `RECORD_TIME` = True  
*Enable time-logging with the 'Log.time' decorator.*
- bool `FULL_DF` = False  
*Print the entire DataFrame to console.*
- str `GREEN` = "\033[32m"  
*ANSI code for green text.*
- str `RED` = "\033[31m"  
*ANSI code for red text.*
- str `YELLOW` = "\033[33m"  
*ANSI code for yellow text.*
- str `BRIGHT` = "\033[93m"  
*ANSI code for bright yellow / cream.*
- str `CYAN` = "\033[96m"  
*ANSI code for light blue.*
- str `GRAY` = "\033[90m"  
*ANSI code for light gray.*
- str `WHITE` = "\033[0m"  
*ANSI code to reset color.*
- str `SUCCESS_COLOR` = `GREEN`  
*ANSI color applied to the prefix of success messages.*
- str `WARNING_COLOR` = `YELLOW`  
*ANSI color applied to the prefix of ignored fail messages.*
- str `FAILURE_COLOR` = `RED`  
*ANSI color applied to the prefix of critical fail messages.*
- str `TIME_COLOR` = `CYAN`  
*ANSI color applied to the prefix of time-elapsed messages.*
- str `CHART_COLOR` = `GRAY`  
*ANSI color applied to the prefix of chart generation messages.*
- str `MSG_COLOR` = `BRIGHT`  
*ANSI color applied to the body of every Log message.*
- f `msg_chart_saved` = lambda title, filename"Saved `chart` '{title}' to {filename}"
- f `msg_elapsed_time` = lambda name, seconds"{name} took {seconds:.3f}s"
- int `run_id` = 1
- str `t_dump` = "[DUMP] "
- f `msg_time_dump` = lambda file\_path"Saved `time` records to '{file\_path}'"
- str `conn_abc` = "BASE CONNECTOR: "
- str `db_conn_abc` = "CONNECTOR: "
- str `rel_db` = "REL DB: "
- str `gr_db` = "GRAPH DB: "
- str `doc_db` = "DOCS DB: "
- str `bad_addr` = "BAD ADDRESS: "
- f `msg_bad_addr` = lambda connection\_string"Failed to connect on {connection\_string}"
- str `bad_path` = "FILE NOT FOUND: "
- f `msg_bad_path` = lambda file\_path"Failed to open file '{file\_path}'"
- f `msg_good_path` = lambda file\_path"Reading contents of file '{file\_path}'"
- f `msg_good_exec_f` = lambda file\_path"Finished executing queries from '{file\_path}'"
- f `msg_bad_exec_f` = lambda file\_path"Error occurred while executing queries from '{file\_path}'"
- f `msg_db_connect` = lambda database\_name"Successfully connected to database: {database\_name}"

- str `good_val` = "VALID RESULT: "
- str `bad_val` = "INCORRECT RESULT: "
- f `msg_compare` = lambda observed, expected"Expected {expected}, got {observed}"
- tuple `msg_result`
- tuple `msg_good_table`
- tuple `msg_good_coll`
- tuple `msg_good_graph`
- f `msg_bad_table` = lambda name"Table '{name}' not found"
- f `msg_bad_coll` = lambda name"Collection '{name}' not found"
- f `msg_bad_graph` = lambda name"Graph '{name}' not found"
- str `test_ops` = "OPERATE: "
- str `test_basic` = "CONNECT: "
- str `test_info` = "DB INFO: "
- str `test_df` = "GET DF: "
- str `test_tmp_db` = "CREATE DB: "
- str `msg_unknown_error` = "An unhandled error occurred."
- str `get_df` = "GET\_DF: "
- str `create_db` = "CREATE\_DB: "
- str `drop_db` = "DROP\_DB: "
- str `run_q` = "QUERY: "
- str `run_f` = "FILE EXEC: "
- str `drop_gr` = "DROP\_GRAPH: "
- f `msg_success_managed_db` = lambda managed, database\_name"Successfully {managed} database '{database\_name}'"
- tuple `msg_fail_manage_db`
- f `msg_success_managed_gr` = lambda managed, database\_name"Successfully {managed} graph '{database\_name}'"
- tuple `msg_fail_manage_gr`
- f `msg_fail_parse` = lambda alias, bad\_value, expected\_type"Could not convert {alias} with value {bad\_value} to type {expected\_type}"
- tuple `msg_multiple_query`
- f `msg_good_exec_q` = lambda query"Executed successfully:\n'{query}'"
- f `msg_good_exec_qr` = lambda query, results"Executed successfully:\n'{query}'\n{Log.msg\_result(results)}"
- f `msg_bad_exec_q` = lambda query"Failed to execute query:\n'{query}'"
- Log `msg_good_df_parse` = lambda df.msg\_result(df)
- f `msg_bad_df_parse` = lambda query"Failed to convert query result to DataFrame:\n'{query}'"
- str `kg` = "KG: "
- str `pytest_db` = "PYTEST (DB): "
- str `db_exists` = "DB\_EXIST: "
- f `msg_db_exists` = lambda database\_name"Database '{database\_name}' already exists."
- f `msg_db_not_found` = lambda database\_name, connection\_string"Could not find database '{database\_name}' using connection '{connection\_string}'"
- f `msg_db_current` = lambda database\_name"Cannot drop database '{database\_name}' while connected to it!"
- str `swap_db` = "SWAP\_DB: "
- str `swap_kg` = "SWAP\_GRAPH: "
- f `msg_swap_db` = lambda old\_db, new\_db"Switched from database '{old\_db}' to database '{new\_db}'"
- f `msg_swap_kg` = lambda old\_kg, new\_kg"Switched from graph '{old\_kg}' to graph '{new\_kg}'"
- str `get_unique` = "UNIQUE: "
- f `msg_none_df` = lambda collection\_type, collection\_name"Unable to fetch DataFrame from {collection\_type} '{collection\_name}' - None"
- str `sub_gr` = "SUBGRAPH: "
- str `gr_rag` = "RAG: "
- f `msg_bad_triples` = lambda graph\_name"No triples found for graph {graph\_name}"

## Static Protected Attributes

- list `_timing_results = []`

### 12.14.1 Detailed Description

The Log class standardizes console output.

### 12.14.2 Member Function Documentation

#### 12.14.2.1 chart()

```
None chart (
 str title,
 str filename,
 bool verbose = True) [static]
```

Print the time taken to complete a function.

##### Parameters

<code>title</code>	The title of the chart.
<code>filename</code>	Where the chart was saved to.
<code>verbose</code>	Whether to actually print. Saves space and reduces nested if statements.

#### 12.14.2.2 chart\_message()

```
None chart_message (
 str prefix = "[CHART] ",
 str msg = "",
 bool verbose = True) [static]
```

A chart message begins with a gray prefix.

##### Parameters

<code>prefix</code>	The context of the message.
<code>msg</code>	The message to print.
<code>verbose</code>	Whether to actually print. Saves space and reduces nested if statements.

#### 12.14.2.3 clear\_timing\_data()

```
clear_timing_data () [static]
```

Clears all recorded timing data.

**12.14.2.4 dump\_timing\_csv()**

```
None dump_timing_csv (
 str file_path = "./logs/elapsed_time.csv") [static]
```

Save timing results to a CSV file, appending if it already exists.

**Parameters**

<i>file_path</i>	Where the saved CSV will be located.
------------------	--------------------------------------

**Returns**

DataFrame with columns: function, elapsed, call\_chain

**12.14.2.5 elapsed\_time()**

```
None elapsed_time (
 str name,
 float seconds,
 str call_chain,
 bool verbose = True) [static]
```

Print the time taken to complete a function.

**Parameters**

<i>name</i>	The name of the function.
<i>seconds</i>	The number of seconds (will be rounded to 3 decimals)
<i>call_chain</i>	The full stack of function calls (for record-keeping)
<i>verbose</i>	Whether to actually print. Saves space and reduces nested if statements.

**12.14.2.6 fail()**

```
None fail (
 str prefix = "ERROR: ",
 str msg = "",
 bool raise_error = True,
 Optional[Exception] other_error = None) [static]
```

A failure message begins with a red prefix.

**Parameters**

<i>prefix</i>	The context of the message.
<i>msg</i>	The message to print.
<i>raise_error</i>	Whether to raise an error.
<i>other_error</i>	Another Exception resulting from this failure.

**Exceptions**

<i>Log.Failure</i>	If raise_error is True
--------------------	------------------------

**12.14.2.7 fail\_legacy()**

```
None fail_legacy (
 str msg = "") [static]
```

A legacy failure message begins with a Red X.

**Parameters**

<i>msg</i>	The message to print.
------------	-----------------------

**12.14.2.8 format\_call\_chain()**

```
str format_call_chain (
 List[FrameInfo] stack,
 str name) [static]
```

Sanitize and concatenate the full call stack for console output.

**Parameters**

<i>stack</i>	The frame stack obtained by inspect.stack().
<i>name</i>	The name of the caller function.

**Returns**

A string representing the full call chain.

**12.14.2.9 get\_merged\_timing()**

```
DataFrame get_merged_timing (
 str file_path = "./logs/elapsed_time.csv") [static]
```

Reads the existing file, deletes rows matching this run\_id, and adds current data.

**Returns**

DataFrame with columns: function, elapsed, call\_chain, run\_id

### 12.14.2.10 get\_timing\_summary()

```
DataFrame get_timing_summary () [static]
```

Returns timing results as a pandas DataFrame.

#### Returns

DataFrame with columns: function, elapsed, call\_chain, run\_id

### 12.14.2.11 print\_timing\_summary()

```
print_timing_summary () [static]
```

Prints a formatted timing summary grouped by function.

### 12.14.2.12 success()

```
None success (
 str prefix = "PASS: ",
 str msg = "",
 bool verbose = True) [static]
```

A success message begins with a green prefix.

#### Parameters

<i>prefix</i>	The context of the message.
<i>msg</i>	The message to print.
<i>verbose</i>	Whether to actually print. Saves space and reduces nested if statements.

### 12.14.2.13 success\_legacy()

```
None success_legacy (
 str msg = "") [static]
```

A legacy success message begins with a Green Plus.

#### Parameters

<i>msg</i>	The message to print.
------------	-----------------------

### 12.14.2.14 time()

```
Callable[..., Any] time (
 Callable[..., Any] func) [static]
```

Logs the time elapsed for a function call.

- Uses an inner wrapper function to capture \*args and \*\*kwargs.

#### Parameters

<i>func</i>	The function to wrap.
-------------	-----------------------

#### Returns

The wrapped function that logs time and forwards the result.

### 12.14.2.15 time\_message()

```
None time_message (
 str prefix = "[TIME] ",
 str msg = "",
 bool verbose = True) [static]
```

A time message begins with a light blue prefix.

#### Parameters

<i>prefix</i>	The context of the message.
<i>msg</i>	The message to print.
<i>verbose</i>	Whether to actually print. Saves space and reduces nested if statements.

### 12.14.2.16 timer()

```
Generator[None, None, None] timer (
 str name = None) [static]
```

Context manager for recording the execution time of code blocks.

#### Parameters

<i>name</i>	Optional name for the timed block. If not provided, uses caller function name. Usage: with Log.timer():
-------------	---------------------------------------------------------------------------------------------------------

## 12.14.3 your code here

### 12.14.3.1 warn()

```
None warn (
 str prefix = "WARN: ",
 str msg = "",
 bool verbose = True) [static]
```

A warning message begins with a yellow prefix.

**Parameters**

<i>prefix</i>	The context of the message.
<i>msg</i>	The message to print.
<i>verbose</i>	Whether to actually print. Saves space and reduces nested if statements.

## 12.14.4 Member Data Documentation

### 12.14.4.1 `_timing_results`

```
list _timing_results = [] [static], [protected]
```

### 12.14.4.2 `bad_addr`

```
str bad_addr = "BAD ADDRESS: " [static]
```

### 12.14.4.3 `bad_path`

```
str bad_path = "FILE NOT FOUND: " [static]
```

### 12.14.4.4 `bad_val`

```
str bad_val = "INCORRECT RESULT: " [static]
```

### 12.14.4.5 `BRIGHT`

```
str BRIGHT = "\033[93m" [static]
```

ANSI code for bright yellow / cream.

### 12.14.4.6 `CHART_COLOR`

```
str CHART_COLOR = GRAY [static]
```

ANSI color applied to the prefix of chart generation messages.

### 12.14.4.7 `conn_abc`

```
str conn_abc = "BASE CONNECTOR: " [static]
```

### 12.14.4.8 `create_db`

```
str create_db = "CREATE_DB: " [static]
```

#### 12.14.4.9 CYAN

```
str CYAN = "\033[96m" [static]
```

ANSI code for light blue.

#### 12.14.4.10 db\_conn\_abc

```
str db_conn_abc = "CONNECTOR: " [static]
```

#### 12.14.4.11 db\_exists

```
str db_exists = "DB_EXIST: " [static]
```

#### 12.14.4.12 doc\_db

```
str doc_db = "DOCS DB: " [static]
```

#### 12.14.4.13 drop\_db

```
str drop_db = "DROP_DB: " [static]
```

#### 12.14.4.14 drop\_gr

```
str drop_gr = "DROP_GRAPH: " [static]
```

#### 12.14.4.15 FAILURE\_COLOR

```
str FAILURE_COLOR = RED [static]
```

ANSI color applied to the prefix of critical fail messages.

#### 12.14.4.16 FULL\_DF

```
bool FULL_DF = False [static]
```

Print the entire DataFrame to console.

#### 12.14.4.17 get\_df

```
str get_df = "GET_DF: " [static]
```

**12.14.4.18 get\_unique**

```
str get_unique = "UNIQUE: " [static]
```

**12.14.4.19 good\_val**

```
str good_val = "VALID RESULT: " [static]
```

**12.14.4.20 gr\_db**

```
str gr_db = "GRAPH DB: " [static]
```

**12.14.4.21 gr\_rag**

```
str gr_rag = "RAG: " [static]
```

**12.14.4.22 GRAY**

```
str GRAY = "\033[90m" [static]
```

ANSI code for light gray.

**12.14.4.23 GREEN**

```
str GREEN = "\033[32m" [static]
```

ANSI code for green text.

**12.14.4.24 kg**

```
str kg = "KG: " [static]
```

**12.14.4.25 msg\_bad\_addr**

```
f msg_bad_addr = lambda connection_string"Failed to connect on {connection_string}" [static]
```

**12.14.4.26 msg\_bad\_coll**

```
f msg_bad_coll = lambda name"Collection '{name}' not found" [static]
```

**12.14.4.27 msg\_bad\_df\_parse**

```
f msg_bad_df_parse = lambda query"Failed to convert query result to DataFrame:\n'{query}'"
[static]
```

**12.14.4.28 msg\_bad\_exec\_f**

```
f msg_bad_exec_f = lambda file_path"Error occurred while executing queries from '{file_path}'"
[static]
```

**12.14.4.29 msg\_bad\_exec\_q**

```
f msg_bad_exec_q = lambda query"Failed to execute query:\n'{query}'"
[static]
```

**12.14.4.30 msg\_bad\_graph**

```
f msg_bad_graph = lambda name"Graph '{name}' not found"
[static]
```

**12.14.4.31 msg\_bad\_path**

```
f msg_bad_path = lambda file_path"Failed to open file '{file_path}'"
[static]
```

**12.14.4.32 msg\_bad\_table**

```
f msg_bad_table = lambda name"Table '{name}' not found"
[static]
```

**12.14.4.33 msg\_bad\_triples**

```
f msg_bad_triples = lambda graph_name"No triples found for graph {graph_name}"
[static]
```

**12.14.4.34 msg\_chart\_saved**

```
f msg_chart_saved = lambda title, filename"Saved chart '{title}' to {filename}"
[static]
```

**12.14.4.35 MSG\_COLOR**

```
str MSG_COLOR = BRIGHT [static]
```

ANSI color applied to the body of every Log message.

**12.14.4.36 msg\_compare**

```
f msg_compare = lambda observed, expected"Expected {expected}, got {observed}" [static]
```

**12.14.4.37 msg\_db\_connect**

```
f msg_db_connect = lambda database_name"Successfully connected to database: {database_name}" [static]
```

**12.14.4.38 msg\_db\_current**

```
f msg_db_current = lambda database_name"Cannot drop database '{database_name}' while connected to it!" [static]
```

**12.14.4.39 msg\_db\_exists**

```
f msg_db_exists = lambda database_name"Database '{database_name}' already exists." [static]
```

**12.14.4.40 msg\_db\_not\_found**

```
f msg_db_not_found = lambda database_name, connection_string"Could not find database '{database_name}' using connection '{connection_string}'" [static]
```

**12.14.4.41 msg\_elapsed\_time**

```
f msg_elapsed_time = lambda name, seconds"{name} took {seconds:.3f}s" [static]
```

**12.14.4.42 msg\_fail\_manage\_db**

```
tuple msg_fail_manage_db [static]
```

**Initial value:**

```
= (
 lambda manage, database_name, connection_string: f"Failed to {manage} database '{database_name}' on connection {connection_string}"
)
```

**12.14.4.43 msg\_fail\_manage\_gr**

```
tuple msg_fail_manage_gr [static]
```

**Initial value:**

```
= (
 lambda manage, database_name, connection_string: f"Failed to {manage} graph '{database_name}' on connection {connection_string}"
)
```

#### 12.14.4.44 msg\_fail\_parse

```
f msg_fail_parse = lambda alias, bad_value, expected_type"Could not convert {alias} with value {bad_value} to type {expected_type}" [static]
```

#### 12.14.4.45 msg\_good\_coll

```
tuple msg_good_coll [static]
```

**Initial value:**

```
= (lambda name, df: f)
```

#### 12.14.4.46 msg\_good\_df\_parse

```
Log msg_good_df_parse = lambda df.msg_result(df) [static]
```

#### 12.14.4.47 msg\_good\_exec\_f

```
f msg_good_exec_f = lambda file_path"Finished executing queries from '{file_path}'" [static]
```

#### 12.14.4.48 msg\_good\_exec\_q

```
f msg_good_exec_q = lambda query"Executed successfully:\n'{query}'" [static]
```

#### 12.14.4.49 msg\_good\_exec\_qr

```
f msg_good_exec_qr = lambda query, results"Executed successfully:\n'{query}'\n{Log.msg_result(results)}" [static]
```

#### 12.14.4.50 msg\_good\_graph

```
tuple msg_good_graph [static]
```

**Initial value:**

```
= (lambda name, df: f)
```

#### 12.14.4.51 msg\_good\_path

```
f msg_good_path = lambda file_path"Reading contents of file '{file_path}'" [static]
```

**12.14.4.52 msg\_good\_table**

```
tuple msg_good_table [static]
```

**Initial value:**

```
= (
 lambda name, df: f
)
```

**12.14.4.53 msg\_multiple\_query**

```
tuple msg_multiple_query [static]
```

**Initial value:**

```
= (
 lambda n_queries, query: f"A combined query ({n_queries} results) was executed as a single query.
Extra results were discarded. Query:\n{query}"
)
```

**12.14.4.54 msg\_none\_df**

```
f msg_none_df = lambda collection_type, collection_name"Unable to fetch DataFrame from {collection←
_type} '{collection_name}' - None" [static]
```

**12.14.4.55 msg\_result**

```
tuple msg_result [static]
```

**Initial value:**

```
= (
 lambda results: f
)
```

**12.14.4.56 msg\_success\_managed\_db**

```
f msg_success_managed_db = lambda managed, database_name"Successfully {managed} database '{database←
_name}'" [static]
```

**12.14.4.57 msg\_success\_managed\_gr**

```
f msg_success_managed_gr = lambda managed, database_name"Successfully {managed} graph '{database←
_name}'" [static]
```

**12.14.4.58 msg\_swap\_db**

```
f msg_swap_db = lambda old_db, new_db"Switched from database '{old_db}' to database '{new←
_db}'" [static]
```

**12.14.4.59 msg\_swap\_kg**

```
f msg_swap_kg = lambda old_kg, new_kg"Switched from graph '{old_kg}' to graph '{new_kg}'"
[static]
```

**12.14.4.60 msg\_time\_dump**

```
f msg_time_dump = lambda file_path"Saved time records to '{file_path}'"
[static]
```

**12.14.4.61 msg\_unknown\_error**

```
str msg_unknown_error = "An unhandled error occurred."
[static]
```

**12.14.4.62 pytest\_db**

```
str pytest_db = "PYTEST (DB): "
[static]
```

**12.14.4.63 RECORD\_TIME**

```
bool RECORD_TIME = True
[static]
```

Enable time-logging with the 'Log.time' decorator.

