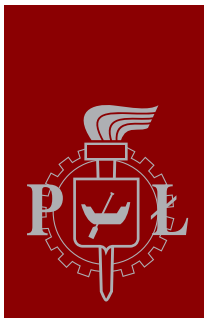

Politechnika Łódzka
Wydział Fizyki Technicznej, Informatyki
i Matematyki Stosowanej

***Marcin Grzelak***
***247022***

Engineering Thesis
Information Technology

# Mechanisms of stimulated context building in conversational recommendation systems based on large language models

***Supervisor:***
*Prof. Dr hab. Inż. Adam Wojciechowski*

ŁÓDŹ 2026

## Abstract

The aim of this work is the design, implementation, and analysis of a conversational recommendation system for dining establishments in the city of **Łódź**. The task of the system is to provide support during the recommendation process, based not only on the compatibility of places but also on the user's preferences. Traditional recommendation systems, mainly based on the analysis of historical data and a passive interaction model, may prove insufficient in situations that go beyond the predicted scenarios of the system. In response to these limitations, conversational systems, which combine natural language processing techniques with recommendation algorithms, are becoming increasingly common.

Within the scope of this work, the architecture of the CRS system was designed based on natural language processing methods and semantic search. The system utilizes vector representations of text (embeddings) generated using transformer-type language models, enabling semantic comparison of user queries with descriptions of elements in the database. To aggregate information from successive statements, pooling methods were applied allowing the construction of a coherent representation of user preferences tailored to the chosen language model. The extraction of the desired location provided in the user's query is handled by a named entity recognition module.

The system architecture is based on the Retrieval-Augmented Generation paradigm, combining semantic search mechanisms with the capabilities of generative large language models. The retrieval module is responsible for selecting the most relevant objects based on the cosine similarity of their vector representations, while the generative component serves as a natural language correspondent.

The conducted analysis showed that the use of advanced semantic representation techniques, the RAG architecture, and multiple functions for specifically defined context construction enables effective matching of recommendations to the user's intent. The results confirm that conversational recommendation systems, characterized by their ease of use and dynamic dialogue, provide new opportunities.



**Keywords**: CRS RAG PLLuM Recommendation Łódź

# Table of Contents

# Glossary and List of Abbreviations

**Conversational Recommender System** A system that combines recommendation algorithms with dialogue interfaces to support users in finding items of interest through natural language interactions.

**Retrieval-Augmented Generation** A technique for enhancing the accuracy and reliability of generative AI models with information fetched from specific and relevant data sources

**Embedding** A dense vector representation of text or other data where the distance between vectors captures semantic similarity.

**Hypothetical Document Embeddings** A query expansion technique where an LLM generates a hypothetical answer to a query, which is then encoded to retrieve real documents.

**Pooling** The process of deriving a fixed-size vector representation (sentence embedding) from the variable-length sequence of token embeddings output by a model (e.g., Mean Pooling, CLS Pooling).

# List of Abbreviations

**CLS** Classification Token

**CRS** Conversational Recommender System

**FAISS** Facebook AI Similarity Search

**HyDE** Hypothetical Document Embeddings

**NER** Named Entity Recognition

**NLP** Natural Language Processing

**PLLuM** Polish Large Language Universal Model

**RAG** Retrieval-Augmented Generation

**POI** Point of Interest

# Introduction

Conversational recommender systems (CRS) have become increasingly popular in recent years. They focus to provide personalized recommendations by interacting with users through natural language dialogue. Unlike traditional recommendation systems, which usually are based on a static list of results, conversational systems allow users to refine their needs with further messages during a conversation. This makes the recommendation process more friendly and natural reminding human communication.

Thanks to constant improvment in technological area and growing presence of chatbots and virtual assistants, conversational recommender systems are now applied in many fields, such as entertainment, e-commerce, tourism, education or services. They allow users to describe their needs and preferences without setting some filters or choosing categories. Thereby trying to find desired product on website, it is not necessary to browsing through dozens of pages. Using CRS it is enough to exchange few sentences with system and best suitable recommendations are there.

However, many of existing solutions are oriented toward a global geographical and thematic scope. This is a suitable approach for domains that are not sensitive to regional characteristics. However, for some applications, this often results in deficiencies related to the level of detail in descriptions or the precision of the provided recommendations. Systems with a broad scope of applicability suffer from a fundamental limitation at their source, namely insufficient domain-specific knowledge on which they are based.

In response to this limitation, this thesis proposes a conversational recommendation system focused on a narrow geographical area - recommendations of gastronomic venues in the city of Łódź. This restriction of the domain allows for the inclusion of more detailed descriptions of venues, taking local conditions into account while preserving the regional context.

The adopted approach enables not only a more precise matching of recommendations with user preferences, but also a more reliable evaluation of system correctness. This evaluation is conducted not only through semantic similarity analysis between user queries and recommended venues but also based on factual real-world conditions. As a result, the system serves as an example of a domain-focused CRS that takes advantage of modern natural language processing and semantic search techniques while addressing specific user needs within a defined context.

## Purpose and Scope of Engineering Thesis

The purpose of this thesis is to design and implement a conversational recommendation system for residents and tourists seeking restaurants and cafes in Łódź. The main aspect of the proposed system is the use of a Polish language model (PLLuM), which not only increases the relevance of recommendations in a local context but also aligns with an ethical approach to the development of artificial intelligence systems by promoting regional technological solutions. The system is based on a conversational interface that enables users to conduct natural dialogue queries and receive personalized recommendations based on their preferences, location, and available data describing gastronomic venues. As part of the thesis, the system was implemented and then evaluated in terms of the quality of recommendations, reliability of generated responses, and overall usability from the perspective of the user. The scope of the thesis also includes an analysis of the system architecture, the applied natural language processing techniques, and the correctness and consistency of the returned recommendations. The system utilizes publicly available data sources describing gastronomic venues. The thesis also discusses the limitations of the proposed approach and possible directions for future development.

