



POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Integrating Semantic and Keyword Search: A Transformer-Based Ap- proach for Content Discovery

TESI DI LAUREA MAGISTRALE IN
COMPUTER SCIENCE AND ENGINEERING
INGEGNERIA INFORMATICA

Author: **Sofia Martellozzo**

Student ID: 996215

Advisor: Prof. Paolo Cremonesi

Co-advisors: Riccardo Biondi, Federico Sallemi

Academic Year: 2023-24

Abstract

In the rapidly evolving domain of streaming platforms, the richness and complexity of content catalogs present both opportunities and challenges for advanced search technologies. The diversity in genre, style, and language, mirrors the content's origins and the platforms' global reach, enhancing user experience, but complicating content discoverability. This scenario sets the stage for our investigation into enhancing content discovery through innovative search methodologies. Our work introduces a novel system implementing a hybrid search mechanism. This approach combines the precision of keyword search with the depth of semantic understanding, offering a more human-like search experience where users can input queries as if conversing with another person. At the heart of our system is a custom embedding model, trained to capture the semantic nuances of user queries and retrieve content that not only matches the keywords but also aligns with the query's underlying intent. The development of our system is based on fine-tuning approach, allowing us to adapt and enhance the capabilities of an existing open-source embedding model for our specific use case. This methodology ensures our model's effectiveness in understanding and processing queries across a broad spectrum of languages, addressing the multilingual needs of global streaming platforms and their diverse user base. Structured as an online service, our solution is designed for easy integration by any streaming company, offering a scalable and adaptable tool to improve content discovery. Through this system, we aim to redefine user interaction with streaming platforms, enabling searches that are more intuitive, efficient, and responsive to the varied ways in which people communicate their content preferences.

Keywords: Semantic Search, Hybrid Search, Embeddings, Transformers, Fine-tuning, Information Retrieval

Abstract in lingua italiana

Nel dominio in rapida evoluzione delle piattaforme di streaming, la ricchezza e la complessità dei cataloghi di contenuti presentano sia opportunità che sfide per le avanzate tecnologie di ricerca. La diversità di genere, stile e linguaggio, che riflette le origini dei contenuti e la portata globale delle piattaforme, migliora l'esperienza dell'utente ma complica la scoperta dei contenuti. Questo scenario pone le basi per la nostra ricerca sul miglioramento della scoperta dei contenuti attraverso metodologie di ricerca innovative. Il nostro lavoro introduce un nuovo sistema che implementa un meccanismo di ricerca ibrido. Questo approccio combina la precisione della ricerca per parole chiave con la profondità della comprensione semantica, offrendo un'esperienza di ricerca più simile a quella umana, in cui gli utenti possono inserire le domande come se stessero conversando con un'altra persona. Il cuore del nostro sistema è un modello di embedding personalizzato, addestrato per catturare le sfumature semantiche delle domande degli utenti e recuperare i contenuti che non solo contengono le parole chiave, ma sono anche in linea con l'intento della domanda. Lo sviluppo del nostro sistema si basa su un approccio di fine-tuning, che ci permette di adattare e migliorare le capacità di un modello esistente e open-source che genera embedding per il nostro caso d'uso specifico. Questa metodologia garantisce l'efficacia del nostro modello nella comprensione e nell'elaborazione di domande in un ampio spettro di lingue, rispondendo alle esigenze multilingua delle piattaforme di streaming globali e della loro variegata base di utenti. Strutturata per essere un servizio online, la nostra soluzione è progettata per essere facilmente integrata da qualsiasi società di streaming, offrendo uno strumento scalabile e adattabile per migliorare la scoperta dei contenuti. Attraverso questo sistema, ci proponiamo di ridefinire l'interazione degli utenti con le piattaforme di streaming, consentendo ricerche più intuitive, efficienti e reattive ai diversi modi in cui le persone comunicano le loro preferenze in fatto di contenuti.

Parole chiave: Ricerca Semantica, Ricerca Ibrida, Embeddings, Transformers, Fine-tuning, Recupero di Informazioni

Contents

Abstract	i
Abstract in lingua italiana	iii
Contents	v
1 Introduction	1
1.1 Context	1
1.2 Scenario	2
1.3 Contributions	3
1.4 Structure of the thesis	4
2 State of the Art	5
2.1 Word Embeddings	5
2.2 Transformers	8
2.2.1 Vanilla Transformers	8
2.2.2 BERT	11
2.2.3 sBERT	14
2.3 Semantic Search	16
2.4 Fine-Tuning	18
2.5 Hybrid Search	22
3 Dataset	25
3.1 Industrial datasets	25
3.1.1 Industrial dataset 1	25
3.1.2 Industrial dataset 2	27
3.1.3 Industrial dataset 3	27
3.2 Synthetic query-answer dataset generation	31
3.2.1 Filtering and splitting	34

3.3	Human query generation	35
4	Methodology	37
4.1	Model selection	37
4.1.1	Approach	40
4.2	Similarity matrix	41
4.3	Fine-tuning strategies	45
4.3.1	Adapter fine-tuning	45
4.3.2	Traditional fine-tuning	46
4.3.3	LoRA fine-tuning	48
5	Implementation	51
5.1	Weaviate	52
5.2	Schema	53
5.3	Search configurations	59
6	Results	61
6.1	Metrics	61
6.1.1	Offline metrics	62
6.2	Visual results	65
6.3	Weaviate hybris search results	72
6.4	Fine tuned model results	73
6.5	Model selection results	76
6.5.1	Tests on dataset 1	77
6.5.2	Tests on dataset 2 and 3	84
7	Conclusions and future developments	89
7.1	Contributions	89
7.2	Limitations	90
7.3	Future Work	91
	Bibliography	93
A	Appendix A	97
A.1	Model selection	97
A.1.1	dataset 1	97

A.1.2 dataset 2 and 3	102
A.2 Fine-tuning	104
A.3 Hybrid search	106
B Appendix B	107
List of Figures	113
List of Tables	115
Acknowledgements	117

1 | Introduction

This thesis work has been developed over a period of six months during an internship program at ContentWise, a software company known for providing its clients with a comprehensive suite of tools and expertise necessary for crafting engaging experiences for end-users. As an innovator within the video streaming industry, ContentWise specializes in the development of advanced recommender systems, continuously exploring new technologies to enhance digital content interaction and personalization.

Building on this spirit of innovation, this thesis work aims to enhance ContentWise's offerings by integrating a novel service into its framework: hybrid search functionality. This novel service is designed to revolutionize the way users interact with streaming platforms, allowing them to conduct searches in natural language. By combining the results of semantic queries with traditional keyword-based search, this service aims to deliver a more intuitive and efficient search experience. This work not only aligns with ContentWise's mission of being a leader in cutting-edge technologies for the video streaming market but also sets a new benchmark for personalization and user engagement in digital content discovery.

1.1. Context

The rapidly evolving domain of streaming platforms, rich with diverse content ranging from films to series, presents a unique challenge and opportunity for enhancing content discoverability. Recognizing this, ContentWise has identified a critical need for a new service that enhances the way users find and interact with content. The proposed solution uses semantic search technology, addressing the limitations of traditional search methods and setting a new standard for content discovery.

This need for advanced search capabilities aligns with the broader research area of Natural Language Processing (NLP), a branch of Artificial Intelligence focused on facilitating meaningful and useful interactions between computers and humans through natural language. NLP encompasses a range of techniques and technologies for understanding,

interpreting, and generating human languages, laying the foundation for this thesis work. More precisely, the NLP's application in this research is Information Retrieval (IR), a subfield focused on extracting relevant information from extensive datasets in response to specific queries.

Semantic search is a particular aspect of IR. This approach involves comparing the embeddings of user queries with those of available content to identify items that are not just relevant but semantically aligned with the user's intentions. By leveraging advancements in embedding technologies, semantic search offers a refined and effective method for connecting users with the content they seek on streaming platforms.

1.2. Scenario

The contemporary landscape of streaming platforms, such as the well-known Netflix, Disney+, and Amazon Prime, offers a rich and complex domain for the application of advanced search technologies. Each streaming company has a distinct catalog of items, reflecting a diversity of content that varies in genre and style in addition to the languages used. These variations are influenced by the origins of the content and the geographical distribution of the platforms. This diversity, while enriching the user experience, also introduces challenges in content discoverability, particularly when users are navigating through vast catalogs.

Traditional recommender systems aim to enhance content discovery by guiding users toward elements that align with their interests and viewing history, by predicting the "ratings" and "preferences" of users. However, this approach may not always satisfy the users' specific needs or desires for new and specific content. Users often have precise content in mind, whether it be a theme, a mood, or a storyline, which cannot be easily articulated through simple keywords or genres. The volume of available content and the complexity of possible user queries necessitate a more sophisticated approach to search, a strategy that transcends the limitations of traditional keyword-based searches.

This is where semantic search emerges as a revolutionary solution, offering a more intuitive and human-like interaction with the platform while improving content discoverability. Importantly, semantic search is adept at overcoming common human errors such as typos or the use of abbreviations, further enhancing its effectiveness. However, a critical obstacle in implementing effective semantic search is the absence of embedding models specifically trained for this task.

To address these complexities, fine-tuning stands out as a promising strategy. By adapting

pre-trained models to the specific nuances of this task through additional training on smaller, targeted datasets, fine-tuning enables the enhancement of model performance in semantic understanding.

Additionally, the performance of models in Information Retrieval (IR) tasks does not guarantee their efficacy in understanding and processing multiple languages. This multilingual capability is crucial, not only for the linguistically diverse nature of content on streaming platforms but also due to user preferences for querying in one language over another. Achieving excellence in semantic search while ensuring multilingual comprehension presents a significant challenge.

1.3. Contributions

Recognizing the power of embedding models to deeply understand text, in this work we decided to adapt these models for their applicability in the domain of streaming platforms for the discovery of movies and series. Drawing from state-of-the-art models and techniques, we propose a novel approach that constructs a hybrid search system, combining semantic search with traditional keyword search mechanisms. This innovative fusion aims to leverage the strengths of both methodologies.

Central to our contribution is the development of our own embedding model, designed to perform queries independently of external services. This strategic move avoids additional costs and potential service discontinuity. By re-training an open-source model, we generate a customized solution for our specific task. This custom model is stored and managed internally, ensuring greater control over its application and optimization.

Nowadays, people are used to adopting a keyword-based search approach across various search engines. Inspired by the rapid increase in popularity of conversational AI technologies, such as chatGPT, our research is driven by the principle that a more human-like approach to machine interaction is highly preferable. This inspiration leads us to adopt similar principles of natural language understanding to revolutionize the way users interact with and navigate streaming platforms.

Through these contributions, this thesis addresses a practical need within a specific application domain and contributes to the research for the integration of human-centric AI technologies in everyday digital interactions. Our work exemplifies the potential of NLP techniques to enhance user experience, paving the way for future innovations in the field.

1.4. Structure of the thesis

This section aims to provide an overview of this document's structure:

- Chapter 2 presents an exploration of the state-of-the-art methodologies and technologies essential for this study, ensuring the essential knowledge to the reader.
- Chapter 3 details the datasets used and employed for training and evaluating our system, highlighting their features and operations to prepare and clean them.
- Chapter 4 explores the model selection phase, the implementation of similarity matrices, and fine-tuning strategies adopted for the development of our system.
- Chapter 5 discusses the final implementation and integration of our model into a proposed hybrid search system, operating as the final solution of our research.
- Chapter 6 offers the most significant results of our research, both through theoretical analysis and schematic representations, complemented by manual evaluations to mirror real-world application scenarios.
- Chapter 7 summarizes the contributions, constraints, and achievements of our study and proposes directions for future investigation.

2 | State of the Art

This chapter attempts to provide a thorough overview of the state of the art in Natural Language Processing (NLP) and Information Retrieval (IR). We explore the mechanisms, challenges, and potential of these technologies, from the fundamental discoveries in word embeddings and Transformer architectures to the sophisticated applications in semantic search and fine-tuning techniques. Our investigation highlights the advances and opens the way for further innovations in this rapidly evolving field.

2.1. Word Embeddings

In Natural Language Processing (NLP), the computer science and linguistic subfield that enables machines to understand human language, word embedding has a pivotal role. Machines work only with numeric elements, they are not able to work directly with words. Unlike images and audio, words are not represented automatically as digital signals. A numerical representation of a word is inadequate; for instance, representing the word "cat" as 37 and "dog" as 135 cannot say anything about each word or their relationship. Moreover, words and texts are sparse data: changing a word in a text can lead to a completely different meaning, diversely than images, which are dense data, in which changing one or few pixels does not change the sense of the image.