**12.14.4.64 RED**

```
str RED = "\033[31m"
[static]
```

ANSI code for red text.

**12.14.4.65 rel\_db**

```
str rel_db = "REL DB: "
[static]
```

**12.14.4.66 run\_f**

```
str run_f = "FILE EXEC: "
[static]
```

**12.14.4.67 run\_id**

```
int run_id = 1
[static]
```

**12.14.4.68 run\_q**

```
str run_q = "QUERY: " [static]
```

**12.14.4.69 sub\_gr**

```
str sub_gr = "SUBGRAPH: " [static]
```

**12.14.4.70 SUCCESS\_COLOR**

```
str SUCCESS_COLOR = GREEN [static]
```

ANSI color applied to the prefix of success messages.

**12.14.4.71 swap\_db**

```
str swap_db = "SWAP_DB: " [static]
```

**12.14.4.72 swap\_kg**

```
str swap_kg = "SWAP_GRAPH: " [static]
```

**12.14.4.73 t\_dump**

```
str t_dump = "[DUMP] " [static]
```

**12.14.4.74 test\_basic**

```
str test_basic = "CONNECT: " [static]
```

**12.14.4.75 test\_df**

```
str test_df = "GET DF: " [static]
```

**12.14.4.76 test\_info**

```
str test_info = "DB INFO: " [static]
```

**12.14.4.77 test\_ops**

```
str test_ops = "OPERATE: " [static]
```

#### 12.14.4.78 test\_tmp\_db

```
str test_tmp_db = "CREATE DB: " [static]
```

#### 12.14.4.79 TIME\_COLOR

```
str TIME_COLOR = CYAN [static]
```

ANSI color applied to the prefix of time-elapsed messages.

#### 12.14.4.80 USE\_COLORS

```
bool USE_COLORS = True [static]
```

Enable ANSI colors in output.

#### 12.14.4.81 WARNING\_COLOR

```
str WARNING_COLOR = YELLOW [static]
```

ANSI color applied to the prefix of ignored fail messages.

#### 12.14.4.82 WHITE

```
str WHITE = "\033[0m" [static]
```

ANSI code to reset color.

#### 12.14.4.83 YELLOW

```
str YELLOW = "\033[33m" [static]
```

ANSI code for yellow text.

The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/src/[util.py](#)

### 12.15 Metrics Class Reference

Utility class for computing and posting evaluation metrics.

## Public Member Functions

- None `__init__` (`self`)
- bool `post_payload` (`self`, `Dict[str, Any]` `payload`)
 

*POST directly to Blazor (soon to be deprecated)*
- None `post_basic_metrics` (`self`, `str book_id`, `str book_title`, `str summary`, `str gold_summary=""`, `str text=""`, `**Any kwargs`)
 

*POST basic evaluation scores to Blazor (ROUGE, BERTScore).*
- None `post_basic_output` (`self`, `str book_id`, `str book_title`, `str summary`)
 

*POST dummy date to Blazor.*

## Static Public Member Functions

- `Dict[str, Any]` `compute_basic_metrics` (`str summary`, `str gold_summary`, `str chunk`)
 

*Compute ROUGE and BERTScore.*
- `Dict[str, Any]` `create_summary_payload` (`str book_id`, `str book_title`, `str summary`, `str gold_summary`, `Dict[str, Any]` `metrics=None`)
 

*Create the full Blazor payload for a single book.*
- `Dict[str, Any]` `generate_default_metrics` (`float rouge1_f1=0.0`, `float rouge2_f1=0.0`, `float rougeL_f1=0.0`, `float rougeLsum_f1=0.0`, `float bert_precision=0.0`, `float bert_recall=0.0`, `float bert_f1=0.0`, `float boooook_score=0.0`, `float questeval_score=0.0`, `str qa_question1="UNKNOWN"`, `str qa_gold1="UNKNOWN"`, `str qa_generated1="UNKNOWN"`, `bool qa_correct1=False`, `float qa_accuracy1=0.0`, `str qa_question2="UNKNOWN"`, `str qa_gold2="UNKNOWN"`, `str qa_generated2="UNKNOWN"`, `bool qa_correct2=False`, `float qa_accuracy2=0.0`)
 

*Generate the metrics sub-payload with customizable default values.*
- `Dict[str, Any]` `generate_example_metrics` ()
 

*Create a placeholder payload with dummy values.*

## Public Attributes

- `HOST`
- `PORT`
- `url`

## Static Public Attributes

- `int timeout_seconds = 900`

### 12.15.1 Detailed Description

Utility class for computing and posting evaluation metrics.

### 12.15.2 Constructor & Destructor Documentation

#### 12.15.2.1 `__init__()`

```
None __init__ (
 self)
```

### 12.15.3 Member Function Documentation

#### 12.15.3.1 compute\_basic\_metrics()

```
Dict[str, Any] compute_basic_metrics (
 str summary,
 str gold_summary,
 str chunk) [static]
```

Compute ROUGE and BERTScore.

##### Parameters

<i>summary</i>	A text string containing a book summary
<i>gold_summary</i>	A summary to compare against
<i>chunk</i>	The original text of the chunk.

##### Returns

Dict containing 'rouge' and 'bertscore' keys. Scores are nested with inconsistent schema.

#### 12.15.3.2 create\_summary\_payload()

```
Dict[str, Any] create_summary_payload (
 str book_id,
 str book_title,
 str summary,
 str gold_summary,
 Dict[str, Any] metrics = None) [static]
```

Create the full Blazor payload for a single book.

##### Parameters

<i>book_id</i>	Unique identifier for one book.
<i>book_title</i>	String containing the title of a book.
<i>summary</i>	String containing a book summary.
<i>gold_summary</i>	Optional summary to compare against.
<i>metrics</i>	Dictionary containing various nested evaluation metrics.

##### Returns

A dictionary with C#-style key names.

#### 12.15.3.3 generate\_default\_metrics()

```
Dict[str, Any] generate_default_metrics (
 float rouge1_f1 = 0.0,
```

```

float rouge2_f1 = 0.0,
float rougeL_f1 = 0.0,
float rougeLsum_f1 = 0.0,
float bert_precision = 0.0,
float bert_recall = 0.0,
float bert_f1 = 0.0,
float boooook_score = 0.0,
float questeval_score = 0.0,
str qa_question1 = "UNKNOWN",
str qa_gold1 = "UNKNOWN",
str qa_generated1 = "UNKNOWN",
bool qa_correct1 = False,
float qa_accuracy1 = 0.0,
str qa_question2 = "UNKNOWN",
str qa_gold2 = "UNKNOWN",
str qa_generated2 = "UNKNOWN",
bool qa_correct2 = False,
float qa_accuracy2 = 0.0) [static]

```

Generate the metrics sub-payload with customizable default values.

#### Parameters

<i>rouge1_f1</i>	The ROUGE-1 evaluation metric.
<i>rouge2_f1</i>	The ROUGE-2 evaluation metric.
<i>rougeL_f1</i>	The ROUGE-L evaluation metric.
<i>rougeLsum_f1</i>	The ROUGE-Lsum evaluation metric.
<i>bert_precision</i>	The BERTScore precision score.
<i>bert_recall</i>	The BERTScore recall score.
<i>bert_f1</i>	The BERTScore F1 score.
<i>boooook_score</i>	The BoooookScore evaluation metric.
<i>questeval_score</i>	The QuestEval evaluation metric.
<i>qa_question1</i>	A question about the book.
<i>qa_gold1</i>	The correct answer to the question.
<i>qa_generated1</i>	A generated answer to judge.
<i>qa_correct1</i>	Whether our answer is correct.
<i>qa_accuracy1</i>	The accuracy score for this QA sample.
<i>qa_question2</i>	A question about the book.
<i>qa_gold2</i>	The correct answer to the question.
<i>qa_generated2</i>	A generated answer to judge.
<i>qa_correct2</i>	Whether our answer is correct.
<i>qa_accuracy2</i>	The accuracy score for this QA sample.

#### Returns

Dictionary containing various nested evaluation metrics.

#### 12.15.3.4 generate\_example\_metrics()

```
Dict[str, Any] generate_example_metrics() [static]
```

Create a placeholder payload with dummy values.

**Returns**

Full payload with nested metrics.

**12.15.3.5 post\_basic\_metrics()**

```
None post_basic_metrics (
 self,
 str book_id,
 str book_title,
 str summary,
 str gold_summary = "",
 str text = "",
 **Any kwargs)
```

POST basic evaluation scores to Blazor (ROUGE, BERTScore).

**Parameters**

<i>book_id</i>	Unique identifier for one book.
<i>book_title</i>	String containing the title of a book.
<i>summary</i>	String containing a book summary.
<i>gold_summary</i>	Optional summary to compare against.
<i>text</i>	A string containing text from the book.
<i>kwargs</i>	Any additional named arguments will be added to the payload.

**12.15.3.6 post\_basic\_output()**

```
None post_basic_output (
 self,
 str book_id,
 str book_title,
 str summary)
```

POST dummy date to Blazor.

**Parameters**

<i>book_id</i>	Unique identifier for one book.
<i>book_title</i>	String containing the title of a book.
<i>summary</i>	String containing a book summary.

**12.15.3.7 post\_payload()**

```
bool post_payload (
 self,
 Dict[str, Any] payload)
```

POST directly to Blazor (soon to be deprecated)

Verify and POST a given payload using the requests API.

**Parameters**

<code>payload</code>	JSON dictionary containing data for a single book.
----------------------	----------------------------------------------------

**Returns**

Whether the POST operation was successful.

## 12.15.4 Member Data Documentation

### 12.15.4.1 HOST

HOST

### 12.15.4.2 PORT

PORT

### 12.15.4.3 timeout\_seconds

```
int timeout_seconds = 900 [static]
```

### 12.15.4.4 url

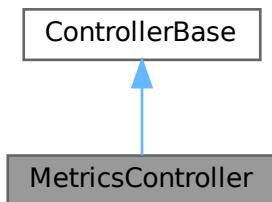
url

The documentation for this class was generated from the following file:

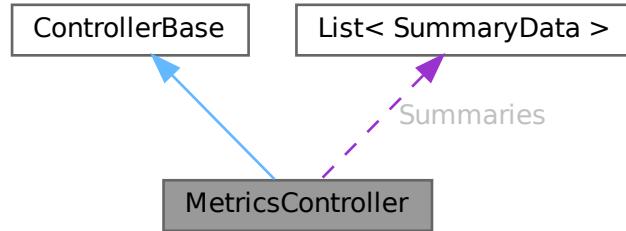
- /home/runner/work/dsci-capstone/dsci-capstone/src/components/[metrics.py](#)

## 12.16 MetricsController Class Reference

Inheritance diagram for MetricsController:



Collaboration diagram for MetricsController:



## Public Member Functions

- `MetricsController (ILogger< MetricsController > logger, IHubContext< MetricsHub > hubContext)`
- `async Task< IActionResult > Post ([FromBody] SummaryData summary)`
- `IActionResult GetIndex (int id)`
- `IActionResult GetAll ()`

## Private Attributes

- `readonly ILogger< MetricsController > _logger`
- `readonly IHubContext< MetricsHub > _hubContext`

## Static Private Attributes

- `static readonly List< SummaryData > Summaries = new()`

## 12.16.1 Constructor & Destructor Documentation

### 12.16.1.1 MetricsController()

```

MetricsController (
 ILogger< MetricsController > logger,
 IHubContext< MetricsHub > hubContext)

```

## 12.16.2 Member Function Documentation

### 12.16.2.1 GetAll()

```
IActionResult GetAll ()
```

### 12.16.2.2 GetIndex()

```
IActionResult GetIndex (
 int id)
```

### 12.16.2.3 Post()

```
async Task< IActionResult > Post (
 [FromBody] SummaryData summary)
```

## 12.16.3 Member Data Documentation

### 12.16.3.1 \_hubContext

```
readonly IHubContext<MetricsHub> _hubContext [private]
```

### 12.16.3.2 \_logger

```
readonly ILogger<MetricsController> _logger [private]
```

### 12.16.3.3 Summaries

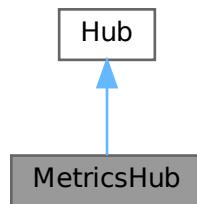
```
readonly List<SummaryData> Summaries = new() [static], [private]
```

The documentation for this class was generated from the following file:

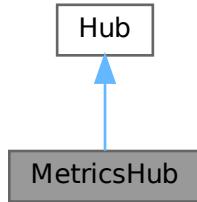
- /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Controllers/MetricsController.cs

## 12.17 MetricsHub Class Reference

Inheritance diagram for MetricsHub:



Collaboration diagram for MetricsHub:



## Public Member Functions

- `MetricsHub (ILogger< MetricsHub >? logger=null)`
- `override async Task OnConnectedAsync ()`
- `override async Task OnDisconnectedAsync (Exception? exception)`

## Private Attributes

- `readonly? ILogger< MetricsHub > _logger`

### 12.17.1 Constructor & Destructor Documentation

#### 12.17.1.1 MetricsHub()

```
MetricsHub (
 ILogger< MetricsHub >? logger = null)
```

### 12.17.2 Member Function Documentation

#### 12.17.2.1 OnConnectedAsync()

```
override async Task OnConnectedAsync ()
```

#### 12.17.2.2 OnDisconnectedAsync()

```
override async Task OnDisconnectedAsync (
 Exception? exception)
```

### 12.17.3 Member Data Documentation

#### 12.17.3.1 `_logger`

```
readonly? ILogger<MetricsHub> _logger [private]
```

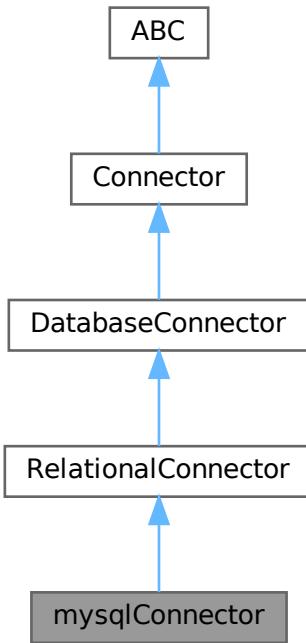
The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Hubs/MetricsHub.cs

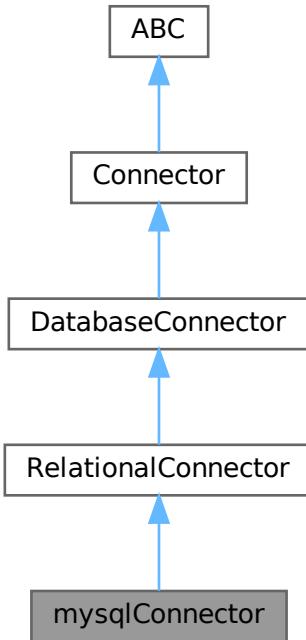
## 12.18 mysqlConnector Class Reference

A relational database connector configured for MySQL.

Inheritance diagram for mysqlConnector:



Collaboration diagram for mysqlConnector:



### Public Member Functions

- None `__init__` (self, bool `verbose=False`)

*Configures the relational connector.*

### Public Member Functions inherited from [RelationalConnector](#)

- "RelationalConnector" `from_env` (cls, bool `verbose=False`)  
*Decides what type of relational connector to create using the .env file.*
- None `change_database` (self, str `new_database`)  
*Update the connection URI to reference a different database in the same engine.*
- bool `test_operations` (self, bool `raise_error=True`)  
*Establish a basic connection to the database, and test full functionality.*
- bool `check_connection` (self, str `log_source`, bool `raise_error=True`)  
*Minimal connection test to determine if our connection string is valid.*
- Optional[DataFrame] `execute_query` (self, str `query`)  
*Send a single command to the database connection.*
- DataFrame `get_dataframe` (self, str `name`, List[str] `columns=[ ]`)  
*Automatically generate and run a query for the specified table using SQLAlchemy.*
- None `create_database` (self, str `database_name`)  
*Use the current database connection to create a sibling database in this engine.*
- None `drop_database` (self, str `database_name=''`)  
*Delete all data stored in a particular database.*
- bool `database_exists` (self, str `database_name`)  
*Search for an existing database using the provided name.*

## Public Member Functions inherited from DatabaseConnector

- None [configure](#) (self, str DB, str database\_name)  
*Read connection settings from the .env file.*
- Generator[None, None, None] [temp\\_database](#) (self, str database\_name)  
*Temporarily switch to a pseudo-database, creating and dropping it if needed.*
- List[Optional[DataFrame]] [execute\\_combined](#) (self, str multi\_query)  
*Run several database commands in sequence.*
- List[Optional[DataFrame]] [execute\\_file](#) (self, str filename)  
*Run several database commands from a file.*

## Static Public Attributes

- dict [specific\\_queries](#)

## Additional Inherited Members

## Public Attributes inherited from RelationalConnector

- database\_name
- verbose
- connection\_string
- db\_type

## Public Attributes inherited from DatabaseConnector

- verbose  
*Whether to print debug messages.*
- db\_type
- db\_engine
- username
- password
- host
- port
- connection\_string

## Protected Member Functions inherited from RelationalConnector

- List[str] [\\_split\\_combined](#) (self, str multi\_query)  
*Divides a string into non-divisible SQL queries using sqlparse.*
- bool [\\_returns\\_data](#) (self, str query)  
*Checks if a query is structured in a way that returns real data, and not status messages.*
- bool [\\_parsable\\_to\\_df](#) (self, Tuple[Any, Any] result)  
*Checks if the result of a SQL query is valid (i.e.*

## Protected Member Functions inherited from DatabaseConnector

- bool [\\_is\\_single\\_query](#) (self, str query)  
*Checks if a string contains multiple queries.*

### 12.18.1 Detailed Description

A relational database connector configured for MySQL.