# 1 Current Conversational Recommender Systems

This section explains the basic concepts underlying conversational recommendation systems. It also cites existing examples of such systems, demonstrating their versatility.

## 1.1 Existing Solutions of Conversational Recommendation Systems

One particularly interesting example of a conversational recommendation system being evaluated is MuseChat (Dong et al., 2024). The system was designed for music recommendation in the filmmaking space, where users interact with the system using natural language dialogue to find suitable background music. Instead of relying solely on explicit me- tadata or predefined tags, MuseChat allows users to describe non-technical properties such as mood, atmosphere, tempo, or emotional tone of a video. The system processes conversational context using a rich language model to extract semantic intent and match it with multimodal embeddings representing music tracks. Audio features and text descriptions are embedded in the semantic space, enabling search based on the dialogue context and the provided music tracks. This solution demonstrates how conversational interaction can effectively capture user preferences that are difficult to express using traditional recommendation systems.

Another instance of a CRS concentrate on a broad domain is the knowledge-enhanced conversational recommendation framework (Fang et al., 2022), developed for the tourism and point-of-interest (POI) recommendation area. In the system, user preferences are incrementally refined during chit chat and combined with structured knowledge graph. The conversational history is encoded using transformer-based language models, while relational information about locations, categories, distances, and contextual attributes is expanded to enrich item representations. The recommendation process connects not only linguistic similarity but also in factual relations. This allows the system to propose locations that satisfy both semantic intent from dialogue and external constraints, such as geographic proximity or functional characteristics of places.

## 1.2 Transformers and Sentence Embeddings

Conversational recommendation systems rely mostly on transformer network architecture (Vaswani et al., 2017) for natural language understanding, beacuse of their efficiency and low cost of traning. Instead of recurrence and convolutions transformers are based on the attention mechanism which allows the model to focus on relevant parts of the input, at the same time being aware of context.

To efficiently compare user queries with dataset document, systems often use sentence embeddings. Models like SBERT or SRoBERTa (Reimers and Gurevych, 2019) encode sentences into vector representations, allowing fast similarity searches with FAISS library. Bi-encoder architectures are particularly useful because they independently encode queries and document

vectors, which improves large-scale data retrieval.

By combining transformers with embeddings, CRS can semantically match user queries with documents in the database, improving recommendation quality with the need to provide document vectors only once.

## 1.3 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) is an architecture that combines retrieval and generation. It allows a model to access external knowledge sources, providing up-to-date information and improving factual accuracy. RAG also helps to reduce hallucinations commonly observed in large language models, making it suitable for NLP tasks (Lewis et al., 2020). Retrieval layer recognize relevant data to support in generating adequate responses. On the other hand there is generative component, which is based on transformer architecture. Generated response combines retrieved information with model output.

## 1.4 Polish Large Language Model

`PLLuM` is a Polish language model (PLLuM Consortium, 2023) developed by a consortium of Polish research institutions, specifically designed for tasks related to the Polish language, such as text generation and comprehension. It has been trained for linguistic nuances to handle grammar and idioms correctly, not forgetting about cultural and contextual details that also have been taken into account. Its specifications make it suitable for implementations aimed at Polish users. Integrating `PLLuM` seems ideal for a conversational recommendation system for Łódź restaurants. Currently, this model is being utilized in various fields, such as `PLLuM Medycyna` and `PLLuM Prawo`, which have been specifically adapted for this area.

## 1.5 Large Language Models in Recommendations

Usage of LLMs in recommendation systems is increasing due to power of context-based understanding (Wu and al., 2023) and coherent responses generation. They excel in capturing specific pieces of information they need. This allows them to expand the context around the user's preferences and finally tailor the response accordingly. Thereby, combination of LLM and RAG can provide both actual reasoning and factual legitimacy.

# 2 Architecture of the system

Architecture of the entire pipeline particularly emphasizes was placed on ensuring ease of use, high response accuracy, adapting to user needs, and improving the reliability of facts. Moreover, various model and technique solutions were considered to achieve the best performance.

## 2.1 Application of LLM

The role of the generative language model is minimized to reduce hallucinations. Its main tasks include location extraction, query expansion, and response formulation. The model also focuses on detecting user intent in an initiated conversation. It distinguishes between casual conversation and a recommendation request.

## 2.2 User Location Detection Module

The location detection system uses a hybrid approach that combines semantic analysis with geocoding. The solution is based on both a language model (PLLuM) and classical Named Entity Recognition (NER) using the spaCy library.

### 2.2.1 Intent Analysis and Query Normalization

Initially, the query is processed by the `CyfraGovPL/PLLuM-12B-nc-chat` polish language model, which interprets the user's intent. The model acts as a 'translator' from informal, colloquial language to formal language, taking into account the appropriate declension form, which is then recognizable by subsequent tools in this layer. This is difficult to achieve with classical algorithms, which is why LLM performs excellently in this area. Desired location name transformations for which analysis is conducted:

- "koło Manu" → "Manufaktura"
- "przy Polibudzie" → "Politechnika Łódzka"

Table 2.1: Analysis of the conversion of location names into direct form

| Input Query | Detected Location | Correctness |
|---|---|---|
| koło Manu | Manufaktura | YES |
| przy Polibudzie | Politechnika Łódzka | YES |
| na pietrynie | Ulica Piotrkowska | YES |
| w okolicach fabrycznego | Fabryczny | YES |
| na lumumbowie | Lumumbowo | YES |
| koło atlas areny | Atlas Arena | YES |
| na offie | — | NO |
| przy ec1 | — | NO |
| blisko galerii | galerii | NO |

Table 2.1 presents exemplary results of location normalization and detection for informal user queries referring to gastronomic places. The table illustrates that colloquial expressions commonly used by local residents can be successfully mapped to their formal location names. Most of the tested phrases were correctly interpreted. However, some cases (like "na offie", "przy ec1") that resulted in failed detections emphasize the limitations when the query is less descriptive.

