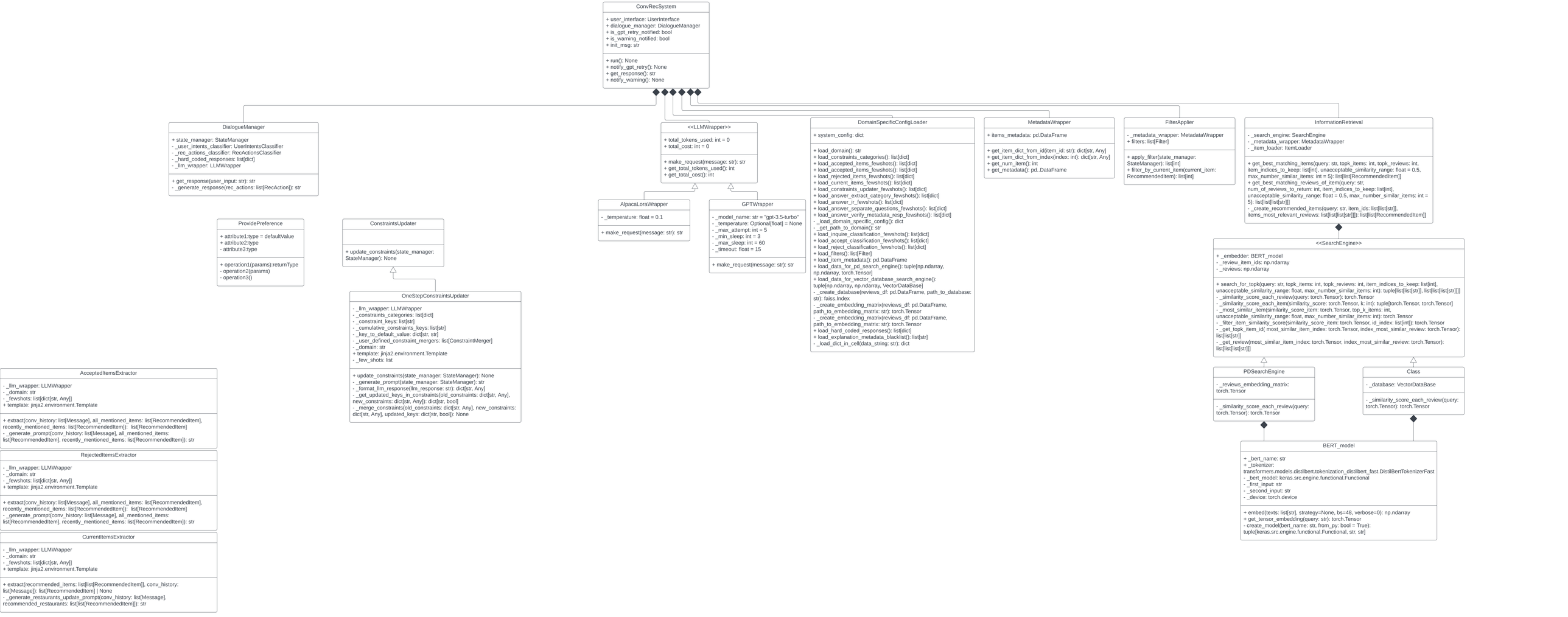
ConvRecSystem
+ user_interface: UserInterface
+ dialogue_manager: DialogueManager
+ is_gpt_retry_notified: bool
+ is_warning_notified: bool
+ init_msg: str
+ run(): None
+ notify_gpt_retry(): None
+ get_response(): str
+ notify_warning(): None
DialogueManager
+ state_manager: StateManager
- _user_intents_classifier: UserIntentsClassifier
- _rec_actions_classifier: RecActionsClassifier
- _hard_coded_responses: list[dict]
- _llm_wrapper: LLMWrapper
+ get_response(user_input: str): str
- _generate_response(rec_actions: list[RecAction]): str
<<LLMWrapper>>
+ total_tokens_used: int = 0
+ total_cost: int = 0
+ make_request(message: str): str
+ get_total_tokens_used(): int
+ get_total_cost(): int
AlpacaLoraWrapper
- _temperature: float = 0.1
+ make_request(message: str): str
GPTWrapper
- _model_name: str = "gpt-3.5-turbo"
- _temperature: Optional[float] = None
- _max_attempt: int = 5
- _min_sleep: int = 3
- _max_sleep: int = 60
- _timeout: float = 15
+ make_request(message: str): str
DomainSpecificConfigLoader
+ system_config: dict
+ load_domain(): str
+ load_constraints_categories(): list[dict]
+ load_accepted_items_fewshots(): list[dict]
+ load_accepted_items_fewshots(): list[dict]
+ load_rejected_items_fewshots(): list[dict]
+ load_current_items_fewshots(): list[dict]
+ load_constraints_updater_fewshots(): list[dict]
+ load_answer_extract_category_fewshots(): list[dict]
+ load_answer_ir_fewshots(): list[dict]
+ load_answer_separate_questions_fewshots(): list[dict]
+ load_answer_verify_metadata_resp_fewshots(): list[dict]
- _load_domain_specific_config(): dict
- _get_path_to_domain(): str
+ load_inquire_classification_fewshots(): list[dict]
+ load_accept_classification_fewshots(): list[dict]
+ load_reject_classification_fewshots(): list[dict]
+ load_filters(): list[Filter]
+ load_item_metadata(): pd.DataFrame
+ load_data_for_pd_search_engine(): tuple[np.ndarray, np.ndarray, torch.Tensor]
+ load_data_for_vector_database_search_engine(): tuple[np.ndarray, np.ndarray, VectorDataBase]
- _create_database(reviews_df: pd.DataFrame, path_to_database: str): faiss.Index
- _create_embedding_matrix(reviews_df: pd.DataFrame, path_to_embedding_matrix: str): torch.Tensor
- _create_embedding_matrix(reviews_df: pd.DataFrame, path_to_embedding_matrix: str): torch.Tensor
+ load_hard_coded_responses(): list[dict]
+ load_explanation_metadata_blacklist(): list[str]
- _load_dict_in_cell(data_string: str): dict
MetadataWrapper