#### Note

Should be hidden from the user using a factory method.

### 12.18.2 Constructor & Destructor Documentation

#### 12.18.2.1 `__init__()`

```
None __init__ (
 self,
 bool verbose = False)
```

Configures the relational connector.

#### Parameters

<code>verbose</code>	Whether to print success and failure messages.
----------------------	------------------------------------------------

Reimplemented from [RelationalConnector](#).

### 12.18.3 Member Data Documentation

#### 12.18.3.1 `specific_queries`

```
dict specific_queries [static]
```

#### Initial value:

```
= {
 "MYSQL": [
 "SELECT DATABASE();", # Single value, name of the current database.
 "SHOW DATABASES;", # List of databases the secondary user can access.
] # List of all databases in the database engine.
}
```

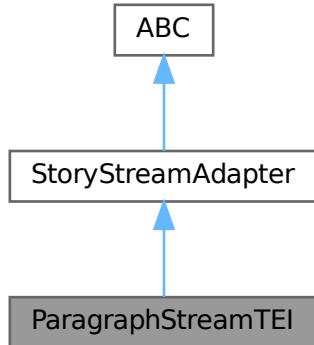
The documentation for this class was generated from the following file:

- [/home/runner/work/dsci-capstone/dsci-capstone/src/connectors/`relational.py`](#)

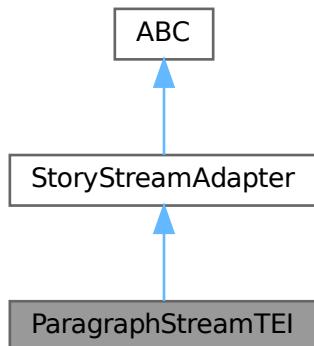
## 12.19 ParagraphStreamTEI Class Reference

Streams paragraphs from a TEI file as Chunk objects.

Inheritance diagram for ParagraphStreamTEI:



Collaboration diagram for ParagraphStreamTEI:



## Public Member Functions

- None `__init__` (self, str `tei_path`, int `book_id`, int `story_id`, list[str] `allowed_chapters`=None, str `start_inclusive`='', str `end_inclusive`='')
- Create a ParagraphStreamTEI object.*
- Iterator[`Chunk`] `stream_segments` (self)  
    *Yields sanitized parts of a book.*
- List[`Chunk`] `pre_compute_segments` (self)  
    *Splits the target book into paragraphs.*

## Public Member Functions inherited from StoryStreamAdapter

- Iterator[Chunk] `stream_paragraphs` (self)  
*Concrete helper method to split segments into paragraphs.*
- Iterator[str] `stream_sentences` (self)  
*Concrete helper method to split paragraphs into sentences.*

## Public Attributes

- `tei_path`
- `book_id`
- `story_id`
- `allowed_chapters`
- `start_inclusive`
- `end_inclusive`
- `lines`
- `root`
- `chunks`
- `xml_namespace`

## Static Public Attributes

- dict `xml_namespace` = {"tei": "http://www.tei-c.org/ns/1.0"}
- str `encoding` = "utf-8"

### 12.19.1 Detailed Description

Streams paragraphs from a TEI file as Chunk objects.

### 12.19.2 Constructor & Destructor Documentation

#### 12.19.2.1 `__init__()`

```
None __init__ (
 self,
 str tei_path,
 int book_id,
 int story_id,
 list[str] allowed_chapters = None,
 str start_inclusive = "",
 str end_inclusive = "")
```

Create a ParagraphStreamTEI object.

#### Parameters

<code>tei_path</code>	Path to an existing TEI XML file.
<code>book_id</code>	ID for this book.
<code>story_id</code>	ID for this story (may be same as <code>book_id</code> ).
<code>allowed_chapters</code>	A list of valid chapter titles. Must exactly match the contents of head.
<code>start_inclusive</code>	(Optional) Unique string representing the start of the book.
<code>end_inclusive</code>	(Optional) Unique string representing the end of the book.

### 12.19.3 Member Function Documentation

#### 12.19.3.1 pre\_compute\_segments()

```
List[Chunk] pre_compute_segments (
 self)
```

Splits the target book into paragraphs.

Yields Chunk objects for each paragraph (

) in the TEI file. Uses etree Element.sourceline to approximate start/end line in TEI. Supports optional start\_inclusive / end\_inclusive boundaries to slice text and stop iteration. Computes progress percentages using character counts:

- story\_percent: progress through the entire story
- chapter\_percent: progress through the current chapter Populates self.chunks so they can be streamed as requested by interface

#### 12.19.3.2 stream\_segments()

```
Iterator[Chunk] stream_segments (
 self)
```

Yields sanitized parts of a book.

- Story segments usually correspond to chapters.
- They serve as borders between chunking operations, ensuring chunks do not span multiple chapters. Implementation is handled by child classes BookStream, etc.
- Segments should be pre-cleaned and must contain 1 paragraph per line with all other newlines removed.

Reimplemented from [StoryStreamAdapter](#).

### 12.19.4 Member Data Documentation

#### 12.19.4.1 allowed\_chapters

allowed\_chapters

#### 12.19.4.2 book\_id

book\_id

#### 12.19.4.3 chunks

chunks

#### 12.19.4.4 encoding

```
str encoding = "utf-8" [static]
```

#### 12.19.4.5 end\_inclusive

```
end_inclusive
```

#### 12.19.4.6 lines

```
lines
```

#### 12.19.4.7 root

```
root
```

#### 12.19.4.8 start\_inclusive

```
start_inclusive
```

#### 12.19.4.9 story\_id

```
story_id
```

#### 12.19.4.10 tei\_path

```
tei_path
```

#### 12.19.4.11 xml\_namespace [1/2]

```
dict xml_namespace = {"tei": "http://www.tei-c.org/ns/1.0"} [static]
```

#### 12.19.4.12 xml\_namespace [2/2]

```
xml_namespace
```

The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/src/components/[book\\_conversion.py](#)

## 12.20 Plot Class Reference

Static plotting helpers for visualization.

### Static Public Member Functions

- None [time\\_elapsed\\_by\\_names](#) (str filename=".~/logs/charts/avg\_runtime.png")  
*Plot average elapsed time per function name, averaging across runs.*

### 12.20.1 Detailed Description

Static plotting helpers for visualization.

### 12.20.2 Member Function Documentation

#### 12.20.2.1 time\_elapsed\_by\_names()

```
None time_elapsed_by_names (
 str filename = ".~/logs/charts/avg_runtime.png") [static]
```

Plot average elapsed time per function name, averaging across runs.

##### Parameters

<i>filename</i>	Where to save the generated chart
-----------------	-----------------------------------

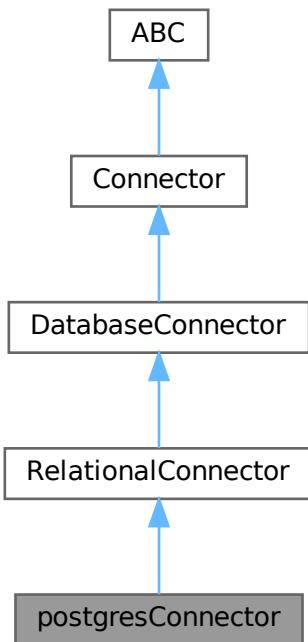
The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/src/[charts.py](#)

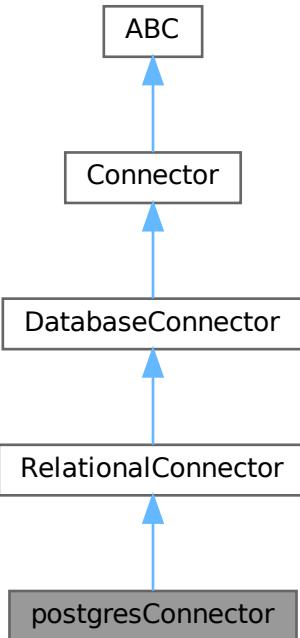
## 12.21 postgresConnector Class Reference

A relational database connector configured for PostgreSQL.

Inheritance diagram for postgresConnector:



Collaboration diagram for postgresConnector:



### Public Member Functions

- None `__init__` (self, bool `verbose=False`)

*Configures the relational connector.*

### Public Member Functions inherited from [RelationalConnector](#)

- "RelationalConnector" `from_env` (cls, bool `verbose=False`)  
*Decides what type of relational connector to create using the .env file.*
- None `change_database` (self, str `new_database`)  
*Update the connection URI to reference a different database in the same engine.*
- bool `test_operations` (self, bool `raise_error=True`)  
*Establish a basic connection to the database, and test full functionality.*
- bool `check_connection` (self, str `log_source`, bool `raise_error=True`)  
*Minimal connection test to determine if our connection string is valid.*
- Optional[DataFrame] `execute_query` (self, str `query`)  
*Send a single command to the database connection.*
- DataFrame `get_dataframe` (self, str `name`, List[str] `columns=[ ]`)  
*Automatically generate and run a query for the specified table using SQLAlchemy.*
- None `create_database` (self, str `database_name`)  
*Use the current database connection to create a sibling database in this engine.*
- None `drop_database` (self, str `database_name=''`)  
*Delete all data stored in a particular database.*
- bool `database_exists` (self, str `database_name`)  
*Search for an existing database using the provided name.*

## Public Member Functions inherited from DatabaseConnector

- None [configure](#) (self, str DB, str database\_name)  
*Read connection settings from the .env file.*
- Generator[None, None, None] [temp\\_database](#) (self, str database\_name)  
*Temporarily switch to a pseudo-database, creating and dropping it if needed.*
- List[Optional[DataFrame]] [execute\\_combined](#) (self, str multi\_query)  
*Run several database commands in sequence.*
- List[Optional[DataFrame]] [execute\\_file](#) (self, str filename)  
*Run several database commands from a file.*

## Static Public Attributes

- dict [specific\\_queries](#)

## Additional Inherited Members

### Public Attributes inherited from RelationalConnector

- database\_name
- verbose
- connection\_string
- db\_type

### Public Attributes inherited from DatabaseConnector

- verbose  
*Whether to print debug messages.*
- db\_type
- db\_engine
- username
- password
- host
- port
- connection\_string

### Protected Member Functions inherited from RelationalConnector

- List[str] [\\_split\\_combined](#) (self, str multi\_query)  
*Divides a string into non-divisible SQL queries using sqlparse.*
- bool [\\_returns\\_data](#) (self, str query)  
*Checks if a query is structured in a way that returns real data, and not status messages.*
- bool [\\_parsable\\_to\\_df](#) (self, Tuple[Any, Any] result)  
*Checks if the result of a SQL query is valid (i.e.*

### Protected Member Functions inherited from DatabaseConnector

- bool [\\_is\\_single\\_query](#) (self, str query)  
*Checks if a string contains multiple queries.*

### 12.21.1 Detailed Description

A relational database connector configured for PostgreSQL.

#### Note

Should be hidden from the user using a factory method.

### 12.21.2 Constructor & Destructor Documentation

#### 12.21.2.1 `__init__()`

```
None __init__ (
 self,
 bool verbose = False)
```

Configures the relational connector.

#### Parameters

<code>verbose</code>	Whether to print success and failure messages.
----------------------	------------------------------------------------

Reimplemented from [RelationalConnector](#).

### 12.21.3 Member Data Documentation

#### 12.21.3.1 `specific_queries`

```
dict specific_queries [static]
```

#### Initial value:

```
= {
 "POSTGRES": [
 "SELECT current_database();", # Single value, name of the current database.
 "SELECT datname FROM pg_database;", # List of ALL databases, even ones we cannot access.
] # List of all databases in the database engine.
}
```

The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/src/connectors/[relational.py](#)

## 12.22 PRF1Metric Class Reference

#### Properties

- string `Name` [get, set]
- double `Precision` [get, set]
- double `Recall` [get, set]
- double `F1Score` [get, set]

## 12.22.1 Property Documentation

### 12.22.1.1 F1Score

```
double F1Score [get], [set]
```

### 12.22.1.2 Name

```
string Name [get], [set]
```

### 12.22.1.3 Precision

```
double Precision [get], [set]
```

### 12.22.1.4 Recall

```
double Recall [get], [set]
```

The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/PRF1Metric.cs

## 12.23 QAItem Class Reference

### Properties

- string [Question](#) [get, set]
- string [GoldAnswer](#) [get, set]
- string [GeneratedAnswer](#) [get, set]
- bool? [IsCorrect](#) [get, set]
- double? [Accuracy](#) [get, set]

### 12.23.1 Property Documentation

#### 12.23.1.1 Accuracy

```
double? Accuracy [get], [set]
```

#### 12.23.1.2 GeneratedAnswer

```
string GeneratedAnswer [get], [set]
```

### 12.23.1.3 GoldAnswer

```
string GoldAnswer [get], [set]
```

### 12.23.1.4 IsCorrect

```
bool? IsCorrect [get], [set]
```

### 12.23.1.5 Question

```
string Question [get], [set]
```

The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/QAItem.cs

## 12.24 QAMetric Class Reference

### Properties

- List<[QAItem](#)> QAItems = new() [get, set]
- double AverageAccuracy [get]

### 12.24.1 Property Documentation

#### 12.24.1.1 AverageAccuracy

```
double AverageAccuracy [get]
```

#### 12.24.1.2 QAItems

```
List<QAItem> QAItems = new() [get], [set]
```

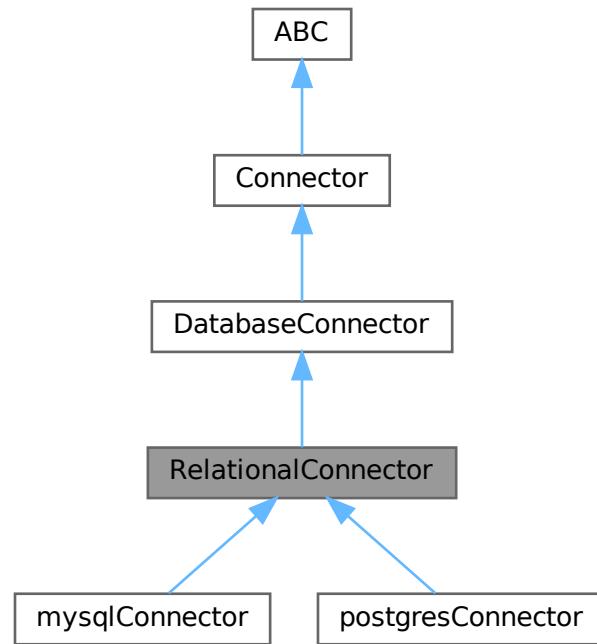
The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/QAMetric.cs

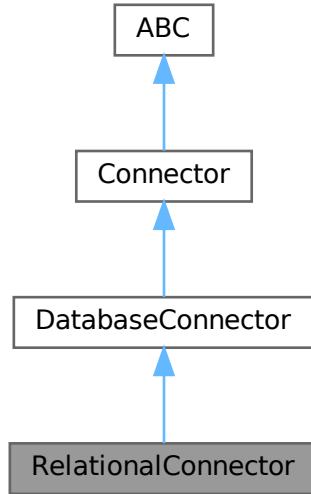
## 12.25 RelationalConnector Class Reference

Connector for relational databases (MySQL, PostgreSQL).

Inheritance diagram for RelationalConnector:



Collaboration diagram for RelationalConnector:



## Public Member Functions

- None `__init__` (self, bool `verbose`, List[str] `specific_queries`)  
*Creates a new database connector.*
- "RelationalConnector" `from_env` (cls, bool `verbose=False`)  
*Decides what type of relational connector to create using the .env file.*
- None `change_database` (self, str `new_database`)  
*Update the connection URI to reference a different database in the same engine.*
- bool `test_operations` (self, bool `raise_error=True`)  
*Establish a basic connection to the database, and test full functionality.*
- bool `check_connection` (self, str `log_source`, bool `raise_error=True`)  
*Minimal connection test to determine if our connection string is valid.*
- Optional[DataFrame] `execute_query` (self, str `query`)  
*Send a single command to the database connection.*
- DataFrame `get_dataframe` (self, str `name`, List[str] `columns=[]`)  
*Automatically generate and run a query for the specified table using SQLAlchemy.*
- None `create_database` (self, str `database_name`)  
*Use the current database connection to create a sibling database in this engine.*
- None `drop_database` (self, str `database_name=""`)  
*Delete all data stored in a particular database.*
- bool `database_exists` (self, str `database_name`)  
*Search for an existing database using the provided name.*

## Public Member Functions inherited from DatabaseConnector

- None `configure` (self, str DB, str database\_name)  
*Read connection settings from the .env file.*
- Generator[None, None, None] `temp_database` (self, str database\_name)  
*Temporarily switch to a pseudo-database, creating and dropping it if needed.*
- List[Optional[DataFrame]] `execute_combined` (self, str multi\_query)  
*Run several database commands in sequence.*
- List[Optional[DataFrame]] `execute_file` (self, str filename)  
*Run several database commands from a file.*

## Public Attributes

- `database_name`
- `verbose`
- `connection_string`
- `db_type`

## Public Attributes inherited from DatabaseConnector

- `verbose`  
*Whether to print debug messages.*
- `db_type`
- `db_engine`
- `username`
- `password`
- `host`
- `port`
- `connection_string`

## Protected Member Functions

- List[str] `_split_combined` (self, str multi\_query)  
*Divides a string into non-divisible SQL queries using sqlparse.*
- bool `_returns_data` (self, str query)  
*Checks if a query is structured in a way that returns real data, and not status messages.*
- bool `_parsable_to_df` (self, Tuple[Any, Any] result)  
*Checks if the result of a SQL query is valid (i.e.*

## Protected Member Functions inherited from DatabaseConnector

- bool `_is_single_query` (self, str query)  
*Checks if a string contains multiple queries.*

### 12.25.1 Detailed Description

Connector for relational databases (MySQL, PostgreSQL).

Uses SQLAlchemy to abstract complex database operations. Hard-coded queries are used for testing purposes, and depend on the specific engine.

## 12.25.2 Constructor & Destructor Documentation

### 12.25.2.1 `__init__()`

```
None __init__ (
 self,
 bool verbose,
 List[str] specific_queries)
```

Creates a new database connector.

Use [src.connectors.relational.RelationalConnector.from\\_env](#) instead (this is called by derived classes).

#### Parameters

<code>verbose</code>	Whether to print success and failure messages.
<code>specific_queries</code>	A list of helpful SQL queries.

Reimplemented from [DatabaseConnector](#).