### 2.2.2 Proper Name Extraction using spaCy

Next step after the name is normalized to desired form. Another model is utilized as a supporting mechanism for the PLLuM and for direct text processing - spaCy library, more specifically the `pl_core_news_lg` model. The input text is preprocessed through conversion to Title Case format to increase the effectiveness of proper name detection by the NER model. The model searches for entities belonging to specific semantic categories: `placeName` (place names), `geogName` (geographic names), `orgName` (organization and landmark names), and `roadName` (street names). In the context of detecting proper place names for restaurants, this can be very misleading for the detection layer. A user describing the type of cuisine might include country or region names in their query. Therefore, a 'blacklist" mechanism has been implemented to reject false positives. The list contains terms that the NER model might incorrectly classify as locations, general qualitative descriptors (e.g., dobra", smaczna"), and names of establishment types (e.g., restauracja", bar", kawiarnia"). This filtering meaningly improves system precision by eliminating information noise.

### 2.2.3 Geocoding using Nominatim API

The detected and verified location name is passed to a geocoding service to transform the textual name into geographic coordinates. The system uses the Nominatim API based on OpenStreetMap data, handled by the `geopy` library. Nominatim provides free access to spatial data for the entire world. Another aspect that improves result precision and avoids name ambiguity across different locations is the suffix added to each query that narrows the search area to the city of Łódź: "{detected_location}, Łódź, Polska". This mechanism enables the elimination of errors resulting from the existence of identical street names in different cities. The geocoding layer returns a pair of geographic coordinates in the format of latitude and longitude. These coordinates then serve as a reference point for calculating the distance between the user's location and recommended restaurants, which is one of the components when calculating the final result.

Table 2.2: Location detection performance summary

| Detection Result | Count | Percentage |
|---|---|---|
| Correctly detected | 32 | 68.1% |
| Not detected | 13 | 27.7% |
| Incorrectly detected | 2 | 4.3% |
| **Total queries tested** | **47** | **100%** |

As shown in Table 2.2, the hybrid detection approach achieved a success rate of 68.1% across all 47 test queries. Thirteen queries were not detected by the PLLuM model, while two cases were either not recognized by spaCy or incorrectly classified. This demonstrates the effectiveness of a hybrid pipeline that integrates large language models with statistical named entity recognition and deterministic geocoding services.

## 2.3 Aggregation Strategies

Before the embeddings are created, the descriptions of POI must be properly prepared to reflect their actual state. Since there are many methods some of them are tested first, to ensure the final result is accurate.

### 2.3.1 Embedding aggregation

Since the model returns a sequence of vectors where each vector represents a single token, and the vector database discussed in later requires a single fixed-size vector representing the entire text, it is necessary to apply an aggregation mechanism. This mechanism reduces the sequence to a single vector representing the entire text chunk. The method used here is pooling, which through a mathematical operation reduces matrices of shape [N, 1024] to a single vector of shape [1, 1024].

For this purpose, two pooling approaches were tested: **mean pooling** and **CLS pooling**.

### 2.3.2 Mean pooling

Mean pooling computes the arithmetic mean of embeddings of all tokens in the sequence. Each token has an equal proportional influence on the final representation.

Mean pooling is simple and computationally efficient, it can result in the loss of important details and nuances. Mean pooling ignores the order and context of chunks, which can be problematic if the sequence of information is important.

### 2.3.3 CLS pooling

The second approach is CLS pooling. It utilizes a special **classification token** <s> (equivalent to `[CLS]` in BERT models). During pretraining, the RoBERTa model was trained so that the

embedding of this token contains a compressed representation of the meaning of the entire sequence.

In the conducted experiments, CLS pooling consistently achieved higher cosine similarity values compared to mean pooling, regardless of the method used to construct textual descriptions.

Table 2.3: Comparison of pooling methods based on average cosine similarity scores

| Pooling Method | Avg. Similarity Score | Median Similarity Score |
|---|---|---|
| CLS Pooling | $\sim 0.78$ | $\sim 0.78$ |
| Mean Pooling | $\sim 0.72$ | $\sim 0.71$ |

As demonstrated in Table 2.3, embeddings created using CLS pooling consistently achieved higher similarity scores compared to mean pooling across all query types.

Based on these results, higher similarity scores in the CLS method correlated with better semantic matching of the retrieved results to the user's intent, providing a clearer separation between relevant and irrelevant documents. However, even a really high similarity value may not guarantee the factual correctness of the result.

## 2.4 Construction of Textual Representations

An important factor affecting the quality of semantic search is the way textual descriptions from which embeddings are created are constructed. Two following approaches were implemented. Since CLS pooling uses the first classification token as a representation of the entire sentence, it performs most effectively for concise, unambiguous, and specific descriptions. A hard limit of 400 characters was applied to ensure that the text fits well within the model's 512-token limit, leaving ample space for special tokens and preventing truncation of key information.

An important element in creating appropriate descriptive structures was the extraction of features that are potentially most frequently used by users when describing a place. A hierarchy of categories was assigned based on popularity. Parameters such as cuisine type, specialization, or atmosphere were given the highest priority. In contrast to characteristics such as Wi-Fi availability or payment methods were placed at the bottom of the list. Thanks to this approach, for places with very rich datasets include only the most important factors.

```
PRIORITY = {
    "types": 1,
    "atmosphere": 4,
    "popular_for": 7,
    "offerings": 2,
    "amenities": 10,
    "service_options": 8,
    "specials": 3,
```

```
    "crowd": 6,
    "accessibility": 5,
    "parking": 9,
    "payments": 11,
}
```

### 2.4.1 Full descriptive context

The first approach is characterized by descriptions in the form of complete sentences in natural language. The description contains information about the place name, type, atmosphere, offerings, amenities, and additional features.

