University of Padova

Department of Department of Mathematics

Master Thesis in Master in Data Science

Integrating subjectivity into relevance measurement for conversational information retrieval

Supervisor
Professor Michele Rossi
University of Padova

Co-supervisor
Sébastien Fournier, Ph.D.
Adrian-Gabriel Chifu, Ph.D.
Laboratoire d'Informatique et des Systèmes (LIS),
Aix-Marseille Université

Master Candidate
Arina Gepalova

Academic Year
2023-2024

Abstract

In today's fast-changing world, people want better and more advanced ways to search for information. Instead of traditional web searches, users are moving toward conversational interfaces that allow natural interactions. To address this need, Conversational Information Retrieval (CIR) was created to provide accurate and relevant results using Natural Language Processing to simulate the natural search style that humans perform.

Large language models like GPT-3 and LLaMA-3 have made significant improvements in understanding and producing human-like text. However, they often produce outdated or general responses rather than ones that are relevant to a given situation. This problem becomes especially noticeable in domains where users ask open-ended questions to express their preferences, emotions, and opinions, such as when recommending books. As a result, CIR systems must be able to process both the explicit content of questions and their underlying characteristics and sentiments.

Aspect-Based Sentiment Analysis (ABSA) is a key method in this context. ABSA helps extract specific aspects from user queries and determines the sentiment associated with each aspect. This detailed analysis is especially important in situations where user preferences and feelings regarding certain features need to be examined.

This study proposes a way to improve CIR systems for book recommendations by adding subjectivity to the search for relevant answers. The project involves extracting aspects and sentiments from user queries and retrieving relevant books using a hybrid retrieval method that combines semantic and full-text search. By considering both semantic similarity and subjective elements of queries, the system aims to standardize document retrieval and center the process on the needs of the person seeking help rather than on the subjective opinions of others.

Contents

Listing of figures

Listing of tables

Listing of acronyms

ABSA ..... Aspect-based Sentiment Analysis

ACD ..... Aspect Category Detection

ATE ..... Aspect Terms Extraction

ATC ..... Aspect Terms Categorization

ATSC ..... Aspect Terms Sentiment Classification

CLEF ..... Conference and Labs of the Evaluation Forum

INEX ..... Initiative for the Evaluation of XML Retrieval

IR ..... Information Retrieval

LT ..... LibraryThing

OTE ..... Opinion Terms Extraction

RRF ..... Reciprocal Rank Fusion

SA ..... Sentiment Analysis

SBS ..... Social Book Search

1 Introduction

Today, in view of such changes in society, the demand for better and more sophisticated means of information search is increasing. The shift from the primitive web search to Web3 UIs such as keyword search or conversational searching is due to natural ways to engage with the systems. The improvement that has embraced the field of dialogue systems is referred to as Conversational Information Retrieval (CIR). They expect to provide sufficient and correct answers in regard to the terms the user is searching using NLP which imitates a natural conversation flow.

The latest models of CIR systems are LLMs such as GPT-3 from OpenAI or the continuation of this LLM – LLaMA3. These models which have learned from large amounts of textual data are capable of understanding and generating human-like text that can be used for conversational agents. However, as to the problem of how the best response is to be made, the balance remains tipped more towards making general and therefore contingent and perforce outmoded responses as opposed to context-sensitive and current ones. In other words, the user queries are semantically unrestricted and are developed from the point of view of the user, especially when it comes to such designated domains as book recommendations, for instance. In fact, people in most cases use CIR to express their preferences, sentiments and opinions. The system therefore not only searches for the query statements but also seeks to look for the inherent attributes and sentiments of the queries.

Aspect-based sentiment analysis (ABSA) has emerged as a crucial technique in this context. ABSA allows for the extraction of specific aspects from user queries and the determination of the sentiment associated with each aspect. This fine-grained analysis is particularly useful in

domains where users' preferences and sentiments towards certain features or characteristics are paramount. By identifying and analyzing these aspects, CIR systems can offer more tailored and contextually relevant results.

The present research proposes a comprehensive approach to enhancing CIR systems for book recommendations by integrating ABSA with advanced retrieval techniques. The project involves the extraction of aspects and sentiments from users' queries, the retrieval of relevant books using a hybrid search mechanism combining semantic and full-text search.

1.1 Motivation

The motivation for this research stems from the increasing complexity and subjectivity of user queries in conversational settings. Traditional retrieval methods often fall short in addressing the nuanced needs of users, leading to irrelevant or unsatisfactory recommendations. By leveraging ABSA and hybrid search techniques, this research aims to bridge this gap, providing a more personalized and effective user experience.

Additionally, the motivation is the search for a universal and standardized method of information retrieval that is based not only on the semantic similarity between the query and relevant documents but also on the analysis of subjectivity in relation to various aspects. As demonstrated in the example above, the LibraryThing forum serves as a space for bringing together people with diverse interests and knowledge. Users recommend books based on their personal experiences, relying on books they have likely read themselves or were recommended by others. Such recommendations are subjective in nature. Implementing a system where the search is conducted according to specific rules helps to standardize document retrieval, limit recommendations to relevant information (in this case, only books available in the catalog), and rely not on the subjective opinion of the person recommending the books, but on the person seeking assistance.

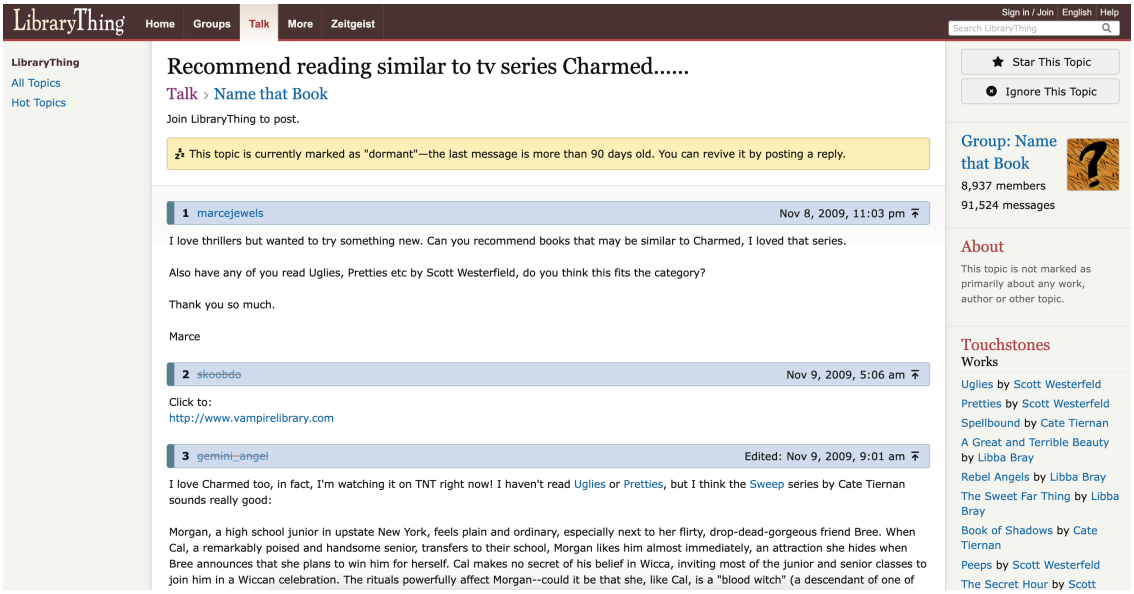

Figure 1.1: Example of a user request on the LT forum

Another motivation for this research is the rapidly changing trends in user behavior when interacting with search systems. Just a few years ago, a standard search query typically did not exceed one sentence. However, with the rise of Large Language Models (LLMs) and various conversational agents (also known as chatbots), people now more frequently formulate their queries in detail. Chatbots are integrated into many websites where the search is conducted over a limited number of documents related to a specific field. Therefore, users now tend to include as much information as possible in their queries, which lengthens the search text but simultaneously narrows down the list of potentially relevant documents through the details and preferences described by the user. For example, with the query

> ”English detective novels”

a user would have to carefully review the recommended documents, examine the content, and decide what suits them best. However, with a query like

> ”Please, recommend English detective novels, but not books by Agatha Christie. I recently read ’Raven Black’ by Ann Cleeves and now I am looking for more detective novels written by English mystery crime writers”

the user would receive a list of books based on the details provided in the query. Even if additional filtering is needed, the user will not have to look through a large number of recommended books.

Moreover, this research allows solving the problem using modern methods.

The task addressed in this work was presented at the 10th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2011)[1], Books and Social Search Track. The book collection was compiled from metadata on Amazon Books [2] and supplemented with user-generated data from the LibraryThing Forum. Additionally, the formulations of user queries and additional search criteria were taken from the 'Talk' section on LT, where users share detailed book recommendations and in-depth responses in a question-and-answer format.

Since public access to the data was closed after the competition ended, the problem was not addressed with advanced methods and technologies, including Transformer Architecture models, advanced models for semantic document search, and modern search engines such as Elasticsearch. In this work, we revisit an already familiar problem using current tools.

1.2 Structure

This thesis is structured as follows: Chapter 1 introduces the problem statement and research objectives, providing an overview of the challenges in CIR and the proposed solutions and delves into the motivation behind this research, discussing the limitations of existing approaches.

Chapter 2 covers the existing solutions in the area of Aspect-Based Sentiment Analysis and Information Retrieval.

Chapter 3 introduces the data. Subsequent chapters will cover the methodological framework, experimental setup, and the results obtained, culminating in a discussion of the findings and their implications for future research.

Concluding remarks are reported in Chapter 6.

2 Related Work

This chapter is divided in four sections: Section 2.1 and Section 2.2 contain an overview of the research papers covering the 2 main stages of this work - Sentiment Analysis and Information Retrieval; Section 2.3 covers solutions of the similar task presented in the previous decade, and Section 2.4 briefly describes our participation in the competition, where we also implemented the information retrieval task using the tools used in the main part of the work, but on a smaller scale with different data.

2.1 Sentiment Analysis

Since the primary objective of this work is the incorporation of subjectivity into information retrieval, the first step is to analyze users queries from the perspective of sentiment. Sentiment Analysis (SA) is popular among the tasks where it is necessary to analyze a statement about something particular: a service, a purchased product, a book read, etc.

Types of Sentiment Analysis

SA can be divided into 3 types [3]: document-level analysis, sentence-level analysis, and aspect-level analysis (Figure 2.1). The first type is often used for analyzing customer reviews of products, market research, and studying employee satisfaction levels. A more detailed analysis — at the sentence level — is usually applied to movie/book reviews, posts, and comments on social

media. The third type, aspect-level analysis, is employed in reviews of restaurants, healthcare processes, and electronic products.

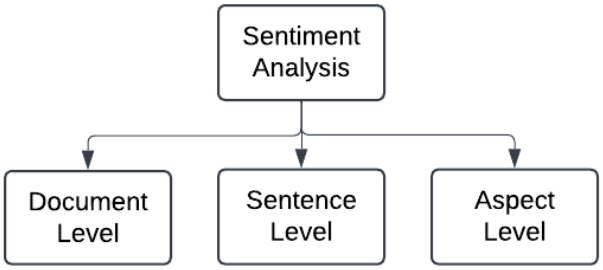

Figure 2.1: SA tasks.

It is the third subtype of SA that we consider the most relevant for this work. Aspect-based sentiment Analysis (ABSA) has become an important area in Natural Language Processing (NLP), offering insights into sentiments expressed towards specific aspects of entities, products, or services.

As in the example with restaurant reviews: where one might give a positive rating to the food but a negative one to the service (for example, *"I really enjoyed the quality of fish, but the waitress was so rude with us all the evening!"*). It would be irrational to evaluate such a review only by the general sentiment because the opinion expressed by the visitor is completely different in relation to various aspects. Or in the case of reviews of electronic products: where one might be satisfied with a laptop's screen quality but dissatisfied with its battery capacity. Similarly, in our task, users on the LT forum provide subjective evaluations of authors, specific books, or genres, and these evaluations can vary significantly. The overall sentiment of a forum query (at the document level) is usually neutral, as the user is asking for help rather than evaluating a particular topic.

Moreover, ABSA implies that we not only extract sentiment towards certain entities, but we extract the entities themselves. So, we obtain specific book titles, authors, or genres to understand user's personal opinions and extract subjectivity from a request. This information is then used to search the book database to retrieve relevant books. If ABSA determines that the user dislikes a certain author, the search can explicitly exclude books by that author from the final list.

The following sections explore rule-based approaches, machine learning methodologies, and

dedicated frameworks that address the challenges of ABSA based on the recent survey by Kandhro et al. [4].

2.1.1 Rule-based and Hybrid Approaches

Rule-based ABSA systems rely on manually defined linguistic rules and pattern-matching techniques to extract aspect terms and classify sentiment. While these systems are straightforward and interpretable, their dependency on domain-specific rules limits scalability and adaptability across different contexts.

Techniques like dependency parsing help identify opinion-bearing phrases and their associated aspects using syntactic structures. However, these systems often cannot correctly handle nuanced language variations and implicit sentiment expressions, making them less effective in complex real-world applications.

One of the best implementations in this category uses dependency tree structures to link opinion words to their aspects. For example, the system effectively identifies dependencies like "$ambiance \rightarrow is \rightarrow poor$" to extract "ambiance" as the aspect and "poor" as the sentiment. This method achieves high accuracy in structured datasets such as SemEval 2014 Task 4 [5].

Recent experiments, even if they include Aspect Terms Extraction based on lexical rules, usually combine this approach with machine learning techniques. For example, Negi et al. [6] formulated ABSA as a *text-to-text task* and used the pre-trained T5 model in addition to the syntactic approach for dataset annotation. Another paper [7] proposes a novel model, Convolution over Dependency Tree. It integrates Bi-directional Long Short Term Memory (Bi-LSTM) networks to capture contextual information and Graph Convolutional Networks (GCN) to leverage syntactic dependencies from dependency trees. This approach enables efficient propagation of opinion information to specific aspect words, improving classification accuracy.

2.1.2 Machine Learning Approach

Machine learning methods, especially those using deep learning, have pushed the boundaries of ABSA with robust models for aspect extraction and sentiment classification.

Aspect Sentiment Quad Prediction (ASQP) with Paraphrase Generation [8] simplifies the ABSA process by transforming it into a paraphrase generation task. This innovative approach generates natural language representations of sentiment quads using a sequence-to-sequence model, leveraging pre-trained generative models like T5 [9]. The input sentence is rephrased to explicitly include all sentiment elements. For instance, the sentence "The pasta is overcooked"

is paraphrased as "Food quality is bad because pasta is overcooked." This method ensures end-to-end processing, reducing error propagation and enhancing performance over traditional pipeline models.