Reimplemented in [mysqlConnector](#), and [postgresConnector](#).

## 12.25.3 Member Function Documentation

### 12.25.3.1 `_parsable_to_df()`

```
bool _parsable_to_df (
 self,
 Tuple[Any, Any] result) [protected]
```

Checks if the result of a SQL query is valid (i.e.

can be converted to a Pandas DataFrame).

- SQLAlchemy CursorResult exposes .returns\_rows and .keys().
- These must be fetched earlier, before the cursor is closed.

#### Parameters

<code>result</code>	The tuple result (rows, columns) of a SQL, Cypher, or JSON query.
---------------------	-------------------------------------------------------------------

#### Returns

Whether the object is parsable to DataFrame.

Reimplemented from [DatabaseConnector](#).

### 12.25.3.2 `_returns_data()`

```
bool _returns_data (
```

```
self,
str query) [protected]
```

Checks if a query is structured in a way that returns real data, and not status messages.

Determines whether a SQL query should yield tabular data.

- Uses an exclusion list - commands that definitely return only status / row count.
- Everything else falls through to execution for validation.

#### Parameters

<i>query</i>	A single pre-validated SQL query string.
--------------	------------------------------------------

#### Returns

Whether the query is intended to fetch data (true) or might return a status message (false).

Reimplemented from [DatabaseConnector](#).

### 12.25.3.3 `_split_combined()`

```
List[str] _split_combined (
 self,
 str multi_query) [protected]
```

Divides a string into non-divisible SQL queries using `sqlparse`.

#### Parameters

<i>multi_query</i>	A string containing multiple queries.
--------------------	---------------------------------------

#### Returns

A list of single-query strings.

Reimplemented from [DatabaseConnector](#).

### 12.25.3.4 `change_database()`

```
None change_database (
 self,
 str new_database)
```

Update the connection URI to reference a different database in the same engine.

#### Parameters

<i>new_database</i>	The name of the database to connect to.
---------------------	-----------------------------------------

Reimplemented from [DatabaseConnector](#).

#### 12.25.3.5 check\_connection()

```
bool check_connection (
 self,
 str log_source,
 bool raise_error = True)
```

Minimal connection test to determine if our connection string is valid.

Connect to our relational database using SQLAlchemy's engine.begin()

##### Parameters

<i>log_source</i>	The Log class prefix indicating which method is performing the check.
<i>raise_error</i>	Whether to raise an error on connection failure.

##### Returns

Whether the connection test was successful.

##### Exceptions

<i>Log.Failure</i>	If <i>raise_error</i> is True and the connection test fails to complete.
--------------------	--------------------------------------------------------------------------

Reimplemented from [Connector](#).

#### 12.25.3.6 create\_database()

```
None create_database (
 self,
 str database_name)
```

Use the current database connection to create a sibling database in this engine.

##### Parameters

<i>database_name</i>	The name of the new database to create.
----------------------	-----------------------------------------

##### Exceptions

<i>Log.Failure</i>	If we fail to create the requested database for any reason.
--------------------	-------------------------------------------------------------

Reimplemented from [DatabaseConnector](#).

### 12.25.3.7 database\_exists()

```
bool database_exists (
 self,
 str database_name)
```

Search for an existing database using the provided name.

#### Parameters

<code>database_name</code>	The name of a database to search for.
----------------------------	---------------------------------------

#### Returns

Whether the database is visible to this connector.

Reimplemented from [DatabaseConnector](#).

### 12.25.3.8 drop\_database()

```
None drop_database (
 self,
 str database_name = "")
```

Delete all data stored in a particular database.

#### Parameters

<code>database_name</code>	The name of an existing database.
----------------------------	-----------------------------------

#### Exceptions

<code>Log.Failure</code>	If we fail to drop the target database for any reason.
--------------------------	--------------------------------------------------------

Reimplemented from [DatabaseConnector](#).

### 12.25.3.9 execute\_query()

```
Optional[DataFrame] execute_query (
 self,
 str query)
```

Send a single command to the database connection.

#### Note

If a result is returned, it will be converted to a DataFrame.

**Parameters**

<i>query</i>	A single query to perform on the database.
--------------	--------------------------------------------

**Returns**

DataFrame containing the result of the query, or None

**Exceptions**

<i>Log.Failure</i>	If the query fails to execute.
--------------------	--------------------------------

Reimplemented from [DatabaseConnector](#).

**12.25.3.10 from\_env()**

```
"RelationalConnector" from_env (
 cls,
 bool verbose = False)
```

Decides what type of relational connector to create using the .env file.

**Parameters**

<i>verbose</i>	Whether to print success and failure messages.
----------------	------------------------------------------------

**Exceptions**

<i>Log.Failure</i>	If the .env file contains an invalid DB_ENGINE value.
--------------------	-------------------------------------------------------

**12.25.3.11 get\_dataframe()**

```
DataFrame get_dataframe (
 self,
 str name,
 List[str] columns = [])
```

Automatically generate and run a query for the specified table using SQLAlchemy.

**Parameters**

<i>name</i>	The name of an existing table or collection in the database.
<i>columns</i>	A list of column names to keep.

**Returns**

Sorted DataFrame containing the requested data

**Exceptions**

<i>Log.Failure</i>	If we fail to create the requested DataFrame for any reason.
--------------------	--------------------------------------------------------------

Reimplemented from [DatabaseConnector](#).

**12.25.3.12 test\_operations()**

```
bool test_operations (
 self,
 bool raise_error = True)
```

Establish a basic connection to the database, and test full functionality.

Can be configured to fail silently, which enables retries or external handling.

**Parameters**

<i>raise_error</i>	Whether to raise an error on connection failure.
--------------------	--------------------------------------------------

**Returns**

Whether the connection test was successful.

**Exceptions**

<i>Log.Failure</i>	If <i>raise_error</i> is True and the connection test fails to complete.
--------------------	--------------------------------------------------------------------------

Reimplemented from [Connector](#).

**12.25.4 Member Data Documentation****12.25.4.1 connection\_string**

connection\_string

**12.25.4.2 database\_name**

database\_name

**12.25.4.3 db\_type**

db\_type

#### 12.25.4.4 verbose

verbose

The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/src/connectors/[relational.py](#)

## 12.26 RelationExtractor Class Reference

### Public Member Functions

- [`\_\_init\_\_`](#) (`self`, `model_name="Babelscape/rebel-large"`, `max_tokens=1024`)
- `List[Tuple[str, str, str]|str] extract` (`self`, `str text`, `bool parse_tuples=False`)

### Public Attributes

- [`nlp`](#)
- [`sentencizer`](#)
- [`tokenizer`](#)
- [`model`](#)
- [`max\_tokens`](#)
- [`tuple\_delim`](#)

### 12.26.1 Constructor & Destructor Documentation

#### 12.26.1.1 `__init__()`

```
__init__ (
 self,
 model_name = "Babelscape/rebel-large",
 max_tokens = 1024)
```

### 12.26.2 Member Function Documentation

#### 12.26.2.1 `extract()`

```
List[Tuple[str, str, str] | str] extract (
 self,
 str text,
 bool parse_tuples = False)
```

### 12.26.3 Member Data Documentation

#### 12.26.3.1 `max_tokens`

`max_tokens`

### 12.26.3.2 model

model

### 12.26.3.3 nlp

nlp

### 12.26.3.4 sentencizer

sentencizer

### 12.26.3.5 tokenizer

tokenizer

### 12.26.3.6 tuple\_delim

tuple\_delim

The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/src/components/[relation\\_extraction.py](#)

## 12.27 ScalarMetric Class Reference

### Properties

- string [Name](#) [get, set]
- double [Value](#) [get, set]

### 12.27.1 Property Documentation

#### 12.27.1.1 Name

string Name [get], [set]

#### 12.27.1.2 Value

double Value [get], [set]

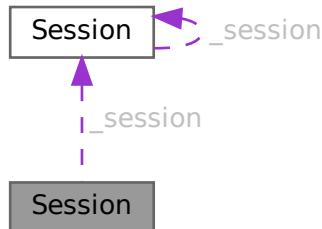
The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/[ScalarMetric.cs](#)

## 12.28 Session Class Reference

Stores active database connections and configuration settings.

Collaboration diagram for Session:



### Public Member Functions

- Self `__new__` (cls, \*Any args, \*\*Any kwargs)  
*Creates a new session at first access, otherwise returns the existing session.*
- None `__init__` (self, bool `verbose=False`)  
*Initializes the session using the .env file.*

### Public Attributes

- `verbose`  
*Enables or disables the components from printing debug info.*
- `relational_db`  
*Stores RDF-compliant semantic triples.*
- `docs_db`  
*Stores input text, pre-processed chunks, JSON intermediates, and final output.*
- `graph_db`  
*Stores entities (nodes) and relations (edges).*
- `main_graph`  
*Main storage for initial pipeline.*
- `metrics`  
*The metrics class needs an instance to read the .env file.*

### Static Protected Attributes

- `_instance = None`
- `bool _created = False`
- `Session _session = None`

## 12.28.1 Detailed Description

Stores active database connections and configuration settings.

- This class implements Singleton design so only one session can be created.

## 12.28.2 Constructor & Destructor Documentation

### 12.28.2.1 `__init__()`

```
None __init__ (
 self,
 bool verbose = False)
```

Initializes the session using the .env file.

- The relational database connector is created using a Factory Method, choosing mysql or postgres based on the .env file.
- The document database connector is created normally since mongo is the only supported option.
- The graph database connector is created normally since neo4j is the only supported option.

## 12.28.3 Member Function Documentation

### 12.28.3.1 `__new__()`

```
Self __new__ (
 cls,
 *Any args,
 **Any kwargs)
```

Creates a new session at first access, otherwise returns the existing session.

#### Parameters

<code>*args</code>	Positional arguments forwarded to <code>init()</code> .
<code>**kwargs</code>	Keyword arguments forwarded to <code>init()</code> .

#### Returns

The new global Session singleton.

## 12.28.4 Member Data Documentation

### 12.28.4.1 `_created`

```
bool _created = False [static], [protected]
```

#### 12.28.4.2 `_instance`

```
_instance = None [static], [protected]
```

#### 12.28.4.3 `_session`

```
Session _session = None [static], [protected]
```

#### 12.28.4.4 `docs_db`

```
docs_db
```

Stores input text, pre-processed chunks, JSON intermediates, and final output.

#### 12.28.4.5 `graph_db`

```
graph_db
```

Stores entities (nodes) and relations (edges).

#### 12.28.4.6 `main_graph`

```
main_graph
```

Main storage for initial pipeline.

#### 12.28.4.7 `metrics`

```
metrics
```

The metrics class needs an instance to read the .env file.

#### 12.28.4.8 `relational_db`

```
relational_db
```

Stores RDF-compliant semantic triples.

#### 12.28.4.9 `verbose`

```
verbose
```

Enables or disables the components from printing debug info.

The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/src/core/[context.py](#)

## 12.29 Story Class Reference

### Public Member Functions

- None `_init_(self, StoryStreamAdapter reader)`
- Iterator[`Chunk`] `stream_chunks_(self)`
- None `pre_split_chunks_(self, int max_chunk_length)`  
*Splits paragraphs into chunks.*

### Public Attributes

- `reader`

### Protected Member Functions

- None `_merge_chunks_(self, List[Chunk] segs, int max_len)`
- `Chunk _make_single_(self, Chunk seg, str text, int max_len, Optional[Chunk] start=None)`

#### 12.29.1 Constructor & Destructor Documentation

##### 12.29.1.1 `_init_()`

```
None __init__ (
 self,
 StoryStreamAdapter reader)
```

#### 12.29.2 Member Function Documentation

##### 12.29.2.1 `_make_single()`

```
Chunk _make_single (
 self,
 Chunk seg,
 str text,
 int max_len,
 Optional[Chunk] start = None) [protected]
```

##### 12.29.2.2 `_merge_chunks()`

```
None _merge_chunks (
 self,
 List[Chunk] segs,
 int max_len) [protected]
```

### 12.29.2.3 pre\_split\_chunks()

```
None pre_split_chunks (
 self,
 int max_chunk_length)
```

Splits paragraphs into chunks.

- Populates self.chunks with Chunk objects that obey max\_chunk\_length.
- Combines adjacent paragraphs when possible.
- Falls back to splitting by sentences if one paragraph is too long.

### 12.29.2.4 stream\_chunks()

```
Iterator[Chunk] stream_chunks (
 self)
```

## 12.29.3 Member Data Documentation

### 12.29.3.1 reader

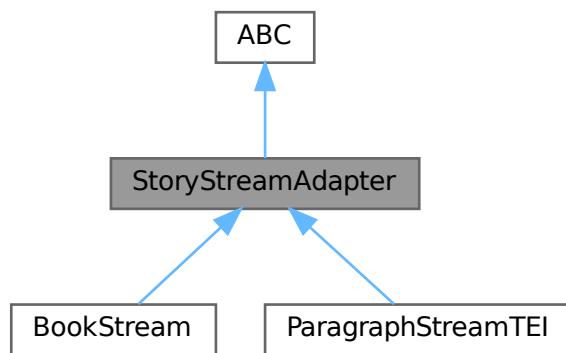
```
reader
```

The documentation for this class was generated from the following file:

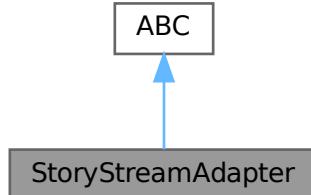
- /home/runner/work/dsci-capstone/dsci-capstone/src/components/book\_conversion.py

## 12.30 StoryStreamAdapter Class Reference

Inheritance diagram for StoryStreamAdapter:



Collaboration diagram for StoryStreamAdapter:



## Public Member Functions

- `Iterator[Chunk] stream_segments (self)`  
*Yields sanitized parts of a book.*
- `Iterator[Chunk] stream_paragraphs (self)`  
*Concrete helper method to split segments into paragraphs.*
- `Iterator[str] stream_sentences (self)`  
*Concrete helper method to split paragraphs into sentences.*

### 12.30.1 Member Function Documentation

#### 12.30.1.1 stream\_paragraphs()

```
Iterator[Chunk] stream_paragraphs (
 self)
```

Concrete helper method to split segments into paragraphs.

The Chunk class is repurposed here so we pass location info. Depending on the Story.pre\_split\_chunks implementation, this might be unnecessary.

#### 12.30.1.2 stream\_segments()

```
Iterator[Chunk] stream_segments (
 self)
```

Yields sanitized parts of a book.

- Story segments usually correspond to chapters.
- They serve as borders between chunking operations, ensuring chunks do not span multiple chapters. Implementation is handled by child classes BookStream, etc.
- Segments should be pre-cleaned and must contain 1 paragraph per line with all other newlines removed.

Reimplemented in [ParagraphStreamTEI](#), and [BookStream](#).

### 12.30.1.3 stream\_sentences()

```
Iterator[str] stream_sentences (
 self)
```

Concrete helper method to split paragraphs into sentences.

Mostly for debugging.

The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/src/components/book\_conversion.py

## 12.31 SummaryData Class Reference

### Properties

- string BookID [get, set]
- string BookTitle [get, set]
- string SummaryText [get, set]
- string GoldSummaryText [get, set]
- **SummaryMetrics Metrics = new()** [get, set]
- List<[QAMetric](#)> QAResults = new() [get, set]

### 12.31.1 Property Documentation

#### 12.31.1.1 BookID

```
string BookID [get], [set]
```

#### 12.31.1.2 BookTitle

```
string BookTitle [get], [set]
```

#### 12.31.1.3 GoldSummaryText

```
string GoldSummaryText [get], [set]
```

#### 12.31.1.4 Metrics

```
SummaryMetrics Metrics = new() [get], [set]
```

#### 12.31.1.5 QAResults

```
List<QAMetric> QAResults = new() [get], [set]
```

### 12.31.1.6 SummaryText

```
string SummaryText [get], [set]
```

The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/[SummaryData.cs](#)

## 12.32 SummaryMetrics Class Reference

### Static Public Member Functions

- static [SummaryMetrics GetDefault \(\)](#)

### Properties

- List<[PRF1Metric](#)> [PRF1Metrics = new\(\)](#) [get, set]
- [QAMetric QA = new\(\)](#) [get, set]
- List<[ScalarMetric](#)> [ScalarMetrics = new\(\)](#) [get, set]

### 12.32.1 Member Function Documentation

#### 12.32.1.1 GetDefault()

```
static SummaryMetrics GetDefault \(\) [static]
```

### 12.32.2 Property Documentation

#### 12.32.2.1 PRF1Metrics

```
List<PRF1Metric> PRF1Metrics = new\(\) [get], [set]
```

#### 12.32.2.2 QA

```
QAMetric QA = new\(\) [get], [set]
```

#### 12.32.2.3 ScalarMetrics

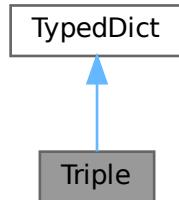
```
List<ScalarMetric> ScalarMetrics = new\(\) [get], [set]
```

The documentation for this class was generated from the following file:

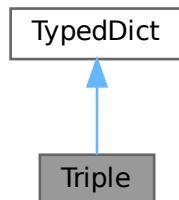
- /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/[SummaryMetrics.cs](#)

## 12.33 Triple Class Reference

Inheritance diagram for Triple:



Collaboration diagram for Triple:



### Static Public Attributes

- str `s`
- str `r`
- str `o`

#### 12.33.1 Member Data Documentation

##### 12.33.1.1 `o`

```
str o [static]
```

##### 12.33.1.2 `r`

```
str r [static]
```

### 12.33.1.3 s

```
str s [static]
```

The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/src/components/[fact\\_storage.py](#)



# Chapter 13

## File Documentation

### 13.1 /home/runner/work/dsci-capstone/dsci-capstone/conftest.py File Reference

#### Namespaces

- namespace `conftest`

#### Functions

- None `pytest_adoption` (Any parser)
- Generator[`Session`, None, None] `session` (pytest.FixtureRequest request)  
*Fixture to create session.*
- Generator[`RelationalConnector`, None, None] `relational_db` (`Session session`)  
*Fixture to get relational database connection.*
- Generator[`DocumentConnector`, None, None] `docs_db` (`Session session`)  
*Fixture to get document database connection.*
- Generator[`GraphConnector`, None, None] `graph_db` (`Session session`)  
*Fixture to get graph database connection.*
- Generator[`KnowledgeGraph`, None, None] `main_graph` (pytest.FixtureRequest request, `GraphConnector graph_db`, `Session session`)  
*Fixture to get main graph database connection.*

### 13.2 /home/runner/work/dsci-capstone/dsci-capstone/src/charts.py File Reference

#### Classes

- class `Plot`  
*Static plotting helpers for visualization.*

## Namespaces

- namespace [src](#)
- namespace [src.charts](#)