Example description structure:

```
The object named X is
A place of type: restaurant.
The atmosphere is described as: cozy.
It is particularly popular for: dinners.
The offering includes: pizza, pasta.
```

This approach is more readable, however, it contains a large number of connecting words and elements that are not essential for embedding creation. These increase semantic noise, resulting in a more verbose description with fewer meaningful information.

### 2.4.2 Keyword-based descriptions

The second approach is based on short, condensed descriptions in a "key: value" format. The specifications contain only information relevant from the perspective of semantic search. A heuristic fallback mechanism was implemented to infer the place type directly from its name (e.g., detecting 'pizza' or 'sushi' strings) when explicit category data is missing.

Example structure:

```
Place type: pizzeria, restaurant.
Offerings: pizza, pasta, salads.
Place character: family-friendly, calm.
Additional features: garden, wheelchair access.
```

The keywords-only approach proved particularly effective for queries about specific attributes (e.g., "vegan options", "wheelchair access). This is because the condensed format maximizes keyword density while eliminating semantic noise.

tutaj tagbele z porownanie

### 2.4.3 Query Expansion

Having selected the best-performing embedding model, pooling method, and textual description construction strategy, a key aspect becomes the analysis of how user queries are represented. It should be noted that all embeddings stored in the vector database are constructed in a schematic and structured manner, in contrast to user queries formulated in natural language without a predefined structure. To reduce this discrepancy, a Query Expansion technique was applied.

The system employs the HyDE (Hypothetical Document Embeddings) method to strengthen the similarity between the user query and the records stored in the vector database. The user query is passed to the language model `CyfraGovPL/PLLuM-12B-nc-chat`, which generates a description of a hypothetical restaurant based on the information contained in the query. A crucial aspect of this process is preserving the same structure used when creating textual chunks in the database ("key: value").

For example, the query "where can I eat good pizza?" is expanded by the LLM into the form: "Name: Pizzeria. Place type: Italian, Pizzeria. Offerings: Pizza. Character: Friendly atmosphere." This format corresponds to the Keywords-Only structure used in the database, which significantly increases semantic alignment.

The generated hypothetical description is then transformed into an embedding vector and used for similarity search in the database instead of the original user query. This technique enables the system to achieve the highest performance in vector similarity retrieval

| Avg. similar(Baseline) | Avg. similarity (HyDE) | Avg. Improvement ($\Delta$) |
|---|---|---|
| 0.7535 | 0.8252 | +0.0717 |

Table 2.4: Average cosine similarity scores for queries with and without the HyDE method

Unfortunately, there's a small drawback to this approach. LLM is used to construct context for the query. If the model is hallucinating, the generated description may be inaccurate, negatively impacting search results.

## 2.5 Comparative Analysis of Representation Strategies

To select the best out of all tested configuration for the recommendation system, a comparative analysis of four different combinations was conducted. The approaches differ in the text representation method (full context vs. keywords) and the aggregation method (CLS pooling vs. mean pooling).

The comparison was based on the cosine similarity scores returned by the model for twenty different test queries. Figure 2.1 presents the results for selected queries representing different user intents, such as specific attributes, various diets, or cuisine types.

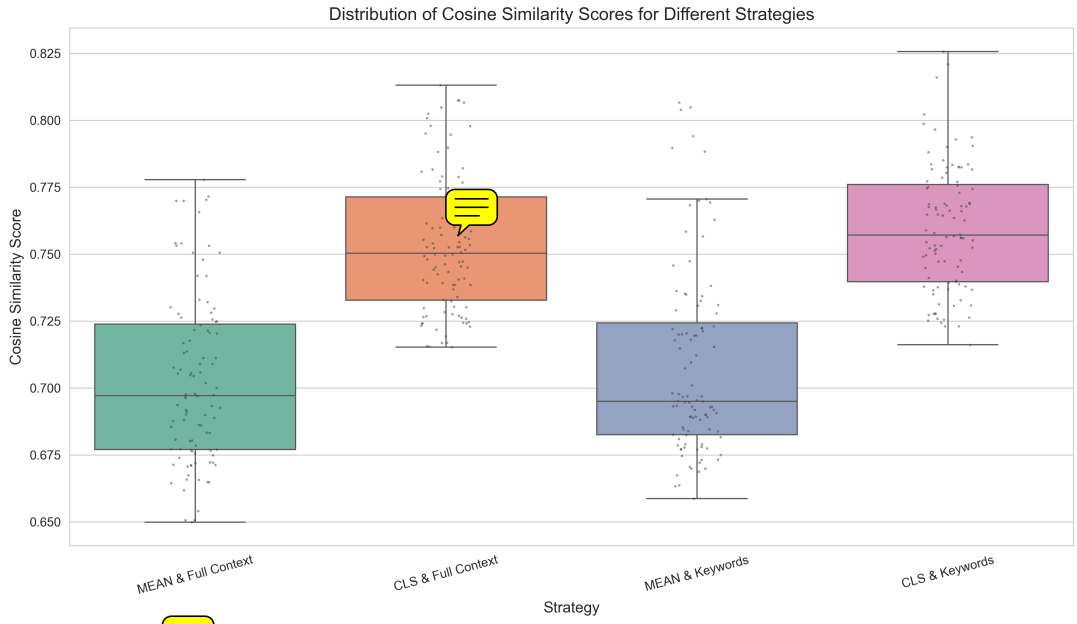


Fig. 2.1: Illustrates boxplot with the distribution of cosine similarity scores

While the difference in context creation is negligible, CLS pooling stands out with significantly better results. This is due to the architecture of the RoBERTa model, in which the classification token is better trained to capturing the semantic meaning of the entire description.

Across all test cases, the CLS pooling method consistently outperformed mean pooling, being better suited for retrieval tasks than averaging method. Which leads to equalization of all tokens in the vector (those less and those more important). It confirms that the adaptation of the model to this method is clearly visible. The next positive aspect of this approach is significantly decreased percentage of "semantic noises" created from extra not really meaningful words from the second method.