A simple solution is a one-hot representation (Li et al. [12]) of words, a sparse discrete vector. Given a text, the list of all the unique words forms a vocabulary, where each word appears as a binary vector of length equal to the vocabulary size. In each vector, the position of the corresponding word in the vocabulary is marked with a 1, and all the other positions are marked with a 0. For instance, with the sentence 'the cat sat on the mat' the vocabulary is ["the", "cat", "sat", "on", "mat"] and the word 'cat' is represented as [0, 1, 0, 0, 0]. The main problems of this technique arise from the loss of the order of the words and the context of the sentence; this representation cannot reflect the semantic meaning and relationship between the words in a sentence. Besides, the resulting vectors are sparse (composed of mostly zeros), and the dimension of the vectors grows with the vocabulary size, leading to memory and computation issues. The

result is an inefficient model, impractical to adopt with large vocabularies.

The N-gram statistic model [29][25] involves the relationship between words, representing the probability of a sentence. This model considers n consecutive words; in a sentence, a '1-gram' takes into consideration each word, a '2-gram' takes every sequence of two consecutive words, and so on. N-gram predicts the probability of a word based on the occurrences of its preceding $N-1$ words, from the Markov assumption that the probability of a word occurring in a text depends on the previous words. The main drawbacks of this model are: as the value n increases, the number of possible combinations in the vocabulary increases, leading to storage and training issues (the phenomenon called 'curse of dimensionality'[2]). Secondly, the $N-1$ window blocks the possibility of capturing longer dependencies in the language.

The breakthrough came with the advent of dense word embedding. Embeddings are vectors that encode semantic and syntactic meanings of words, mapping textual words into a multi-dimensional continuous space. In particular, word embedding models convert the semantics of words into geometric properties. Vectors representing similar words are positioned closely in the feature space, and the difference between two embeddings denotes the relationship between the corresponding words. For instance, it is possible to derive analogies in the relations between "man: king = woman: queen" from the difference between the vectors for "queen", "king", and "woman", "man"; since they share the same direction, we have two similar relational meaning denoted by 'gender' purple arrows in the Figure 2.1. Similar to the result obtained by subtracting "king", "man", and "queen", "woman" as the 'royal' yellow arrow in Figure 2.1.

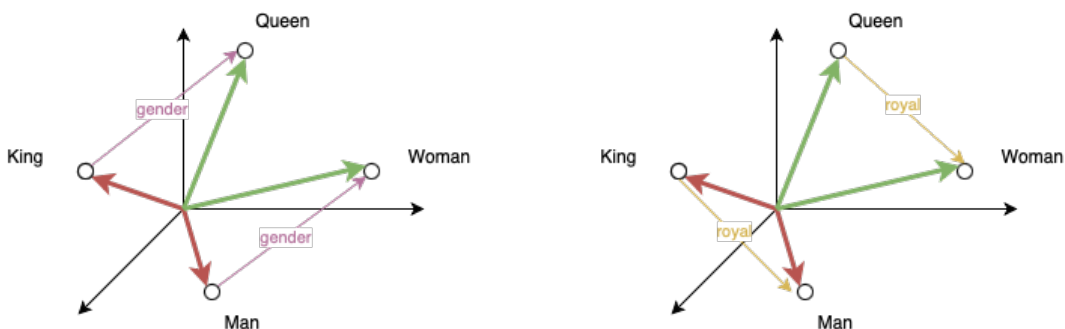


Figure 2.1: Semantic analogies derived from subtraction between word embeddings.

The concept of word embedding is associated with the contribution made by Bengio et al. [3] with a neural network able to learn a distributed representation of words without incurring the curse of dimensionality[2] of the statistical language models. Instead of learning every possible combination of a probability distribution, this model embeds the

sequence of preceding words in a smaller dimensional space to obtain the probability distribution of the subsequent word. The model has two hidden layers: a projection layer, which is linear and contains the word representation in the features space, and a hyperbolic tangent layer. The output comes from the final computation of the softmax function. The introduction of word embedding and neural approach to language modeling opened the way for the deep learning revolution in NLP.

Inspired by the work of Bengio [3], Maikolov et al. [15] focus on a shallower model called Word2Vec, trained on more data to achieve better performances. The main idea is to move from the prediction of the next word in a sentence to two new approaches: Continuous Bag of Words (CBOW) learns to encode a word based on its context, composed by the surrounding words, and Skip-gram learns the representation of the context, given a word. To reduce the computational complexity of the model, the hidden layer of Bengio's model is removed and the projection layer is shared among all the inputs. The two models offered by Word2Vec are shown in the following Figure 2.2.

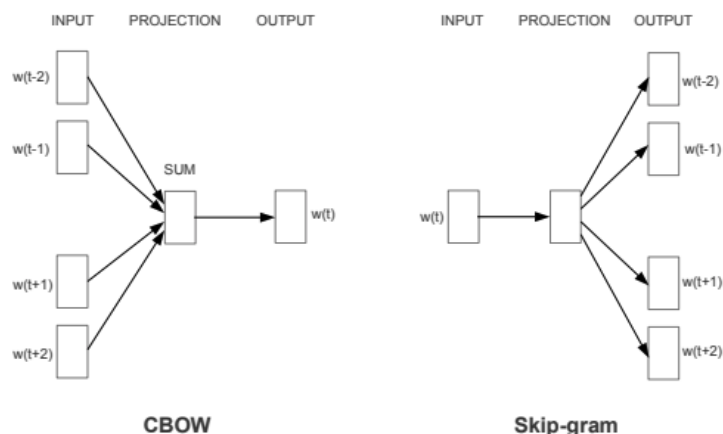


Figure 2.2: Word2Vec models

Word2Vec achieves higher performance than its predecessors. However, this model encounters challenges in processing long sentences or entire documents.

An alternative primary word embedding model is Global Vectors for word representations (GloVe), proposed by Pennington et al. [18], which combines matrix factorization and shallow window-based methods. Matrix factorization produces low-dimensional word representation by employing low-rank approximations to break down expansive matrices that encapsulate statistical information about a corpus. The window-based method involves acquiring word representations designed to facilitate predictions within local context windows, such as Word2Vec [15]. In contrast to matrix factorization techniques,

shallow window-based methods face a drawback in that they don't directly leverage the co-occurrence statistics of the corpus. Rather than operating directly on the corpus's co-occurrence patterns, these models scan context windows throughout the entire corpus, missing out on the substantial repetition present in the data. This hybrid approach allows GloVe to capture more complex word relationships, by learning word vectors from word co-occurrences (the counts of how often words appear together in a given context, giving a global statistical overview of word associations) and local context.

2.2. Transformers

Embedding has a fundamental role in Natural Language Processing (NLP) because machines necessitate a way to represent words and text to elaborate natural language results. Statistical techniques, like the known N-gram model[29], can process natural language but are outperformed by even simple deep learning techniques, such as multi-layer perceptron. With the process in this research area, new and more complex models, like Word2Vec [15] and GloVe [18], set the fundamentals of language models. Their ability to represent words in vectors, called embeddings, that contain both the syntactic and the semantic meaning, brings them to achieve state-of-the-art in many NLP tasks. Their weakness is their context-independent nature. With the advent of the Transformer, introduced by Vaswani et al. [26], a revolution in the way sentences are read and understood by machines is made in NLP.

2.2.1. Vanilla Transformers

Recurrent Neural Networks (RNNs), Long Short-Term Memory Networks (LSTMs) [9], and Gated Recurrent Neural Networks [5] at first, represented the state-of-the-art approaches for addressing problems such as language modeling and machine translation. Their structure includes feedback connections, or cycles, to incorporate in each computation the previous ones, simulating a memory, allowing for more context compared to the classic Feed Forward Neural Networks (FFNNs). Despite having the concept of context, these models are limited: provided with a long sentence as input, they tend to forget the beginning while still completing the processing of the text. Furthermore, their sequential nature precludes parallelization within training examples. Another issue only RNNs present is that they are known for suffering from exploding or vanishing gradient problems.

With the work of Vaswani et al. [26], they propose a new model called Transformer. This model maintains the sequential structure of the previously mentioned models. The

key innovation of this model is that it relies entirely on the attention mechanism to link the dependencies between inputs and outputs, admitting more extended contexts. The transformer model can parallelize the process, leading to faster training and enhanced performance, and is considered the actual state-of-the-art in translation quality. Despite being initially developed for text processing, today its application extends to various types of data, including images, videos, audio, or virtually any other sequential data.

The preliminary step to using the Transformer model involves converting raw data into a structured format to facilitate machine understanding. This process is called *tokenization*, followed by *encoding*. Tokenization is the process of segmenting text into meaningful smaller chunks, referred to as tokens. These tokens often represent words but can also include subword units. Afterward, the procedure of encoding translates each token into numerical values. The choice of the tokenization method and the encoding scheme is crucial and can impact the model's performance.

The Transformer model can process sentences and produce word embeddings of each word, vectors containing different aspects of the word itself based on its context. The methodology used by the model to extract the context of each word is by analyzing the entire input sentence and focusing on different parts of it. This technique is called the attention mechanism [26].

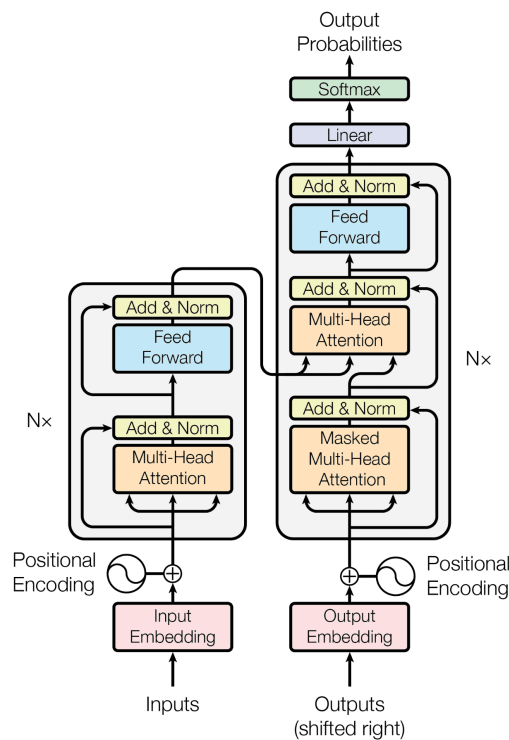


Figure 2.3: Transformer model architecture.

Transformer architecture follows the Encoder-Decoder structure as illustrated in Figure 2.3. The encoder maps the input sequence to a continuous representation that the decoder uses to generate an output sequence of symbols. The model is auto-regressive, computing one part of the output at each step taking into consideration the previously generated part of the output as additional input.

The encoder is composed of six identical layers, each with two sub-layers: the first is a multi-head self-attention mechanism and the second one is a simple fully connected feed-forward network. Around each sub-layer is built a residual connection, followed by a layer normalization. The dimension of the output of the embedding layer and the sub-layers is 512. (Vaswani et al. [26])

The decoder is also composed of six identical layers, as the encoder, with an additional third sub-layer that performs multi-head attention over the output of the encoder. Another modification is applied on the self-attention sub-layer to avoid positions from attending to consecutive positions. In the decoder, as in the encoder, each sub-layer has a residual connection followed by a layer normalization. (Vaswani et al. [26])

Current Transformer models can be either an encoder, a decoder, or both, depending on the specific task they are designed to accomplish.

Since the Transformer does not have recurrence or convolution, which gives the ability to consider the order of a sequence to RNNs, it adds this information through **positional encodings**. It works by injecting a combination of sine and cosine functions with different frequencies about the position of the tokens in the sentence. Because positional encodings share identical dimensions with the embeddings, it is possible to aggregate them through summation at the bottom of the encoder and decoder stacks.

Self-attention, also called intra-attention, is the key element to empower the model to select which tokens of a sentence are the most relevant to understand the meaning of a single token. This mechanism differs from vanilla attention, which focuses specifically on attention between encoders and decoders. With self-attention, the transformer calculates all words of a sentence simultaneously and not sequentially as RNNs, which allows for parallelizing the computation. Another advantage is the consideration of long-range dependencies, which leads to the creation of contextualized word embeddings: representations of words taking into account their meaning within sentences.