## 13.3 /home/runner/work/dsci-capstone/dsci-capstone/src/\_\_init\_\_.py File Reference

### Namespaces

- namespace [src](#)

## 13.4 /home/runner/work/dsci-capstone/dsci-capstone/src/components/\_\_init\_\_.py File Reference

### Namespaces

- namespace [src](#)
- namespace [src.components](#)

## 13.5 /home/runner/work/dsci-capstone/dsci-capstone/src/connectors/\_\_init\_\_.py File Reference

### Namespaces

- namespace [src](#)
- namespace [src.connectors](#)

## 13.6 /home/runner/work/dsci-capstone/dsci-capstone/src/core/\_\_init\_\_.py File Reference

### Namespaces

- namespace [src](#)
- namespace [src.core](#)

## 13.7 /home/runner/work/dsci-capstone/dsci-capstone/tests/\_\_init\_\_.py File Reference

### Namespaces

- namespace [tests](#)

## 13.8 /home/runner/work/dsci-capstone/dsci-capstone/src/components/book\_conversion.py File Reference

### Classes

- class [Chunk](#)  
*Lightweight container for a span of story text.*
- class [StoryStreamAdapter](#)
- class [Story](#)
- class [ParagraphStreamTEI](#)  
*Streams paragraphs from a TEI file as Chunk objects.*
- class [Book](#)
- class [BookStream](#)
- class [BookFactory](#)
- class [EPUBToTEI](#)  
*Converts EPUB files to XML format (TEI specification).*

### Namespaces

- namespace [src](#)
- namespace [src.components](#)
- namespace [src.components.book\\_conversion](#)

### Variables

- `nlp` = spacy.blank("en")
- `sentencizer` = nlp.add\_pipe("sentencizer")

## 13.9 /home/runner/work/dsci-capstone/dsci-capstone/src/components/corpus.py File Reference

### Namespaces

- namespace [src](#)
- namespace [src.components](#)
- namespace [src.components.corpus](#)

### Functions

- `load_booksum ()`
- `to_df_booksum (ds)`
- `load_narrativeqa ()`
- `to_df_nqa (ds)`
- `normalize_title (t)`
- `merge_dataframes (df1, df2, suffix1, suffix2, key_columns)`
- `fuzzy_merge_titles (df1, df2, suffix1, suffix2, key="title", threshold=90, scorer=fuzz.token_sort_ratio)`  
*Perform a two-way fuzzy merge between two DataFrames on a text column (e.g., book titles).*

## Variables

- `df_booksum = load_booksum()`
- `df_nqa = load_narrativeqa()`
- `df = fuzzy_merge_titles(df_booksum, df_nqa, "_booksum", "_nqa", key="title", threshold=70)`
- `index`
- `m = Metrics()`

## 13.10 /home/runner/work/dsci-capstone/dsci-capstone/src/components/fact\_storage.py File Reference

### Classes

- class `Triple`
- class `KnowledgeGraph`

*Manages a single graph within Neo4j.*

### Namespaces

- namespace `src`
- namespace `src.components`
- namespace `src.components.fact_storage`

## 13.11 /home/runner/work/dsci-capstone/dsci-capstone/src/components/metrics.py File Reference

### Classes

- class `Metrics`

*Utility class for computing and posting evaluation metrics.*

### Namespaces

- namespace `src`
- namespace `src.components`
- namespace `src.components.metrics`

### Functions

- Dict[str, Any] `run_questeval` (Dict[str, Any] chunk, \*str qeval\_task="summarization", bool use\_cuda=False, bool use\_question\_weighter=True)
 

*Run QuestEval metric calculation.*
- Dict[str, Any] `run_bookscore` (Dict[str, Any] chunk, \*str model="gpt-3.5-turbo", int batch\_size=10, bool use\_v2=True)
 

*Run Bookscore metric for long-form summarization.*
- str `chunk_bookscore` (str book\_text, str book\_title='book', int chunk\_size=2048)
 

*Chunk a book into Bookscore segments.*

**13.12 /home/runner/work/dsci-capstone/dsci-capstone/src/components/README.md File Reference****13.13 /home/runner/work/dsci-capstone/dsci-capstone/src/connectors/README.md File Reference****13.14 /home/runner/work/dsci-capstone/dsci-capstone/src/core/README.md File Reference****13.15 /home/runner/work/dsci-capstone/dsci-capstone/src/README.md File Reference****13.16 /home/runner/work/dsci-capstone/dsci-capstone/tests/examples-db/README.md File Reference****13.17 /home/runner/work/dsci-capstone/dsci-capstone/tests/README.md File Reference****13.18 /home/runner/work/dsci-capstone/dsci-capstone/src/components/relation\_extraction.py File Reference****Classes**

- class [RelationExtractor](#)

**Namespaces**

- namespace [src](#)
- namespace [src.components](#)
- namespace [src.components.relation\\_extraction](#)

**13.19 /home/runner/work/dsci-capstone/dsci-capstone/src/connectors/base.py File Reference****Classes**

- class [Connector](#)  
*Abstract base class for external connectors.*
- class [DatabaseConnector](#)  
*Abstract base class for database engine connectors.*

## Namespaces

- namespace `src`
- namespace `src.connectors`
- namespace `src.connectors.base`

## 13.20 /home/runner/work/dsci-capstone/dsci-capstone/src/connectors/document.py File Reference

### Classes

- class `DocumentConnector`  
*Connector for MongoDB (document database)*

### Namespaces

- namespace `src`
- namespace `src.connectors`
- namespace `src.connectors.document`

### Functions

- `MongoHandle mongo_handle (str host, str alias)`  
*Establish a temporary connection to MongoDB.*
- `DataFrame _flatten_recursive (DataFrame df)`  
*Explode all list columns and flatten dict columns until only scalars remain.*
- `str _sanitize_json (str text)`  
*Remove comments and other non-JSON content from a MongoDB query string.*
- `Dict[str, Any] _sanitize_document (Dict[str, Any] doc, Dict[str, Set[Type[Any]]] type_registry)`  
*Normalize document fields to consistent types for DataFrame construction.*
- `DataFrame _docs_to_df (List[Dict[str, Any]] docs, bool merge_unspecified=True)`  
*Convert raw MongoDB documents to a Pandas DataFrame.*
- `str _find_compatible_nested_key (Type[Any] value_type, Dict[str, Set[Type[Any]]] nested_schema, bool merge_unspecified)`  
*Find a nested column compatible with the given primitive type.*

### Variables

- `MongoHandle = Generator["Database[Any]", None, None]`

## 13.21 /home/runner/work/dsci-capstone/dsci-capstone/src/connectors/graph.py File Reference

### Classes

- class `GraphConnector`  
*Connector for Neo4j (graph database).*

## Namespaces

- namespace [src](#)
- namespace [src.connectors](#)
- namespace [src.connectors.graph](#)

## Functions

- DataFrame [\\_filter\\_to\\_db](#) (DataFrame df, str database\_name)  
*Filter a DataFrame by database context.*
- DataFrame [\\_tuples\\_to\\_df](#) (List[Tuple[Any,...]] tuples, List[str] meta)  
*Convert Neo4j query results (nodes and relationships) into a Pandas DataFrame.*
- DataFrame [\\_normalize\\_elements](#) (DataFrame df)  
*Convert Neo4j query results (nodes and relationships) into a Pandas DataFrame.*

## 13.22 /home/runner/work/dsci-capstone/dsci-capstone/src/connectors/llm.py File Reference

### Classes

- class [LLMConnector](#)  
*Connector for prompting and returning LLM output (raw text/JSON) via LangChain.*

## Namespaces

- namespace [src](#)
- namespace [src.connectors](#)
- namespace [src.connectors.llm](#)

## 13.23 /home/runner/work/dsci-capstone/dsci-capstone/src/connectors/relational.py File Reference

### Classes

- class [RelationalConnector](#)  
*Connector for relational databases (MySQL, PostgreSQL).*
- class [mysqlConnector](#)  
*A relational database connector configured for MySQL.*
- class [postgresConnector](#)  
*A relational database connector configured for PostgreSQL.*

## Namespaces

- namespace [src](#)
- namespace [src.connectors](#)
- namespace [src.connectors.relational](#)

## 13.24 /home/runner/work/dsci-capstone/dsci-capstone/src/core/boss.py File Reference

### Namespaces

- namespace `src`
- namespace `src.core`
- namespace `src.core.boss`

*Boss microservice for orchestrating distributed task processing.*

### Functions

- Dict[str, str] `load_worker_config` (List[str] `task_types`)
 

*Load worker service URLs from environment variables.*
- None `clear_task_data` (`MongoHandle mongo_db`, str `collection_name`, str `chunk_id`, str `task_name`)
 

*Clear any existing task data before assigning new task to worker.*
- bool `assign_task_to_worker` (str `worker_url`, str `database_name`, str `collection_name`, str `chunk_id`)
 

*Assign a task to a worker microservice.*
- Flask `create_app` (`DocumentConnector docs_db`, str `database_name`, str `collection_name`, Dict[str, str] `worker_urls`)
 

*Create and configure Flask application for boss service.*
- None `create_boss_thread` (str `DB_NAME`, int `BOSS_PORT`, str `COLLECTION`)
 

*Helpers to interact with the Flask boss thread.*
- requests.models.Response `post_story_status` (int `boss_port`, int `story_id`, str `task`, str `status`)
 

*Send a chunk-level update to the boss Flask app.*
- requests.models.Response `post_chunk_status` (int `boss_port`, str `chunk_id`, int `story_id`, str `task`, str `status`)
 

*Process all chunks in MongoDB matching the provided story ID.*
- requests.models.Response `post_process_full_story` (int `boss_port`, int `story_id`, str `task_type`)
 

*Process all chunks in MongoDB matching the provided story ID.*

### Variables

- `MongoHandle` = Generator["Database[Any]", None, None]

## 13.25 /home/runner/work/dsci-capstone/dsci-capstone/src/core/context.py File Reference

### Classes

- class `Session`

*Stores active database connections and configuration settings.*

### Namespaces

- namespace `src`
- namespace `src.core`
- namespace `src.core.context`

## Functions

- `Session get_session` (\*Any args, \*\*Any kwargs)  
*Lazily creates a session on first call, otherwise returns the existing session.*
- Any `__getattr__` (str name)  
*Lazy attribute resolution for module-level imports.*

## Variables

- `Session session`  
*The global instance of the singleton Session class.*

## 13.26 /home/runner/work/dsci-capstone/dsci-capstone/src/core/stages.py File Reference

### Namespaces

- namespace `src`
- namespace `src.core`
- namespace `src.core.stages`

### Functions

- str `task_01_convert_epub` (str epub\_path, Optional[`EPUBToTEI`] converter=None)  
*Will revisit later - Book classes need refactoring #####.*
- `task_02_parse_chapters` (tei\_path, book\_chapters, book\_id, story\_id, start\_str, end\_str)
- `task_03_chunk_story` (story, max\_chunk\_length=1500)
- `task_10_random_chunk` (chunks)
- `task_10_sample_chunks` (chunks, n\_sample)
- `task_11_send_chunk` (c, collection\_name, book\_title)
- `task_12_relation_extraction_rebel` (text, max\_tokens=1024, parse\_tuples=True)
- `task_13_concatenate_triples` (extracted)
- `task_14_relation_extraction_llm` (triples\_string, text)
- str `task_15_sanitize_triples_llm` (str llm\_output)
- `task_20_send_triples` (triples)
- `group_21_1_describe_graph` (top\_n=3)
- `group_21_2_send_statistics` ()
- `group_21_3_post_statistics` ()
- `task_22_verbalize_triples` (mode="triple")
- `task_30_summarize_llm` (triples\_string)  
*Prompt LLM to generate summary.*
- `task_31_send_summary` (summary, collection\_name, chunk\_id)
- `task_40_post_summary` (book\_id, book\_title, summary)  
*Send book info to Blazor.*
- `task_40_post_payload` (book\_id, book\_title, summary, gold\_summary, chunk, bookscore, questeval)  
*Send metrics to Blazor.*

## 13.27 /home/runner/work/dsci-capstone/dsci-capstone/src/core/worker.py File Reference

### Namespaces

- namespace `src`
- namespace `src.core`
- namespace `src.core.worker`

*Generic Flask worker microservice for distributed task processing.*

### Functions

- None `process_task` (`MongoHandle mongo_db, str collection_name, str chunk_id, str task_name, Dict[str, Any] chunk_doc, str boss_url, Callable[[Dict[str, Any]], Dict[str, Any]] task_handler, Any task_kwargs=None`)  
*Perform the assigned task in a background thread.*
- str `load_mongo_config` (`str database`)  
*Load MongoDB configuration from environment variables.*
- str `load_boss_config` ()  
*Load boss service callback URL from environment variables.*
- Tuple[Callable[[Dict[str, Any]], Dict[str, Any]], Dict[str, Any]] `get_task_info` (`str task_name`)  
*Dynamically import and return the appropriate task handler function.*
- `load_imports` (`func`)  
*Pre-warm the task by importing requirements.*
- None `mark_task_in_progress` (`MongoHandle mongo_db, str collection_name, str chunk_id, str task_name`)  
*Mark a task as in-progress in MongoDB before processing begins.*
- None `save_task_result` (`MongoHandle mongo_db, str collection_name, str chunk_id, str task_name, Dict[str, Any] result`)  
*Save completed task results to MongoDB.*
- None `notify_boss` (`str boss_url, str chunk_id, str task_name, str status`)  
*Send completion notification to boss service.*
- Flask `create_app` (`str task_name, str boss_url`)  
*Create and configure Flask application for task processing.*

### Variables

- `MongoHandle` = Generator["Database[Any]", None, None]
- `parser` = argparse.ArgumentParser(description="Flask worker microservice")
- `required`
- `True`
- `help`
- `args` = parser.parse\_args()
- str `task_queue` = Queue()  
*Background threading system for non-blocking task handling.*
- `daemon`
- str `boss_url` = `load_boss_config()`
- `PORT` = int(os.environ[f"{args.task.upper()}\_PORT"])
- Flask `app` = `create_app(args.task, boss_url)`
- `host`
- `port`
- `use_reloader`

## 13.28 /home/runner/work/dsci-capstone/dsci-capstone/src/main.py File Reference

### Namespaces

- namespace `src`
- namespace `src.main`

### Functions

- `pipeline_A` (epub\_path, book\_chapters, start\_str, end\_str, `book_id`, `story_id`)  
*Connects all components to convert an EPUB file to a book summary.*
- `pipeline_B` (collection\_name, `chunks`, `book_title`)  
*Extracts triples from a random chunk.*
- `pipeline_C` (`json_triples`)  
*Generates a LLM summary using Neo4j triples.*
- `pipeline_D` (collection\_name, `triples_string`, `chunk_id`)  
*Generate chunk summary.*
- None `pipeline_E` (str `summary`, str `book_title`, str `book_id`, str `chunk=""`, str `gold_summary=""`, float `bookscore=None`, float `questeval=None`)  
*Compute metrics and send available data to Blazor.*
- `full_pipeline` (collection\_name, epub\_path, book\_chapters, start\_str, end\_str, `book_id`, `story_id`, `book_title`)
- `old_main` (collection\_name)

### Variables

- `DB_NAME` = os.environ["DB\_NAME"]
- `BOSS_PORT` = int(os.environ["PYTHON\_PORT"])
- `COLLECTION` = os.environ["COLLECTION\_NAME"]
- bool `load_from_checkpoint` = False
- str `checkpoint_path` = "./datasets/checkpoint.pkl"
- `exist_ok`
- int `story_id` = 1
- int `book_id` = 2
- str `book_title` = "The Phoenix and the Carpet"
- `data` = pickle.load(f\_read)
- `triples` = `data`["triples"]
- `chunk` = `data`["chunk"]
- `chunks`
- `chunk_id` = `chunk.get_chunk_id()`
- `triples_string` = `pipeline_C(triples)`
- `summary` = `pipeline_D(COLLECTION, triples_string, chunk.get_chunk_id())`
- `response` = `post_process_full_story(BOSS_PORT, story_id, task_type)`

## 13.29 /home/runner/work/dsci-capstone/dsci-capstone/src/util.py File Reference

### Classes

- class [Log](#)  
*The Log class standardizes console output.*
- class [Log.Failure](#)  
*User-facing base class for custom error handling.*
- class [Log.BadAddressFailure](#)  
*Raised when a database connection string or address is invalid.*

### Namespaces

- namespace [src](#)
- namespace [src.util](#)

### Functions

- DataFrame [df\\_natural\\_sorted](#) (DataFrame df, List[str] ignored\_columns=[], List[str] sort\_columns=[])  
*Sort a DataFrame in natural order using only certain columns.*
- bool [check\\_values](#) (List[Any] results, List[Any] expected, bool verbose, str log\_source, bool raise\_error)  
*Safely compare two lists of values.*

## 13.30 /home/runner/work/dsci-capstone/dsci-capstone/tests/test\_db\_basic.py File Reference

### Namespaces

- namespace [tests](#)
- namespace [tests.test\\_db\\_basic](#)

### Functions

- None [test\\_db\\_relational\\_minimal](#) (RelationalConnector relational\_db)  
*Tests if the RelationalConnector has a valid connection string.*
- None [test\\_db\\_docs\\_minimal](#) (DocumentConnector docs\_db)  
*Tests if the DocumentConnector has a valid connection string.*
- None [test\\_db\\_graph\\_minimal](#) (GraphConnector graph\_db)  
*Tests if the GraphConnector has a valid connection string.*
- None [test\\_db\\_relational\\_comprehensive](#) (RelationalConnector relational\_db)  
*Tests if the GraphConnector is working as intended.*
- None [test\\_db\\_docs\\_comprehensive](#) (DocumentConnector docs\_db)  
*Tests if the GraphConnector is working as intended.*
- None [test\\_db\\_graph\\_comprehensive](#) (GraphConnector graph\_db)  
*Tests if the GraphConnector is working as intended.*

## 13.31 /home/runner/work/dsci-capstone/dsci-capstone/tests/test\_db\_files.py File Reference

### Namespaces

- namespace `tests`
- namespace `tests.test_db_files`

### Functions

- Generator[None, None, None] `load_examples_relational` (`RelationalConnector` relational\_db)  
*Fixture to create relational tables using engine-specific syntax.*
- None `test_sql_example_1` (`RelationalConnector` relational\_db, Generator[None, None, None] `load_examples_relational`)  
*Run queries contained within test files.*
- None `test_sql_example_2` (`RelationalConnector` relational\_db, Generator[None, None, None] `load_examples_relational`)  
*Run queries contained within test files.*
- None `test_mongo_example_1` (`DocumentConnector` docs\_db)  
*Run queries contained within test files.*
- None `test_mongo_example_2` (`DocumentConnector` docs\_db)  
*Run queries contained within test files.*
- None `test_mongo_example_3` (`DocumentConnector` docs\_db)  
*Run queries contained within test files.*
- None `test_cypher_example_1` (`GraphConnector` graph\_db)  
*Run queries contained within test files.*
- None `test_cypher_example_2` (`GraphConnector` graph\_db)  
*Test social network graph with relationships and mixed query patterns.*
- None `test_cypher_example_3` (`GraphConnector` graph\_db)  
*Test scene and dialogue graphs with proper isolation.*
- None `test_cypher_example_4` (`GraphConnector` graph\_db)  
*Test event graph with property mutations and multi-hop traversal.*
- None `_exec_query_file` (`DatabaseConnector` db\_fixture, str filename, List[str] valid\_files)  
*Run queries from a local file through the database.*