Based on this analysis, the **CLS Pooling with Keywords-based descriptions** strategy was selected as the primary method for the CRS system, as it offers the best balance between precision for specific attributes and overall semantic matching quality.

## 2.6 Vector Database and Retrieval

This section presents the dual nature of the retrieval system approach and the benefits of using each layer. The focus was also on how to calculate the similarity of the final results.

### ~~2.6.1~~ FAISS as a Vector Database

To enable efficient searching within the generated vector space, the **FAISS** (Facebook AI Similarity Search) library was utilized. FAISS is an open-source library designed for high-performance similarity search and clustering of dense vectors, designed by Meta Research.

Unlike cloud-based solutions, FAISS operates locally, providing full control over index configuration and memory management. This characteristic makes it particularly suitable for research projects and applications where data privacy and low latency are critical, without the need for external infrastructure. In this case that is not that noticeable due to the small database (over 200 records).

### 2.6.2 Bi-Encoder Layer

The system's task is to implement the dense retrieval strategy, using Bi-Encoder architecture. Chosen embedding model `sdadas/mmlw-retrieval-roberta-large`, a variant of RoBERTa intended for retrieval tasks, is responsible for mapping both stored restaurant descriptions and user queries into a shared vector space of 1024 dimensions. Each input whether it is document or a query is processed independently by the same model, producing a fixed-size vector representation without any direct interaction between them two.

Bi-Encoder stands out for its ability to encode independently. It allows document vectors to be precomputed once and stored in the vector database in advance. When user makes interaction, only the user's query needs to be encoded, after which the resulting vector is compared against all stored document vectors using a similarity metric. This design makes the Bi-Encoder highly efficient for large-scale retrieval cases.

After sequences are tokenized using Byte-Pair Encoding (BPE) algorithm, the underlying Transformer architecture utilizes a Self-Attention mechanism to build comprehensive representations. Unlike traditional sequential models that process text word-by-word, self-attention enables parallel processing of all tokens simultaneously. This allows the model to capture the dependencies between tokens and distinguish the relationships between all inputs before compressing them into a single vector.

However, independent encoding comes with some limitations. Because the query and document are never directly compared at the token level during encoding, the Bi-Encoder may miss subtle relevance details, such as differentiating the meaning of a word depending on the context or irrelevant correlations between similar fragments. While this is sufficient for the initial retrieval stage, where speed and scalability are the priority. The efficiency can be further increased by introducing a reranking layer to refine the obtained results.

### 2.6.3 Cross-Encoder Reranker

To further improve the quality of retrievals, a Cross-Encoder model is implemented as a reranker after the initial Bi-Encoder search. The Cross-Encoder characterizes slightly different. Instead of encoding the query and document independently, it takes both simultaneously, concatenates them, and passes to Transformer network.operates on a fundamentally different principle: instead of encoding the query and document independently, it takes both inputs simultaneously, concatenates them, and passes the pair through a single Transformer network.

This allows the model to perform self-attention directly on tokens, which then outputs a single score between 0 and 1 indicating how relevant the document is for the given query.

The advantage of Cross-Encoders is the higher performance. Elaborating plenty of pairs (query and document description) would be really slow. Hence, the Bi-Encoder is used to create a initial set of feasible candidates which are then re-ranked by the Cross-Encoder, prioritizing top-k results. Using only one encoder would result in deficiencies in one of the domains. Bi-encoding alone would lose accuracy. On the other hand, cross encoding is much slower.

Consequently, the optimal approach is to employ both encoders complementary. Treating the Bi-Encoder as a fast first-pass filter that efficiently retrieves a broad set of candidates, and the Cross-Encoder as a precise second-stage reranker that scores only this narrowed results (Reimers and Gurevych, 2019).

### 2.6.4 Similarity Metrics and Normalization

The core of the retrieval process involves comparing the query vector $q$ with document vectors $d_i$. The metric used to determine similarity is **Cosine Similarity**, which measures the cosine of the angle between two vectors. Mathematically, it is defined as:

$$\text{similarity}(q, d) = \frac{q \cdot d}{\|q\|\|d\|} = \frac{\sum_{i=1}^{n} q_i d_i}{\sqrt{\sum_{i=1}^{n} q_i^2}\sqrt{\sum_{i=1}^{n} d_i^2}} \tag{2.1}$$

In the implementation, the `IndexFlatIP` (Inner Product) index from FAISS is used, to ensure that the inner product calculation is mathematically equivalent to cosine similarity, a normalization step is applied - `L2 Normalization`. All vectors after being processed are of length 1 ($\|v\| = 1$). Under this condition, the denominator in the formula becomes 1, and the similarity simplifies to the dot product of both queries:

$$q_{\text{norm}} \cdot d_{\text{norm}} = \cos(\theta) \tag{2.2}$$

This optimization allows for extremely fast matrix multiplication operations while preserving the semantic properties of cosine similarity.

### 2.6.5 Embedding Model Selection and Evaluation

In the process of selecting the optimal embedding model, a comparative evaluation of three different solutions dedicated to the Polish language was conducted for:

- `sdadas/mmlw-retrieval-roberta-large`
- `sdadas/mmlw-retrieval-roberta-large-v2`

- `sdadas/stella-pl-retrieval`

The experiments involved retrieval effectiveness tests on a prepared reference dataset. Although the RoBERTa-large-v2 model achieved the highest raw cosine similarity values, factual verification of the results revealed a tendency towards overconfidence. It assigned high scores to thematically unrelated documents, which hindered effective filtering. Analysis of the RoBERTa-large results showed that it offers the best balance between precision and stability. On the other hand, the Stella model, when using the keyword strategy, exhibited a critical error, returning an identical set of results regardless of the query content. Based on the performance and reliability of models , the `sdadas/mmlw-retrieval-roberta-large` model was selected for the final system implementation.