Multiple-Element Joint Detection (MEJD) model [10] focuses on detecting target-aspect-sentiment triples by leveraging a combination of BERT embeddings, Bi-LSTM for capturing context, and a Graph Attention Convolutional Network (GACN) to capture dependencies between sentence elements and aspect categories. The GACN introduces an attention mechanism to enhance the flow of sentiment information between aspect terms and contextual words. Additionally, the approach uses Part-Of-the-Speech (POS)-tagging information to improve target detection. The model's architecture effectively handles complex relationships, including implicit targets and overlapping sentiment associations.

2.1.3 Frameworks

Modern ABSA frameworks that gained popularity during the last 2 years provide comprehensive solutions by integrating multiple subtasks, such as Aspect Terms Extraction and Aspect Sentiment Classification, into a unified system. They combine modularity, multilingual capabilities, and user-friendliness.

PyABSA

PyABSA is an open-source framework providing modular components for various ABSA tasks, enabling fast development. It is structured to allow users to easily train, evaluate, and deploy ABSA models with minimal setup. The PyABSA framework is modular, consisting of 5 main components (Figure 2.2):

- Configuration Manager: Handles all settings related to training, hyperparameters, and other configurations. It ensures the training process runs smoothly by checking the validity of all parameters before execution.
- Trainer: This module facilitates the training of ABSA models, supporting multiple prebuilt models and custom loss functions for optimization. It also enables k-fold cross-validation and efficient handling of insufficient datasets.
- Dataset Manager: Manages built-in and custom datasets, supporting automatic downloading and easy combination of datasets for training and testing. It includes tools for annotating datasets automatically or manually.

- Checkpoint Manager: Standardizes the inference process by managing pre-trained models and checkpoints. It supports models based on both GloVe and transformer architectures, allowing users to quickly deploy models for inference.
- Metric Visualizer: Automates the evaluation and visualization of model performance using plots like box plots, trajectory plots, and others, reducing the manual effort needed to interpret metrics.

Fast-LSA-T-V2 Model: Integrated into PyABSA, this model combines efficient dataset handling and pre-trained transformers to achieve high performance on ASC tasks, as demonstrated in restaurant and laptop review datasets.

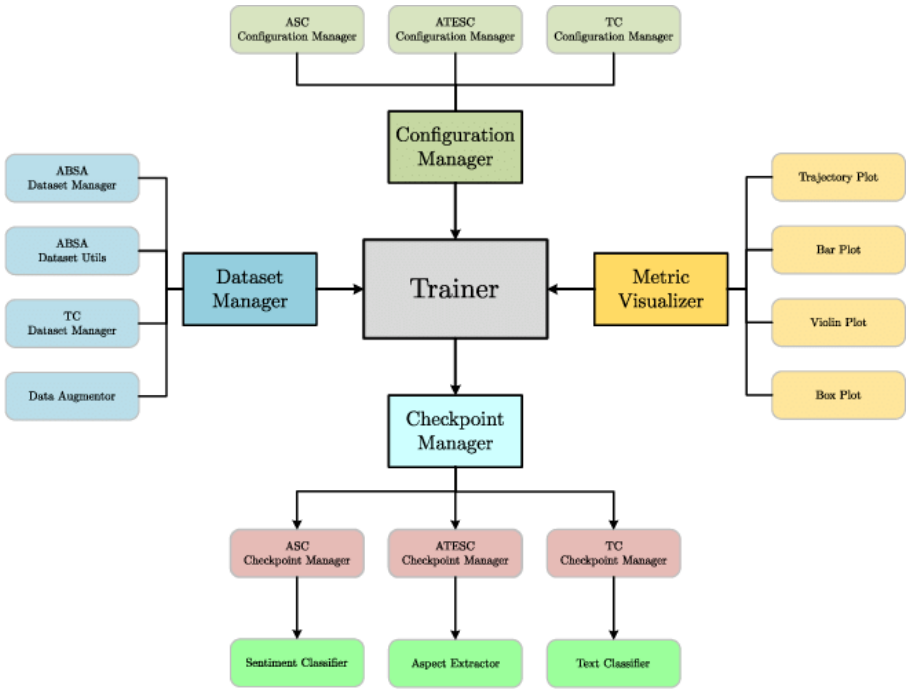

Figure 2.2: PyABSA architecture.[11]

SetFit

Few-shot learning has become an important area in natural language processing because it helps solve problems where there is very little labeled data. Traditional methods like parameter-efficient fine-tuning (PEFT) and pattern-exploiting training (PET) often need large models and carefully designed prompts. These approaches work well but can be slow, expensive, and inconsistent due to the manual effort required.

In contrast, SetFit (Sentence Transformer Fine-Tuning) introduces an efficient, prompt-free framework specifically designed for few-shot text classification. It is based on Sentence Transformers (ST), which leverage Siamese and triplet network structures and create meaningful sentence embeddings—compact representations of text that capture their meaning. SetFit uses a two-step process (Figure 2.3): first, it fine-tunes the transformer in a contrastive manner (on pairs of similar and dissimilar sentences) on a small set of labeled examples, helping the model understand relationships between them. The fine-tuning process is then followed by training a lightweight classification head on the embeddings produced by the fine-tuned transformer. This two-step process ensures high accuracy while maintaining computational efficiency, as demonstrated in multiple experiments. This method doesn't need prompts or very large models and works much faster than other techniques. The framework's developers compared SetFit to other models like GPT-3, T-FEW, and ADAPET. SetFit is much lighter and easier to use. It doesn't require huge computational resources and still achieves similar or better results in many cases.

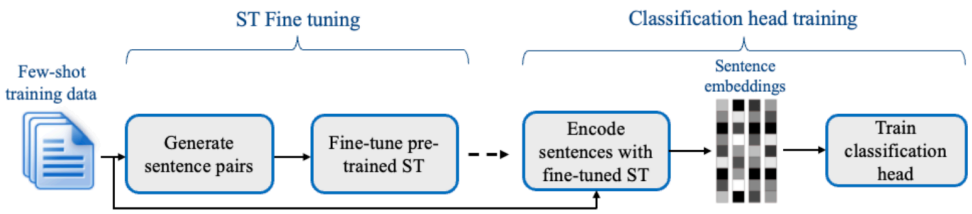

Figure 2.3: SetFit architecture.[12]

Moreover, SetFit provides convenient tools for implementing ABSA. The model leverages its few-shot capabilities to analyze textual sentiment toward specific aspects or entities. It provides a module `setfit[absa]` that includes `ABSAModel` class along with `AbsaTrainer` to fine-tune the model for a specific domain. ABSAModel can be used to implement aspect terms extraction (ATE) and aspect sentiment classification (ASC) tasks simultaneously or `AspectModel` can be used only for ATE and `PolarityModel` - only for ASC. ABSAModel model can be fine-tuned with custom data and it doesn't require a lot of labeled data as well.

Developers highlight that SetFit performs better when the number of labeled data is low and it surpasses T5 model (that is 2x bigger) and GPT2-medium (3x bigger). In the 'Implementation' part of this work, we will compare the open-source models.

2.2 Information Retrieval

The objective of the second part of the work is to retrieve relevant documents from a collection of books.

Information retrieval (IR) helps to find useful documents or data based on a user's search query. Over time, IR has improved, introducing new methods to make searches faster, more accurate, and smarter. Techniques like full-text search, vector search, and hybrid search are used to do this. Each method works differently and has its advantages and disadvantages. [13]

2.2.1 Full-Text Search

Full-text search is one of the oldest ways to find documents. It looks for exact words from the query in the documents. Methods used for this type of search (like BM25 and TF-IDF) rank documents by counting how often words appear and how important they are.

This method is simple and works well for basic searches. However, it has problems when a word has many meanings (polysemy) or when different words mean the same thing (synonymy). For example, a user searching for "car" may not see results for "automobile."

To improve, techniques like query expansion and document expansion are used:

- Query expansion adds similar or related words to the search terms.
- Document expansion adds more details to the documents to match different types of queries.

Even with these improvements, full-text search struggles with understanding the meaning of words.

2.2.2 Vector Search

Vector search solves some problems of full-text search by looking at the meaning of words, not just the words themselves. It uses vectors to represent words, sentences, or even documents. These vectors help the system compare how similar two things are.

Dense Vector Search

In dense vector search, queries and documents are turned into high-dimensional vectors using machine learning models like BERT. These vectors capture the context and meaning of words.

Dense vector search is very good at finding meaning, but it needs a lot of computing power. To handle large data, it uses methods like approximate nearest neighbor (ANN) to search faster.

Sparse Vector Search

Sparse vector search uses fewer numbers to represent queries and documents. It keeps only the important parts, which makes it faster and easier to store. Models like SPLADE improve sparse search by making it smarter about which words and meanings to focus on.

Sparse methods are not as detailed as dense methods, but they are quicker and use less computing power. This makes them useful when speed is more important than deep understanding.

2.2.3 Hybrid Search

Hybrid search combines full-text search with vector search to get the best of both. It uses:

- Full-text methods for quick and exact word matches.
- Vector methods for understanding meaning and context.

In hybrid systems, results from both methods are combined to give better answers. For example, they might mix sparse vectors (for speed) with dense vectors (for meaning). This makes hybrid search powerful for tasks like question answering, where both accuracy and understanding are needed.

2.2.4 Search Engines

Just as there are different approaches to IR methods, there are also different search engines that are suitable for specific purposes. For a long time, the most popular and widely used search engines for academic purposes were Indri [14] and Terrier [15]. More suitable for industrial applications and offering advanced capabilities for IR is Elasticsearch [16].

Indri

Indri, developed as part of the Lemur Project, combines probabilistic inference networks and language modeling to support structured queries and retrieval across diverse document types. It models relevance as the probability $P(I|D, \alpha, \beta)$, where:

- I: Information need expressed in the query.
- D: A document being evaluated.
- α, β: Smoothing parameters to adjust probabilities for sparse data.

The retrieval model employs a multiple Bernoulli framework:

$$P(r_i|D, \alpha, \beta) = \frac{\#(r_i, D) + \alpha}{|D| + \alpha + \beta} \tag{2.1}$$

where:

- $\#(r_i, D)$: Frequency of the term or feature r_i in document D.
- $|D|$: Length of the document.

To adjust for sparse data, Dirichlet smoothing is applied:

$$P(r_i|D, \mu) = \frac{\#(r_i, D) + \mu P(r_i|C)}{|D| + \mu} \tag{2.2}$$

where:

- $P(r_i|C)$: The probability of feature r_i in the entire collection C.
- μ: A smoothing parameter that balances document and collection-level probabilities.

Indri supports proximity operators like `#uwN` (unordered window) and `#odN` (ordered window) to approximate contextual semantics. However, its native support for semantic search is limited, requiring external preprocessing or extensions for hybrid capabilities.

Terrier

Terrier, based on the Divergence From Randomness (DFR) framework, measures term informativeness by its divergence from a random distribution:

$$w(t, d) = \text{Info}(t) \cdot \text{Normalization} \tag{2.3}$$

where:

- $\text{Info}(t) = -\log_2(P(t|\text{random}))$: Quantifies how much the term t stands out in the document.
- Normalization: Adjusts weights for document length.

For semantic search, Terrier integrates dense retrieval models, computing similarity between query and document embeddings as:

$$\text{Sim}(\mathbf{q}, \mathbf{d}) = \frac{\mathbf{q} \cdot \mathbf{d}}{\|\mathbf{q}\|\|\mathbf{d}\|} \tag{2.4}$$

where:

- $\mathbf{q}$: Query vector.
- $\mathbf{d}$: Document vector.
- $\|\mathbf{q}\|$: Magnitude of the query vector.
- $\|\mathbf{d}\|$: Magnitude of the document vector.

Hybrid search combines dense vector scores and traditional DFR or BM25 scores:

$$\text{HybridScore}(D) = \lambda \cdot \text{DFRScore}(D) + (1 - \lambda) \cdot \text{SemanticScore}(D) \tag{2.5}$$

where λ balances lexical and semantic relevance.

Elasticsearch

Elasticsearch is an open-source search engine designed for real-time, large-scale search and analytics. Its primary ranking function is BM25, which evaluates documents as:

$$\text{Score}(Q, D) = \sum_{t \in Q} IDF(t) \cdot \frac{f(t, D) \cdot (k_1 + 1)}{f(t, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avg}(|D|)}\right)} \tag{2.6}$$

where:

- Q: Query terms.
- $f(t, D)$: Frequency of term t in document D.
- $|D|$: Length of document D.

- avg($|D|$): Average document length in the collection.
- k_1: A parameter controlling the impact of term frequency.
- b: A parameter controlling the level of document length normalization.
- $IDF(t)$: Inverse document frequency of term t, calculated as:

$$IDF(t) = \log \frac{N - n(t) + 0.5}{n(t) + 0.5} \tag{2.7}$$

  where N is the total number of documents, and $n(t)$ is the number of documents containing term t.

Elasticsearch excels in semantic search by natively supporting dense vector embeddings. It calculates similarity between vectors using cosine similarity score same as in Terrier (Formula 2.4).

To implement hybrid search, the search engine provides various mathematical tools to combine search results across multiple fields and calculate the final result. For example, it combines BM25 and semantic similarity scores:

$$\text{HybridScore}(D) = \lambda \cdot \text{BM25Score}(D) + (1 - \lambda) \cdot \text{SemanticScore}(D) \tag{2.8}$$

where λ controls the balance between lexical and semantic components.

Nowadays, the most commonly used method for calculating the final score is the algorithm Reciprocal Rank Fusion (RRF). RRF gives each document a score depending on its position (rank) in each list (if we use different types of search). Higher-ranked documents (those appearing near the top) receive higher scores. The RRF score for a document is calculated using the formula:

$$\text{Score}(d) = \sum_s \frac{1}{k + \text{rank}_s(d)} \tag{2.9}$$

where:

- Score(d): Total score for document d.
- s: Each individual ranked list or source.
- $\text{rank}_s(d)$: Rank position of document d in list s (with 1 being the top).
- k: A constant (usually set to 60) that controls the influence of rank positions.

It summarizes the scores from all lists for each document and sorts documents based on their total RRF scores to create a unified list that favors highly ranked documents across multiple lists.

2.3 INEX2011: Books and Social Search Track

As was mentioned in Section 1.1, all the necessary resources were obtained from the INEX2011 workshop. The organizers also published the results of the teams[1], which allowed for the study of approaches that had already been applied to solving this problem in order to understand what would be considered innovative. The difference is that in the competition, teams were evaluated by metrics for extracting ranked documents. In our case, we have data where it is only known that a document is relevant or nonrelevant to a query. That is, we extract an unranked list of documents.