## 13.32 /home/runner/work/dsci-capstone/dsci-capstone/tests/test\_kg\_triples.py File Reference

### Namespaces

- namespace `tests`
- namespace `tests.test_kg_triples`

## Functions

- None `test_knowledge_graph_triples` (`KnowledgeGraph main_graph`)
 

*Test KnowledgeGraph triple operations using add\_triple and get\_all\_triples.*
- Generator[`KnowledgeGraph`, `None`, `None`] `nature_scene_graph` (`KnowledgeGraph main_graph`)
 

*Create a scene graph with multiple location-based communities for testing.*
- None `test_get_subgraph_by_nodes` (`KnowledgeGraph nature_scene_graph`)
 

*Test filtering triples by specific node IDs.*
- None `test_get_neighborhood` (`KnowledgeGraph nature_scene_graph`)
 

*Test k-hop neighborhood expansion around a central node.*
- None `test_get_random_walk_sample` (`KnowledgeGraph nature_scene_graph`)
 

*Test random walk sampling starting from specified nodes.*
- None `test_get_neighborhood_comprehensive` (`KnowledgeGraph nature_scene_graph`)
 

*Comprehensive test for k-hop neighborhood expansion.*
- None `test_get_random_walk_sample_comprehensive` (`KnowledgeGraph nature_scene_graph`)
 

*Comprehensive test for random walk sampling.*
- None `test_detect_community_clusters_minimal` (`KnowledgeGraph nature_scene_graph`)
 

*Test basic community detection functionality.*
- None `test_detect_community_clusters_comprehensive` (`KnowledgeGraph nature_scene_graph`)
 

*Comprehensive test for community detection with various parameters.*
- None `test_ranked_degree` (`KnowledgeGraph nature_scene_graph`)
 

*Test filtering triples by ranked node degree.*
- None `test_ranked_degree_ties` (`KnowledgeGraph main_graph`)
 

*Test that degree ranking correctly handles ties with minimal data.*

## 13.33 /home/runner/work/dsci-capstone/dsci-capstone/tests/test\_pipeline.py File Reference

### Namespaces

- namespace `tests`
- namespace `tests.test_pipeline`

### Functions

- `book_data` (`request`)
 

*Fixtures.*
- `book_1_data` ()
 

*Example data for Book 1: Five Children and It.*
- `book_2_data` ()
 

*Example data for Book 2: The Phoenix and the Carpet - realistic pipeline data.*
- `test_job_01_convert_epub` (`book_data`)
 

*Tests.*
- `test_job_02_parse_chapters` (`book_data`)
 

*Test TEI -> Story parsing for multiple books.*
- `test_job_03_chunk_story` (`book_data`)
 

*Test Story -> chunks splitting for multiple books.*
- `test_job_10_sample_chunks` (`book_data`)
 

*Test sampling multiple chunks from a list.*

- [test\\_job\\_10\\_random\\_chunk \(book\\_data\)](#)  
*Test selecting a single random chunk.*
- [test\\_job\\_11\\_send\\_chunk \(docs\\_db, book\\_data\)](#)  
*Test inserting chunk into MongoDB collection.*
- [test\\_job\\_13\\_concatenate\\_triples \(book\\_data\)](#)  
*Test converting extracted triples to newline-delimited string.*
- [test\\_job\\_15\\_sanitize\\_triples\\_llm \(book\\_data\)](#)  
*Test parsing LLM output JSON into triples list.*
- [test\\_job\\_20\\_send\\_triples \(main\\_graph, book\\_data\)](#)  
*Test inserting triples into knowledge graph.*
- [test\\_job\\_21\\_describe\\_graph \(main\\_graph, book\\_data\)](#)  
*Test generating edge count summary of knowledge graph.*
- [test\\_job\\_22\\_verbalize\\_triples \(main\\_graph, book\\_data\)](#)  
*Test converting high-degree triples to string format.*
- [test\\_job\\_31\\_send\\_summary \(docs\\_db, book\\_data\)](#)  
*Test updating chunk with summary in MongoDB.*
- [test\\_pipeline\\_A\\_minimal \(book\\_data\)](#)  
*Minimal aggregate test.*
- [test\\_pipeline\\_A\\_from\\_csv \(\)](#)  
*Read example CSV and run pipeline\_A for each row.*
- [test\\_pipeline\\_C\\_minimal \(main\\_graph, book\\_data\)](#)  
*Test running pipeline\_C with smoke test data.*
- [test\\_pipeline\\_E\\_minimal\\_summary\\_only \(book\\_data\)](#)  
*Test running pipeline\_E with summary-only mode.*
- [test\\_pipeline\\_E\\_minimal\\_full\\_payload \(book\\_data\)](#)  
*Test running pipeline\_E with full payload including metrics.*

- 13.34 /home/runner/work/dsci-capstone/dsci-capstone/web-app/Blazor ←  
App/Components/\_Imports.razor File Reference
- 13.35 /home/runner/work/dsci-capstone/dsci-capstone/web-app/Blazor ←  
App/Components/App.razor File Reference
- 13.36 /home/runner/work/dsci-capstone/dsci-capstone/web-app/Blazor ←  
App/Components/Layout/MainLayout.razor File Reference
- 13.37 /home/runner/work/dsci-capstone/dsci-capstone/web-app/Blazor ←  
App/Components/Layout/NavMenu.razor File Reference
- 13.38 /home/runner/work/dsci-capstone/dsci-capstone/web-app/Blazor ←  
App/Components/Pages/Error.razor File Reference
- 13.39 /home/runner/work/dsci-capstone/dsci-capstone/web-app/Blazor ←  
App/Components/Pages/Graph.razor File Reference
- 13.40 /home/runner/work/dsci-capstone/dsci-capstone/web-app/Blazor ←  
App/Components/Pages/Home.razor File Reference
- 13.41 /home/runner/work/dsci-capstone/dsci-capstone/web-app/Blazor ←  
App/Components/Pages/Metrics.razor File Reference
- 13.42 /home/runner/work/dsci-capstone/dsci-capstone/web-app/Blazor ←  
App/Components/Routes.razor File Reference
- 13.43 /home/runner/work/dsci-capstone/dsci-capstone/web-app/Blazor ←  
App/Controllers/MetricsController.cs File Reference

#### Classes

- class [MetricsController](#)

#### Namespaces

- namespace [BlazorApp](#)
- namespace [BlazorApp.Controllers](#)

## 13.44 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Hubs/MetricsHub.cs File Reference

### Classes

- class [MetricsHub](#)

### Namespaces

- namespace [BlazorApp](#)
- namespace [BlazorApp.Hubs](#)

## 13.45 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/PRF1Metric.cs File Reference

### Classes

- class [PRF1Metric](#)

### Namespaces

- namespace [BlazorApp](#)
- namespace [BlazorApp.Models](#)

## 13.46 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/QAItem.cs File Reference

### Classes

- class [QAItem](#)

### Namespaces

- namespace [BlazorApp](#)
- namespace [BlazorApp.Models](#)

## 13.47 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/QAMetric.cs File Reference

### Classes

- class [QAMetric](#)

## Namespaces

- namespace [BlazorApp](#)
- namespace [BlazorApp.Models](#)

# 13.48 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/ScalarMetric.cs File Reference

## Classes

- class [ScalarMetric](#)

## Namespaces

- namespace [BlazorApp](#)
- namespace [BlazorApp.Models](#)

# 13.49 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/SummaryData.cs File Reference

## Classes

- class [SummaryData](#)

## Namespaces

- namespace [BlazorApp](#)
- namespace [BlazorApp.Models](#)

# 13.50 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/SummaryMetrics.cs File Reference

## Classes

- class [SummaryMetrics](#)

## Namespaces

- namespace [BlazorApp](#)
- namespace [BlazorApp.Models](#)

# Index

/home/runner/work/dsci-capstone/dsci-capstone/conftest.py, 211  
201 /home/runner/work/dsci-capstone/dsci-capstone/src/util.py,  
/home/runner/work/dsci-capstone/dsci-capstone/src/README.md, 212  
205 /home/runner/work/dsci-capstone/dsci-capstone/tests/README.md,  
/home/runner/work/dsci-capstone/dsci-capstone/src/\_\_init\_\_.py, 205  
202 /home/runner/work/dsci-capstone/dsci-capstone/tests/\_\_init\_\_.py,  
/home/runner/work/dsci-capstone/dsci-capstone/src/charts.py, 202  
201 /home/runner/work/dsci-capstone/dsci-capstone/tests/examples-  
/home/runner/work/dsci-capstone/dsci-capstone/src/components/README.md, 205  
205 /home/runner/work/dsci-capstone/dsci-capstone/tests/test\_db\_basic.py,  
/home/runner/work/dsci-capstone/dsci-capstone/src/components/\_\_init\_\_.py,  
202 /home/runner/work/dsci-capstone/dsci-capstone/tests/test\_db\_files.py,  
/home/runner/work/dsci-capstone/dsci-capstone/src/components/bookconversion.py,  
203 /home/runner/work/dsci-capstone/dsci-capstone/tests/test\_kg\_triples.py,  
/home/runner/work/dsci-capstone/dsci-capstone/src/components/corpustools.py,  
203 /home/runner/work/dsci-capstone/dsci-capstone/tests/test\_pipeline.py,  
/home/runner/work/dsci-capstone/dsci-capstone/src/components/factstorage.py,  
204 /home/runner/work/dsci-capstone/dsci-capstone/web-  
/home/runner/work/dsci-capstone/dsci-capstone/src/components/metamodels/BlazorApp/Components/App.razor, 216  
204 /home/runner/work/dsci-capstone/dsci-capstone/web-  
/home/runner/work/dsci-capstone/dsci-capstone/src/components/relationships/BlazorApp/Components/Layout/MainLayout.razor,  
205 216  
/home/runner/work/dsci-capstone/dsci-capstone/src/connections/BlazorApp/Components/Layout/NavMenu.razor,  
205 /home/runner/work/dsci-capstone/dsci-capstone/web-  
/home/runner/work/dsci-capstone/dsci-capstone/src/connectors/\_\_init\_\_.py, 216  
202 /home/runner/work/dsci-capstone/dsci-capstone/web-  
/home/runner/work/dsci-capstone/dsci-capstone/src/connectors/baseapp/BlazorApp/Components/Pages/Error.razor,  
205 216  
/home/runner/work/dsci-capstone/dsci-capstone/src/connectors/documents/BlazorApp/Components/Pages/Graph.razor,  
206 /home/runner/work/dsci-capstone/dsci-capstone/web-  
/home/runner/work/dsci-capstone/dsci-capstone/src/connectors/graphs/BlazorApp/Components/Pages/Home.razor,  
206 216  
/home/runner/work/dsci-capstone/dsci-capstone/src/connectors/statistics/BlazorApp/Components/Pages/Metrics.razor,  
207 216  
/home/runner/work/dsci-capstone/dsci-capstone/src/core/BlazorApp/Components/Routes.razor, 216  
205 /home/runner/work/dsci-capstone/dsci-capstone/web-  
/home/runner/work/dsci-capstone/dsci-capstone/src/core/\_\_init\_\_.py, 216  
202 /home/runner/work/dsci-capstone/dsci-capstone/web-  
/home/runner/work/dsci-capstone/dsci-capstone/src/core/blocks/BlazorApp/Components/\_Imports.razor,  
208 /home/runner/work/dsci-capstone/dsci-capstone/web-  
/home/runner/work/dsci-capstone/dsci-capstone/src/core/context.py, 216  
208 /home/runner/work/dsci-capstone/dsci-capstone/web-  
/home/runner/work/dsci-capstone/dsci-capstone/src/core/stages.py, 216  
209 /home/runner/work/dsci-capstone/dsci-capstone/web-  
/home/runner/work/dsci-capstone/dsci-capstone/src/core/webhooks/BlazorApp/Hubs/MetricsHub.cs, 217  
210 /home/runner/work/dsci-capstone/dsci-capstone/src/main.py, 216

app/BlazorApp/Models/PRF1Metric.cs, 217  
 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/QAItem.cs, 217  
 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/QAMetric.cs, 217  
 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/ScalarMetric.cs, 218  
 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/SummaryData.cs, 218  
 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/SummaryMetrics.cs, 218  
`__getattr__`  
 src.core.context, 40  
`__init__`  
 Book, 71  
 BookStream, 74  
 Chunk, 75  
 DatabaseConnector, 83  
 DocumentConnector, 93  
 EPUBToTEI, 100  
 GraphConnector, 107  
 KnowledgeGraph, 119  
 LLMConnector, 133  
 Log.BadAddressFailure, 71  
 Log.Failure, 103  
 Metrics, 155  
 mysqlConnector, 167  
 ParagraphStreamTEI, 169  
 postgresConnector, 176  
 RelationalConnector, 182  
 RelationExtractor, 188  
 Session, 191  
 Story, 193  
`__new__`  
 Session, 191  
`__repr__`  
 Chunk, 76  
`__str__`  
 Log.Failure, 104  
`_auth_suffix`  
 DocumentConnector, 99  
`_created`  
 Session, 191  
`_docs_to_df`  
 src.connectors.document, 31  
`_exec_query_file`  
 tests.test\_db\_files, 58  
`_execute_retag_db`  
 GraphConnector, 107  
`_fetch_latest`  
 GraphConnector, 107  
`_filter_to_db`  
 src.connectors.graph, 34  
`_find_compatible_nested_key`  
 src.connectors.document, 31  
`_first_insert`  
 KnowledgeGraph, 130  
`_flatten_recursive`  
 src.connectors.document, 31  
`_graph_name`  
 GraphConnector, 117  
`_hubContext`  
 MetricsController, 162  
`_instance`  
 Session, 191  
`_is_single_query`  
 DatabaseConnector, 84  
`_logger`  
 MetricsController, 162  
 MetricsHub, 164  
`_make_single`  
 Story, 193  
`_merge_chunks`  
 Story, 193  
`_normalize_elements`  
 src.connectors.graph, 34  
`_parsable_to_df`  
 DatabaseConnector, 84  
 DocumentConnector, 93  
 GraphConnector, 109  
 RelationalConnector, 182  
`_prune_bad_tags`  
 EPUBToTEI, 100  
`_returns_data`  
 DatabaseConnector, 84  
 DocumentConnector, 94  
 GraphConnector, 109  
 RelationalConnector, 182  
`_sanitize_document`  
 src.connectors.document, 32  
`_sanitize_ids`  
 EPUBToTEI, 100  
`_sanitize_json`  
 src.connectors.document, 32  
`_session`  
 Session, 192  
`_split_combined`  
 DatabaseConnector, 85  
 DocumentConnector, 94  
 GraphConnector, 110  
 RelationalConnector, 183  
`_timing_results`  
 Log, 145  
`_tuples_to_df`  
 src.connectors.graph, 35  
  
 Accuracy  
 QAItem, 177  
`add_triple`  
 KnowledgeGraph, 120  
`add_triples_json`  
 KnowledgeGraph, 120  
`allowed_chapters`  
 ParagraphStreamTEI, 170  
`app`  
 src.core.worker, 49

args  
    src.core.worker, 49  
assign\_task\_to\_worker  
    src.core.boss, 36  
AverageAccuracy  
    QAMetric, 178  
  
bad\_addr  
    Log, 145  
bad\_path  
    Log, 145  
bad\_val  
    Log, 145  
BlazorApp, 21  
BlazorApp.Controllers, 21  
BlazorApp.Hubs, 21  
BlazorApp.Models, 21  
Book, 71  
    \_\_init\_\_, 71  
    stream\_chapters, 71  
book  
    BookStream, 74  
book\_1\_data  
    tests.test\_pipeline, 65  
book\_2\_data  
    tests.test\_pipeline, 65  
book\_data  
    tests.test\_pipeline, 65  
book\_id  
    Chunk, 77  
    ParagraphStreamTEI, 170  
    src.main, 53  
book\_title  
    src.main, 53  
BookFactory, 72  
    create\_book, 72  
BookID  
    SummaryData, 196  
BookStream, 73  
    \_\_init\_\_, 74  
    book, 74  
    stream\_segments, 74  
BookTitle  
    SummaryData, 196  
BOSS\_PORT  
    src.main, 53  
boss\_url  
    src.core.worker, 49  
BRIGHT  
    Log, 145  
  
change\_database  
    DatabaseConnector, 85  
    DocumentConnector, 94  
    GraphConnector, 110  
    RelationalConnector, 183  
chapter\_number  
    Chunk, 77  
chapter\_percent