Table 2.5: Aggregated summary of average cosine similarity scores for the tested embedding models

| **Embedding Model** | **Avg. Similarity Score** |
|---|---|
| RoBERTa-large | 0.731 |
| Stella | 0.601 |
| RoBERTa-large-v2 | 0.762 |

## 2.7 Architecture Summary

Each component of the system architecture was carefully selected based on numerous tests and comparative analyses aimed at optimizing the entire pipeline. The applied methods for generating textual representations, including the use of descriptions based on keywords and CLS pooling, proved to be best suited for retrieval techniques based on vector similarity.

The system architecture is built upon a multi-stage retrieval process that combines fast similarity search using a local FAISS index with a more precise re-ranking layer based on Cross-Encoder models. Additionally, the application of the HyDE technique enabled query expansion and improved semantic matching, particularly in the case of short or ambiguous user queries.

An important element of the system is also the location detection mechanism, which employs an approach combining language models with classical named entity recognition methods, enabling more accurate filtering of results.The entire architecture is designed to be scalable while striking a balance between achieving high semantic accuracy and maintaining realistic, authentic responses.

# 3 Implementation of the conversational recommendation system

This chapter illustrates how the system actually works, why particular solutions were chosen and whar is it background flow of the pipeline.

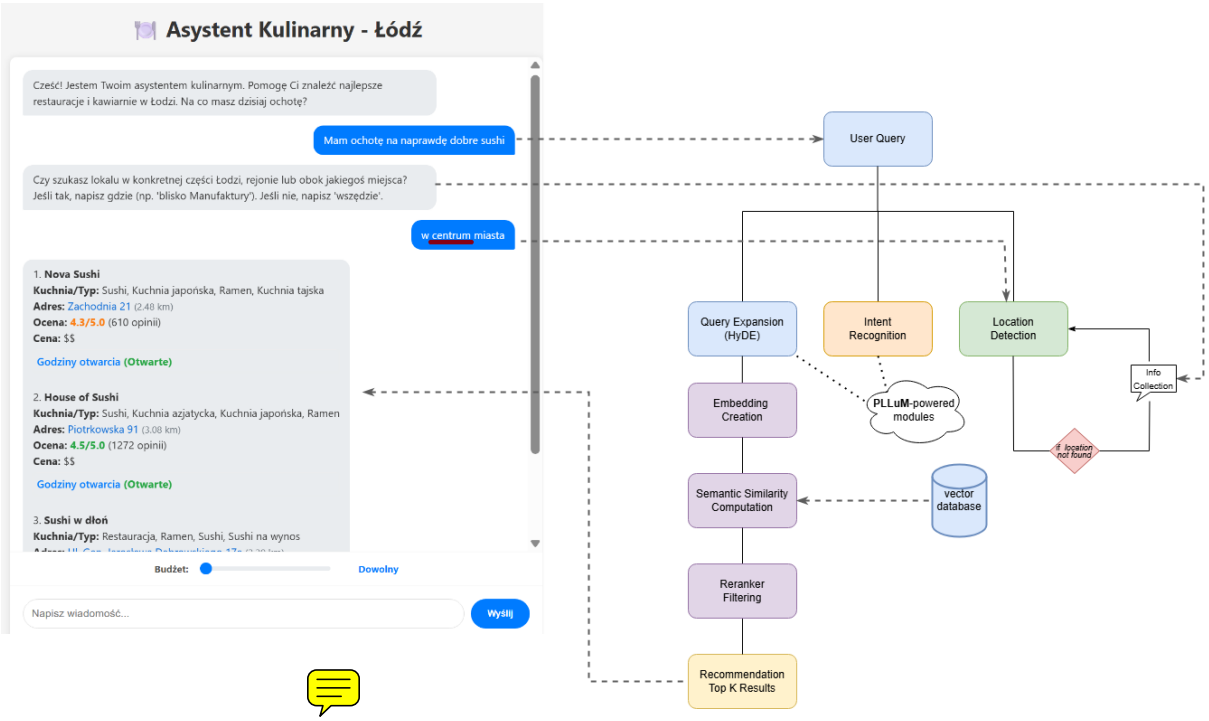


Fig. 3.1: General architecture overview and data flow

Figure 3.1 presents the complete data flow of the CRS. The interaction is initiated by a query from the user. Unlike simple chatbots, the system does not pass the query directly to the generator. Instead, it employs a multi-stage analysis pipeline. First, a PLLuM acts as a reasoning engine to classify the user's intent and extract structured parameters (cuisine type, location) into a JSON format. Simultaneously, the system tries to detect the user's location and expands the query using the HyDE technique. The transformed query is then used to retrieve candidates from the FAISS vector database. The final selection is not based solely on semantic similarity but utilizes a weighted scoring algorithm that considers the Cross-Encoder score, Google Maps ratings, and distance.

$$\text{Score}_{final} = 0.35 \cdot S + 0.35 \cdot Ra + 0.10 \cdot Re + 0.20 \cdot D \tag{3.1}$$

$S$ – semantic similarity score (after re-ranker usage),
$Ra$ – normalized average google rating,
$Re$ – normalized number of google reviews,
$D$ – normalized distance from the user.

Although the semantic factor accounts for only 35% of the final score, this weighting strategy is a deliberate design choice. It accepts a trade-off in semantic similarity to prioritize attributes of higher practical utility, such as proximity and restaurant ratings. Since the retrieval stage ensures that all candidates are already contextually relevant, this approach optimizes the system for user satisfaction in a real-world gastronomic context.

## 3.1 Architecture and Data Flow

Figure 3.2 presents the general architecture and data flow of the proposed conversational recommendation system. The system follows a modular pipeline design, where each component is responsible for a defined task, which improves both scalability and maintainability.

The process begins with the acquisition of data about gastronomic venues using *OverPass-API* and *SerpAPI*. The collected data are structured, cleaned, and divided into textual chunks containing key information. Each chunk is converted into a dense vector representation using a bi-encoder model and stored in a *FAISS* index for efficient similarity search.