The top-performing team[17] used pseudo-relevance feedback on an index using 50 terms from the top 10 results. Indri[14] was used for indexing the data. Based on various experiments with data indexing, the best result was achieved when only reviews and tags were used from the book collection, and only basic title information was used from the query data.

Another high-performing team[18] filtered out fields from the book collection that were irrelevant to the search, leaving 19 fields, combining user reviews into a single text, and merging fields containing quotes and excerpts from the book into a single content field. This team also used Indri for search.

Inspiration and initial steps in data exploration and processing were taken from the participants' solutions, and approaches that did not lead to good results were also analyzed. However, the main objective of this work is to propose a new and modern approach to information retrieval by means that have not yet been solved and to extract new information from queries (such as subjectivity) and use it in document retrieval.

2.4 CLEF 2024: Conference and Labs of the Evaluation Forum

While working on the main project, we also took part in a competition CLEF 2024: JokerLab [19]. The goal of the track was to analyze short documents and extract those that are relevant for a query and also have pun or wordplay. In this lab there were less data than in our main

project, and queries included only text and had just a few words, but the idea behind the information retrieval is similar.

We experimented with the T5 transformer model, that was trained for various NLP tasks and has an architecture designed for efficient fine-tuning on a specific dataset. The main steps of the work includeв query extension, tokenization, similarity scoring, and the use of pre-trained models to identify texts with puns.

Data description

The organizers provided a set of 61,104 documents with short texts and queries of 1 to 5 words. The training data consists of 12 queries accompanied by the corresponding relevance judgments, where relevant documents are marked 0 or 1 to indicate the absence or the presence of a pun in the text, respectively.

Our solution

The task was divided into several stages: working with queries, expanding queries with synonyms, finding the best method for tokenization of queries and documents, choosing a threshold for the similarity score, and working with a pre-trained model to filter texts with puns.

1. Queries extension.
   Each query was extended with its synonyms and compared similarity not only with an initial query but also with the synonyms found with `WordNet` database. If the original request is a proper noun, it is left in the original single version.

   Some examples of extended queries include:

   - qid_test_52: Tom - [Tom]
   - qid_test_50: vein - [vein, mineral vein, venous blood vessel]

2. Tokenization.

   - bert-base-uncased[20]: embeddings were calculated for extended queries and document texts. Cosine similarity was used as the similarity score.
   - all-MiniLM-L6-v2[21]: this model returns embeddings in a suitable format: for each synonym, its vector was calculated separately. As a result, a query of dimension *(n, 384)* was obtained, where *n* - is the number of synonyms obtained in the

previous step. Text of a document was processed in the initial format and a vector of dimension *(1, 384)* was obtained. The similarity was calculated for each synonym, and the maximum value found was taken as the final similarity score.

The model with the highest accuracy, all-MiniLM-L6-v2, was chosen as the final model.

3. Similarity score.
   The next step was to find the optimal threshold for the similarity score to include all relevant documents. Accuracy was calculated for the range (0.6, 0.95) with the step of 0.05. The final threshold = **0.35** was chosen based on calculations where the F1-score reached its highest value compared to other thresholds higher than 0.35.

4. Wordplay detection.
   To identify texts containing puns/wordplay, the `flan-T5-base` model [**?**] was taken as the basis. Since 2,389 labeled documents could be obtained from the training data, the model was additionally trained on these data. 250 documents were excluded from training process to evaluate the performance on labeled data.

   Prompt used to pass the model:
   `“Does the following text has pun/wordplay? {Text} “`

Results

The following metrics were calculated for 250 documents extracted from the labeled data:

- accuracy: 75.2%
- precision: 18.75%
- recall: 5.77%
- specificity: 93.43%

The imbalanced dataset leads to a higher number of false negatives, as the model is more likely to predict the majority class (’no wordplay’). It results in many missed actual positives, lowers the recall, and increases specificity.

The model was mostly confused in texts containing definitions. The model showed better performance on identifying the correct class (no pun/wordplay) on short texts of 1 sentence up to 15 words.

3

Data Description

This section contains a description of the data used in the work. Section 3.1 includes an exploratory analysis of users requests data from the LT forum. Section 3.2 describes the collection of books from Amazon. Section 3.3 provides a description of the data used to evaluate solutions in Information Retrieval problems.

3.1 User Requests

There are 211 queries in total. The data is collected in a single XML file (Figure 3.1), where each query contains the following information:

<title> – the title of the topic thread;

<group> – the name of the discussion group;

<narrative> – the narrative field contains the first message of the topic thread;

<type> – a topic type label assigned by the dataset creators;

<genre> – genre of books a user is looking for;

<specificity> – shows if the request is broad/narrow.

```
<topic id="3835">
  <title>General Gardening Book Recommendations</title>
  <group>Gardening</group>
  <narrative>
    I've been asked to recommend a good "general purpose"
        gardening book for someone who wants to know the
        basics of flowers & shrubs and when to do various
        tasks. This would be for someone gardening in zone 6.
        Does anyone have any suggestions?
  </narrative>
  <types>
    <type>subject</type>
  </types>
  <genres>
    <genre>science - botany</genre>
    <genre>agriculture</genre>
  </genres>
  <specificity>broad</specificity>
</topic>
```

Figure 3.1: Example of user request with additional information.

The main fields for searching are the <title> and <narrative>, as well as <group>. Titles always summarize the question being asked or even contain the main user's intent. For example,

<title>Marquis de Sade</title>
<group>Alternative Sexuality</group>
<narrative>
So...has anyone read any of his work? Looking for recommendations on where to start. Thanks!
</narrative>

HTML tags were preliminarily removed from the textual data, and from the links in the <a> tags, only the text content was retained without the URLs. Below are statistics for the <narrative>. Table 3.2 shows the average length of a query. In most cases, the number of sentences in a query does not exceed 9, and the number of words does not exceed 180. Anomalies typically include queries where users list a large number of books or authors they have already read. For example,

”hey all, my department’s annual book auction is coming up (faculty members sell books they no longer want to us broke grad students, basically)... and this year there is a huge selection of nabokov on the list of available books. i’m a lover of most classic literature, but haven’t read any nabokov... so i was wondering if you experts might look at this list and tell me which few you think would be best for a nabokov-virgin? thanks in advance for any help, and here’s the list i’ve got (sorry about the formatting, it’s just a cut-n-paste).
Vladimir Nabokov, Ada.
Vladimir Nabokov, The Defense.
Vladimir Nabokov, The Eye.
Vladimir Nabokov, The Gift.
Vladimir Nabokov, Invitation to a Beheading.
Vladimir Nabokov, The Song of Igor’s Campaign.
Vladimir Nabokov, Spring in Fialta.”

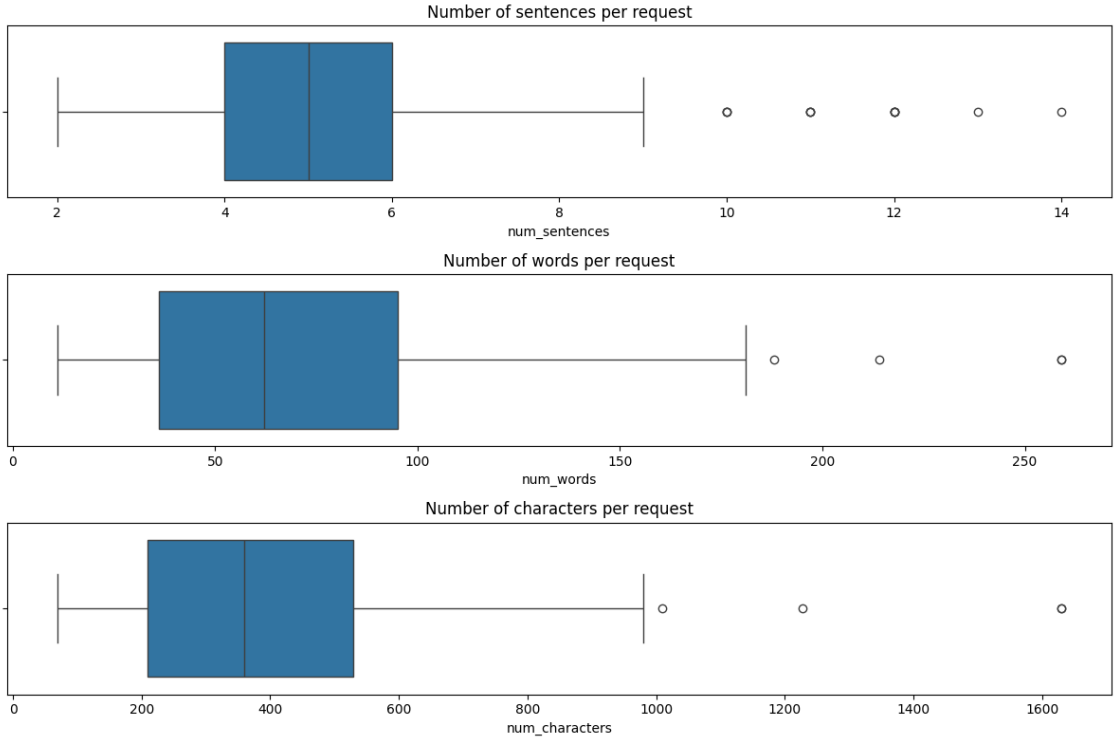

Figure 3.2: Number of sentences, words and characters in requests.

Most users search for 1 genre per request (Figure 3.3). In some cases, there are more than 5 genres in one request. Then, as a rule, these are genres that are similar to each other (religion,

history, social science, etc.). Literature and History are among the most frequently searched genres (Figure 3.3).

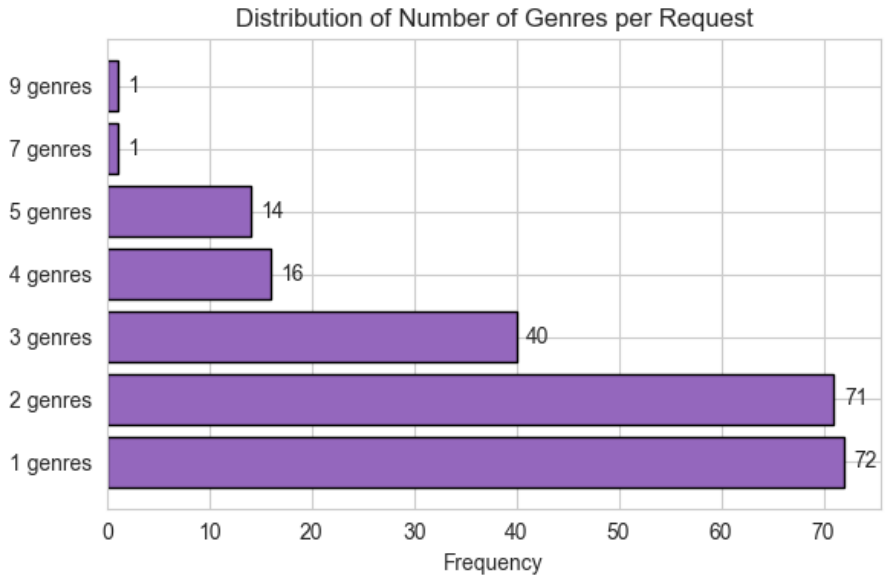

Figure 3.3: Number of genres per request.

3.2 Books Collection

The collection consists of 800,000 books and is obtained from the Amazon website [2]. In addition to automatically generated information, books in the collection contain user-generated fields (such as book review, summary, dedication, epigraph etc). The full initial list of tags from the collection is shown on Table 3.1.

In Chapter **??** we will explain how we preprocess tags that we consider important for our task. For now, we look at some statistics and distributions over the dataset.

For example, the percentage of different codes from the Dewey list, which can be interpreted as genres (Figure 3.6), or by categories – automatically generated from the Amazon collection (Figure 3.7. As in the list with user queries, literature is the most popular genre. Next in popularity are social sciences and non-fiction books.

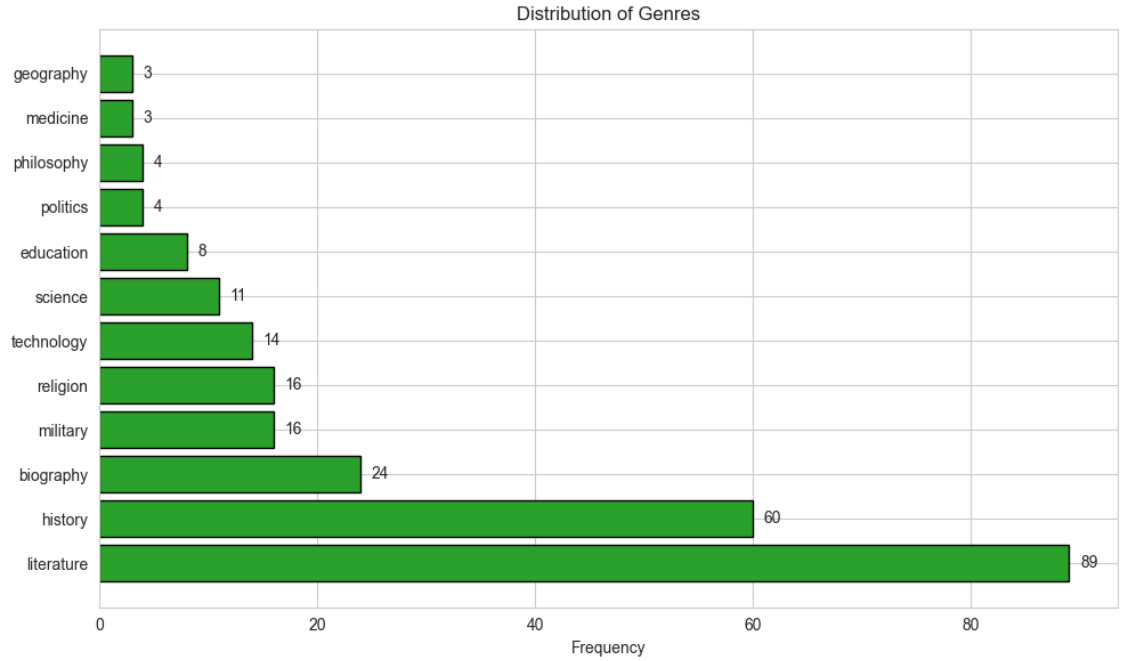

Figure 3.4: Genres distribution in requests.