Self-attention refers to a single attention layer in the **multi-head attention** block shown in Figure 2.3, where each node maps a query and a pair of key-value to an output, all of which are vectors. The Transformer does not rely on just one attention head but uses

multiple attention heads in parallel, as shown in Figure 2.4. This technique allows the model to achieve a more comprehensive and accurate understanding considering different relationship representations.

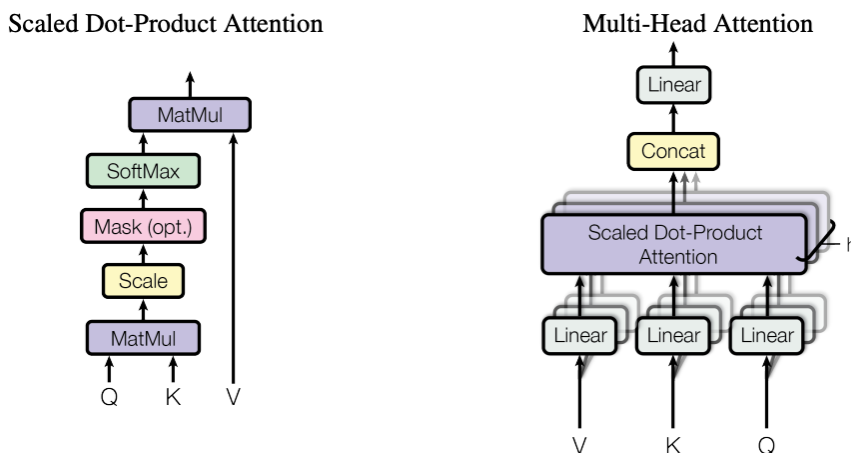


Figure 2.4: (left) Single head attention (right) multiple-head attention

Despite their revolutionary influence, Transformers have certain drawbacks; when processing longer texts, the self-attention mechanism can lead to quadratic computational costs with sequence length. Furthermore, in some situations, the need for significant computational resources and extensive pre-training datasets can be a limitation. These challenges highlight the need for more advanced models that improve and expand upon the Transformer design.

2.2.2. BERT

The scarcity of training data in Machine Learning (ML) is one of the main obstacles. To fill this gap, researchers designed a technique called pre-training. In the NLP research area, pre-training enables general language representation models to be trained on a large corpus of unannotated web content. Compared to training on smaller and more specific datasets from scratch, the pre-trained model can be improved significantly in accuracy by fine-tuning on NLP tasks with small amounts of data, such as question answering and sentiment analysis.

Pre-trained representations may be contextual or context-free, and contextual representation can be further classified as bidirectional or unidirectional. Word2Vec [15] and GloVe [18] are examples of context-free models that provide a word embedding representation for every word in the vocabulary. For instance, without considering the context, the

term "bank" would be represented the same way in "bank account" and "bank of the river". Contextualized unidirectionally models, such as GPT, consider the context from the preceding or following tokens. The word representation of "bank" in the sentence "I accessed the bank account" considers the phrase "I accessed the" rather than "account". The introduction of Bidirectional Encoder Representations from Transformers (BERT) by Devlin et al. [8] [24] marks a paradigm change in the way models address language's bidirectionality. It is a bidirectional contextual model that allows transitioning from word embeddings to contextualized word embeddings, in which the context of each word determines its unique representation from both the preceding and subsequent words. In the above instance, the word "bank" is represented by the entire sentence "I accessed the...account". (Devlin et al. [7])

BERT relies on the Transformer's neural network architecture, where the self-attention mechanism is bidirectional, allowing BERT to evaluate each word according to its previous and subsequent context, to determine its meaning. BERT can produce contextualized word embeddings in this manner. BERT relies on the masked language model (MLM) to generate high-quality contextual word embeddings. The masked language model aims to estimate the original vocabulary ID of a masked word based only on its context, by randomly masking a portion of the input tokens. The MLM aim permits the representation to combine the left and right context, which enables pre-train a deep bidirectional Transformer, in contrast to the left-to-right language model pre-training.

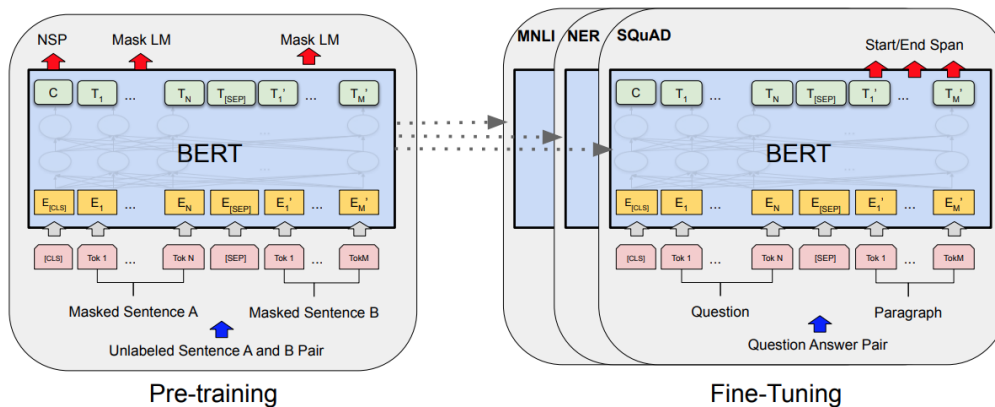


Figure 2.5: (left) BERT pre-training (right) BERT fine-tuning

Pre-training and fine-tuning are the two stages of developing BERT, as illustrated in Figure 2.5. BERT stands out for having a consistent design, suitable for a wide range of activities. The final downstream architecture and the pre-trained architecture differ slightly. The input can represent a single sentence or a pair of phrases (e.g. <question,

answer>) to allow the model to perform different tasks. BERT uses WordPiece tokenization to divide text into smaller units called tokens. In each sequence, the first token is a classification token ([CLS]). Sentence pairs are packed into one lengthy sequence. First, a unique token ([SEP]) is added to the input to distinguish them. Secondly, a learned embedding is assigned to each token, indicating to which sentence it belongs. The corresponding token, segment, and position embeddings are summed to obtain the input representation. The Figure 2.6 shows a description of this structure.

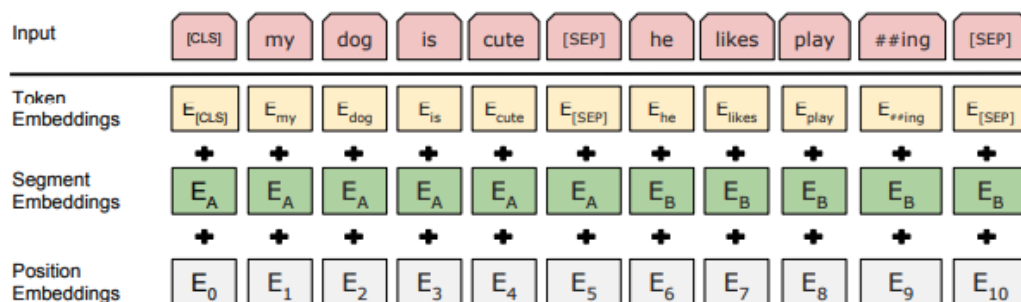


Figure 2.6: BERT input representation

The pre-training consists of two unsupervised tasks, the masked language model (MLM) and the next sentence prediction (NSP), based on plain text from a Wikipedia corpus [8]. In masked LM, part of the input is masked with a special token ([MASK]). The model aims to predict these masked words. The problem is that this leads to a discrepancy between the pre-training and the fine-tuning, as the [MASK] token does not appear in the fine-tuning. To mitigate this, the masked words are not always replaced by the actual [MASK] token: in 80% of cases, the randomly selected token is replaced by the [MASK] token, in 10% of cases it is replaced by a random token, and in 10% of cases it is not replaced. The model in NSP is trained to comprehend the relationships between sentences. The text corpus is binary, meaning that each pair of sentences in the corpus is assigned a label, either 'isNext' or 'isNotNext'. The training set contains half of the pairs that are truly one subsequent to the other. These pre-training are helpful for subsequent fine-tuning on tasks such as natural language inference (NLI) and question answering (QA).

Following pre-training, BERT can be fine-tuned for a specific language task, like question-answering (QA) and sentiment analysis (SA), as shown in Figure 2.5. It is possible to train the inner layers of the model in addition to the last layer during fine-tuning. Furthermore, it can handle words outside its vocabulary (OOV) by breaking them down into smaller chunks or using a unique token [UNK]. This possibility to refine the model for a specific task gives rise to numerous BERT variants. RoBERTa[13], ALBERT, distilBERT, and

others are some of the BERT variants.

The BERT model has some issues despite being an effective model. The drawback of this model is that its self-supervised pre-training focuses primarily on generating high-quality contextual **word** embeddings, resulting in sentence embeddings not as well-built. BERT is also incompatible with multilingual processing as it has been pre-trained solely on English text. Additionally, there is a token limit for the input, which means that text too long may be removed.

2.2.3. sBERT

Sentence-BERT (sBERT) by Reimers & Gurevych [21], is a BERT [7] model modification that derives semantically relevant **sentence** embeddings compared via cosine-similarity using siamese and triplet network architectures. Semantically meaningful refers to the proximity of semantically related statements in vector space. In terms of accuracy and time spent, sBERT performs better on the semantic text similarity (STS) problem than the BERT [7] and RoBERTa [13] models.

BERT uses a cross-encoder approach for tasks such as sentence-pair regression. Unfortunately, because there might be too many possible combinations this strategy is often not practicable. For tasks such as clustering and semantic search, where each sentence is mapped to a vector space in which sentences that are semantically related are close to each other, BERT has problems. Using the output of the first token ([CLS]), or output layer averaging is the most frequently used method to obtain the fixed-size BERT vector from each sentence. The sentence representations obtained with this method are extremely poor and often worse than averaging the GloVe [18] embedding.

These problems are resolved by sBERT [21], a Siamese network architecture that produces input sentence vectors of a certain size. Semantically similar sentences can be identified using similarity metrics such as Euclidean distance or cosine similarity. With sBERT the process of identifying the most similar pair among 10,000 sentences is simplified from 65 hours with BERT to about 5 seconds.

To generate fixed-size sentence embeddings, the sBERT model architecture combines a pooling layer with the output of the BERT or RoBERTa network. A Siamese and triplet network is developed to update the weights of the fundamental block model, BERT or RoBERTa, so that the resulting sentence embeddings are semantically meaningful and comparable. This Figure 2.7 illustrates the structure.

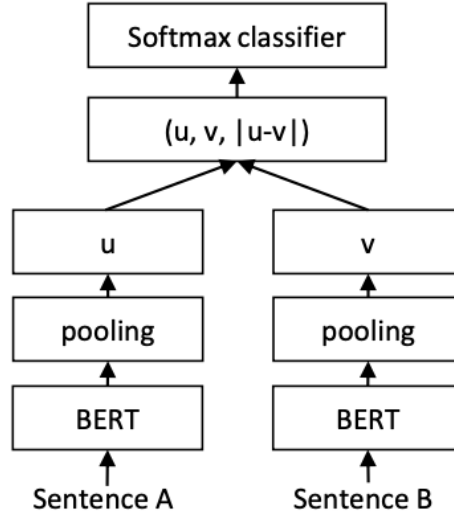


Figure 2.7: sBERT pre-training

A family of neural networks, called Siamese neural networks, consists of two or more subnetworks that are the same, meaning that the parameters and weights are identical. The parameter updates are replicated in both sub-networks. Two sentence embeddings u and v are generated by the Siamese network, then concatenated with the element-wise difference $|u - v|$ and multiply it with the trainable weight $W_t \in \mathbb{R}^{3n \times k}$:

$$o = \text{softmax}(W_t(u, v, |u - v|)) \quad (2.1)$$

where n is the dimension of the sentence embeddings and k is the number of labels (Reimers & Gurevych [21]). The cross-entropy loss is calculated based on the true label expected between entailment, neutral or contradiction, and softmax function. At inference time, the similarity score is directly calculated using the two sentence embeddings (u and v), eliminating the concatenation and classification layer. Regression tasks are also performed with this layout; in this case, during training, the mean squared-error loss is used.