During user interaction, the input query is analyzed and converted into an embedding using the same encoding mechanism. Semantic retrieval is performed by comparing the query vector with stored vectors using cosine similarity, selecting the most relevant candidates. The top retrieved results are subsequently processed by a re-ranking model based on a cross-encoder architecture, which evaluates the semantic relevance more precisely.

Finally, the selected results are passed to the response generation module, where a large language model just generates appropriately formulated response.

nie wiem, czy nie lepiej tego typu opisy umieścić wcześniej jako wprowadzenie do rozwiązań szczegółowych

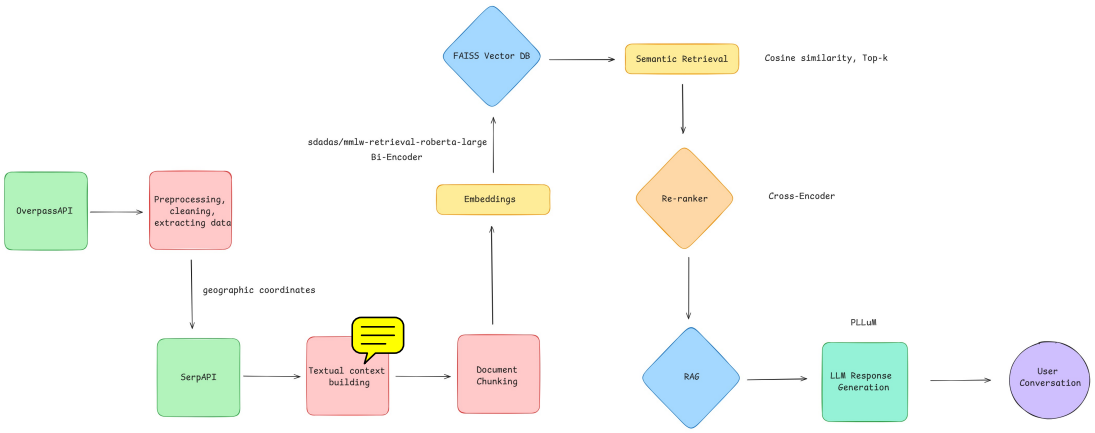


Fig. 3.2: Comprehensive modular architecture of the system.

## 3.2 Design Decisions and System Rationale

Is it necessary to use a combination of a bi-encoder and a cross-encoder for such a small dataset? In the case of approximately 200 records, only a cross-encoder architecture would indeed not be time-consuming. However, the system was designed with a strong emphasis on scalability and the ability to handle large datasets. Relying just on a cross-encoder would

significantly limit the application's usability. Therefore, the system achieves a balance between high precision and low latency, allowing for database growth without potential performance degradation.

## 3.3 System Limitations

Despite its advantages, the system has certain limitations. The quality of recommendations is strictly dependent on the validity of the external data. Since the system operates on a snapshot of data (fetched via API), information such as "current opening hours" or "temporary closures" relies on the last update. Additionally, the hybrid location module depends on external geocoding providers (Nominatim) 2.2, which may occasionally fail to resolve colloquial location names without sophisticated context.

rozdział miał dotyczyć implementacji a de facto dotyczył architektury systemu. Implementacja to stos technologiczny, struktura klas, itp.

# 4 Evaluation Metrics

To assess the quality of the recommender system, standard metrics commonly used in that field were employed (Manning, Raghavan, and Schütze, 2008; Gunawardana and Shani, 2009). The evaluation is conducted on the top-k results returned by the model.

## 4.1 Precision@K

Precision measures the number of relevant items among the first $K$ results returned by the system. This metric allows evaluating the level of noise in search and recommendation results.

$$\text{Precision@K} = \frac{|Rel_K|}{K} \tag{4.1}$$

where:

- $|Rel_K|$ – the number of relevant items in the first $K$ results,
- $K$ – the number of considered results (in this case, $K = 5$).

A high *Precision@K* value indicates that the system provides the user with accurate recommendations, which affects the perceived quality of the system and user trust. As indicated by studies (Chen and Pu, 2014), key factors influencing the adoption of conversational recommender systems include perceived recommendation quality, usefulness, ease of use, trust, and enjoyment of interaction (Baizal, Murti, and Adiwijaya, 2017a; Baizal, Murti, and Adiwijaya, 2017b), as well as a sense of control and user satisfaction.

## 4.2 Recall@K

Recall (Recall@K), also called completeness, measures the system's ability to find relevant items among all relevant items available in the dataset. This metric shows what portion of all possible correct results the system included among the top $K$ recommendations.

$$\text{Recall@K} = \frac{|Rel_K|}{|G_q|} \tag{4.2}$$

where:

- $|Rel_K|$ – the number of relevant items in the first $5$ results,
- $|G_q|$ – the total number of relevant items in the dataset for query $q$.

In the context of recommendation systems, users often expect a single accurate suggestion rather than many overlapping results. A commonly used simplified measure of recall is *Hit Rate@K*. This metric determines the proportion of queries for which at least one relevant item

appears among the top $5$ recommendations, indicating whether any relevant result was found rather than counting the total number of relevant results.

## 4.3 Mean Reciprocal Rank

Mean Reciprocal Rank (MRR) is a metric that considers the order of results. It identifies the highest-ranked result that is considered relevant.

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i} \tag{4.3}$$

where:

- $|Q|$ – the number of test queries (here, 20),
- $rank_i$ – the position of the first relevant item in the results list for the $i$-th query; if no relevant result is found, the value is set to $0$.

Experiments were conducted to evaluate the effectiveness of the search module. A test set of 20 queries was defined, potentially reflecting typical user formulations. Example queries: "Gdzie zjem dobrą pizzę na grubym cieście?", "Szukam autentycznej kuchni azjatyckiej". For each query, a reference set (Ground Truth) was created, containing a list of candidates (5 for each query) best matching the query, based on restaurant metadata (cuisine type, offerings, description). This process allowed determining the accuracy of the search results. A document is considered appropriate for a given query if the returned result matches an item defined in the reference set. To identify the optimal system configuration, the performance of different text representation strategies (pooling: MEAN vs CLS) and input preparation methods (full context vs keywords) was compared.