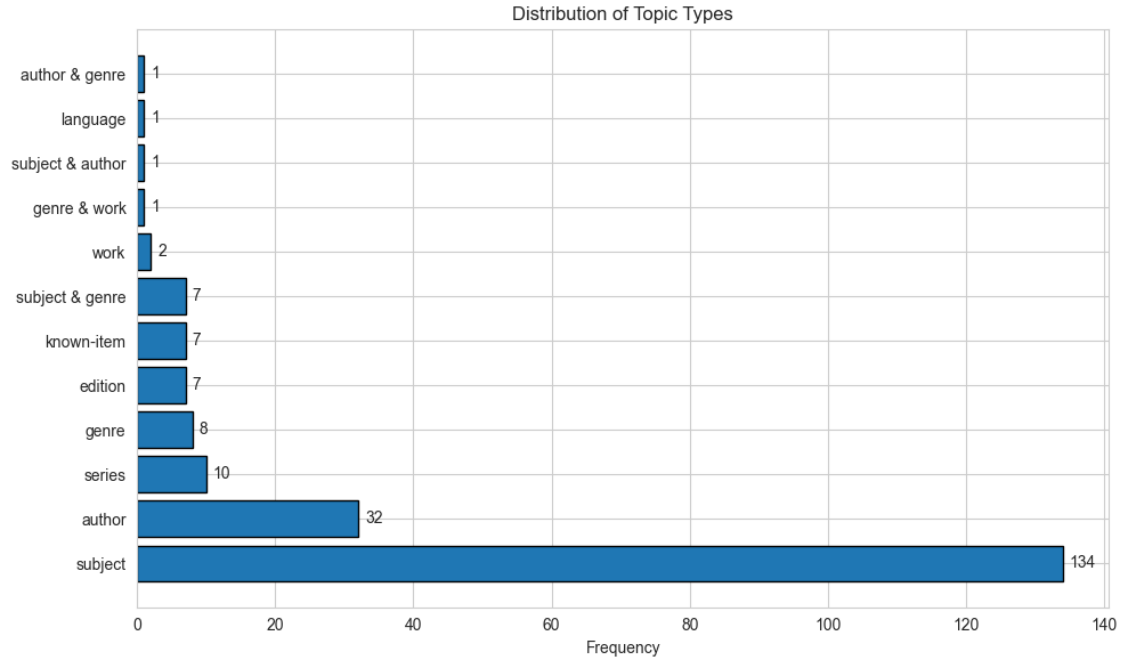

Figure 3.5: Topic types distribution in requests.

Tag Name	Tag Name	Tag Name	Tag Name
book	studio	title	imagecategory
dimensions	tags	edition	name
reviews	isbn	dewey	role
editorialreviews	ean	creator	blurber
images	binding	review	dedication
creators	label	rating	epigraph
blurbers	listprice	authorid	firstwordsitem
dedications	manufacturer	totalvotes	lastwordsitem
epigraphs	numberofpages	helpfulvotes	quotation
firstwords	publisher	date	seriesitem
lastwords	height	summary	award
quotations	width	editorialreview	browseNode
series	length	content	character
awards	weight	source	place
browseNodes	readinglevel	image	subject
characters	releasedate	imageCategories	tag
places	publicationdate	url	data
subjects			

Table 3.1: Tags list in the initial books collection.

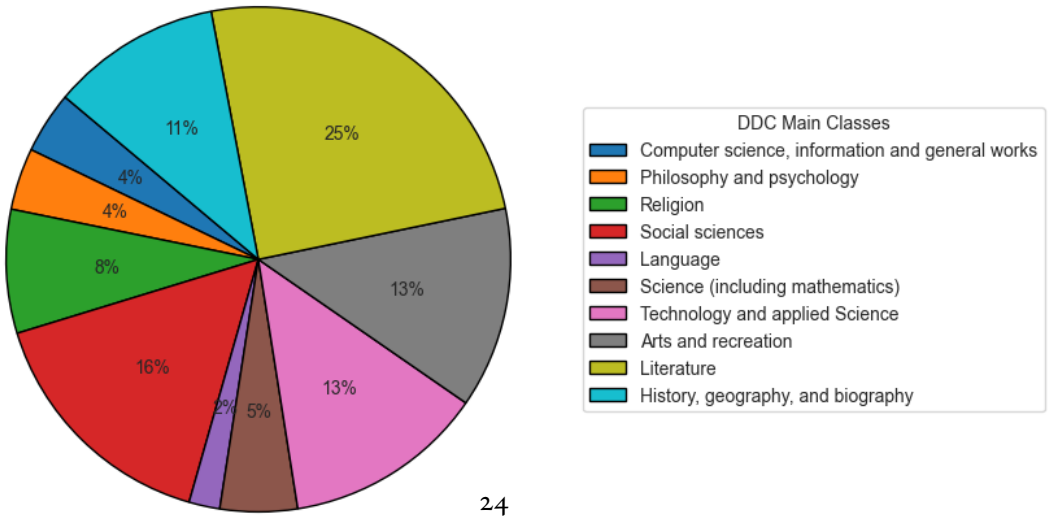

Figure 3.6: Dewey Decimal Code distribution.

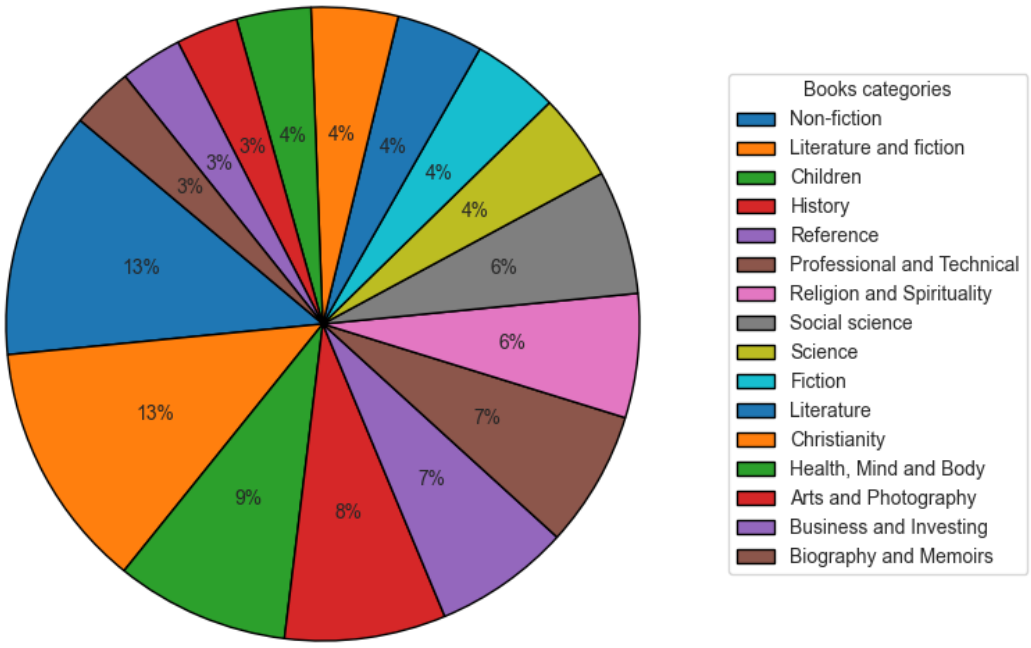

Figure 3.7: Books categories distribution.

3.3 QRELS Files

In addition to the data containing the collection of books and the list of queries, we are also provided with a file in the '.qrels' format (Query Relevance Judgments). This file represents relevance judgments of documents with respect to specific queries.

As a rule, this is a plain text file with the following attributes:

- Query ID: A unique identifier for the query.
- Document ID: A unique identifier for the document being judged.
- Relevance Score: A numeric score indicating the relevance of the document to the query.
- Additional Column: optional column for any specific use case.

Relevance scores can be represented in binary format: 0 (not relevant) or 1 (relevant). Alternatively, there can be graded relevance (e.g., 0, 1, 2, 3), where higher numbers indicate higher relevance.

An example of the .qrels file provided to us is presented on Figure 3.8.

```
1    74  0 0822942542 1
2    74  0 0822959399 1
3    399 0 0307236722 1
4    399 0 0333511824 1
5    399 0 0333511832 1
6    399 0 0618443363 1
7    399 0 0688044026 1
8    399 0 0688122612 1
9    399 0 0941034062 1
10   530 0 0060155825 1
```

Figure 3.8: Example of .qrels format.

Since the collection of documents for retrieval is very large, it is not practical to use the relevance score in binary format. In our case, the file includes only those query-document pairs that are relevant. By default, we assume that if a pair is not found in the file, then the document is not relevant to the query. It is also worth noting that the score is unranked; that is, the order of the retrieved documents does not matter to us.

The second column in our case is not used.

4

Methodology

This chapter provides a general overview of the work: the steps taken, additional data identified for task implementation, its preparation, and the methods used to evaluate the system's results.

Figure 4.1 illustrates the higl-level representation of the full process. It is important to emphasize that these stages are not executed simultaneously. Each type of data is processed separately. Additionally, it should be clarified that the goal of the final solution is to generate a list of relevant documents for an already existing set of user queries. At this point, we are not developing a full end-to-end application where the system processes data in real time, performs subjectivity extraction, and then forwards the data to a search engine for document retrieval. We approach this problem primarily from an academic perspective, focusing on exploring various methodologies. In some cases, we aim for results that yield the best metrics, even if they require more time to execute.

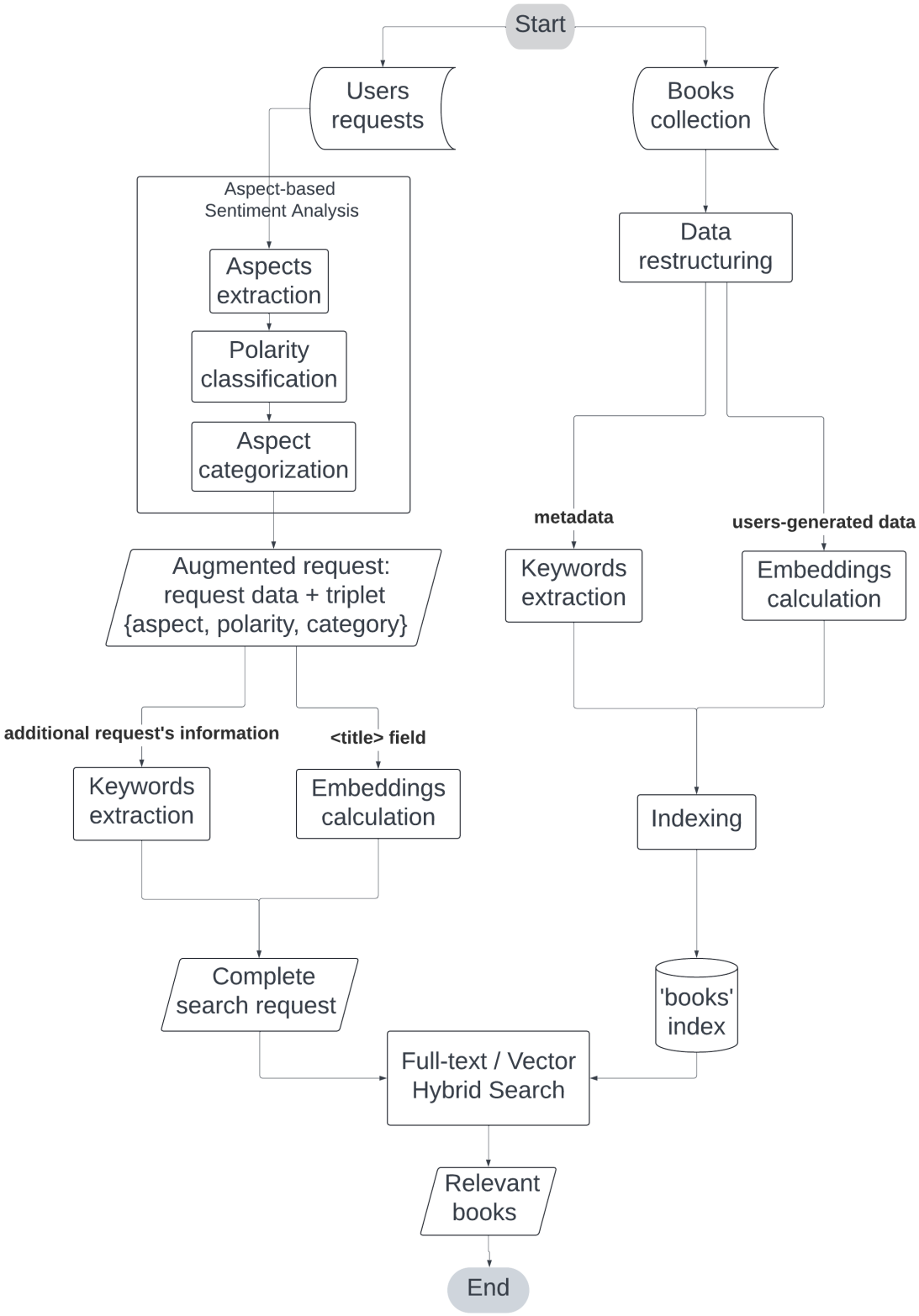

Figure 4.1: Project workflow.

The first part of the work is entirely dedicated to extracting subjectivity from user queries. By subjectivity, we mean a person's attitude to some aspect. Attitude is often expressed by positive or negative emotions. That is why we devote the entire first part of the work to the analysis of sentiment. Forum queries contain various types of information, but sentiment is extracted only from text written in a conversational format. As a result of this process, we obtain triplets consisting of an aspect, its sentiment, and its category in addition to the provided request's information.

The second part focuses on information retrieval. However, in addition to the retrieval itself, this part also involves preparing the data for the search process. Therefore, tasks such as indexing the document collection and formatting the query appropriately for the search are also included in the information retrieval phase.

As output for a specific user query, we get a list of relevant books from the indexed document collection. This is the final step. After that, we can only calculate metrics and evaluate our system.

4.1 Subjectivity exploration

4.1.1 Sentiment Analysis

As was mentioned in Section 2.1, sentiment analysis on the aspect level is the type of SA that we need.

When we dive into the details of ABSA, we see that it also includes various sub-tasks and we need to choose which of them we should implement. It depends on the type of information that needs to be extracted [22]: aspect extraction, opinion term extraction, sentiment classification, and aspect categorization into specific groups (Figure 4.2).

The aspect itself will be used in the search for relevant books, making the ATE (Aspect Term Extraction) task a primary step in this part of the work. At this stage, we extract book titles, author names, genre names that the reader may or may not like.