    Chunk, 77  
char\_count  
    Chunk, 76  
chart  
    Log, 140  
CHART\_COLOR  
    Log, 145  
chart\_message  
    Log, 140  
check\_connection  
    Connector, 79  
    DocumentConnector, 95  
    GraphConnector, 110  
    LLMConnector, 133  
    RelationalConnector, 184  
check\_values  
    src.util, 55  
checkpoint\_path  
    src.main, 53  
Chunk, 74  
    \_\_init\_\_, 75  
    \_\_repr\_\_, 76  
    book\_id, 77  
    chapter\_number, 77  
    chapter\_percent, 77  
    char\_count, 76  
    get\_chunk\_id, 76  
    line\_end, 77  
    line\_start, 77  
    story\_id, 77  
    story\_percent, 77  
    text, 77  
    to\_mongo\_dict, 77  
chunk  
    src.main, 54  
chunk\_bookscore  
    src.components.metrics, 27  
chunk\_id  
    src.main, 54  
chunks  
    ParagraphStreamTEI, 170  
    src.main, 54  
clean\_tei  
    EPUBToTEI, 101  
clean\_tei\_content  
    EPUBToTEI, 101  
clear\_task\_data  
    src.core.boss, 37  
clear\_timing\_data  
    Log, 140  
COLLECTION  
    src.main, 54  
Components Module Overview, 1  
compute\_basic\_metrics  
    Metrics, 156  
configure  
    DatabaseConnector, 86  
    LLMConnector, 133

conftest, 22  
 docs\_db, 22  
 graph\_db, 22  
 main\_graph, 22  
 pytest\_addoption, 22  
 relational\_db, 22  
 session, 23  
 conn\_abc  
 Log, 145  
 connection\_string  
 DatabaseConnector, 89  
 DocumentConnector, 99  
 GraphConnector, 117  
 RelationalConnector, 187  
 Connector, 78  
 check\_connection, 79  
 execute\_file, 80  
 execute\_query, 80  
 test\_operations, 80  
 Connectors Module Overview, 3  
 convert\_to\_teii  
 EPUBToTEI, 101  
 Core Module Overview, 5  
 create\_app  
 src.core.boss, 37  
 src.core.worker, 46  
 create\_book  
 BookFactory, 72  
 create\_boss\_thread  
 src.core.boss, 38  
 create\_database  
 DatabaseConnector, 86  
 DocumentConnector, 95  
 GraphConnector, 111  
 RelationalConnector, 184  
 create\_db  
 Log, 145  
 create\_summary\_payload  
 Metrics, 156  
 CYAN  
 Log, 145  
 daemon  
 src.core.worker, 49  
 data  
 src.main, 54  
 database  
 KnowledgeGraph, 131  
 Database Example Datasets, 9  
 database\_exists  
 DatabaseConnector, 86  
 DocumentConnector, 96  
 GraphConnector, 111  
 RelationalConnector, 184  
 database\_name  
 DocumentConnector, 99  
 GraphConnector, 117  
 RelationalConnector, 187  
 DatabaseConnector, 81  
 \_\_init\_\_, 83  
 \_is\_single\_query, 84  
 \_parsable\_to\_df, 84  
 \_returns\_data, 84  
 \_split\_combined, 85  
 change\_database, 85  
 configure, 86  
 connection\_string, 89  
 create\_database, 86  
 database\_exists, 86  
 db\_engine, 89  
 db\_type, 89  
 drop\_database, 87  
 execute\_combined, 87  
 execute\_file, 87  
 execute\_query, 88  
 get\_dataframe, 88  
 host, 89  
 password, 90  
 port, 90  
 temp\_database, 89  
 username, 90  
 verbose, 90  
 db\_conn\_abc  
 Log, 146  
 db\_engine  
 DatabaseConnector, 89  
 db\_exists  
 Log, 146  
 DB\_NAME  
 src.main, 54  
 db\_type  
 DatabaseConnector, 89  
 RelationalConnector, 187  
 delete\_dummy  
 DocumentConnector, 96  
 GraphConnector, 112  
 detect\_community\_clusters  
 KnowledgeGraph, 120  
 df  
 src.components.corpus, 26  
 df\_booksum  
 src.components.corpus, 26  
 df\_natural\_sorted  
 src.util, 56  
 df\_nqa  
 src.components.corpus, 26  
 doc\_db  
 Log, 146  
 docs\_db  
 conftest, 22  
 Session, 192  
 DocumentConnector, 90  
 \_\_init\_\_, 93  
 \_auth\_suffix, 99  
 \_parsable\_to\_df, 93  
 \_returns\_data, 94  
 \_split\_combined, 94

change\_database, 94  
check\_connection, 95  
connection\_string, 99  
create\_database, 95  
database\_exists, 96  
database\_name, 99  
delete\_dummy, 96  
drop\_database, 96  
execute\_query, 97  
get\_dataframe, 97  
get\_unmanaged\_handle, 98  
test\_operations, 98  
verbose, 99  
drop\_database  
    DatabaseConnector, 87  
    DocumentConnector, 96  
    GraphConnector, 112  
    RelationalConnector, 185  
drop\_db  
    Log, 146  
drop\_gr  
    Log, 146  
drop\_graph  
    GraphConnector, 112  
dump\_timing\_csv  
    Log, 140  
elapsed\_time  
    Log, 141  
encoding  
    EPUBToTEI, 101  
    ParagraphStreamTEI, 170  
end\_inclusive  
    ParagraphStreamTEI, 171  
epub\_path  
    EPUBToTEI, 101  
    EPUBToTEI, 99  
        \_\_init\_\_, 100  
        \_prune\_bad\_tags, 100  
        \_sanitize\_ids, 100  
        clean\_tei, 101  
        clean\_tei\_content, 101  
        convert\_to\_teい, 101  
        encoding, 101  
        epub\_path, 101  
        pandoc\_xml\_path, 101  
        raw\_teи\_content, 102  
        tei\_path, 102  
        xml\_namespace, 102  
execute\_combined  
    DatabaseConnector, 87  
execute\_file  
    Connector, 80  
    DatabaseConnector, 87  
    LLMConnector, 134  
execute\_full\_query  
    LLMConnector, 134  
execute\_query  
    Connector, 80  
DatabaseConnector, 88  
DocumentConnector, 97  
GraphConnector, 113  
LLMConnector, 134  
RelationalConnector, 185  
exist\_ok  
    src.main, 54  
extract  
    RelationExtractor, 188  
F1Score  
    PRF1Metric, 177  
fail  
    Log, 141  
fail\_legacy  
    Log, 142  
FAILURE\_COLOR  
    Log, 146  
find\_element\_names  
    KnowledgeGraph, 121  
format\_call\_chain  
    Log, 142  
from\_env  
    RelationalConnector, 186  
FULL\_DF  
    Log, 146  
full\_pipeline  
    src.main, 52  
fuzzy\_merge\_titles  
    src.components.corpus, 25  
generate\_default\_metrics  
    Metrics, 156  
generate\_example\_metrics  
    Metrics, 157  
GeneratedAnswer  
    QAItem, 177  
get\_all\_triples  
    KnowledgeGraph, 122  
get\_by\_ranked\_degree  
    KnowledgeGraph, 122  
get\_chunk\_id  
    Chunk, 76  
get\_community\_subgraph  
    KnowledgeGraph, 123  
get\_dataframe  
    DatabaseConnector, 88  
    DocumentConnector, 97  
    GraphConnector, 113  
    RelationalConnector, 186  
get\_degree\_range  
    KnowledgeGraph, 123  
get\_df  
    Log, 146  
get\_edge\_counts  
    KnowledgeGraph, 124  
get\_merged\_timing  
    Log, 142  
get\_neighborhood

KnowledgeGraph, 124  
 get\_node\_context  
     KnowledgeGraph, 125  
 get\_random\_walk\_sample  
     KnowledgeGraph, 126  
 get\_relation\_summary  
     KnowledgeGraph, 126  
 get\_session  
     src.core.context, 40  
 get\_subgraph\_by\_nodes  
     KnowledgeGraph, 126  
 get\_summary\_stats  
     KnowledgeGraph, 127  
 get\_task\_info  
     src.core.worker, 46  
 get\_timing\_summary  
     Log, 142  
 get\_triple\_properties  
     KnowledgeGraph, 127  
 get\_unique  
     GraphConnector, 114  
     Log, 146  
 get\_unmanaged\_handle  
     DocumentConnector, 98  
 GetAll  
     MetricsController, 161  
 GetDefault  
     SummaryMetrics, 197  
 GetIndex  
     MetricsController, 161  
 GoldAnswer  
     QAItem, 177  
 GoldSummaryText  
     SummaryData, 196  
 good\_val  
     Log, 147  
 gr\_db  
     Log, 147  
 gr\_rag  
     Log, 147  
 graph\_db  
     conftest, 22  
     Session, 192  
 graph\_exists  
     GraphConnector, 114  
 graph\_name  
     KnowledgeGraph, 131  
 GraphConnector, 104  
     \_\_init\_\_, 107  
     \_execute\_retag\_db, 107  
     \_fetch\_latest, 107  
     \_graph\_name, 117  
     \_parsable\_to\_df, 109  
     \_returns\_data, 109  
     \_split\_combined, 110  
     change\_database, 110  
     check\_connection, 110  
     connection\_string, 117  
     create\_database, 111  
     database\_exists, 111  
     database\_name, 117  
     delete\_dummy, 112  
     drop\_database, 112  
     drop\_graph, 112  
     execute\_query, 113  
     get\_dataframe, 113  
     get\_unique, 114  
     graph\_exists, 114  
     IS\_DUMMY\_, 116  
     NOT\_DUMMY\_, 116  
     SAME\_DB\_KG\_, 116  
     temp\_graph, 116  
     test\_operations, 117  
     verbose, 117  
 GRAY  
     Log, 147  
 GREEN  
     Log, 147  
 group\_21\_1\_describe\_graph  
     src.core.stages, 42  
 group\_21\_2\_send\_statistics  
     src.core.stages, 42  
 group\_21\_3\_post\_statistics  
     src.core.stages, 42  
 help  
     src.core.worker, 50  
 HOST  
     Metrics, 160  
 host  
     DatabaseConnector, 89  
     src.core.worker, 50  
 index  
     src.components.corpus, 26  
 IS\_DUMMY\_  
     GraphConnector, 116  
 IsCorrect  
     QAItem, 178  
 kg  
     Log, 147  
 KnowledgeGraph, 118  
     \_\_init\_\_, 119  
     \_first\_insert, 130  
     add\_triple, 120  
     add\_triples\_json, 120  
     database, 131  
     detect\_community\_clusters, 120  
     find\_element\_names, 121  
     get\_all\_triples, 122  
     get\_by\_ranked\_degree, 122  
     get\_community\_subgraph, 123  
     get\_degree\_range, 123  
     get\_edge\_counts, 124  
     get\_neighborhood, 124  
     get\_node\_context, 125

get\_random\_walk\_sample, 126  
get\_relation\_summary, 126  
get\_subgraph\_by\_nodes, 126  
get\_summary\_stats, 127  
get\_triple\_properties, 127  
graph\_name, 131  
print\_nodes, 128  
print\_triples, 128  
to\_contextualized\_string, 128  
to\_narrative, 129  
to\_triples\_string, 129  
triples\_to\_names, 130  
verbose, 131

line\_end  
    Chunk, 77

line\_start  
    Chunk, 77

lines  
    ParagraphStreamTEI, 171

llm  
    LLMConnector, 136

LLMConnector, 131  
    \_\_init\_\_, 133  
    check\_connection, 133  
    configure, 133  
    execute\_file, 134  
    execute\_full\_query, 134  
    execute\_query, 134  
    llm, 136  
    model\_name, 136  
    normalize\_triples, 135  
    system\_prompt, 136  
    test\_operations, 135

load\_booksum  
    src.components.corpus, 25

load\_boss\_config  
    src.core.worker, 46

load\_examples\_relational  
    tests.test\_db\_files, 59

load\_from\_checkpoint  
    src.main, 54

load\_imports  
    src.core.worker, 47

load\_mongo\_config  
    src.core.worker, 47

load\_narrativeqa  
    src.components.corpus, 25

load\_worker\_config  
    src.core.boss, 38

Log, 136  
    \_timing\_results, 145  
    bad\_addr, 145  
    bad\_path, 145  
    bad\_val, 145  
    BRIGHT, 145  
    chart, 140  
    CHART\_COLOR, 145  
    chart\_message, 140

clear\_timing\_data, 140  
conn\_abc, 145  
create\_db, 145  
CYAN, 145  
db\_conn\_abc, 146  
db\_exists, 146  
doc\_db, 146  
drop\_db, 146  
drop\_gr, 146  
dump\_timing\_csv, 140  
elapsed\_time, 141  
fail, 141  
fail\_legacy, 142  
FAILURE\_COLOR, 146  
format\_call\_chain, 142  
FULL\_DF, 146  
get\_df, 146  
get\_merged\_timing, 142  
get\_timing\_summary, 142  
get\_unique, 146  
good\_val, 147  
gr\_db, 147  
gr\_rag, 147  
GRAY, 147  
GREEN, 147  
kg, 147  
msg\_bad\_addr, 147  
msg\_bad\_coll, 147  
msg\_bad\_df\_parse, 147  
msg\_bad\_exec\_f, 148  
msg\_bad\_exec\_q, 148  
msg\_bad\_graph, 148  
msg\_bad\_path, 148  
msg\_bad\_table, 148  
msg\_bad\_triples, 148  
msg\_chart\_saved, 148  
MSG\_COLOR, 148  
msg\_compare, 148  
msg\_db\_connect, 149  
msg\_db\_current, 149  
msg\_db\_exists, 149  
msg\_db\_not\_found, 149  
msg\_elapsed\_time, 149  
msg\_fail\_manage\_db, 149  
msg\_fail\_manage\_gr, 149  
msg\_fail\_parse, 149  
msg\_good\_coll, 150  
msg\_good\_df\_parse, 150  
msg\_good\_exec\_f, 150  
msg\_good\_exec\_q, 150  
msg\_good\_exec\_qr, 150  
msg\_good\_graph, 150  
msg\_good\_path, 150  
msg\_good\_table, 150  
msg\_multiple\_query, 151  
msg\_none\_df, 151  
msg\_result, 151  
msg\_success\_managed\_db, 151

msg\_success\_managed\_gr, 151  
 msg\_swap\_db, 151  
 msg\_swap\_kg, 151  
 msg\_time\_dump, 152  
 msg\_unknown\_error, 152  
 print\_timing\_summary, 143  
 pytest\_db, 152  
 RECORD\_TIME, 152  
 RED, 152  
 rel\_db, 152  
 run\_f, 152  
 run\_id, 152  
 run\_q, 152  
 sub\_gr, 153  
 success, 143  
 SUCCESS\_COLOR, 153  
 success\_legacy, 143  
 swap\_db, 153  
 swap\_kg, 153  
 t\_dump, 153  
 test\_basic, 153  
 test\_df, 153  
 test\_info, 153  
 test\_ops, 153  
 test\_tmp\_db, 153  
 time, 143  
 TIME\_COLOR, 154  
 time\_message, 144  
 timer, 144  
 USE\_COLORS, 154  
 warn, 144  
 WARNING\_COLOR, 154  
 WHITE, 154  
 YELLOW, 154  
 Log.BadAddressFailure, 69  
     \_\_init\_\_, 71  
 Log.Failure, 102  
     \_\_init\_\_, 103  
     \_\_str\_\_, 104  
 msg, 104  
 prefix, 104

m

src.components.corpus, 27  
 main\_graph  
     conftest, 22  
     Session, 192  
 mark\_task\_in\_progress  
     src.core.worker, 47  
 max\_tokens  
     RelationExtractor, 188  
 merge\_dataframes  
     src.components.corpus, 26  
 Metrics, 154  
     \_\_init\_\_, 155  
     compute\_basic\_metrics, 156  
     create\_summary\_payload, 156  
     generate\_default\_metrics, 156  
     generate\_example\_metrics, 157

HOST, 160  
 PORT, 160  
 post\_basic\_metrics, 158  
 post\_basic\_output, 158  
 post\_payload, 158  
 SummaryData, 196  
 timeout\_seconds, 160  
 url, 160

metrics

Session, 192  
 MetricsController, 160  
     \_hubContext, 162  
     \_logger, 162  
     GetAll, 161  
     GetIndex, 161  
     MetricsController, 161  
     Post, 162  
     Summaries, 162

MetricsHub, 162

    \_logger, 164  
     MetricsHub, 163  
     OnConnectedAsync, 163  
     OnDisconnectedAsync, 163

model

RelationExtractor, 188  
 model\_name  
     LLMConnector, 136

mongo\_handle

src.connectors.document, 33

MongoHandle

src.connectors.document, 33  
 src.core.boss, 39  
 src.core.worker, 50

msg

Log.Failure, 104  
 msg\_bad\_addr  
     Log, 147  
 msg\_bad\_coll  
     Log, 147  
 msg\_bad\_df\_parse  
     Log, 147  
 msg\_bad\_exec\_f  
     Log, 148  
 msg\_bad\_exec\_q  
     Log, 148  
 msg\_bad\_graph  
     Log, 148  
 msg\_bad\_path  
     Log, 148  
 msg\_bad\_table  
     Log, 148  
 msg\_bad\_triples  
     Log, 148  
 msg\_chart\_saved  
     Log, 148  
 MSG\_COLOR  
     Log, 148  
 msg\_compare

Log, 148  
msg\_db\_connect  
    Log, 149  
msg\_db\_current  
    Log, 149  
msg\_db\_exists  
    Log, 149  
msg\_db\_not\_found  
    Log, 149  
msg\_elapsed\_time  
    Log, 149  
msg\_fail\_manage\_db  
    Log, 149  
msg\_fail\_manage\_gr  
    Log, 149  
msg\_fail\_parse  
    Log, 149  
msg\_good\_coll  
    Log, 150  
msg\_good\_df\_parse  
    Log, 150  
msg\_good\_exec\_f  
    Log, 150  
msg\_good\_exec\_q  
    Log, 150  
msg\_good\_exec\_qr  
    Log, 150  
msg\_good\_graph  
    Log, 150  
msg\_good\_path  
    Log, 150  
msg\_good\_table  
    Log, 150  
msg\_multiple\_query  
    Log, 151  
msg\_none\_df  
    Log, 151  
msg\_result  
    Log, 151  
msg\_success\_managed\_db  
    Log, 151  
msg\_success\_managed\_gr  
    Log, 151  
msg\_swap\_db  
    Log, 151  
msg\_swap\_kg  
    Log, 151  
msg\_time\_dump  
    Log, 152  
msg\_unknown\_error  
    Log, 152  
mysqlConnector, 164  
    \_\_init\_\_, 167  
    specific\_queries, 167