Triplet loss is frequently used when training Siamese networks. It is a loss function where a baseline input a is compared to a positive input p and a negative input n . The goal is to reduce the distance between the anchor input and the positive input and to maximize the distance between the anchor input and the negative input. The loss function that is

minimized mathematically is:

$$\max(\|sa - sp\| - \|sa - sn\| + \varepsilon, 0) \quad (2.2)$$

where s_x is the sentence embedding for $a/n/p$, $\|\cdot\|$ is a general distance metric and ε the margin that ensures that s_p is at least ε closer to s_a than s_n . (Reimers & Gurevych [21])

Benchmark tests show that sBERT performs better on semantic similarity tasks than BERT and other sentence embedding algorithms. Despite its advances, sBERT still has drawbacks, including its dependence on the quality of pre-training data and difficulties in multilingual environments.

2.3. Semantic Search

Conventional search engines, such as Elasticsearch, are usually based on a bag-of-words strategy (keyword search) that combines the ranking functions TF-IDF or BM25 with an inverted index. With the introduction of semantic search, the capabilities of search engines have evolved significantly. Semantic search is an advanced method for information retrieval that goes beyond keyword-based techniques by interpreting the meaning and intent of the user's query. Instead of looking for exact word matches, this method matches a query and a text document that are semantically related. For this reason, it is sometimes referred to as a natural language search.

Semantic search technology can be distinguished between two alternative approaches: symmetric and asymmetric. A scenario known as symmetric semantic search occurs when the length and contextual information content of the user query and the corpus documents are similar. An asymmetric semantic search, on the other hand, occurs when the length and information content of the query and the searched pages differ. Typically, the query in this scenario is short and concise, often formulated as a brief question or a set of keywords, while the documents containing the answers or relevant information are much more extensive.

Semantic search relies on the concept of embeddings and Natural Language Process (NLP) models. Technically, a model is trained to generate question and item vector embeddings using a machine-learning approach known as dense retrieval. The items, or parts of the text corpus to be retrieved, can vary depending on the application and can include different formats, including web pages, podcast episodes, and movies/series. The objective is to obtain vectors of query and relevant items close to each other in the embedding space, as shown in Figure 2.8.

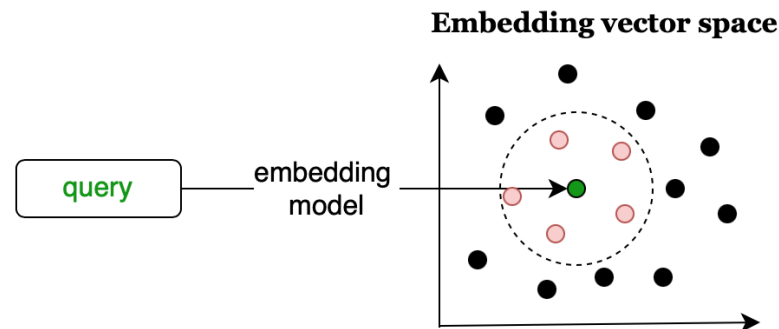


Figure 2.8: The retrieving process where query and relevant items (red points) are close to each other in the vector embedding space.

Assuming that the model is robust and produces an efficient embedding, the retrieval step involves using an algorithm that determines the similarity between the query and object vectors and then ranks the most relevant vectors according to the highest similarity score. The distance between these vector representations is used to calculate the similarity between the elements: the closer the vectors are in the embedding space, the higher their similarity score. Manhattan, Euclidean, Cosine, Dot product, and Chebyshev are the most commonly used distance metrics; the selection is based on the specific use case. It's important to note that when embedding vectors are normalized, cosine similarity and the dot product return equivalent results.

An effective method to find the closest vectors in a space for a given query vector is K nearest neighbors (k-NN) [19] [6]. The hyperparameter k specifies the number of nearest neighbors to be retrieved. One major limitation of k-NN is that to find the nearest vectors for a given query, it must first compute the distance between each vector in the database. If the database contains a large number of elements, this is extremely inefficient, due to the curse of dimensionality [2]. The curse of dimensionality describes the phenomenon that as the dimensionality of data increases, the volume of space increases so exponentially that the available data becomes sparse. Because of this sparsity, it is challenging to determine the precise nearest neighbors without incurring significant computational costs.

A variant of the original algorithm can obtain a good estimate instead of computing the distance between all the items in the corpus, even at the cost of losing accuracy. One common k-NN variant in the field of data analysis and retrieval is Approximate Nearest Neighbors Search (ANNS) [11]. ANNS is a technique that searches for an object in a reference dataset that is approximately the closest to a query object. Without performing an exact nearest neighbor, this algorithm does not incur complex and time-consuming

elaborations. This enhancement is particularly useful for a variety of applications where searching for multiple similar objects is required.

2.4. Fine-Tuning

The performance of semantic search is directly linked to the quality of the embedding model used. It is important to remember that embedding models — such as BERT — were not originally developed for semantic similarity. Instead, they were trained to predict words with masks in huge text corpora; as a result, they learned that texts with comparable meanings also tend to have similar embeddings.

The ability of LLMs to generalize well over a wide range of domains is due to their extensive training with large datasets. Nevertheless, domain-specific precision is sometimes sacrificed in the interest of this broad application. To overcome this limitation, fine-tuning procedures are used to more accurately fit these models to the specific domain. Fine-tuning is the process of improving a pre-trained LLM by training it on a smaller, more specialized dataset to tune it to a specific task or to improve its performance, as illustrated in Figure 2.9. To optimize the model for tasks such as semantic search, this strategy ensures that the model gains a greater understanding of domain-specific material while maintaining its adaptability.

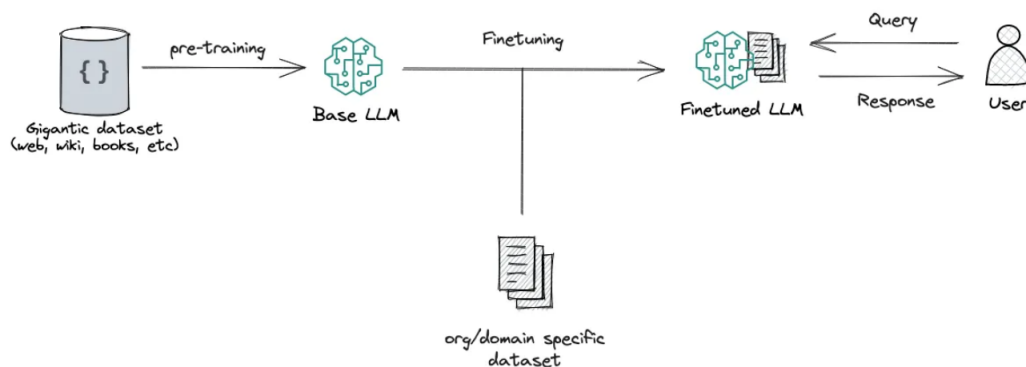


Figure 2.9: Fine-tuning pipeline for Information Retrieval.

Sentence-BERT (sBERT) fine-tuning provides a practically accurate solution to improve the performance of semantic search in specific domains. Fine-tuning this model enables a targeted augmentation of its semantic understanding and retrieval performance, as sBERT is designed to generate semantically rich phrase embeddings. To better capture the intricacy and complexity of domain-specific language and purpose, this procedure may

alter not only the final layer of sBERT but also its intermediate layers.

The efficiency and effectiveness of fine-tuning embedding models for semantic search tasks is influenced by the choice of loss function. This function is essential for training the model to recognize the semantic similarity of sentence pairs and, consequently, their proximity in the embedded vector space. This is accomplished by the triplet loss function, augmented with the cosine distance as the metric to determine the space between the vectors.

Contrastive Learning (CL) has become a key strategy for improving natural language processing (NLP) task performance. This self-supervised learning paradigm aims to improve the semantic extraction of sentences by maximizing the distance between dissimilar samples and decreasing the distance between semantically similar samples. Formally, given a sample x that shares similarity with x_1 but not with x_2 , contrastive learning aims to learn an encoder function f , ensuring that

$$\text{sim}(f(x), f(x_1)) \gg \text{sim}(f(x), f(x_2)) \quad (2.3)$$

where $\text{sim}(\cdot, \cdot)$ denotes a similarity function, such as cosine similarity. This methodology involves training models with batches composed of a similar sample x_1 and $N-1$ dissimilar samples x_2, x_3, \dots, x_N , emphasizing the contrast between the 'anchor' sample and a multitude of 'negative' samples [33], as illustrated in Figure 2.10.



Figure 2.10: Visualization of triplet loss goal, reduce the distance of an anchor to the positive example and increase the distance to the negative one.

A key advantage of CL over generative self-supervised learning approaches is its focused attention on the high-level features of the samples. Unlike generative models that unintentionally focus more on fine-grained details, which could compromise the model's ability to capture broader semantic meanings, CL ensures that the model's learning process focuses on understanding and differentiating the high-level semantic relationships among samples. Due to this feature, CL performs better in semantic search applications, when used for tasks that require a sophisticated understanding of semantic similarity.

Multiple Negative Ranking Loss (MNRL) is a variant of triplet loss, and operates on the principle of contrastive learning, using sentence pairs $[(a_1, b_1), \dots, (a_n, b_n)]$ where each a_i, b_i pair consists of semantically similar sentences, and a_i, b_j (for $i \neq j$) are considered semantically dissimilar. The objective of MNRL is double: to minimize the distance between embeddings of similar sentence pairs (a_i, b_i) while maximizing the distance between dissimilar pairs (a_i, b_j) . This dual focus aligns with the core methodology of contrastive learning, promoting a more refined and effective embedding space for semantic search applications. This framework can be directly applied to semantic search by treating query-satisfying items in the corpus as positive examples and all others as negatives.

To further refine the effectiveness of MNRL, it is possible to introduce triplets $[(a_1, b_1, c_1), \dots, (a_n, b_n, c_n)]$ into the training data. Here, c_i represents a hard negative for each a_i, b_i pair — sentences that are semantically distinct to a_i and b_i , ensuring that they are far from each other in the embedding space.

However, the effectiveness of CL is not without its challenges. One notable issue is its sensitivity to the size and selection of negative samples. The balance and diversity of these samples are crucial for the effective training of the model, as they directly impact the model’s ability to accurately discern semantic similarities and differences across a wide range of contexts.

Updating the entire model, including all its weights, can lead to significant performance improvements, but it has one possible drawback: a phenomenon known as catastrophic forgetting. This phenomenon occurs when, after fine-tuning, the model loses the knowledge it acquired in the initial pre-training phase. The source of this problem is the change in the embedding layers. Since the learned information is embedded in each layer, any modifications during fine-tuning can lead to the loss of this pre-trained knowledge.

To mitigate the risk of catastrophic forgetting, a possible strategy is to use **adapter** modules. In this approach, one or more additional layers are integrated on top of the existing architecture of the embedding model, with fine-tuning applied exclusively to these newly added components. By maintaining the layers of the original model, the adapter technique not only protects the previously learned knowledge but also adapts the representations of the model to a new latent space, specific to the retrieval tasks related to the particular dataset and queries. Nevertheless, there are disadvantages associated with this approach. While catastrophic forgetting is avoided, the improvement in model performance can be modest. Furthermore, adding additional layers can cause the model to respond more slowly.

Hu et al. [10] present a novel approach called Low-Rank Adaptation (LoRA) that ad-

dresses the weaknesses of the previously discussed fine-tuning methods. This method integrates trainable low-rank decomposition matrices into the layers of the Transformer architecture while preserving the pre-trained model weights. This strategy drastically reduces the number of trainable parameters needed for downstream tasks. By optimizing the rank-decomposition matrices of these layers, LoRA facilitates the indirect training of certain dense layers in a neural network while keeping the pre-trained weights unchanged, as shown in Figure 2.11.

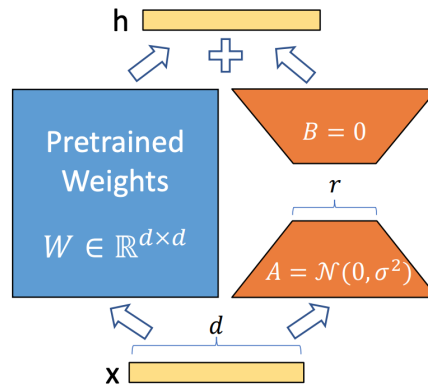


Figure 2.11: Visualization of low-rank decomposition.

The dense layers of a neural network that perform matrix multiplications usually have full-rank weight matrices. Aghajanyan et al. [1] have shown that pre-trained language models have a low "intrinsic dimension" during task-specific adaptation, which enables effective learning even in the case of random projection into a smaller subspace. Inspired by this finding, Hu et al. [10] suggested that during adaptation, the updates of the weights also have a low "intrinsic rank".