The ATSC (Aspect Term Sentiment Classification) task can be formulated differently depending on the requirements. Sentiment in product reviews is often determined on a 5-point scale (from 1 to 5), while sentiment in social media comments can be divided into custom classes — for example, different types of aggression or happiness. In our task, it is more important to determine whether the user has a positive or negative attitude toward a mentioned aspect, as this distinction directly affects whether a particular book should be included in the

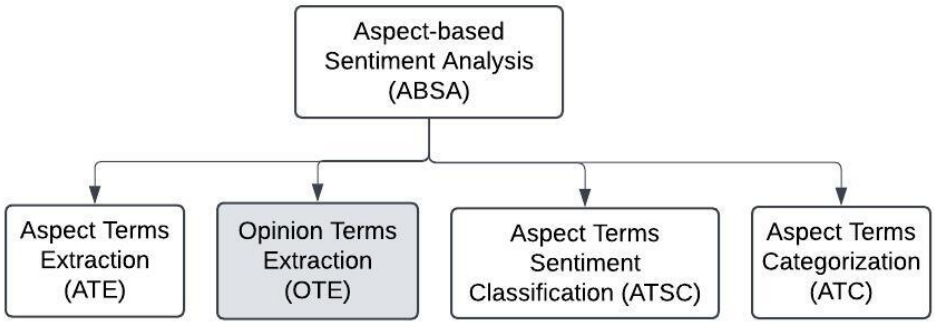

Figure 4.2: ABSA sub-tasks.

final list. However, since users may also express a neutral opinion by simply mentioning an aspect, a third class must be introduced. Thus, sentiment classification is performed across three classes: Positive, Neutral, and Negative.

The ATC (Aspect Term Categorization) task and its specific categories are determined by the requirements and nature of the task. Unlike ATSC, there are no universally accepted categories for this type of classification. In the following sections, we will explain how we selected the categories and the reasoning behind these decisions. For now, it is worth noting that this task is critical, as we need to map the fields from the book collection to the fields in user queries. While some fields are already explicitly known from the initial set of queries (e.g., tag `<genre>`), other fields need to be extracted from the user query and assigned to one of the categories (e.g., fields like `<author>` and `<book_title>`).

Finally, ABSA also includes the OTE (Opinion Term Extraction) task. This involves identifying the word or phrase upon which the sentiment class was determined. Let's say a user mentions *"I am a huge fan of Harry Potter"*. OTE implies the extraction of the word *"fan"*. But we are not interested in how sentiment is expressed. Much more important is that the class of sentiment is *Positive* in relation to the *"Harry Potter"* aspect. For our project, this information is not particularly significant, as we cannot perform a meaningful search using these terms or match them with the book collection. Additionally, it is an extra step of a model calling. OTE also requires an additional call to an Opinion Term extraction model, which wastes resources and takes time. Given the above arguments, this step is not included in the final process.

4.1.2 Annotated reviews data

One of the challenges we faced during this work was the lack of annotated data in our specific domain—book descriptions [23]. Standard datasets used to evaluate sentiment analysis quality include social media posts from Twitter [24] (for document- or sentence-level analysis) and restaurant or electronic product reviews [5, 25] (for aspect-level analysis). However, these annotated datasets are not suitable for our purpose. When analyzing the subjectivity of queries from LT, it is essential to identify proper nouns (author names or book titles) as aspects. Most of the datasets focus on simple nouns, pronouns or compound nouns.

There is a dataset of book reviews collected from Amazon [26], containing over 200,000 reviews. This dataset is the most suitable for our domain. However, the data can be used to train models only on document-level sentiment analysis. In this dataset, reviews are rated on a 5-point scale. While it is possible to convert the sentiment ratings into a 3-class format, the absence of annotated aspects in the data makes it impossible to fully utilize this dataset for aspect-level analysis.

For these reasons, the only dataset found for ABSA evaluation is a collection of 295 reviews from 40 different books, gathered from Amazon/LT reviews [27]. We will use it to train models and evaluate the obtained results. However, this dataset does impose limitations, as the amount of data (300 reviews) is still small enough to fully train or fine-tune a machine learning model. Moreover, the fact that the reviews are collected from data on 40 books greatly limits the list of unique aspects. The reviews contain many repeated author names and genre titles.

The 300 reviews are divided into 2977 sentences. All reviews include 3504 aspects in total. Below is an example of one sentence from one review (Figure 4.3). Each review has the following tags:

`<Review>`: full data of a review.

`<sentence>`: there are one or more sentences in each review. Each is extracted one by one.

`<text>`: plain text of the sentence.

`<Opinion>`: tag with subjectivity information. Include a category of an aspect, it's polarity, and an aspect itself. Occurrence shows ordinal information.

In the following section, we explain how we prepare annotated data for Sentiment Analysis implementation.

```
<Review rid="000_0007210000_22">
    <sentences>
        <sentence id="000_0007210000_22:1">
        <text>The fourth psychological thriller from Michael
            Marshall Smith (writing under his truncated
            pseudonym) is a departure from his Straw Men
            universe</text>
        <Opinions>
            <Opinion category="CONTENT#GENRE" occurrence="1"
                polarity="positive" target="psychological
                thriller"/>
            <Opinion category="BOOK#AUTHOR" occurrence="1"
                polarity="neutral" target="Michael Marshall
                Smith"/>
        </Opinions>
        </sentence>
    </sentences>
<Review>
```

Figure 4.3: Example of an annotated review.

Data cleaning

Despite the fact that this dataset was specifically collected for ABSA, we had to reconsider the inclusion of certain aspects as potentially useful for our task. Since all ABSA subtasks will depend on the aspects correctly identified at the very beginning, it is necessary to analyze which aspects should be included. It is most convenient to analyze aspects using examples of their categories.

The dataset creators identified 13 categories, some of which are irrelevant to our task (Figure 4.4).

We believe that for retrieving books based on queries, we cannot rely on rare categories (such as price, book length, or period). The categories of book structure and book quality are more frequent; however, based on the book collection data, we lack tags in the dataset that would allow us to search for quality or structure. Also, there are very few aspects of these categories in the dataset, and we will not be able to teach the model to correctly identify these classes. Furthermore, we observe that the ”general” category appears very frequently. However, after manual inspection of the data, it was found that this category typically includes terms such as ”book,” ”series,” or ”edition.” For now, we retain them since they are technically aspects

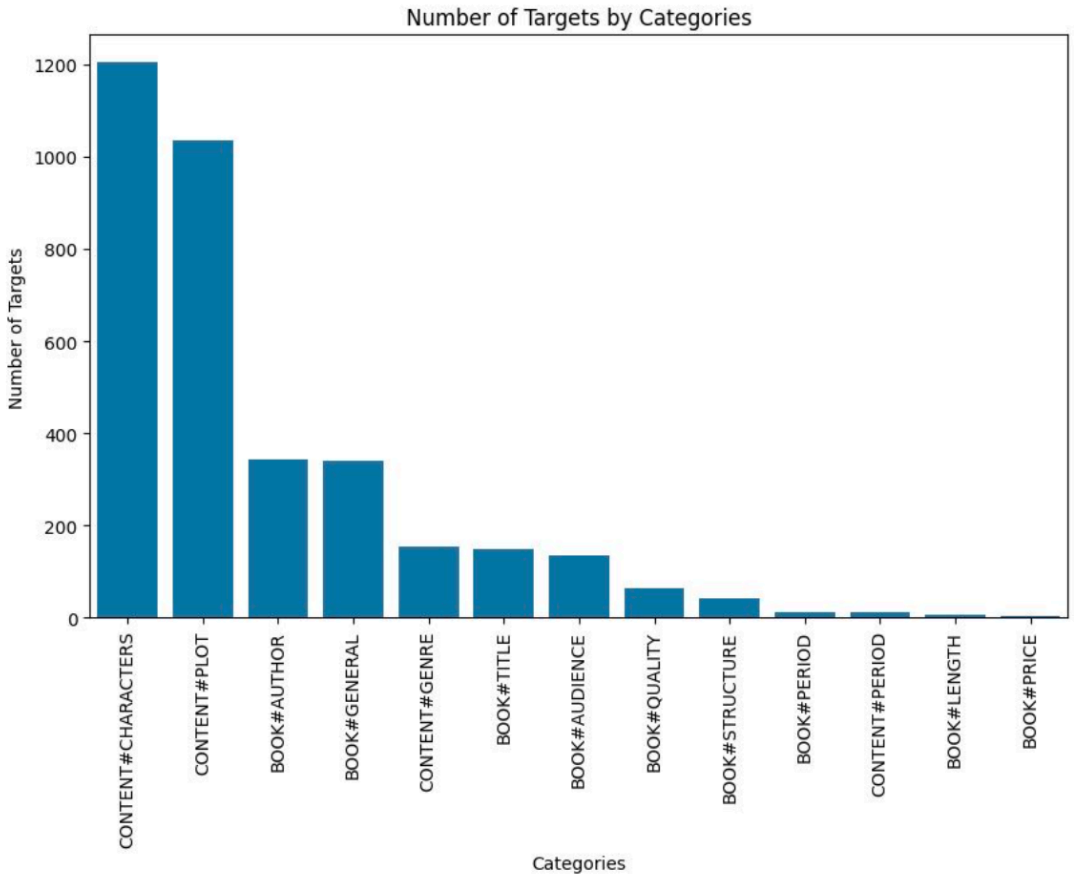

Figure 4.4: Initial aspect terms categories in the annotated dataset.

according to the rules, but in future query formulations for the book collection, we will exclude such values.

After analyzing the necessary categories, we are left with 7 categories:

CONTENT#CHARACTERS

CONTENT#PLOT

BOOK#GENERAL

BOOK#AUTHOR

BOOK#GENRE

BOOK#TITLE

BOOK#AUDIENCE

The classes of aspect term sentiments are standard. There are three classes: POSITIVE, NEUTRAL, and NEGATIVE. These classes remain unchanged as they provide the level of granularity we need to describe sentiments.

Data preparation

Now, we have data annotated with aspects that are relevant to our task. To implement ABSA, the XML format is flattened into a standard dataframe, where each occurrence of an aspect corresponds to a separate row (Table 4.1).

ID	text	target	polarity	category	ordinal
1	I think Kolbler does a good job of detailing the rise of the Mafia	Kolbler	POSITIVE	BOOK#AUTHOR	1
2	Zindel could've done better	Zindel	NEGATIVE	BOOK#AUTHOR	1

Table 4.1: Example of restructured annotated data.

The data is divided into training and testing sets in a 3:1 ratio. Table 4.2 provides statistics on the divided data.

Dataset	#Sentences	#POSITIVE	#NEGATIVE	#NEUTRAL	#Total
Train	2232	720	271	1487	2478
Test	745	233	91	485	809

Table 4.2: Train and Test sets statistics.

For the aspect category classification problem, we also check that the class distribution between the training and test sets is approximately the same (Figure 4.5).

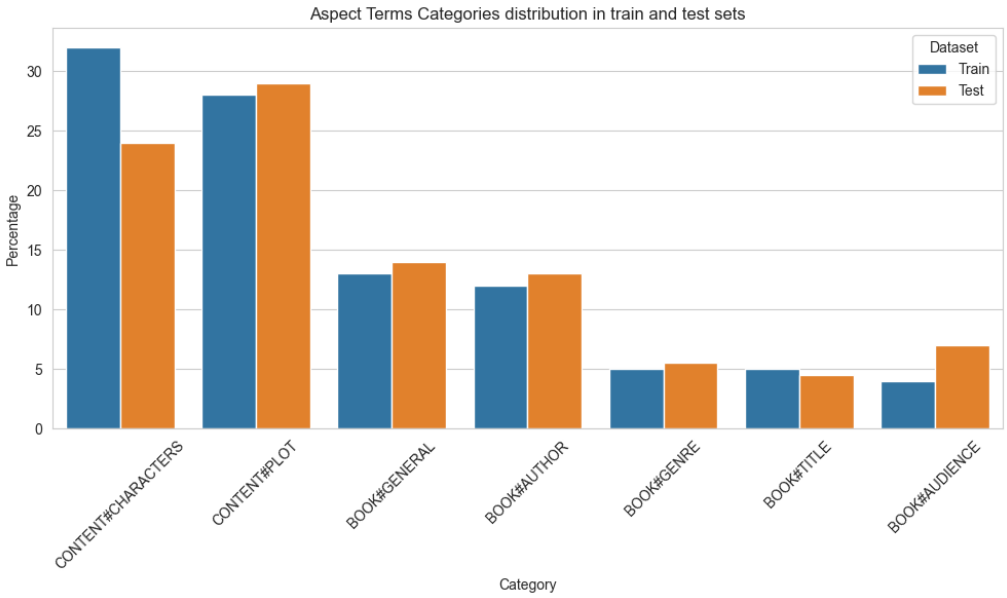

Figure 4.5: Distribution of different categories among the Train and Test sets.

4.1.3 Evaluation

Each of the subtasks is evaluated using commonly accepted metrics for ABSA. The creators of the annotated dataset described in the previous section have already provided results, which we will consider as the baseline model's performance.

For comparing ABSA solutions, it is standard practice to use the metrics Precision, Recall, and F1-score. They are calculated using the counts of True Positives (TP), False Positives (FP), and False Negatives (FN). Next, we will look at how these metrics can be interpreted for each task.

Aspect Terms Extraction

The tricky part of this task is that for each sentence, the model may predict the correct number of aspects, as well as more or fewer aspects than required. However, after all we summarize the TP, FN, FP and TN values among all data to get the final scores. Table 4.3 will help to understand how classes are defined depending on predicted aspect terms.

Next, we see the example of a class assignment: *"Monday gives Arthur the minute hand to the 'Key to the Kingdom' that he possesses"*.

In this example, $TP = 2$; $FN = 1$; $FP = 1$.

Actual/Predicted	Aspect Term (Relevant)	Not an Aspect Term (Non-relevant)
Predicted as Aspect Term	True Positives (TP): Correctly predicted aspect terms	False Positives (FP): Terms incorrectly predicted as aspect terms
Not Predicted as Aspect Term	False Negatives (FN): Missed aspect terms	True Negatives (TN): Correctly ignored non-aspect terms

Table 4.3: Contingency table for ATE task.

true aspects	Monday	Arthur		Key to the Kingdom
predicted aspects		Arthur	minute hand	Key to the Kingdom
assigned class	FN	TP	FP	TP

Table 4.4: Example of caption for the table.

For the final calculation of the scores among multiple sentences, we aggregate the results among all the sentences.

Let the number of sentences be N. We first calculate the total True Positives, False Positives, and False Negatives across all sentences:

$$\text{TP}_{\text{total}} = \sum_{i=1}^{N} \text{TP}_i, \quad \text{FP}_{\text{total}} = \sum_{i=1}^{N} \text{FP}_i, \quad \text{FN}_{\text{total}} = \sum_{i=1}^{N} \text{FN}_i \tag{4.1}$$

Using these aggregated values, we compute the metrics as follows:

$$\text{Precision (P)} = \frac{\text{TP}_{\text{total}}}{\text{TP}_{\text{total}} + \text{FP}_{\text{total}}} \tag{4.2}$$

$$\text{Recall (R)} = \frac{\text{TP}_{\text{total}}}{\text{TP}_{\text{total}} + \text{FN}_{\text{total}}} \tag{4.3}$$

$$\text{F1-Score} = 2 \cdot \frac{P \cdot R}{P + R} \tag{4.4}$$

High Precision indicates few false positives, meaning most predicted aspects are correct. High recall means most actual aspects are correctly identified, even if precision is lower. F1-score balances the trade-off between precision and recall.