Name  
    PRF1Metric, 177  
    ScalarMetric, 189  
nature\_scene\_graph

tests.test\_kg\_triples, 61  
nlp  
    RelationExtractor, 189  
    src.components.book\_conversion, 24  
normalize\_title  
    src.components.corpus, 26  
normalize\_triples  
    LLMConnector, 135  
NOT\_DUMMY\_  
    GraphConnector, 116  
notify\_boss  
    src.core.worker, 48

o

    Triple, 198  
old\_main  
    src.main, 52  
OnConnectedAsync  
    MetricsHub, 163  
OnDisconnectedAsync  
    MetricsHub, 163

pandoc\_xml\_path  
    EPUBToTEI, 101  
ParagraphStreamTEI, 167  
    \_\_init\_\_, 169  
    allowed\_chapters, 170  
    book\_id, 170  
    chunks, 170  
    encoding, 170  
    end\_inclusive, 171  
    lines, 171  
    pre\_compute\_segments, 170  
    root, 171  
    start\_inclusive, 171  
    story\_id, 171  
    stream\_segments, 170  
    tei\_path, 171  
    xml\_namespace, 171

parser  
    src.core.worker, 50

password  
    DatabaseConnector, 90

pipeline\_A  
    src.main, 52

pipeline\_B  
    src.main, 52

pipeline\_C  
    src.main, 52

pipeline\_D  
    src.main, 53

pipeline\_E  
    src.main, 53

Plot, 172  
    time\_elapsed\_by\_names, 172

PORT  
    Metrics, 160  
    src.core.worker, 50

port

**DatabaseConnector**, 90  
**src.core.worker**, 50  
**Post**  
 MetricsController, 162  
**post\_basic\_metrics**  
 Metrics, 158  
**post\_basic\_output**  
 Metrics, 158  
**post\_chunk\_status**  
 src.core.boss, 38  
**post\_payload**  
 Metrics, 158  
**post\_process\_full\_story**  
 src.core.boss, 38  
**post\_story\_status**  
 src.core.boss, 39  
**postgresConnector**, 172  
 \_\_init\_\_, 176  
 specific\_queries, 176  
**pre\_compute\_segments**  
 ParagraphStreamTEI, 170  
**pre\_split\_chunks**  
 Story, 193  
**Precision**  
 PRF1Metric, 177  
**prefix**  
 Log.Failure, 104  
**PRF1Metric**, 176  
 F1Score, 177  
 Name, 177  
 Precision, 177  
 Recall, 177  
**PRF1Metrics**  
 SummaryMetrics, 197  
**print\_nodes**  
 KnowledgeGraph, 128  
**print\_timing\_summary**  
 Log, 143  
**print\_triples**  
 KnowledgeGraph, 128  
**process\_task**  
 src.core.worker, 48  
**pytest\_addoption**  
 conftest, 22  
**pytest\_db**  
 Log, 152  
**QA**  
 SummaryMetrics, 197  
**QAItem**, 177  
 Accuracy, 177  
 GeneratedAnswer, 177  
 GoldAnswer, 177  
 IsCorrect, 178  
 Question, 178  
**QAItems**  
 QAMetric, 178  
**QAMetric**, 178  
 AverageAccuracy, 178  
**QAItems**, 178  
**QAResults**  
 SummaryData, 196  
**Question**  
 QAItem, 178  
**r**  
 Triple, 198  
**raw\_tei\_content**  
 EPUBToTEI, 102  
**reader**  
 Story, 194  
**Recall**  
 PRF1Metric, 177  
**RECORD\_TIME**  
 Log, 152  
**RED**  
 Log, 152  
**rel\_db**  
 Log, 152  
**relational\_db**  
 conftest, 22  
 Session, 192  
**RelationalConnector**, 179  
 \_\_init\_\_, 182  
 \_parsable\_to\_df, 182  
 \_returns\_data, 182  
 \_split\_combined, 183  
 change\_database, 183  
 check\_connection, 184  
 connection\_string, 187  
 create\_database, 184  
 database\_exists, 184  
 database\_name, 187  
 db\_type, 187  
 drop\_database, 185  
 execute\_query, 185  
 from\_env, 186  
 get\_dataframe, 186  
 test\_operations, 187  
 verbose, 187  
**RelationExtractor**, 188  
 \_\_init\_\_, 188  
 extract, 188  
 max\_tokens, 188  
 model, 188  
 nlp, 189  
 sentencizer, 189  
 tokenizer, 189  
 tuple\_delim, 189  
**required**  
 src.core.worker, 50  
**response**  
 src.main, 54  
**root**  
 ParagraphStreamTEI, 171  
**run\_bookscore**  
 src.components.metrics, 28  
**run\_f**

Log, 152  
run\_id  
    Log, 152  
run\_q  
    Log, 152  
run\_questeval  
    src.components.metrics, 29

s  
    Triple, 198  
SAME\_DB\_KG\_  
    GraphConnector, 116  
save\_task\_result  
    src.core.worker, 49  
ScalarMetric, 189  
    Name, 189  
    Value, 189  
ScalarMetrics  
    SummaryMetrics, 197  
sentencizer  
    RelationExtractor, 189  
    src.components.book\_conversion, 24  
Session, 190  
    \_\_init\_\_, 191  
    \_\_new\_\_, 191  
    \_created, 191  
    \_instance, 191  
    \_session, 192  
    docs\_db, 192  
    graph\_db, 192  
    main\_graph, 192  
    metrics, 192  
    relational\_db, 192  
    verbose, 192  
session  
    conftest, 23  
    src.core.context, 41  
Source Directory Overview, 7  
specific\_queries  
    mysqlConnector, 167  
    postgresConnector, 176  
src, 23  
src.charts, 23  
src.components, 23  
src.components.book\_conversion, 24  
    nlp, 24  
    sentencizer, 24  
src.components.corpus, 24  
    df, 26  
    df\_booksum, 26  
    df\_nqa, 26  
    fuzzy\_merge\_titles, 25  
    index, 26  
    load\_booksum, 25  
    load\_narrativeqa, 25  
    m, 27  
    merge\_dataframes, 26  
    normalize\_title, 26  
    to\_df\_booksum, 26

    to\_df\_nqa, 26  
src.components.fact\_storage, 27  
src.components.metrics, 27  
    chunk\_bookscore, 27  
    run\_bookscore, 28  
    run\_questeval, 29  
src.components.relation\_extraction, 29  
src.connectors, 29  
src.connectors.base, 30  
src.connectors.document, 30  
    \_\_docs\_to\_df, 31  
    \_\_find\_compatible\_nested\_key, 31  
    \_\_flatten\_recursive, 31  
    \_\_sanitize\_document, 32  
    \_\_sanitize\_json, 32  
    mongo\_handle, 33  
    MongoHandle, 33  
src.connectors.graph, 33  
    \_\_filter\_to\_db, 34  
    \_\_normalize\_elements, 34  
    \_\_tuples\_to\_df, 35  
src.connectors.llm, 35  
src.connectors.relational, 35  
src.core, 36  
src.core.boss, 36  
    assign\_task\_to\_worker, 36  
    clear\_task\_data, 37  
    create\_app, 37  
    create\_boss\_thread, 38  
    load\_worker\_config, 38  
    MongoHandle, 39  
    post\_chunk\_status, 38  
    post\_process\_full\_story, 38  
    post\_story\_status, 39  
src.core.context, 40  
    \_\_getattr\_\_, 40  
    get\_session, 40  
    session, 41  
src.core.stages, 41  
    group\_21\_1\_describe\_graph, 42  
    group\_21\_2\_send\_statistics, 42  
    group\_21\_3\_post\_statistics, 42  
    task\_01\_convert\_epub, 42  
    task\_02\_parse\_chapters, 42  
    task\_03\_chunk\_story, 42  
    task\_10\_random\_chunk, 42  
    task\_10\_sample\_chunks, 43  
    task\_11\_send\_chunk, 43  
    task\_12\_relation\_extraction\_rebel, 43  
    task\_13\_concatenate\_triples, 43  
    task\_14\_relation\_extraction\_llm, 43  
    task\_15\_sanitize\_triples\_llm, 43  
    task\_20\_send\_triples, 43  
    task\_22\_verbalize\_triples, 43  
    task\_30\_summarize\_llm, 44  
    task\_31\_send\_summary, 44  
    task\_40\_post\_payload, 44  
    task\_40\_post\_summary, 44

src.core.worker, 45  
 app, 49  
 args, 49  
 boss\_url, 49  
 create\_app, 46  
 daemon, 49  
 get\_task\_info, 46  
 help, 50  
 host, 50  
 load\_boss\_config, 46  
 load\_imports, 47  
 load\_mongo\_config, 47  
 mark\_task\_in\_progress, 47  
 MongoHandle, 50  
 notify\_boss, 48  
 parser, 50  
 PORT, 50  
 port, 50  
 process\_task, 48  
 required, 50  
 save\_task\_result, 49  
 target, 50  
 task\_queue, 50  
 task\_worker, 50  
 True, 51  
 use\_reloader, 51  
 src.main, 51  
     book\_id, 53  
     book\_title, 53  
     BOSS\_PORT, 53  
     checkpoint\_path, 53  
     chunk, 54  
     chunk\_id, 54  
     chunks, 54  
     COLLECTION, 54  
     data, 54  
     DB\_NAME, 54  
     exist\_ok, 54  
     full\_pipeline, 52  
     load\_from\_checkpoint, 54  
     old\_main, 52  
     pipeline\_A, 52  
     pipeline\_B, 52  
     pipeline\_C, 52  
     pipeline\_D, 53  
     pipeline\_E, 53  
     response, 54  
     story\_id, 55  
     summary, 55  
     triples, 55  
     triples\_string, 55  
 src.util, 55  
     check\_values, 55  
     df\_natural\_sorted, 56  
 start\_inclusive  
     ParagraphStreamTEI, 171  
 Story, 193  
     \_\_init\_\_, 193  
         \_make\_single, 193  
         \_merge\_chunks, 193  
         pre\_split\_chunks, 193  
         reader, 194  
         stream\_chunks, 194  
 story\_id  
     Chunk, 77  
     ParagraphStreamTEI, 171  
     src.main, 55  
 story\_percent  
     Chunk, 77  
 StoryStreamAdapter, 194  
     stream\_paragraphs, 195  
     stream\_segments, 195  
     stream\_sentences, 195  
 stream\_chapters  
     Book, 71  
 stream\_chunks  
     Story, 194  
 stream\_paragraphs  
     StoryStreamAdapter, 195  
 stream\_segments  
     BookStream, 74  
     ParagraphStreamTEI, 170  
     StoryStreamAdapter, 195  
 stream\_sentences  
     StoryStreamAdapter, 195  
 sub\_gr  
     Log, 153  
 success  
     Log, 143  
 SUCCESS\_COLOR  
     Log, 153  
 success\_legacy  
     Log, 143  
 Summaries  
     MetricsController, 162  
 summary  
     src.main, 55  
 SummaryData, 196  
     BookID, 196  
     BookTitle, 196  
     GoldSummaryText, 196  
     Metrics, 196  
     QAResults, 196  
     SummaryText, 196  
 SummaryMetrics, 197  
     GetDefault, 197  
     PRF1Metrics, 197  
     QA, 197  
     ScalarMetrics, 197  
 SummaryText  
     SummaryData, 196  
 swap\_db  
     Log, 153  
 swap\_kg  
     Log, 153  
 system\_prompt

LLMConnector, 136  
t\_dump  
    Log, 153  
target  
    src.core.worker, 50  
task\_01\_convert\_epub  
    src.core.stages, 42  
task\_02\_parse\_chapters  
    src.core.stages, 42  
task\_03\_chunk\_story  
    src.core.stages, 42  
task\_10\_random\_chunk  
    src.core.stages, 42  
task\_10\_sample\_chunks  
    src.core.stages, 43  
task\_11\_send\_chunk  
    src.core.stages, 43  
task\_12\_relation\_extraction\_rebel  
    src.core.stages, 43  
task\_13\_concatenate\_triples  
    src.core.stages, 43  
task\_14\_relation\_extraction\_llm  
    src.core.stages, 43  
task\_15\_sanitize\_triples\_llm  
    src.core.stages, 43  
task\_20\_send\_triples  
    src.core.stages, 43  
task\_22\_verbalize\_triples  
    src.core.stages, 43  
task\_30\_summarize\_llm  
    src.core.stages, 44  
task\_31\_send\_summary  
    src.core.stages, 44  
task\_40\_post\_payload  
    src.core.stages, 44  
task\_40\_post\_summary  
    src.core.stages, 44  
task\_queue  
    src.core.worker, 50  
task\_worker  
    src.core.worker, 50  
tei\_path  
    EPUBToTEI, 102  
    ParagraphStreamTEI, 171  
temp\_database  
    DatabaseConnector, 89  
temp\_graph  
    GraphConnector, 116  
Test Suite Overview, 11  
test\_basic  
    Log, 153  
test\_cypher\_example\_1  
    tests.test\_db\_files, 59  
test\_cypher\_example\_2  
    tests.test\_db\_files, 59  
test\_cypher\_example\_3  
    tests.test\_db\_files, 59  
test\_cypher\_example\_4  
    tests.test\_db\_files, 59  
tests.test\_db\_files, 59  
test\_db\_docs\_comprehensive  
    tests.test\_db\_basic, 57  
test\_db\_docs\_minimal  
    tests.test\_db\_basic, 57  
test\_db\_graph\_comprehensive  
    tests.test\_db\_basic, 57  
test\_db\_graph\_minimal  
    tests.test\_db\_basic, 57  
test\_db\_relational\_comprehensive  
    tests.test\_db\_basic, 57  
test\_db\_relational\_minimal  
    tests.test\_db\_basic, 57  
test\_detect\_community\_clusters\_comprehensive  
    tests.test\_kg\_triples, 61  
test\_detect\_community\_clusters\_minimal  
    tests.test\_kg\_triples, 62  
test\_df  
    Log, 153  
test\_get\_neighborhood  
    tests.test\_kg\_triples, 62  
test\_get\_neighborhood\_comprehensive  
    tests.test\_kg\_triples, 62  
test\_get\_random\_walk\_sample  
    tests.test\_kg\_triples, 62  
test\_get\_random\_walk\_sample\_comprehensive  
    tests.test\_kg\_triples, 63  
test\_get\_subgraph\_by\_nodes  
    tests.test\_kg\_triples, 63  
test\_info  
    Log, 153  
test\_job\_01\_convert\_epub  
    tests.test\_pipeline, 65  
test\_job\_02\_parse\_chapters  
    tests.test\_pipeline, 65  
test\_job\_03\_chunk\_story  
    tests.test\_pipeline, 66  
test\_job\_10\_random\_chunk  
    tests.test\_pipeline, 66  
test\_job\_10\_sample\_chunks  
    tests.test\_pipeline, 66  
test\_job\_11\_send\_chunk  
    tests.test\_pipeline, 66  
test\_job\_13\_concatenate\_triples  
    tests.test\_pipeline, 66  
test\_job\_15\_sanitize\_triples\_llm  
    tests.test\_pipeline, 66  
test\_job\_20\_send\_triples  
    tests.test\_pipeline, 66  
test\_job\_21\_describe\_graph  
    tests.test\_pipeline, 67  
test\_job\_22\_verbalize\_triples  
    tests.test\_pipeline, 67  
test\_job\_31\_send\_summary  
    tests.test\_pipeline, 67  
test\_knowledge\_graph\_triples  
    tests.test\_kg\_triples, 63  
test\_mongo\_example\_1

tests.test\_db\_files, 59  
 test\_mongo\_example\_2  
     tests.test\_db\_files, 60  
 test\_mongo\_example\_3  
     tests.test\_db\_files, 60  
 test\_operations  
     Connector, 80  
     DocumentConnector, 98  
     GraphConnector, 117  
     LLMConnector, 135  
     RelationalConnector, 187  
 test\_ops  
     Log, 153  
 test\_pipeline\_A\_from\_csv  
     tests.test\_pipeline, 67  
 test\_pipeline\_A\_minimal  
     tests.test\_pipeline, 67  
 test\_pipeline\_C\_minimal  
     tests.test\_pipeline, 67  
 test\_pipeline\_E\_minimal\_full\_payload  
     tests.test\_pipeline, 68  
 test\_pipeline\_E\_minimal\_summary\_only  
     tests.test\_pipeline, 68  
 test\_ranked\_degree  
     tests.test\_kg\_triples, 63  
 test\_ranked\_degree\_ties  
     tests.test\_kg\_triples, 64  
 test\_sql\_example\_1  
     tests.test\_db\_files, 60  
 test\_sql\_example\_2  
     tests.test\_db\_files, 60  
 test\_tmp\_db  
     Log, 153  
 tests, 56  
 tests.test\_db\_basic, 56  
     test\_db\_docs\_comprehensive, 57  
     test\_db\_docs\_minimal, 57  
     test\_db\_graph\_comprehensive, 57  
     test\_db\_graph\_minimal, 57  
     test\_db\_relational\_comprehensive, 57  
     test\_db\_relational\_minimal, 57  
 tests.test\_db\_files, 58  
     \_exec\_query\_file, 58  
     load\_examples\_relational, 59  
     test\_cypher\_example\_1, 59  
     test\_cypher\_example\_2, 59  
     test\_cypher\_example\_3, 59  
     test\_cypher\_example\_4, 59  
     test\_mongo\_example\_1, 59  
     test\_mongo\_example\_2, 60  
     test\_mongo\_example\_3, 60  
     test\_sql\_example\_1, 60  
     test\_sql\_example\_2, 60  
 tests.test\_kg\_triples, 61  
     nature\_scene\_graph, 61  
     test\_detect\_community\_clusters\_comprehensive,  
         61  
     test\_detect\_community\_clusters\_minimal, 62  
         test\_get\_neighborhood, 62  
         test\_get\_neighborhood\_comprehensive, 62  
         test\_get\_random\_walk\_sample, 62  
         test\_get\_random\_walk\_sample\_comprehensive,  
             63  
         test\_get\_subgraph\_by\_nodes, 63  
         test\_knowledge\_graph\_triples, 63  
         test\_ranked\_degree, 63  
         test\_ranked\_degree\_ties, 64  
 tests.test\_pipeline, 64  
     book\_1\_data, 65  
     book\_2\_data, 65  
     book\_data, 65  
     test\_job\_01\_convert\_epub, 65  
     test\_job\_02\_parse\_chapters, 65  
     test\_job\_03\_chunk\_story, 66  
     test\_job\_10\_random\_chunk, 66  
     test\_job\_10\_sample\_chunks, 66  
     test\_job\_11\_send\_chunk, 66  
     test\_job\_13\_concatenate\_triples, 66  
     test\_job\_15\_sanitize\_triples\_llm, 66  
     test\_job\_20\_send\_triples, 66  
     test\_job\_21\_describe\_graph, 67  
     test\_job\_22\_verbalize\_triples, 67  
     test\_job\_31\_send\_summary, 67  
     test\_pipeline\_A\_from\_csv, 67  
     test\_pipeline\_A\_minimal, 67  
     test\_pipeline\_C\_minimal, 67  
     test\_pipeline\_E\_minimal\_full\_payload, 68  
     test\_pipeline\_E\_minimal\_summary\_only, 68  
 text  
     Chunk, 77  
 time  
     Log, 143  
 TIME\_COLOR  
     Log, 154  
 time\_elapsed\_by\_names  
     Plot, 172  
 time\_message  
     Log, 144  
 timeout\_seconds  
     Metrics, 160  
 timer  
     Log, 144  
 to\_contextualized\_string  
     KnowledgeGraph, 128  
 to\_df\_booksum  
     src.components.corpus, 26  
 to\_df\_nqa  
     src.components.corpus, 26  
 to\_mongo\_dict  
     Chunk, 77  
 to\_narrative  
     KnowledgeGraph, 129  
 to\_triples\_string  
     KnowledgeGraph, 129  
 tokenizer  
     RelationExtractor, 189

Triple, 198  
    o, 198  
    r, 198  
    s, 198  
triples  
    src.main, 55  
triples\_string  
    src.main, 55  
triples\_to\_names  
    KnowledgeGraph, 130  
True  
    src.core.worker, 51  
tuple\_delim  
    RelationExtractor, 189  
  
url  
    Metrics, 160  
USE\_COLORS  
    Log, 154  
use\_reloader  
    src.core.worker, 51  
username  
    DatabaseConnector, 90  
  
Value  
    ScalarMetric, 189  
verbose  
    DatabaseConnector, 90  
    DocumentConnector, 99  
    GraphConnector, 117  
    KnowledgeGraph, 131  
    RelationalConnector, 187  
    Session, 192  
  
warn  
    Log, 144  
WARNING\_COLOR  
    Log, 154  
WHITE  
    Log, 154  
  
xml\_namespace  
    EPUBToTEI, 102  
    ParagraphStreamTEI, 171  
  
YELLOW  
    Log, 154