+ items_metadata: pd.DataFrame
+ get_item_dict_from_id(item_id: str): dict[str, Any]
+ get_item_dict_from_index(index: int): dict[str, Any]
+ get_num_item(): int
+ get_metadata(): pd..DataFrame
FilterApplier
- _metadata_wrapper: MetadataWrapper
+ filters: list[Filter]
+ apply_filter(state_manager: StateManager): list[int]
+ filter_by_current_item(current_item: RecommendedItem): list[int]
InformationRetrieval
- _search_engine: SearchEngine
- _metadata_wrapper: MetadataWrapper
- _item_loader: ItemLoader
+ get_best_matching_items(query: str, topk_items: int, topk_reviews: int, item_indices_to_keep: list[int], unacceptable_similarity_range: float = 0.5, max_number_similar_items: int = 5): list[list[RecommendedItem]]
+ get_best_matching_reviews_of_item(query: str, num_of_reviews_to_return: int, item_indices_to_keep: list[int], unacceptable_similarity_range: float = 0.5, max_number_similar_items: int = 5): list[list[list[str]]]
- _create_recommended_items(query: str, item_ids: list[list[str]], items_most_relevant_reviews: list[list[list[str]]]): list[list[RecommendedItem]]
<<SearchEngine>>
+ _embedder: BERT_model
- _review_item_ids: np.ndarray
- _reviews: np.ndarray
+ search_for_topk(query: str, topk_items: int, topk_reviews: int, item_indices_to_keep: list[int], unacceptable_similarity_range: float, max_number_similar_items: int): tuple[list[list[str]], list[list[list[str]]]]
- _similarity_score_each_review(query: torch.Tensor): torch.Tensor
- _similarity_score_each_item(similarity_score: torch.Tensor, k: int): tuple[torch.Tensor, torch.Tensor]
- _most_similar_item(similarity_score_item: torch.Tensor, top_k_items: int, unacceptable_similarity_range: float, max_number_similar_items: int): torch.Tensor
- _filter_item_similarity_score(similarity_score_item: torch.Tensor, id_index: list[int]): torch.Tensor
- _get_topk_item_id(most_similar_item_index: torch.Tensor, index_most_similar_review: torch.Tensor): list[list[str]]
- _get_review(most_similar_item_index: torch.Tensor, index_most_similar_review: torch.Tensor): list[list[list[str]]]
PDSearchEngine
- _reviews_embedding_matrix: torch.Tensor
- _similarity_score_each_review(query: torch.Tensor): torch.Tensor
Class
- _database: VectorDataBase
- _similarity_score_each_review(query: torch.Tensor): torch.Tensor
BERT_model
+ _bert_name: str
+ _tokenizer: transformers.models.distilbert.tokenization_distilbert_fast.DistilBertTokenizerFast
- _bert_model: keras.src.engine.functional.Functional
- _first_input: str
- _second_input: str