For an original weight matrix W_0 , its update is represented by a low-rank decomposition $W_0 + \Delta W = W_0 + BA$. During the training process, W_0 remains constant and does not receive gradient updates, whereas A and B consist of trainable parameters. Both W_0 and $\Delta W = BA$ are multiplied by the same input, and their outputs are aggregated on a coordinate-wise basis. Consequently for $h = W_0x$ the forward pass is modified as follows:

$$h = W_0x + \Delta Wx = W_0x + BAx \quad (2.4)$$

as shown in Figure 2.11. The initialization of the two low-rank matrices decomposing W_0 is as follows: A is initialized with a random Gaussian distribution, and B is initialized to zero. (Hu et al. [10])

One of the many advantages of LoRA is that a trained model can be used for multiple tasks by deploying several small LoRA modules, each customized for a specific task. This feature significantly reduces the overhead related to task switching and memory requirements. In addition, LoRA eliminates the need to calculate gradients and maintain an optimization state for most parameters, improving training efficiency and reducing hardware requirements by about three times when using adaptive optimizers. Unlike a fully optimized model, the simplicity and linear design of LoRA allow for easy integration of the trainable matrices with the static weights at deployment, preventing the introduction of additional inference latency. This feature ensures that LoRA maintains operational efficiency and practicality for real-world applications.

2.5. Hybrid Search

The development of sophisticated embedding models represents a step forward in the realization of powerful semantic search functions. Despite these developments, traditional term-based search techniques — such as BM25 by Robertson and Zaragoza [22] — remain relevant and often outperform even the most advanced deep learning models for certain tasks. This discrepancy presents an interesting idea: the combination of term-based and neural models in a single hybrid multi-retriever system that utilizes the advantages of both methods.

Vector search is based on semantic similarity, and keyword search is based on the frequency of query terms in texts; both have advantages and disadvantages: Vector search is superior in terms of semantic similarity, but keyword search is more precise. To maximize search results, the hybrid search model combines the "best-of-both-worlds" strategy by using both sparse and dense vector search. At its core, a hybrid search system has two work processes. A vector search is performed to find elements that come close to the vector representation of the search query. At the same time, a keyword search sorts the results according to the frequency of occurrence of the query words. This dual method not only combines the results of the two search processes but also attempts to summarize several score measures (such as cosine similarity and BM25) in a single rating, that cannot be directly combined due to different scales and statistical distribution.

The hybrid search uses a fusion algorithm to incorporate the model scores. These algorithms offer a new ranking approach by balancing and combining the results of vector and keyword searches, as illustrated in Figure 2.12.

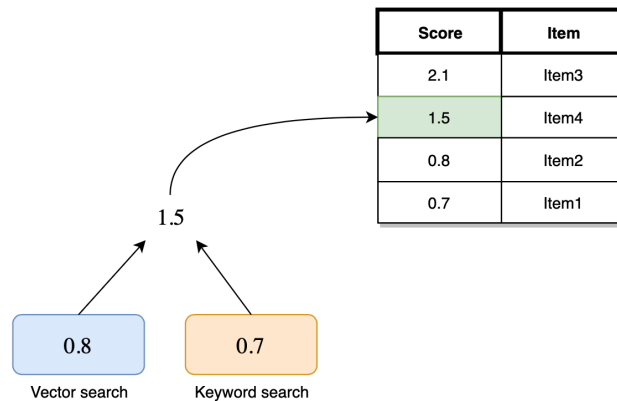


Figure 2.12: Visual schema of fusion algorithm.

The `relativeScoreFusion` and `rankedFusion` algorithms represent two different approaches to score fusion. The `rankedFusion` method assigns a score to each object depending on where it appears in the search results. The highest score is given to the object that comes first, and the scores drop down in order of the ranking. These rank-based scores from the vector and keyword searches are added together to determine the final score. However, with `relativeScoreFusion`, the score of each object is determined by normalizing the metrics generated by the vector and keyword searches, respectively. According to this scale, the highest value becomes 1, the lowest value becomes 0, and the remaining values fall somewhere in between. The final result is therefore determined by a scaled sum of the normalized vector similarity and the normalized keyword search.

3 | Dataset

In the realm of Machine Learning research, the quality and diversity of the underlying data have a direct impact on the effectiveness and generalization of the results. The potential achievements of any model, especially in terms of performance and applicability, depend on how relevant and rich the data used for training and testing is. The purpose of this chapter is to provide insight into the available data and a description of the datasets used to train and evaluate our search model. We provide a detailed overview of the structure of the dataset and describe the pre-processing steps used to provide insight into the applicability of the dataset in real-world situations as well as its suitability for our academic goals. It is important to point out that the data we used for this study comes from internal company sources and industry partners, rendering it not publicly available.

3.1. Industrial datasets

In this section, we detail three datasets containing the metadata employed in our study. Originating from diverse companies, these datasets exhibit both similarities and differences in their structure and content.

3.1.1. Industrial dataset 1

The dataset provided by Contentwise, assembled with the open-source community database known as The Movie Database (TMDB), includes 9531 entries, each representing either a movie or an episode of a series. Initially, these entries come with a comprehensive set of attributes. However, for the purposes of this study, we have curated the dataset by excluding some attributes that are not useful to our research objectives. The dataset is structured as a JSON file, with each object representing a unique entity defined by key-value pairs that specify attributes and their corresponding values. Property keys are intuitively named and allow for easy interpretation. Here is an example of one sample:

```

{
  "Id": 3514,
  "Title": "Gilmore Girls",
  "Summary": "Christopher talks
  Lorelai into visiting Rory during
  Parents' Weekend at Yale; Luke meets
  April's swimming coach, who convinces
  him to take her adult swimming class.",
  "Episode name": "Go, Bulldogs!",
  "Country of origin": "USA",
  "Score": "8.1",
  "Show type": "Episode",
  "Release year": "2000",
  "Duration": "43",
  "Cast": [
    "Lauren Graham",
    "Alexis Bledel",
    ...
    "Alia Rhiana Eckerman"
  ],
  "Genres": [
    "Comedy drama"
  ],
  "Award": [
    "Screen Actors Guild Awards",
    "Golden Globe"
  ],
  "Mood": [
    "Amusing",
    "Endearing",
    "Emotional"
  ],
  "Settings": [
    "Connecticut",
    "Small town",
    "Household",
    "Inn"
  ],
  ...
}
...
"Director": [
  "Wil Shriner"
],
"Keywords": [
  "Mother/daughter relationship",
  "Small-town life",
  "Friendship",
  "Family dysfunction",
  "Romance"
],
"Characters": [
  "Mother",
  "Daughter",
  "Friend",
  "Love interest"
],
"Subjects": [
  "Mother/daughter relationship",
  "Small-town life",
  "Friendship",
  "Family dysfunction",
  "Romance"
],
"Semantic categories": [
  "FemaleLead"
]
}

```

3.1.2. Industrial dataset 2

This dataset comes from a large Northern European company operating in Denmark, Finland, Norway, and Sweden and specializing in satellite television and broadband services. The dataset under consideration consists of 23410 entries, all of which are distinctive examples of visual media content such as TV shows or movies. Carefully categorized and tagged with several descriptive attributes, these entries allow for a closer examination of the content collection. Also in this case it is structured as a JSON file. A unique aspect of the dataset is its organization into five parallel versions, each suitable for one of the five main languages spoken in the provider's service regions: English, Danish, Finnish, Norwegian, and Swedish. It is noteworthy that each element is consistently available in all language versions of the database, even if the completeness of the attributes varies depending on the language.

3.1.3. Industrial dataset 3

The third dataset used in this study comes from a top global telecommunications company operating in Europe and the Americas. The dataset under consideration consists of 1158509 unique entries for movies or TV shows originating from eight different Latin American countries: Brazil, Chile, Peru, Argentina, Colombia, Uruguay, Ecuador, and Mexico. The attribute 'TenantId' facilitates the differentiation of the entries based on country by serving as an alias for the geographical origin of the content. This dataset is singular and unified, unlike the previously discussed dataset (Section 3.1.2), which is divided into five versions based on language resulting in duplicate entries across each version. In order to classify the content by country, this dataset is segmented based on the 'TenantId' attribute after collection. The adjacent bar plot (Figure 3.1) provides a visual representation of the distribution of entries across these countries.

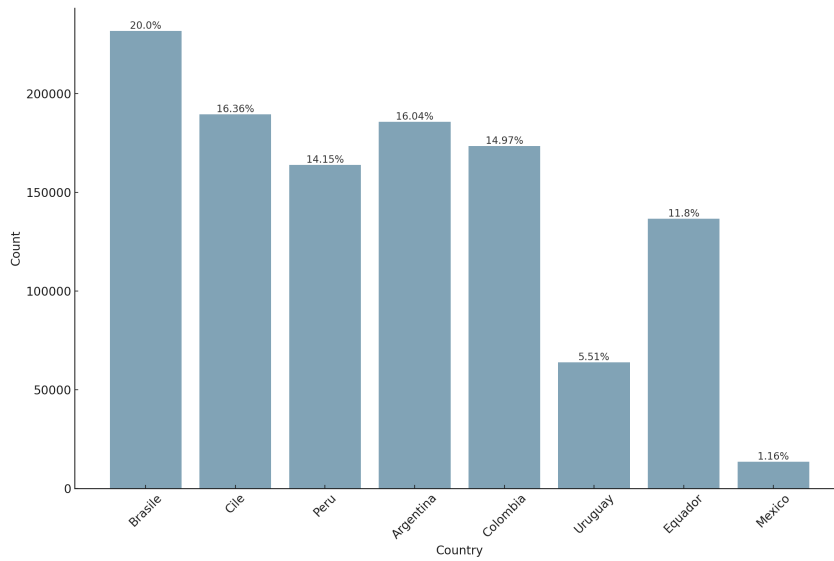


Figure 3.1: Item distribution based on TenantId

The subset that corresponds to Brazil is chosen for testing because it is the biggest portion, encompassing approximately 20% of the dataset with 231743 entries. The dataset predominantly features attributes in Portuguese and is structured as a JSON file.

Attributes and comparison

In Table 5.1 we outline the attributes across three industrial datasets related to streaming content metadata. Each attribute is briefly described, followed by three columns indicating its presence (✓) or absence in each dataset. This format offers a clear comparison, highlighting unique and shared attributes.

Attributes	Description	dataset1	dataset2	dataset3
Id	A unique identifier for each content	✓	✓	✓
Title	The title of the movie or serie	✓	✓	✓
Episode name	For series, the title of the specific episode	✓	✓	
Episode number	The sequence number of the episode within its serie		✓	
Season number	The season number to which the episode belongs		✓	
Country	The originating country	✓	✓	

Original language	The originating language	✓		✓
Show type	Description of the content type	✓		✓
Release year	The release year	✓	✓	✓
Decade	The decade in which the content is set			✓
Duration	The duration of the content expressed in minutes	✓	✓	
Summary	A brief outline of the content's storyline	✓	✓	
Actors	A list of actors	✓	✓	✓
Director	The name of the director	✓	✓	✓
Genres	A list of the categorized style	✓	✓	
Mood	A list of emotional tone or atmosphere	✓		
Scenario	The settings and contextual background	✓		
Theme	The central topic or underlying subject	✓		
Settings	The time and location	✓		
Score	Ratings given by users	✓		
Award	A set of award the content has received	✓		
Keyword	A set of terms associated with the content	✓	✓	
Characters	Key figures and personas within the narrative	✓		
Subjects	Main topics or elements explored in the content	✓		
Subgenres	Categories that further refine the genres	✓		
Genres2	An additional list of genres	✓		
Semantic categories	Conceptual groups for classify the content	✓		

IsKid	A boolean attribute indicating whether the content is suitable for children		✓	
TenantId	An identification number for the country in which is distributed			✓

Table 3.1: Comparison of attributes in industrial datasets

From the table, it becomes immediately evident that dataset 1 possesses the biggest set of attributes in comparison to datasets 2 and 3, particularly dataset 3, which exhibits a significant deficiency in content attributes.

An additional evaluation of the three datasets is placed, with a particular emphasis on their utility to semantic search. Beyond the universally present title attribute, the presence of an overview assumes a significant role. The overview provides the most general and comprehensive description of the content, its presence enriches each item with pertinent information that can potentially enhance performance. We quantified the number of items with and without an overview for each dataset in order to provide a more thorough explanation of the expressiveness and completeness of the datasets, especially for semantic tasks. In Figure 3.2 below, a graph visually represents the proportion of items featuring an overview relative to the total number of items within each dataset.