Aspect Terms Sentiment Classification and Aspect Terms Categorization

In both ATSC and ATC tasks, we deal with multi-class classification. For multi-class classification, precision, recall, and F1-score are computed per class and can be averaged. Table 4.5 shows the general contingency matrix for multi-class classification.

Actual/Predicted	**Category 1**	**Category 2**	**...**	**Category N**
Predicted as Category 1	True Positives (TP)	False Positives (FP)	...	False Positives (FP)
Predicted as Category 2	False Positives (FP)	True Positives (TP)	...	False Positives (FP)
...	...	...	...	...
Predicted as Category N	False Positives (FP)	False Positives (FP)	...	True Positives (TP)

Table 4.5: Contingency table for ATSC and ATC tasks.

First, metrics are calculated for each category:

$$P_{\text{category}} = \frac{\text{TP}_{\text{category}}}{\text{TP}_{\text{category}} + \text{FP}_{\text{category}}} \tag{4.5}$$

$$R_{\text{category}} = \frac{\text{TP}_{\text{category}}}{\text{TP}_{\text{category}} + \text{FN}_{\text{category}}} \tag{4.6}$$

$$F1_{\text{category}} = 2 \cdot \frac{P_{\text{category}} \cdot R_{\text{category}}}{P_{\text{category}} + R_{\text{category}}} \tag{4.7}$$

Averaging Methods for Overall Metrics: To evaluate the performance of the system across all categories, two common averaging methods are used:

- **Micro-Averaged Metrics**:
  - Aggregate TP, FP, and FN across all categories and compute precision, recall, and F1-score based on the aggregated counts:

$$P_{\text{micro}} = \frac{\sum_{i=1}^{N} \text{TP}_i}{\sum_{i=1}^{N} (\text{TP}_i + \text{FP}_i)} \tag{4.8}$$

$$R_{\text{micro}} = \frac{\sum_{i=1}^{N} \text{TP}_i}{\sum_{i=1}^{N} (\text{TP}_i + \text{FN}_i)} \tag{4.9}$$

$$F1_{\text{micro}} = 2 \cdot \frac{P_{\text{micro}} \cdot R_{\text{micro}}}{P_{\text{micro}} + R_{\text{micro}}} \tag{4.10}$$

- **Macro-Averaged Metrics**:
  - Calculate precision, recall, and F1-score for each category independently, then take the average:

$$P_{\text{macro}} = \frac{1}{N} \sum_{i=1}^{N} P_{\text{category}_i} \tag{4.11}$$

$$R_{\text{macro}} = \frac{1}{N} \sum_{i=1}^{N} R_{\text{category}_i}, \tag{4.12}$$

$$F1_{\text{macro}} = \frac{1}{N} \sum_{i=1}^{N} F1_{\text{category}_i} \tag{4.13}$$

High precision for a sentiment class (e.g., Positive) means fewer aspects are incorrectly classified as Positive.
High recall for a class indicates most actual sentiments of that class are correctly classified.
F1-score summarizes the overall performance for each sentiment class.

4.2 Books retrieval

4.2.1 Search engine

In Section 2 we discussed the various search engines available for information retrieval. Since Indri and Terrier are typically used in academic research and have also been employed in previous years to address this task, we decided to experiment with Elasticsearch. Moreover, after analyzing tools for implementing hybrid search, we concluded that Elasticsearch is the most suitable tool for this task.

4.2.2 Data Preparation

For our chosen search engine, we prepare the data accordingly. Detailed information on how book collections are indexed into the database is described in Chapter **??**.

We draw attention to the fact that we carefully filter the data that we enter into the database. And if we extract additional information from the data with the user's request (Aspect-Sentiment-Category triplets from ABSA), then from the book data we select only the necessary tags.

Requests Data

Changes made to user request data:

After analyzing the tags `<title>` and `<group>`, it was decided to combine them into one field, since they often complement each other in meaning, but are not interchangeable.

The `<types>` and `<genres>` tags were originally nested tags or could include a list of multiple types/groups. We decided to flatten both of them to list to apply full-text search.

Tag `<narrative>` was removed, since we use it only to extract information for ABSA. In general, narrative text is long and besides aspects include nonrelevant information.

New tags were added: `sentiment` tags include dictionaries, where key is a category and value is a list of aspects related to this category. There might be 0, 1 or more aspects per each category.

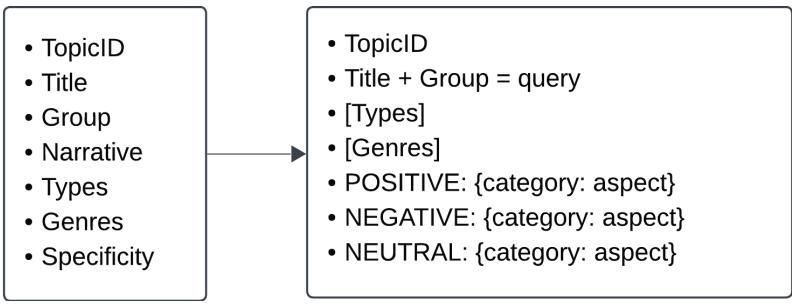

Figure 4.6: Requests Data. Tags selection.

Books data

A detailed description of working with a book collection will be presented in Chapter **??**

The book collection consists of queries generated by users (from the LT forum) and metadata from the Amazon website. Metadata is automatically generated individual pointers by which you can quickly find a book. We use such data for searching by keywords.

Since we must also process information generated by users, all text information is converted to vector form, since in free-style queries it is necessary to understand the semantic closeness of texts.

4.2.3 Evaluation

To describe the evaluation of the IR system and the logic of metric selection, we refer to Professor Manning's book [28], which is a fundamental basis in the field of information retrieval. In view of the data provided, to evaluate the results contained in the *.qrel* files, we use non-ranked list scores, where for each query we only know the list of relevant documents. Then, by default, if a document is not in this list, it is considered nonrelevant. We solve the binary classification problem *"document d is relevant for query q"*. We focus on fundamental metrics that assess the quality of the retrieved document set: Precision, Recall, and the combined F-Measure. These metrics were described in the section above, but in this part of the work we formulate them from the point of view of information retrieval. These metrics are computed using the number of relevant and non-relevant documents retrieved that are organized as shown in the contingency table below.

	Relevant	**Nonrelevant**
Retrieved	true positives (TP)	false positives (FP)
Not retrieved	false negatives (FN)	true negatives (TN)

Table 4.6: Contingency table for IR systems.

Precision

If we consider the proportion of correctly predicted positive objects among all objects predicted by the positive class, we obtain a Precision metric.

$$\text{Precision} = \frac{\#(\text{relevant items retrieved})}{\#(\text{retrieved items})} = P(\text{relevant} \mid \text{retrieved}) = \frac{TP}{TP + FP} \quad (4.14)$$

Intuitively, the metric shows the share of relevant documents among all those found by the classifier. The fewer false positives the model allows, the higher its Precision will be.

Recall

If we consider the proportion of correctly found positive objects among all objects of the positive class, then we obtain a Recall metric.

$$\text{Recall} = \frac{\#(\text{relevant items retrieved})}{\#(\text{relevant items})} = P(\text{retrieved} \mid \text{relevant}) = \frac{TP}{TP + FN} \quad (4.15)$$

Intuitively, the metric shows the proportion of documents found out of all relevant ones. The fewer false negatives, the higher the recall of the model.

F1-score

It is convenient to compare solutions when their quality is expressed by a single number. In the case of the Precision-Recall pair, there is a popular way to combine them into a single metric – take their harmonic mean – to get F1-score.

$$\text{F1} = 2 \cdot \frac{\text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}} = \frac{\text{TP}}{\text{TP} + \frac{\text{FP}+\text{FN}}{2}} \quad (4.16)$$

The F1 measure assumes equal importance for Precision and Recall.

Accuracy (not suitable)

It may seem intuitive to evaluate performance using accuracy, defined as the fraction of correct classifications:

$$\text{Accuracy} = \frac{\#(\text{correctly classified documents})}{\#(\text{total documents})} = \frac{TP + TN}{TP + FP + FN + TN} \quad (4.17)$$

This metric is not suitable for information retrieval due to the typically skewed nature of datasets, where the vast majority of documents are non-relevant. Optimizing for accuracy would lead to trivial solutions where all documents are classified as non-relevant, thus failing the purpose of retrieval.

Overall, by employing Precision, Recall and F1-score, we ensure that the evaluation captures both the relevance and completeness of the retrieval process, enabling meaningful comparisons and system optimization.

5

Implementation. Sentiment Analysis

Let us recall that aspect-based sentiment analysis in this work includes 3 standard stages: aspect term extraction, aspect category detection and aspect sentiment classification. Unlike similar ABSA problems, we do not need to define a word or expression expressing sentiment (Aspect-Opinion Pair Extraction task). And since in the future, the search will be performed precisely by aspect term, the most important thing is the correct extraction of the aspect itself. This chapter is devoted to working with user queries, from which we will obtain subjectivity.

In the following sections we describe experiments with ABSA sub-tasks. We first explain the approaches, and only at the end compare and analyze results altogether. The data that is used to train the models and calculate the metrics was described in Chapter 4, so this chapter pays minimal attention to the data (only if we have to change the structure of the data for a particular method). We first describe the experiments and at the end of this chapter (Section 5.3 we evaluate different approaches.

5.1 Aspect Terms Extraction and Sentiment Classification

The first step in determining subjectivity in user queries is to identify the aspects towards which the user expresses one or another point of view.

5.1.1 Rule-based extraction

The most trivial way to identify aspects is to define lexical rules by which they will be determined. In this case, it is defined by parts of speech (POS). Typically, aspects are taken to be nouns, proper nouns, pronouns, and their composite forms. To identify opinion terms, one simply uses an adjective or participle grammatically linked to the identified group of words.

Despite the naivety of this method, it can still be used for a variety of tasks with the simplest texts. Its advantage lies in the ability to independently define rules relevant to the task: for example, it is possible to explicitly specify that only proper nouns can serve as aspects, while nouns cannot. Additionally, this is the most lightweight approach, as it does not involve the use of complex machine learning algorithms.

For our task, we define a set of parts of the speech that we think might be potential aspects and a set of dependencies. So, the general rule is: *if a* $word_1$ *(defined POS) is connected to another* $word_2$ *(defined POS) by a defined dependency relation, then we can consider* $word_1$ *an aspect term and* $word_2$ *an opinion term.* We refer to a recent paper [6] to define some syntactic rules but adjust them for our specific task.

Table 5.1 shows the connection scheme.

Dependency Relation	Aspect Term	Opinion Word
$AT{-}DEP{-}O$	AT	O
$AT{-}DEP_1{-}M{-}DEP_2{-}O$	AT	O
$AT_1{-}DEP_3{-}AT_2$	AT_1, AT_2	–

Table 5.1: Dependency relations rules for ABSA [6]

Aspect terms and Opinion can be one of the following POS:

$$AT \in \{\text{noun, proper noun}\}$$
$$O \in \{\text{adjecitve, verb, adverb}\}$$
$$M \in \{\text{coordinating conjunction}\}$$

We define dependencies and aspect terms as follow:

$$DEP \in \{\text{amod, nsubj, xcomp, obl, obj, nmod, dep}\}$$
$$DEP_1 \in \{\text{amod, nsubj, nmod}\}$$
$$DEP_2 \in \{\text{amod, nsubj, xcomp, advmod, nmod}\}$$
$$DEP_3 \in \{\text{conj, compound}\}$$

where

amod: Adjectival modifier (e.g., "blue" in "blue car").

nsubj: Nominal subject (e.g., "She" in "She runs").

xcomp: Open clausal complement (e.g., "to swim" in "He wants to swim").

compound: Compound modifier (e.g., "Chocolate cake" - "chocolate" modifies "cake.")

obl: Oblique nominal (e.g., "in the park" in "He runs in the park").

obj: Object (e.g., "apples" in "She eats apples").

nmod: Nominal modifier (e.g., "of the book" in "The cover of the book").

dep: Unspecified dependency (a fallback for undefined relations).

advmod: Adverbial modifier (e.g., "quickly" in "She runs quickly").

conj: Conjunct (e.g., "and" in "John and Mary").

For syntactic analysis, the `SpaCy` library [29] was used: part of speech was determined using Part-of-Speech tagging, and dependencies between words were analyzed using Dependency Parsing.

As expected, the use of lexical tools produced both correct and incorrect results. Figure 5.1 shows an example where the phrase "gardening book," defined by the relation $NOUN \xrightarrow{\text{compound}} NOUN$, is correctly attributed to an aspect, allowing the genre of the book to be identified.

An incorrect aspect identification is shown in Figure 5.2. From this sentence, we expect to identify the author's name, "Michael Connelly" as the aspect. The noun "fan" should indicate that the user has a POSITIVE sentiment toward this author. However, since "Michael Connelly" was parsed as a modifier of "fan," the aspect is identified as "Michael Connelly fan." Moreover, such aspect identification leads to an incorrect sentiment determination, as "fan" will not be attributed to the Opinion Term.

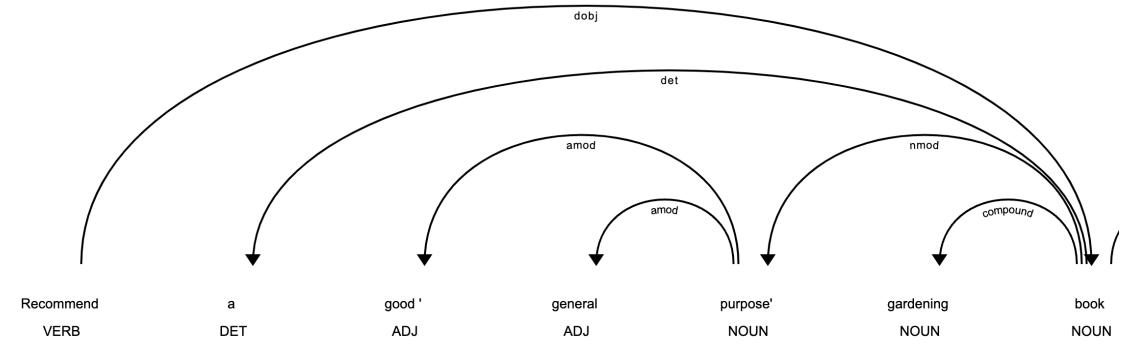

Figure 5.1: Correct syntactic ATE.