- _device: torch.device
+ embed(texts: list[str], strategy=None, bs=48, verbose=0): np.ndarray
+ get_tensor_embedding(query: str): torch.Tensor
- create_model(bert_name: str, from_py: bool = True): tuple[keras.src.engine.functional.Functional, str, str]
ProvidePreference
+ attribute1:type = defaultValue
+ attribute2:type
- attribute3:type
+ operation1(params):returnType
- operation2(params)
- operation3()
ConstraintsUpdater
+ update_constraints(state_manager: StateManager): None
OneStepConstraintsUpdater
- _llm_wrapper: LLMWrapper
- _constraints_categories: list[dict]
- _constraint_keys: list[str]
- _cumulative_constraints_keys: list[str]
- _key_to_default_value: dict[str, str]
- _user_defined_constraint_mergers: list[ConstraintMerger]
- _domain: str
+ template: jinja2.environment.Template
- _few_shots: list
+ update_constraints(state_manager: StateManager): None
- _generate_prompt(state_manager: StateManager): str
- _format_llm_response(llm_response: str): dict[str, Any]
- _get_updated_keys_in_constraints(old_constraints: dict[str, Any], new_constraints: dict[str, Any]): dict[str, bool]
- _merge_constraints(old_constraints: dict[str, Any], new_constraints: dict[str, Any], updated_keys: dict[str, bool]): None
AcceptedItemsExtractor
- _llm_wrapper: LLMWrapper
- _domain: str
- _fewshots: list[dict[str, Any]]
+ template: jinja2.environment.Template
+ extract(conv_history: list[Message], all_mentioned_items: list[RecommendedItem], recently_mentioned_items: list[RecommendedItem]): list[RecommendedItem]
- _generate_prompt(conv_history: list[Message], all_mentioned_items: list[RecommendedItem], recently_mentioned_items: list[RecommendedItem]): str
RejectedItemsExtractor
- _llm_wrapper: LLMWrapper
- _domain: str
- _fewshots: list[dict[str, Any]]
+ template: jinja2.environment.Template
+ extract(conv_history: list[Message], all_mentioned_items: list[RecommendedItem], recently_mentioned_items: list[RecommendedItem]): list[RecommendedItem]
- _generate_prompt(conv_history: list[Message], all_mentioned_items: list[RecommendedItem], recently_mentioned_items: list[RecommendedItem]): str
CurrentItemsExtractor
- _llm_wrapper: LLMWrapper
- _domain: str
- _fewshots: list[dict[str, Any]]
+ template: jinja2.environment.Template
+ extract(recommended_items: list[list[RecommendedItem]], conv_history: list[Message]): list[RecommendedItem] | None
- _generate_restaurants_update_prompt(conv_history: list[Message], recommended_restaurants: list[list[RecommendedItem]]): str