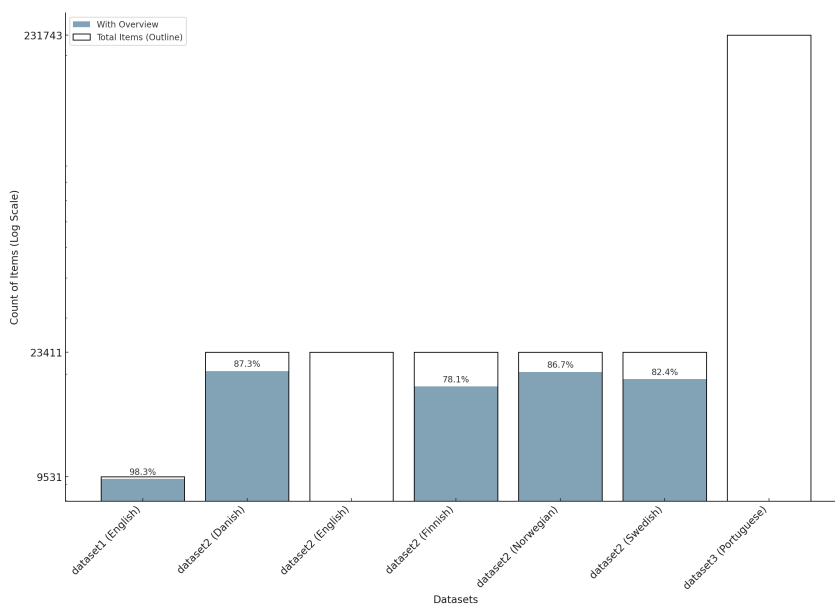


Figure 3.2: Comparison of items with overview and total items per dataset

3.2. Synthetic query-answer dataset generation

The availability and cost of training and evaluation data is one of the challenges in building models in the field of Deep Learning. Due to the sensitive and valuable nature of the data in many sectors, it can be protected by law or business interest. Health data and private company information are typical examples. There is also a possibility that there is no dataset for a certain downstream task and that creating one by hand is impractical. It describes our scenario: we lack an open-source dataset applicable to our specific task, and our data remain confidential, sourced directly from our partners. Furthermore, manually curating the most relevant items in response to a variety of potential user queries—based on the semantic essence of each query—would demand an excessive amount of time and effort. A strategy for resolving these data problems is the preparation of synthetic data, which is a labeled dataset designed for a particular use case utilizing an existing trained model or advanced data manipulation techniques [16] [20].

In the following Figure 3.3, we represent the pipeline employed to generate distinct versions of synthetic datasets, each comprising query-ranked list answers (QA), through the utilization of `gpt-3.5-turbo` and customized for the respective industrial datasets.

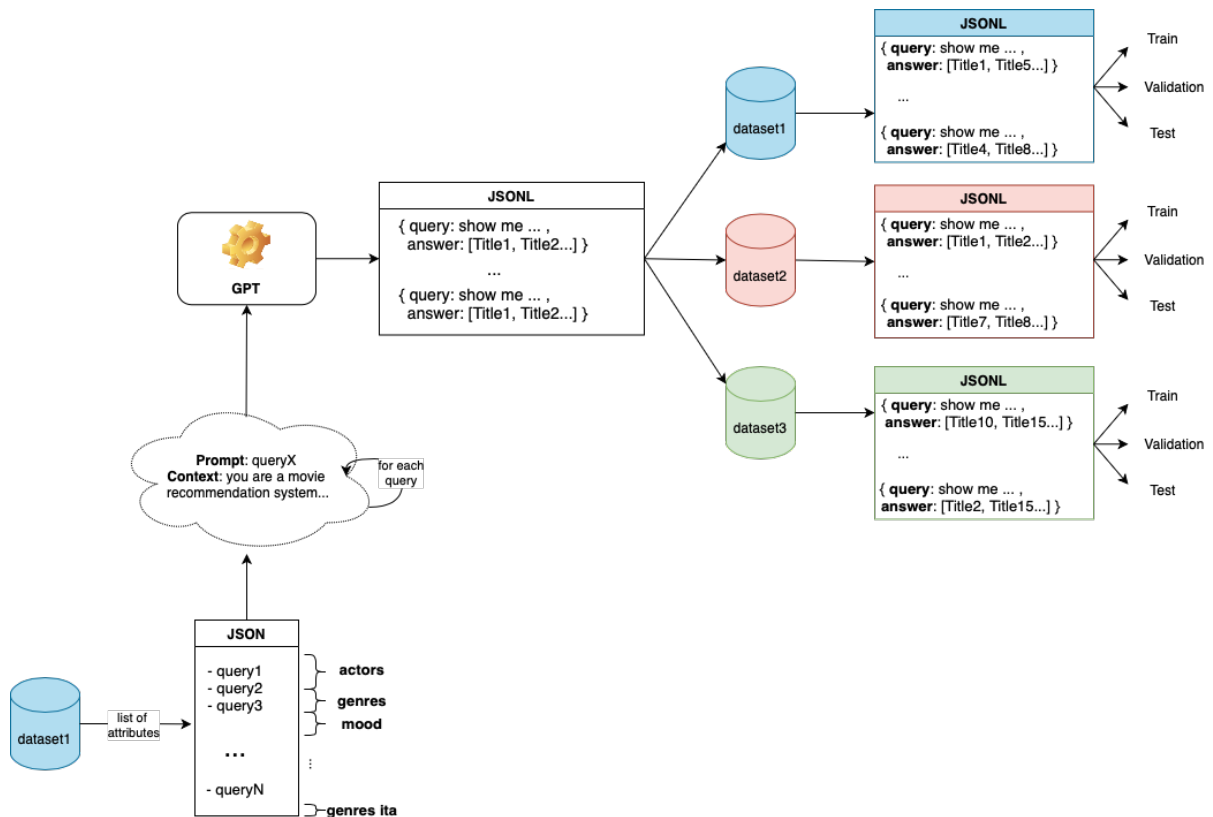


Figure 3.3: Workflow diagram for synthetic Question-Answer dataset generation

To generate our synthetic dataset, we start by creating a diverse array of queries, distinguished from one another through four distinct methods. First, we generated a basic type of query utilizing the varying values present in industrial dataset 1, each categorized according to corresponding attributes.

```
"show me something with Derek Johns"
```

Second, we formulate more complex queries by combining different values across various keys (in pairs, triplets, or quartets).

```
"show me something rousing thrilling",  
"show me something bad choices dreams come true living with mental illness"
```

The third category of queries employs '*dynamic categories*', attributes retrieved from an algorithm developed by Contentwise. In this context, a demo environment was populated with fictitious users, each associated with a history of content viewed from the catalog and their respective ratings for those items, enabling user profiling. Based on the users' ratings, the algorithm generates these '*dynamic categories*' to customize content selections to the users' preferences.

```
"Drama movies starring Matthew McConaughey",  
"2000s Dark movies shot in Hospital and New York City"
```

Lastly, considering all queries previously mentioned are in English and our objective is to evaluate the model across multiple languages, we translated the queries into several languages, including Italian, Spanish, French, German, Portuguese, and Chinese.

```
"Encuentrame algo con Jake Lloyd",  
"Trovami un vincitore del Premio Robert",  
"Trouvez-moi quelque chose cuisine arts martiaux medical",  
"Finde mich etwas erschreckend ironisch stilvoll"
```

Each language maintained an equal number of translated queries for each query type, ensuring no repetitions.

The resulting list is composed of a total of 46358 queries. Below, a bar graph Figure 3.4 illustrates the distribution of queries by type.

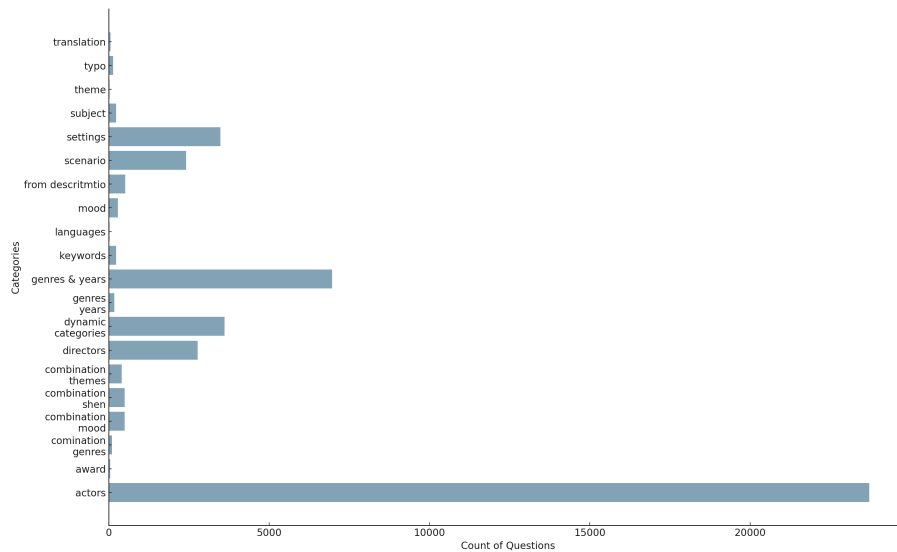


Figure 3.4: Query distribution per type

With the power of the language model ChatGPT, developed by OpenAI, we build the dataset by ranking movies or series on the previously constructed set of queries. The objective is to establish a rating of twenty streaming items for each query, based on the semantic similarity between the query and possible suggestions. This is accomplished through the `gpt-3.5-turbo` model via OpenAI's API. The requests are structured to provide the model with explicit context, which improves the relevance and precision of the output. The API requests are formulated as follows:

```
jobs = [{
  "model": "gpt-3.5-turbo",
  "messages": [
    {"role": "system",
     "content": "You are a movie recommendation system. Provide a ranked list of 20
     movies or series. The answer must be in json format with the movies title in a
     list of this type {movies:[]} , in the order of the ranking."},
    {"role": "user",
     "content": question}]] for question in questions]
```

The final step in answers construction involves cleaning the responses obtained from GPT. Firstly, we exclude any responses that contain error messages. Secondly, the procedure requires the removal of any ranking positions prefixed to each movie or series title (from '1. Interstellar' to 'Interstellar'). Additionally, we eliminate any explanatory text following the titles that may have been provided by the model to justify its selections. This step ensures that the answer lists in the dataset solely consist of the titles of movies or series.

3.2.1. Filtering and splitting

A data filtering process is implemented afterward to ensure the integrity and utility of our synthetic dataset. Each title generated by GPT is scrutinized to verify the presence in each industrial dataset, resulting in three versions of the query-answer (QA) synthetic dataset. Titles found within the dataset are maintained in the response, accompanied by their identification number ('Id'). Contrarily, titles not present in the dataset are excluded from the response list. The final step in the filtering process involves eliminating queries for which the response is an empty list, indicating either an inability of GPT to provide an answer or the absence of any generated titles within the dataset. The uniqueness of each query is guaranteed by hashing the query string with the SHA-256 function, generating a 256-bit alphanumeric identifier ('itemID') for each query. Thus, the QA dataset is formatted as a JSONL file. An example is provided below:

```
{
  "question": "Dark Sci-Fi movies",
  "answer":
    [
      {"Title": "Blade Runner", "Id": 1716},
      {"Title": "The Matrix", "Id": 1693},
      {"Title": "Inception", "Id": 70784},
      {"Title": "Ex Machina", "Id": 33544},
      {"Title": "Interstellar", "Id": 33415},
      {"Title": "The Terminator", "Id": 702},
      {"Title": "A Clockwork Orange", "Id": 2386},
      {"Title": "Mad Max: Fury Road", "Id": 33411},
      {"Title": "Eternal Sunshine of the Spotless Mind", "Id": 199},
      {"Title": "The Fifth Element", "Id": 1129},
      {"Title": "Minority Report", "Id": 955},
      {"Title": "Akira", "Id": 1215},
      {"Title": "Metropolis", "Id": 2008},
      {"Title": "Ghost in the Shell", "Id": 38818},
      {"Title": "Children of Men", "Id": 3171},
      {"Title": "Upgrade", "Id": 71032}
    ],
  "itemID":
    "b8bfd297351368655a5744672f1c34b7e6c9b6c89e2b40059bb12adb1be204e9"
}
```

Every entry in this file consists of a query and a list of titles and IDs, designed as the ground truth for that query. The dataset shows variation in the number of responses per query, ranging from 1 to 20, with an average of 7.6 responses.