Table 4.1: Comparison of evaluation results for different strategies

| Method | Hit Rate@5 | MRR | Precision@5 |
|---|---|---|---|
| MEAN (full context) | 0.80 | 0.5600 | 0.45 |
| CLS (full context) | 0.85 | 0.6308 | 0.41 |
| MEAN (keywords) | 0.75 | 0.5917 | 0.39 |
| CLS (keywords) | **0.90** | **0.7625** | **0.49** |

The best results were achieved using the CLS strategy on data limited to keywords. An MRR of *0.7625* indicates that, in most cases, a relevant result appeared as the first returned item (highest semantic match). Moreover, the very high Hit Rate of 0.9 shows that in 90% of cases, a result from the reference set was returned. The obtained results indicate high accuracy and relevance of the recommendations provided.

## 4.4 Metrics Analysis

In the conducted measurements, the reference set (ground truth) was created from sample queries. Due to the lack of a strictly defined, unambiguous measure, the defined elements may introduce minor biases. Another factor contributing to metric drops is limitations arising from available information. Publicly available information about establishments varies significantly in detail for locations, some descriptions are extensive and cover many aspects, whereas for others they are sparse and generic. Similar discrepancies also occur in the distribution of object categories, where types such as pizzerias or Polish cuisine restaurants appear much more frequently than, for example, Georgian cuisine.

## Conclusions

Considering the available project resources, including limited access to external data sources and hardware constraints, the results achieved are satisfactory. The conducted tests and experiments confirm that the adopted architecture ensures accurate matching of user intent with relevant food establishments while maintaining high quality of the generated responses. But the modular design of the system still leaves space for future development and optimization.

**Database Standardization and Enrichment:** A key observation during the evaluation was that the retrieval module tends to favor places with richer textual descriptions. To eliminate this bias, the knowledge base can be even better normalized and supplemented by missing categories. Thus, expanding the database itself would allow for much more diverse and detailed recommendations

**Hardware Infrastructure Optimization:** Zastosowanie wydajniejszej infrastruktury sprzętowej pozwoliłoby na redukcję czasu odpowiedzi systemu oraz optymalizację działania etapów pipeline, które wymagają charakteryzują się większą złożonością obliczeniową.

**Embedding Model Research:** Nie sposób wybrac "ten najlepszy" model embeddigowy gdyz jest ich mnóstwo. Także i tutaj zostaje pole do poprawy, znajdując bardziej kompatybilny i wydajnosciowy model retrievalowy.

## Summary

The primary objective of this thesis is to design and implement a conversational recommendation system for gastronomic places in Łódź, which has been successfully realized. The developed solution utilizes a modular RAG architecture, ensuring high scalability. Proposed design not only facilitates future improvements but also easily allows for expansion to other geographic locations or domains. The solution have a real impact on the quality of life for Łódź residents. Due to convenience of use it may save time potentially wasted on searching for the desired cuisine via internet manually. One of the crucial facet is the integration of the Polish language model - PLLuM, for the text-operation tasks where it is used. This choice ensures correct handling of the nuances of the Polish language, while simultaneously supporting national technological initiative. Furthermore, the project demonstrates that effective recommendation system requires a balanced approach. By limiting the generative model's role to a reasoning engine, responsible for intent detection, query expansion and location extraction. Moreover, it relies on an external vector database for information retrieval, ensuring that all recommendations are strictly grounded in verified data. Consequently, the possibility of returning hallucinated results is eliminated to almost zero, as the system is technically constrained to retrieve only those items present in the structured dataset. The combination of a powerful language model with an adjusted knowledge base is an increasingly popular solution, mainly due to its universality across various domains.

# List of Tables

# List of Figures

# List of Equations

## References


Baizal, Abdurahman, Z. K. Murti, and Y. R. Adiwijaya (2017a). "Evaluating Functional Requirements-Based Compound Recommender Systems". In: *Proceedings of the International Conference on Advanced Informatics: Concepts, Theory and Applications (ICAICTA)*, pp. 1–6.

– (2017b). "User Experience Evaluation of Conversational Recommender Systems". In: *International Journal of Electrical and Computer Engineering* 7.5, pp. 2789–2796.

Chen, Li and Pearl Pu (2014). "Experiments on User Experiences with Recommender Interfaces". In: *Behaviour & Information Technology* 33.9, pp. 871–885.

Dong, Z. et al. (2024). "MuseChat: A Conversational Music Recommendation System for Videos". In: *CVPR*.

Fang, Hui et al. (2022). "DTCRSKG: A Deep Travel Conversational Recommender System Incorporating Knowledge Graph". In: *Mathematics* 10.9, p. 1402. DOI: 10.3390/math10091402.

Gunawardana, Asela and Guy Shani (2009). "A Survey of Accuracy Evaluation Metrics of Recommendation Tasks". In: *Journal of Machine Learning Research* 10, pp. 2935–2962.

Lewis, Patrick et al. (2020). "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks". In: *NeurIPS* 33, pp. 9459–9474.

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze (2008). *Introduction to Information Retrieval*. Cambridge University Press.

PLLuM Consortium (2023). *Polish Large Language Model (PLLuM)*. https://pllum.org.pl.

Reimers, Nils and Iryna Gurevych (2019). "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Proceedings of EMNLP*, pp. 3982–3992.

Vaswani, Ashish et al. (2017). "Attention Is All You Need". In: *Advances in Neural Information Processing Systems*, pp. 5998–6008.

Wu, Shuo and et al. (2023). "A Survey on Large Language Models for Recommendation". In: *AI Review Journal* 15.1, pp. 1–20.