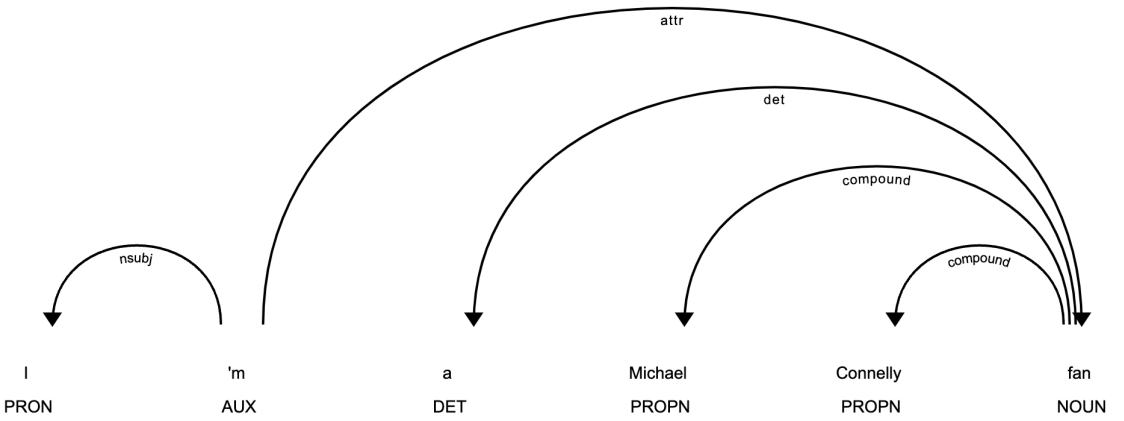

Figure 5.2: Incorrect syntactic ATE.

Interim conclusion

We have tried this approach out of curiosity — to see if it was possible to write rules for dependency parsing tailored to a specific domain to achieve competitive results without relying on significant computational resources. As expected, the results were low 5.4. We decided to stop exploring syntactic rules and not consider this option as a viable method for extracting Opinion Terms and their subsequent classification.

5.1.2 Extraction using LLM

Referring to the same paper as in the previous section, we now try to look at the aspect extraction task as a text-to-text task. The T5 model [9] is suitable for this. T5 (Text-to-Text Transfer Transformer) is an encoder-decoder transformer model. The encoder processes the input text

to understand its meaning, while the decoder generates the desired output text. It handles a wide range of language tasks by converting them all into a single format: text input and text output. Inputs and outputs for tasks are represented as text, whether it's translation, summarization, classification, or question answering.

At the moment, the T5 model has many improved models based on it. The most popular models are FLAN-T5 [30]. FLAN-T5 uses the same number of parameters as the original T5 model, but in addition, it has been trained on more than 1000 additional tasks and works with a larger number of languages. There are such subtypes as flan-t5-small, flan-t5-base and flan-t5-large. We will work with the flan-t5-base model as it is not the biggest model in terms of the number of parameters that give good results.

Data Structure

Since we want to get a list of aspects for each sentence, we can pass a prompt (instruction on what to do) to the model along with the sentence from which we want to extract aspects. Authors [6] suggest the prompt described in Figure 5.3.

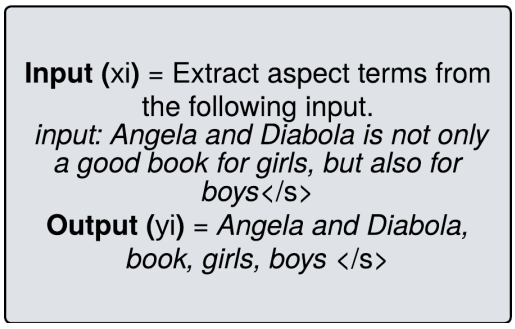

Figure 5.3: LLM Prompt for ATE.

We will use the dataset described in Section 4.1.2. For each sentence in the labeled data, we have a list with 0, 1, or more aspects (Table 5.2).

sentence	**aspects**
This is a very good book that answers the many questions of young adults as they are faced with them	[book, young adults]
My two year old loves the book	[two year old, book]

Table 5.2: Example of data for aspect terms extraction with LLM.

Since we want to pass text data to the model, it must be in a format that the model understands - that is, numerical. To do this, we generate vector representations of sentences, which are also called embeddings. Since we are using the flan-t5-base model, embeddings are also calculated using the tokenizer used by the same model. Next, we pass the vectorized prompt along with the sentence to the T5ForConditionalGeneration class, which is provided by the HuggingFace library. "T5ForConditionalGeneration - is a T5 Model with a language modeling head on top", as described on the library page.

Zero-shot generation

First, we test how well the model performs the task without additional information.

Zero-shot generation refers to the ability of the model to generate outputs for tasks it has not explicitly been fine-tuned for. Instead, it uses its pretrained knowledge to perform new tasks based on textual instructions or prompts. In this case, we simply pass the model a prompt with sentences from the test set to get the result.

After manually checking several results, we noticed that the model was doing very poorly. Here are a couple of examples:

- Input = Extract aspect terms from the following input. Input: A well-written novel by Lynn Reid Banks weaves good and evil into a touching and unforgettable novel.
  output = []
- Input = When finished, you will be so familiar with these characters, that you will wish there was a second book!
  output = [you]

Results calculated on the full dataset can be found in Section 5.3.1

Fine-tuned model

The model was additionally fine-tuned with available train data described in Section 4.1.2. We took flan-t5-base for embedding representation as well to compare zero-shot and fine-tuned models with the same vectors. We fine-tuned the model on 16 epochs and chose the best at the end of the process.

Thanks to additional training, some aspects have become better defined, but the overall result still leaves much to be desired:

- **Input** = Extract aspect terms from the following input. Input: A well-written novel by Lynn Reid Banks weaves good and evil into a touching and unforgettable novel.
  output = [novel]
- **Input** = When finished, you will be so familiar with these characters, that you will wish there was a second book!
  output = [you, characters]

Interim conclusion

We managed to improve the results compared to syntactic analysis, but not all aspects are extracted correctly yet. We think that the T5 model needs more data for effective additional training. Moreover, the annotated data is collected only for 40 books, which means that many aspects are repeated, which does not allow the model to learn to identify different aspects.

We also think that this method is very resource-intensive. If the first part of the task should extract all aspects from a sentence in one call to the model, then to determine the sentiment and category, you need to call the model for each aspect separately. For example, with this prompt:

Input (xi) = Given the aspect term and the sentence. Predict if the aspect term in the sentence has a positive, negative or neutral sentiment expressed on it.
aspect term: Angela and Diabola
sentence: Angela and Diabola is not only a good book for girls, but also for boys</s>
Output (yi) = positive</s>

Figure 5.4: LLM Prompt for ATE.

Therefore, we consider the model unsuitable for ASC and ATC tasks. In the next section, we will try another variant based on Sentence-Transformers.

5.1.3 Extraction using SetFit Framework

Over the past two years, two frameworks have emerged that provide tools specifically for ABSA tasks: PyABSA and SetFit. The first offers a wide range of tools to automatically annotate datasets, format data for subsequent work, and address various ABSA subtasks (ATE, OTE,

ATSC, ATC). However, its drawback is the lack of support for fine-tuning the model for specific domains to extract aspects. It relies on the FAST LSA V2 model for embedding calculation without the option for replacement. The framework only allows fine-tuning the second part — sentiment classification.

Given that our task involves many non-standard aspects (such as authors' names or book titles), we opted for the other modern framework, SetFit [12]. This framework allows for fine-tuning vector representations of textual data for a specific domain with a minimal amount of annotated data.

Data Structure

This framework requires a specific data structure, so our data, instead of storing multiple aspects for a single sentence, were flattened by aspects. Now, each aspect corresponds to its own entry in the dataset, and to avoid confusion with the aspect order, the data were supplemented with an attribute called 'ordinal'.

text	span	label	ordinal
This is a very good book that answers the many questions of young adults as they are faced with them	book	positive	0
My two year old loves the book	book	neutral	0

Table 5.3: Example of data for aspect terms extraction with setFit.

ABSA with SetFit

SetFit is based on fine-tuning pre-trained Sentence Transformers for specific tasks using few-shot examples. For ABSA, this involves generating aspect-sentiment pairs and using contrastive learning for fine-tuning.

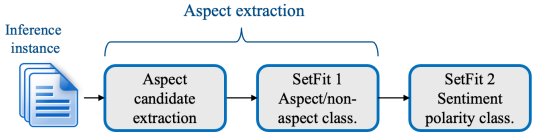

Figure 5.5: SetFit for ABSA.

By using SetFit, we leverage its capability to fine-tune a pre-trained Sentence Transformer model to a target domain with only a limited number of labeled examples. The process relies on contrastive learning, where samples are presented in pairs (e.g., "aspect:sentence" pairs) that the model must learn to distinguish based on their labels. This approach reshapes the representation space so that valid aspect-sentiment combinations cluster closely together, while non-aspects or differing sentiment categories remain clearly separated.

Concretely, once potential aspects are extracted (e.g., via SpaCy), each candidate aspect is paired with the original sentence. Then, the task of aspect extraction is transformed into a binary classification problem. A pair is formed in the format 'potential aspect:sentence'. If the potential aspect matches the actual aspect (known during the training phase), it is assigned a label of '1'. If the aspect extracted using SpaCy is not actually an aspect, it is assigned a label of '0'.

These pairs serve as inputs to the model, which then adjusts its embedding space so that correct aspect terms align more tightly with their source sentences. During training, positive examples (aspect terms known to be correct) are pulled closer, and negative examples (incorrect or irrelevant terms) are pushed away. Similarly, when dealing with sentiment classification, each "aspect:sentence" pair is associated with a sentiment category, allowing the model to cluster instances with identical sentiments together.

Thus, after annotation, the framework generates texts in the following format:

> British woman : he book is by a British woman, the characters are mainly British;
> label '1'
>
> level : The interest level is rated at grade five through eight;
> label '0'

Sentences for ATSC are formed in a similar way. Only instead of 0 or 1 there will be a sentiment class.

A key strength of SetFit lies in its minimal data requirements. Even with a very small set of annotated examples, the model can exploit these contrastive pairs to refine its embeddings. This adaptability makes it ideal for domains where labeled data may be scarce, inconsistent, or specialized. For instance, aspects like unusual author names or unique book titles can be effectively integrated into the model's representation space, even if they occur infrequently, simply by including them as labeled examples.

During inference, the process is straightforward. For each new sentence, candidate aspects are extracted and paired with the sentence. The fine-tuned model's embeddings then help deter-

mine which candidates are genuine aspects and, subsequently, identify their sentiments. This unified approach simplifies what might otherwise be a more fragmented pipeline: instead of building separate modules for aspect recognition and sentiment analysis, both tasks are integrated into a single embedding-based framework.

Compared to some earlier frameworks—where domain adaptation often required extensive modifications or was limited in scope—SetFit's method of contrastive fine-tuning offers a cleaner, more efficient solution. As a result, it can easily be adjusted whenever new aspects or sentiment classes emerge, keeping the model flexible and up-to-date for ongoing ABSA requirements.

Aspect Extraction and Sentiment Classification

Thanks to the module `setfit[absa]` we can fine-tune 2 models at once. The module provides ABSAModel class that includes AspectModel class and PolarityModel class. We use these classes and experiment with different sentence-transformers models for embeddings calculation. We referred to the models leaderboard [31] to chose not too large, but still efficient models.

We compare 3 models:

- **all-MiniLM-L6-v2** [21]: a smaller and faster distilled version of BERT. It has 6 transformer layers. Best for general-purpose sentence embeddings when speed and low resource usage are critical.
- **all-mpnet-base-v2** [32]: a model blending BERT and Transformer-XL ideas, improving context understanding and sentence-level embeddings. Suitable for tasks needing higher precision, like similarity search, ranking, or dense retrieval. Preferred when more compute resources are available.
- **BAAI/bge-small-en-v1.5** [33]: a compact embedding model, optimized for embedding generation with smaller model size and custom training. Focused on embeddings for retrieval and search tasks.

5.2 Aspect Terms Categorization

After achieving impressive results in sentiment classification using the SetFit framework, we decided to apply the same approach to classify aspect categories. This decision was motivated

by the fact that this task involves even more classes while still having limited data. The data format we use remains the same, with the only difference being that aspect categories now serve as the classes. Let us recall that there are a total of 7 categories, and they are evenly split between the training and test sets.

Thus, our texts for category classification are structured as follows:

> boys : Well-liked by both boys and girls in my middle school classes, this series is favored by boys;
> label: BOOK#AUDIENCE

> MISTER MONDAY : MISTER MONDAY is the first in a new series by Garth Nix, author of THE SEVENTH TOWER;
> label: BOOK#TITLE

> the Will : Monday, one of the seven trusties entrusted with the Will while GA is off gallivanting about;
> label: CONTENT#CHARACTERS

Each entry is formatted as 'aspect term: sentence'. The label corresponds to one of the 7 predefined categories.

Unlike the previous one, this task is single-stage, so we simply use the SetFitModel class with additional training for our task. As a model for obtaining embeddings, we took the same one as in the previous part of the work - all-mpnet-base-v2, with which we managed to get the best metrics.

5.3 Evaluation

5.3.1 Quantitative analysis

All metrics were calculated on the data described in Section 4.1.2.

Approach	Embedding model	Fine-tuned	Precision	Recall	F1	#Params
Syntactic Parsing	-	-	25.48	36.28	29.94	-
LLM	flan-t5-base	no	20.35	22.63	21.43	248M
LLM	flan-t5-base	yes	56.48	53.37	54.88	248M
SetFit	all-MiniLM-L6-v2	yes	87.55	74.58	80.55	22.7M
SetFit	**all-mpnet-base-v2**	yes	89.21	82.44	85.69	109M
SetFit	BAAI/bge-small-en-v1.5	yes	81.18	79.45	80.31	33.4M

Table 5.4: ATE task. Results comparison.

Syntactic Parsing serves as a baseline, employing a traditional approach for aspect extraction. It achieves low precision, recall, and F1-score. This outcome indicates that while syntactic methods might capture a wide range of aspect terms (higher recall than LLM without fine-tuning), they lack precision and robustness.

The results demonstrate the impact of fine-tuning on LLMs. Without fine-tuning, flan-t5-base underperforms compared to syntactic parsing. However, fine-tuning significantly boosts performance across all metrics, with an F1-score increase of 33.45 points.

SetFit with all-mpnet-base-v2 emerges as the top performer, achieving the highest F1-score (85.69), striking an excellent balance between accuracy and parameter efficiency. SetFit models consistently outperform LLMs in this task, showcasing their capability for effective and efficient aspect term extraction.