Following the data filtering, the dataset is divided into three segments: training, validation, and testing, assigned in proportions of 60%, 10%, and 30%, respectively. The development phase of the thesis saw the inclusion of an additional 1389 translated queries, enhancing the diversity of the dataset and improving evaluation outcomes, thus expanding the total number of queries to 47747.

The distribution of dataset samples across the training, validation, and testing sets, is illustrated in the Table 3.2. The additional version of the QA synthetic dataset with more translated queries is detailed exclusively for the industrial dataset 1 to avoid redundancy, as the distribution proportions are consistent across all dataset segments.

	dataset1 v1	dataset2	dataset3	dataset1 v2
train	25105	25518	27482	25792
validation	3976	2835	3053	4103
test	11802	11573	12737	12166
empty	5475	6432	3086	5686
Total valid	40883	39926	43272	42061

Table 3.2: Distribution of query-answer pairs across Train, Validation, and Test sets for each industrial dataset, including counts of excluded queries with Empty answers. The Total valid is the aggregate number of samples, excluding the empty samples.

3.3. Human query generation

The objective of this thesis covers both research pursuits and practical applications, necessitating manual validation to address and benefit from GPT’s limitations. Despite its impressive capabilities and broad knowledge base, GPT’s information is current only up to 2021. Consequently, it may not recognize a movie or series released after this year that could provide an ideal answer to a query. Moreover, the QA dataset processed includes responses ranging from one to twenty items, from catalogs with thousands of entities, suggesting that additional items may still represent optimal responses.

To emulate an online application environment while performing human validation we develop a form. This form first clarifies the principle behind semantic search with a particular focus on its application to films and series. Participants were asked to submit queries they would ask to such a search engine. This method allowed for manual evaluation of our system response, because of a resulting smaller number of questions compared to the size of the test set generated. The form collect a total of 50 questions.

4 | Methodology

This chapter presents a comprehensive examination of the model selection process, the implementation of similarity matrices, and fine-tuning strategies adopted. These are key components in the development of semantic search and information retrieval systems. The effectiveness of the search engine is directly correlated with the performance of the model and the effectiveness of the corresponding generated embeddings in semantic search tasks, highlighting the central role in making the right choice of the model and training in the proper way.

4.1. Model selection

The first step in this part of our work consists of the model selection. The objective is to identify an embedding model that demonstrates high performance on the question-answer (QA) synthetic test set, delineated in Section 3.2.1. Our selection criterion relies on the examination of open-source models that are both available for download and offer the flexibility for customization through fine-tuning processes. The choice of open-source models is motivated by the desire for customizable solutions that avoid additional costs and reduce dependency on external services, which may compromise system availability in case of service problems.

Our search for suitable models primarily targets the Hugging Face model repository, renowned for its extensive collection of pre-trained models. The performance of these models on established benchmark datasets is one of the main criteria for our selection. Specifically, models that demonstrate impressive performance in semantic search and information retrieval (IR) tasks.

Another important consideration is the size of the model. Given that the end product of our research is planned to be a real-time application, it is imperative to select models that are not excessively large and fast. The rationale behind this is twofold: ensure the responsiveness of the service by minimizing latency in model inference and mitigate the cost associated with storage requirements.

In addition to exploring HuggingFace models, we include the embedding model called **Titan**, from AWS. It is important to clarify that the inclusion of **Titan** in our analysis is not to adopt it as the exclusive model for our application. Rather, its evaluation serves as an auxiliary benchmark, offering an extra comparative viewpoint about the models under consideration. This method differs from our application of **gpt-3.5-turbo** as a benchmark, which is used in a different ability. Specifically, **gpt-3.5-turbo** is used to generate the QA test set through an API call, functioning as a chatbot. This mechanism facilitates the selection of the top K items based on similarity, without necessitating the computation of embeddings by the model itself.

The model selection involves the following:

- **multi-qa-MiniLM-L6-cos-v1**: This is a sentence transformer model from Hugging Face, designed for semantic search. It maps phrases and paragraphs to a 384-dimensional vector space. The model is trained on 215 million (question, answer) pairs from various sources and uses MiniLM, a variant of BERT, as its base architecture. During training, a self-supervised contrastive learning objective is used where the model identifies matching sentence pairs from a set of candidates. The Multiple Negative Ranking Loss (MNRL) is employed during training. The model produces normalized embeddings, and the preferred scoring function in this case is the dot product due to its efficiency when dealing with normalized vectors.
- **multi-qa-mpnet-base-dot-v1**: Similar to multi-qa-MiniLM-L6-cos-v1, this model is from the sentence-transformers library and designed for semantic search. However, it uses mpnet-base (a Transformer-based architecture) as its base model and produces embeddings with a higher dimensionality (768). Unlike the previous model, the embeddings here are not normalized, making the dot product the only suitable scoring function. It also utilizes a self-supervised contrastive learning objective during training and has a maximum input window of 512 tokens.
- **all-mpnet-base-v2**: This model, a sentence transformer (sBERT) built on the pre-trained microsoft/mpnet-base model, is specifically designed for tasks like information retrieval, clustering, and sentence similarity. The pre-trained model is fine-tuned on a massive dataset exceeding 1 billion sentence pairs. Cross-entropy loss guides the fine-tuning process by comparing the model's predictions with the actual matching sentences. It generates 768-dimensional dense vectors as sentence embeddings, capturing the semantic meaning of the input text. It can handle text with a maximum length of 384 tokens, truncating longer inputs.
- **msmarco-roberta-base-ance-firstp**: This model is designed for semantic search

tasks. It builds upon RoBERTa by adding pooling, dense, and normalization layers. This ensures the generated 768-dimensional sentence embeddings have a consistent unit length, making them suitable for tasks like dot product similarity calculations. The model is trained using Approximate Nearest Neighbor Negative Contrastive Estimation (ANCE) [32]. This training method utilizes the corpus's approximate nearest neighbor (ANN) index to create more realistic negative examples for improved model performance.

- **all-MiniLM-L6-v2**: This model is the result of fine-tuning the nreimers/MiniLM-L6-H384-uncased pre-trained model. It is particularly suitable for tasks like clustering or semantic search. It produces a 384-dimensional dense vector representation for each input text. The model is designed to handle short text inputs due to its limited token input window. Inputs exceeding 256 tokens will be truncated.
- **bge-large-en-v1.5, bge-base-en-v1.5, bge-small-en-v1.5**: The BGE (Big Gradient Embedding) models belong to the FlagEmbedding family, which focuses on retrieval-augmented Large Language Models (LLMs). These models are trained using contrastive learning on large-scale datasets of English text pairs [31]. Hugging Face offers three BGE models for English text processing: the largest model, generating 1024-dimensional embedding vectors, the medium-sized model (base) produces 768-dimensional embedding vectors, and the smallest model, generating 384-dimensional embedding vectors. All three models can handle inputs up to 512 tokens and cosine similarity is the recommended scoring function to compare retrieved embeddings. A separate BGE model version (bge-large-zh-v1.5) is also available for Chinese text processing.
- **multilingual-e5-base** and **multilingual-e5-large**: These are multilingual text embedding models from Wang et al. [27]. They are based on the xlm-roberta architecture and come in two sizes: base (12 layers, 768-dimensional embeddings) and large (24 layers, 1024-dimensional embeddings). Both models are trained in two stages: first with contrastive pre-training using weak supervision, followed by supervised training on a mix of multilingual datasets. This approach allows them to capture semantic similarities across 100 different languages.
- **Titan**: This model, a text embedding model from AWS Bedrock, is designed for tasks involving semantic similarity, such as information retrieval, recommendation systems, and document clustering. It excels in text retrieval tasks, supporting Retrieval Augmented Generation (RAG) use cases. Titan converts textual data into numerical representations (embeddings) for efficient searching of relevant passages

within a large dataset. The model handles a wide range of text lengths, including very long inputs of up to 8192 tokens, and supports over 25 languages including English, Chinese, and Spanish. Offered through a serverless API, Titan prioritizes low latency and cost-effectiveness, eliminating the need for infrastructure management. However, fine-tuning this model is not supported.

4.1.1. Approach

A bifurcated testing methodology intends to measure the effect of input data granularity on the performance of the selected models, and empirically evaluate the effectiveness of the models. The models are put through two different experimental setups, with the metadata from the industrial dataset 1 (Section 3.1.1) and the list of queries in the corresponding QA synthetic dataset:

1. All-inclusive Metadata Input: In this scenario, models received a comprehensive collection of all item metadata. This method is based on the theory that a greater variety in the input context may improve the capacity to produce more precise embeddings of the model, which consequently may increase the relevance of items that are retrieved in response to a query.
2. Condensed Input (Title and Overview Only): This experimental setup, in contrast to the first condition, restricts the input to just the item titles and overviews. This condition looks at the performance of the model in a scenario with limited input, which is representative of a typical use case where precise metadata might not be easily available.

It is essential to keep in mind that despite the availability of rich metadata, the embedding models' intrinsic architecture places a cap on the maximum input length, expressed in tokens. As a result, inputs larger than the maximum token capacity of the model are truncated.

The evaluation approach proposed in Chapter 6, employs a wide range of metrics to measure the efficacy of the models. This structured methodology enables a comprehensive evaluation of the selected embedding models in various input conditions, with metadata in English derived from industrial dataset 1.

To further explore this area, we intend to evaluate the cross-lingual robustness and generalization capabilities of the top six models across multilingual datasets. These models include the previously mentioned Titan from Amazon and the open-source Hugging Face models `multi-qa-mpnet-base-dot-v1`, the large and base versions of `bge-*-en-v1.5`,

and `multilingual-e5-base`. This evaluation involves conducting tests on a secondary dataset versioned into six catalogs, referred to as dataset 2 (Section 3.1.2). Each catalog corresponds to a distinct language, facilitating an examination of how well the models perform when used with metadata in different languages.

Furthermore, is conducted an evaluation of a third dataset (dataset 3, Section 3.1.3), characterized by its metadata entirely in Portuguese. This specific dataset adds another level of analysis, which furthers our understanding of how adaptable the models are to languages other than English.

In addition to the automated tests, we conduct a manual evaluation. This involves selecting particular queries that originate from an established form (Section 3.3), and then examining the search results that each model produces. Such manual testing plays a crucial role in detecting details in the models' output that automated metrics might miss, offering an accurate diagnosis of their effectiveness in semantic search and information retrieval tasks across diverse linguistic contexts.

4.2. Similarity matrix

Nowadays, recommender systems are essential tools on a wide range of online platforms, such as YouTube, Facebook, Netflix, and Amazon. These systems help decision-making for users by customizing recommendations based on their preferences. The most commonly employed strategies among the multitude of techniques utilized are collaborative filtering (CF), content-based filtering (CBF), and hybrid approaches [4][23].

Sharing similar tastes and preferences among users is the foundation of Collaborative Filtering (CF). It assumes that people who have shown comparable preferences in the past will probably continue to have similar interests. Therefore, if two users are judged similar and one of them expresses a preference for a specific item, that item is then suggested to the other user with the expectation that it will be appreciated.

Conversely, Content-Based Filtering (CBF) focuses on the attributes of the items themselves, along with the user's preferences. This approach makes recommendations for products by finding similarities between the attributes of products the user has liked in the past and those of novel, unknown products. If a user expresses a preference for a particular item, CBF looks for and recommends other items that have the same characteristics. This allows recommendations to be customized to each user's individual preferences.

Hybrid models combine the strengths of both CF and CBF approaches to improve the diversity and accuracy of recommendations. The majority of research in the field of

recommender systems has focused on CF because of its ability to capture user preference patterns from past data.

Embeddings play an important role in the development and improvement of these techniques, especially for content-based filtering. It is also essential to investigate the construction and utility of similarity matrices. The foundation of this method is the idea that objects can be represented in a multidimensional embedding space, where the similarity between two points is determined by their distance from each other, as well explained in detail in Section 2.1.

Central to our methodology is the construction of similarity matrices, wherein the similarity or distance between items is stored. Specifically, for a given item A, its corresponding row in the similarity matrix encapsulates the degree of similarity between A and every other item in the corresponding column. This structure facilitates the identification of items that are similar to A. Motivated by this utility, we have constructed various matrices utilizing different embedding models. This approach allows us to assess the informational richness of the resulting embeddings, thereby providing insights into their effectiveness in capturing the nuances of item similarities.