Metric	Precision	Recall	F1-Score	Support
Positive	0.830	0.802	0.816	91
Negative	0.933	0.973	0.953	485
Neutral	0.940	0.867	0.902	233
Accuracy		0.923		
Macro Avg	0.901	0.881	0.890	809
Weighted Avg	0.923	0.923	0.923	809

Table 5.5: ATSC task. Metrics.

The classification for positive sentiment achieves relatively lower scores compared to other classes, with the lowest precision, recall, and F1-score. This indicates some difficulty in accurately identifying positive sentiment, likely due to a smaller support size (91), which may limit model generalizability for this class.

Negative sentiment classification shows the highest performance across all metrics. The high recall (0.973) demonstrates the model's effectiveness in correctly identifying most instances of negative sentiment, with precision (0.933) ensuring minimal false positives. The large support (485) provides sufficient data to achieve reliable classification.

Neutral sentiment classification performs well, with high precision (0.940), suggesting minimal false positives for neutral predictions. However, recall (0.867) is slightly lower, indicating some neutral instances are misclassified. Despite this, the F1-score (0.902) remains strong, benefiting from a balanced support size.

Metric	Precision	Recall	F1-Score	Support
CONTENT#CHARACTERS	0.593	0.561	0.577	57
CONTENT#PLOT	0.581	0.600	0.591	125
BOOK#GENERAL	0.720	0.776	0.747	116
BOOK#AUTHOR	0.484	0.652	0.556	46
BOOK#GENRE	0.814	0.875	0.843	40
BOOK#TITLE	0.844	0.818	0.831	192
BOOK#AUDIENCE	0.900	0.811	0.853	233
Accuracy		0.752		
Macro Avg	0.705	0.728	0.714	809
Weighted Avg	0.762	0.752	0.755	809

Table 5.6: ATC task. Metrics.

The model excels in categories with larger support, such as "BOOK#AUDIENCE" and "BOOK#TITLE," achieving high precision, recall, and F1-scores.

Performance is weak in smaller categories like "CONTENT#CHARACTERS" and "BOOK#AUTHOR," likely due to insufficient examples for training.

The macro average shows the model's overall ability across all categories, but the weighted average indicates its reliance on better-performing, larger categories.

5.3.2 Qualitative analysis

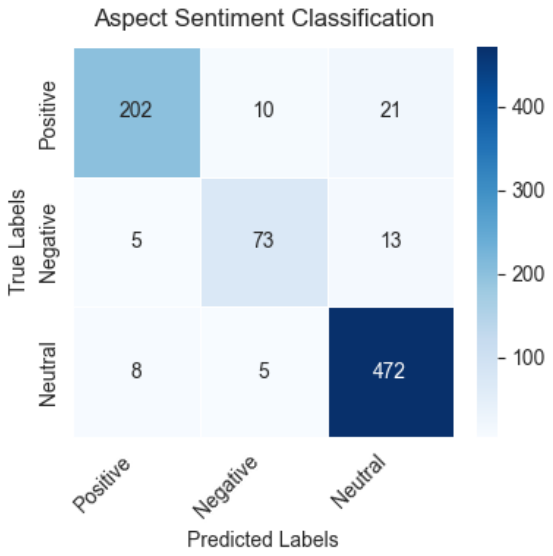

Figure 5.6: Confusion matrix for sentiment classification.

The model demonstrates strong overall performance, particularly in recognizing neutral and negative sentiments. However, it shows some difficulty distinguishing between positive and neutral classes, likely due to subtle or mixed sentiment expressions. By addressing these challenges, the model's robustness and interpretability can be further enhanced.

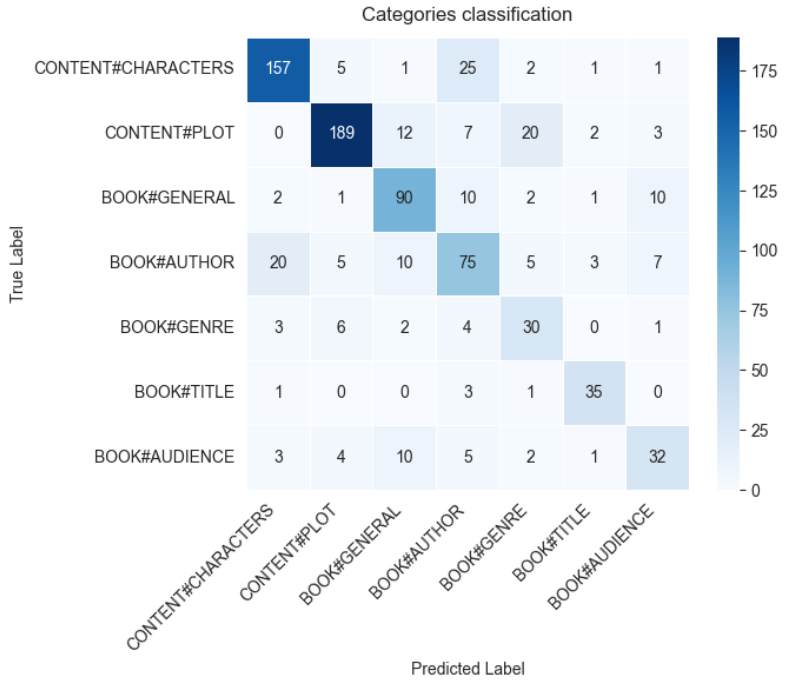

Figure 5.7: Confusion matrix for terms categorization.

”CONTENTCHARACTERS”: The model performs strongly for this category, with most instances classified correctly. However, there is a notable confusion with ”BOOKAUTHOR,” likely due to overlapping semantic features (e.g., authors often write about characters).

”CONTENTPLOT”: The model demonstrates strong performance but struggles with distinguishing ”PLOT” from related categories like ”CHARACTERS” and ”AUTHOR.” These errors may stem from instances where plot descriptions mention characters or authors, creating ambiguity.

”BOOKAUTHOR”: This category is relatively challenging for the model, with frequent confusion between ”AUTHOR” and ”CHARACTERS.” This suggests that the model struggles to differentiate between the author as a creator and characters within the content.

The model shows strong categorization performance for clear and well-defined categories but struggles with ambiguous or semantically overlapping ones.

6 Conclusion

In this master's thesis, we implemented a process that allowed us to extract relevant documents from a book collection for user queries in a conversational form, where users usually express their opinions, share impressions on some aspect.

The first part of the work was devoted to sentiment analysis, or rather its subtype - Aspect-based Sentiment Analysis. In this part, we were able to extract subjective user assessments in order to then improve the search engine for extracting books. This approach allowed us to recommend books/authors or genres that the user likes, and not include in the recommendations those books that are not interesting to him.

This thesis is a new solution to an already old problem. In this master's thesis, the problem was solved using modern systems (for example, the Elasticsearch search engine), as well as using new machine learning models.

References

[1] M. Koolen, G. Kazai, J. Kamps, A. Doucet, and M. Landoni, "Overview of the inex 2011 books and social search track," in *Focused Retrieval of Content and Structure: 10th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2011, Saarbrücken, Germany, December 12-14, 2011, Revised Selected Papers 10*. Springer, 2012, pp. 1–29.

[2] Amazon. Amazon books. [Online]. Available: https://www.amazon.com/b/ref=txtb_surl_textbooks/?node=465600

[3] P. Balaji, O. Nagaraju, and D. Haritha, "Levels of sentiment analysis and its challenges: A literature review," in *2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDAC)*. IEEE, 2017, pp. 436–439.

[4] I. A. Kandhro, F. Ali, M. Uddin, A. Kehar, and S. Manickam, "Exploring aspect-based sentiment analysis: an in-depth review of current methods and prospects for advancement," *Knowledge and Information Systems*, pp. 1–31, 2024.

[5] M. Pontiki, D. Galanis, J. Pavlopoulos, H. Papageorgiou, I. Androutsopoulos, and S. Manandhar, "SemEval-2014 task 4: Aspect based sentiment analysis," in *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, P. Nakov and T. Zesch, Eds. Dublin, Ireland: Association for Computational Linguistics, Aug. 2014, pp. 27–35. [Online]. Available: https://aclanthology.org/S14-2004

[6] G. Negi, R. Sarkar, O. Zayed, and P. Buitelaar, "A hybrid approach to aspect based sentiment analysis using transfer learning," *arXiv preprint arXiv:2403.17254*, 2024.

[7] K. Sun, R. Zhang, S. Mensah, Y. Mao, and X. Liu, "Aspect-level sentiment analysis via convolution over dependency tree," in *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, 2019, pp. 5679–5688.

[8] W. Zhang, Y. Deng, X. Li, Y. Yuan, L. Bing, and W. Lam, "Aspect sentiment quad prediction as paraphrase generation," *arXiv preprint arXiv:2110.00796*, 2021.

[9] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," 2023. [Online]. Available: https://arxiv.org/abs/1910.10683

[10] C. Wu, Q. Xiong, H. Yi, Y. Yu, Q. Zhu, M. Gao, and J. Chen, "Multiple-element joint detection for aspect-based sentiment analysis," *Knowledge-Based Systems*, vol. 223, p. 107073, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705121003361

[11] H. Yang and K. Li, "Pyabsa: open framework for aspect-based sentiment analysis," *arXiv preprint arXiv:2208.01368*, vol. 10, 2022.

[12] L. Tunstall, N. Reimers, U. E. S. Jo, L. Bates, D. Korat, M. Wasserblat, and O. Pereg, "Efficient few-shot learning without prompts," *arXiv preprint arXiv:2209.11055*, 2022.

[13] K. A. Hambarde and H. Proenca, "Information retrieval: recent advances and beyond," *IEEE Access*, 2023.

[14] Indri. Indri. [Online]. Available: https://www.lemurproject.org/indri/

[15] Terrier. Terrier. [Online]. Available: http://terrier.org/

[16] Elasticsearch. Elasticsearch. [Online]. Available: https://www.elastic.co/elasticsearch

[17] F. Adriaans, J. Kamps, and M. Koolen, "University of amsterdam at inex 2011: Book and data centric tracks," in *Copyright cc2011 remains with the author/owner (s). The unreviewed pre-proceedings are collections of work submitted before the December workshops. They are not peer reviewed, are not quality controlled, and contain known errors in content and editing. The proceedings, published after the Workshop, is the authoritative reference for the work done at INEX.*, 2011, p. 36.

[18] T. Bogers, K. W. Christensen, and B. Larsen, "Rslis at inex 2011: social book search track," in *Focused Retrieval of Content and Structure: 10th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2011, Saarbrücken, Germany, December 12-14, 2011, Revised Selected Papers 10*. Springer, 2012, pp. 45–56.

[19] L. Ermakova, A.-G. Bosser, T. Miller, T. Thomas, V. M. P. Preciado, G. Sidorov, and A. Jatowt, "Clef 2024 joker lab: Automatic humour analysis," in *Advances in Information Retrieval: 46th European Conference on Information Retrieval, ECIR 2024, Glasgow, UK, March 24–28, 2024, Proceedings, Part VI*. Berlin, Heidelberg: Springer-Verlag, 2024, p. 36–43. [Online]. Available: https://doi.org/10.1007/978-3-031-56072-9_5

[20] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[21] HuggingFace. sentence-transformers/all-minilm-l6-v2. [Online]. Available: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

[22] W. Zhang, X. Li, Y. Deng, L. Bing, and W. Lam, "A survey on aspect-based sentiment analysis: Tasks, methods, and challenges," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 11, pp. 11 019–11 038, 2022.

[23] S. U. S. Chebolu, F. Dernoncourt, N. Lipka, and T. Solorio, "A review of datasets for aspect-based sentiment analysis," in *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023, pp. 611–628.

[24] F. Barbieri, J. Camacho-Collados, L. Neves, and L. Espinosa-Anke, "Tweeteval: Unified benchmark and comparative evaluation for tweet classification," *arXiv preprint arXiv:2010.12421*, 2020.

[25] M. Pontiki, D. Galanis, H. Papageorgiou, I. Androutsopoulos, S. Manandhar, M. Al-Smadi, M. Al-Ayyoub, Y. Zhao, B. Qin, O. De Clercq *et al.*, "Semeval-2016 task 5: Aspect based sentiment analysis," in *International workshop on semantic evaluation*, 2016, pp. 19–30.

[26] P. Keung, Y. Lu, G. Szarvas, and N. A. Smith, "The multilingual amazon reviews corpus," *arXiv preprint arXiv:2010.02573*, 2020.

[27] T. Álvarez-López, M. Fernández-Gavilanes, E. Costa-Montenegro, and P. Bellot, "A proposal for book oriented aspect based sentiment analysis: Comparison over domains,"

in *Natural Language Processing and Information Systems: 23rd International Conference on Applications of Natural Language to Information Systems, NLDB 2018, Paris, France, June 13-15, 2018, Proceedings 23*. Springer, 2018, pp. 3–14.

[28] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[29] SpaCy. Spacy. [Online]. Available: https://spacy.io/

[30] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, E. Li, X. Wang, M. Dehghani, S. Brahma, A. Webson, S. S. Gu, Z. Dai, M. Suzgun, X. Chen, A. Chowdhery, S. Narang, G. Mishra, A. Yu, V. Zhao, Y. Huang, A. Dai, H. Yu, S. Petrov, E. H. Chi, J. Dean, J. Devlin, A. Roberts, D. Zhou, Q. V. Le, and J. Wei, "Scaling instruction-finetuned language models," 2022. [Online]. Available: https://arxiv.org/abs/2210.11416

[31] HuggingFace. leaderboard. [Online]. Available: https://huggingface.co/spaces/mteb/leaderboard

[32] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu, "Mpnet: Masked and permuted pre-training for language understanding," *Advances in neural information processing systems*, vol. 33, pp. 16 857–16 867, 2020.

[33] S. Xiao, Z. Liu, P. Zhang, and N. Muennighoff, "C-pack: Packaged resources to advance general chinese embedding," 2023.

Acknowledgments

In conclusion of this work I would like to express my gratitude to the unique Erasmus Mundus Joint Master's Program in Big Data Management and Analytics, its leader Esteban Zimányi, and local coordinators Oscar Romero and Massimiliano de Leoni.

Without the past two years of challenging but very interesting studies, I would not have been able to imagine working on such an exciting task in the field of natural language processing as a master's thesis. This research work would not have been possible without my supervisors Sébastien Fournier and Adrian-Gabriel Chifu from Laboratoire d'Informatique et des Systèmes. And also without the support and help of Professor Michele Rossi.

Finally, I would like to express my gratitude to my classmates and family for their moral support throughout my studies.