The initial step involves embedding each item in the catalog, referred to as industrial dataset1, into a high-dimensional space. This process is executed by each of the embedding models selected from the previous section of this chapter. Once embedded, these representations are normalized (if not already done by the last layer of the model itself) and organized into a matrix, where each row corresponds to an item's embedding vector. This matrix serves as the foundation for constructing the similarity matrix. For simplicity, we will refer to it as E .

The similarity matrix is constructed by performing a matrix multiplication between E and its transpose. Mathematically, given S the similarity matrix it is computed as:

$$S = EE^T \tag{4.1}$$

The resulting matrix has the following properties: it is symmetric, with diagonal elements equal to 1, it is square, its values are in a range from -1 to 1, and is positive semidefinite. Each element in the matrix $s_{i,j}$ denotes the similarity between items i and j , where each diagonal element represents self-similarity, thereby justifying the value of 1.

Following the generation of the similarity matrix, a refinement procedure is implemented to enhance its usability and computational efficiency. This procedure entails the removal of negative values and the application of sparsity techniques: we remove negative values

from the matrix by simply clipping it within the range from 0 to 1. In the sparsification process, a sorting of similarity scores within each row of the matrix is performed. Only the top 100 scores are subsequently kept. The selection criterion is based on the idea that, for each item, the most significant connections are those with the highest similarity scores, reflecting the closest item-to-item relationships. Consequently, every other entry in the matrix is set to zero. By concentrating on the most relevant similarities, this reduction not only simplifies the matrix but also drastically lowers processing and storage needs, increasing system efficiency.

The refined similarity matrix serves as the basis for an empirical demonstration of the embedding model capabilities. To this end, the matrix is uploaded to a bucket to be accessed by an AWS EC2 instance, which has been configured to replicate the demo environment of Contentwise that emulates a real-world application scenario. This environment is populated with the catalog data from dataset1, including a wide variety of items, such as films and series episodes.

Within this demo environment, each item's visualization page is enriched with multiple carousels, each corresponding to the results obtained by the similarity matrices derived from different embedding models. This includes the following models: `ada` from OpenAI, `Titan`, `bge-*-en-v1.5` (both base and large variants), and `multilingual-e5-base`. These carousels dynamically showcase the ten most similar items, the item itself excluded, as determined by the sparsified similarity matrix, offering a visual and easy illustration of the richness in the information of the generated embeddings. This deployment offers a tangible platform for assessing the effectiveness and suitability of different embedding models in an actual environment, in addition to serving as a proof of concept for the theoretical concepts described in previous sections. We can evaluate the qualitative effects of various embeddings on recommendation diversity and accuracy by looking at the recommendations made in this demo context. An example is provided in Figure 4.1.

This methodology provides a compelling visualization for comprehending the content understanding capabilities of these models. However, because of the complex nature of the embeddings' performance, choosing the best model outright turns out to be difficult, if not impossible. Furthermore, fully interpreting the reasons behind the observed results remains an intricate work. Navigation through the demo for each item is often unclear the correlations among the displayed item and its ten most similar counterparts within the carousels. These similarities may result from several attributes, such as genre, cast, directorship, or the setting of the content itself. Thus, while this approach offers useful initial observations, it cannot be considered the definitive metric for determining the superiority of one model over others.

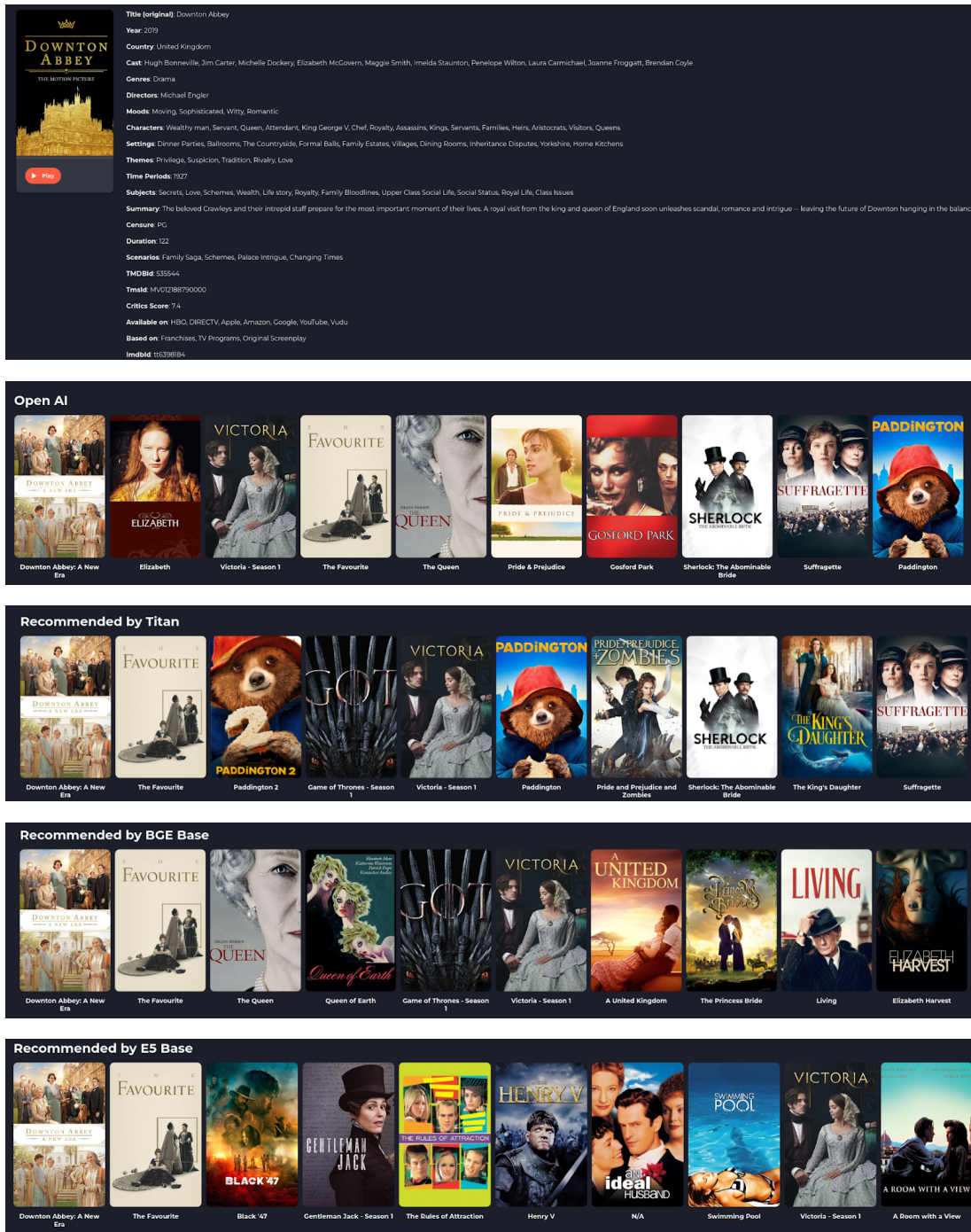


Figure 4.1: Interface snapshot within demo environment, showing carousels of items similar to the selected one, based on similarity matrices developed with different embedding models

4.3. Fine-tuning strategies

Within the domain of deep learning, fine-tuning is a process where a pre-trained model is further optimized to enhance its performance on a more specific task. This approach maintains the original knowledge the model has acquired during its initial training phase, applying it to a smaller, task-specific dataset. The objective of fine-tuning is to refine the model's capabilities, enabling it to generate outputs that are more aligned with the particular needs of the task at hand.

Fine-tuning plays a crucial role in our study, which focuses on the Large Language Model (LLM) for Information Retrieval (IR) in the field of semantic search. Our approach involves fine-tuning two embedding models: `bge-base-en-v1.5` and `multilingual-e5-base`, with a focus on semantic search in retrieving relevant movies or series based on queries. The selection of the model is due to their excellent ability to generate rich embeddings, as explained in details in Section 6.4.

4.3.1. Adapter fine-tuning

The first approach to fine-tune our models involved the application of an adapter mechanism. Adapters facilitate the improvement of pre-trained models by introducing one or more additional layers. This allows specific updates to these extra weights without changing the original model's parameters. This technique allows the specialization of a high-performing model to a specific domain or task, without the need for comprehensive retraining of the entire model architecture.

We implement this method with the `llama-index` library, which simplifies incorporating adapter layers into models built using the Hugging Face framework. The default adapter configuration is characterized by a single fully connected linear layer added on top of the base model. Additionally, the `llama-index` provides the option to employ a two-layer adapter, which requires the specification of an input dimension (corresponding to the output dimension of the underlying model), a hidden dimension, and the desired output dimension. The flexibility of the `llama-index` library also allows for the development of customized adapter configurations, allowing the developer to define any preferred number of layers.

We adopt in our experiments both the default single-layer adapter configuration and the more complex two-layer adapter setup. Each configuration is subject to a fine-tuning process of a single epoch. In particular, we assigned the dimensionality of 1024 to the hidden layer of the two-layer adapter.

For the preparation of our dataset, we utilize the `EmbeddingQAFinetuneDataset` class from the llama-index library, which is adept at formatting data for our specific training needs. This class transforms our dataset, originally stored in a JSON format, into a format compatible with PyTorch's DataLoader. The structured dataset comprises a dictionary labeled 'queries', mapping each query ID to its respective string representation ('query'). Additionally, it includes a dictionary for the 'corpus', linking each corpus ID to a string that concatenates all metadata of an item. Furthermore, a 'relevant_docs' dictionary associates each query ID with a list of relevant corpus IDs, establishing the ground truth for training, based on our QA synthetic training set.

We use the `EmbeddingAdapterFinetuneEngine` class from the llama-index library to start our fine-tuning. This class is specifically designed to facilitate the fine-tuning of our model by leveraging the pre-existing structure and adapter configurations. It requires the original model, a DataLoader containing the formatted dataset, and the name of the folder where the newly fine-tuned model will be saved. Within the `finetune()` method of this class, a customized **MultipleNegativeRankingLoss** is embedded. This loss function is akin to the one available in the `sentence_transformer` library, discussed in Section 2.4. The optimization of the model during the fine-tuning process is handled by the Adam optimizer, a widely used optimization algorithm that adapts the learning rate for each parameter, contributing to a more effective and efficient training phase.

To enhance the fine-tuning strategy for our models, we integrate Automatic Mixed Precision (AMP) into our training process, by setting to True the specific parameter in the `finetune()` method. AMP is a technique that allows for faster training and reduced memory usage by dynamically scaling the numerical precision of floating-point calculations. This method achieves a balance between computational efficiency and model accuracy, facilitating more efficient training iterations.

A significant advantage of fine-tuning with an adapter is its computational efficiency: it does not require extensive computational resources or prolonged periods. Consequently, this optimization is executed locally on a standard personal computer with the completion of one epoch of training in a few hours.

4.3.2. Traditional fine-tuning

As a second method to refining our models, we develop the fine-tuning of the entire model architectures. This method diverges from the adapter fine-tuning method by updating all the model parameters during the training phase and does not add any additional layer to the model structure. Within the context of semantic search for the information re-

trieval area, the **Multiple Negative Ranking Loss** (MNRL) is the most appropriate loss function for this task. MNRL facilitates contrastive learning, wherein the training dataset comprises question-answer pairs (movies or series matching the query). In a training batch, each pair considers the query and the corresponding answer as a positive (similar) instance, whereas the same query and all the other answers as negative (dissimilar) instances. A depth explanation of this learning procedure is described in Section 2.4. This methodological choice addresses the absence of explicit negative samples, a common challenge in Natural Language Processing (NLP) tasks, using the intrinsic diversity within the batch as a source of contrastive examples.

In the implementation of this fine-tuning process, we have opted for a batch size of 5 elements. This decision is driven by the computationally demanding nature of the process and the strategic consideration of contrastive learning dynamics. A smaller batch size mitigates the computational load, ensuring that the training remains feasible on available hardware resources. By maintaining all other answers in a batch as negatives, except for the one corresponding to the query, we inherently increase the model's ability to discern relevant from irrelevant information. However, with larger batch sizes, there is a higher probability of including another answer that may still bear substantial similarity to the query's corresponding answer. This potential overlap may inadvertently introduce ambiguity in the negative samples.

We use the `NoDuplicatesDataLoader` class from the `sentence-transformer` library to improve our training procedure. This class ensures the uniqueness of queries and answers within each batch by subdividing the training set into batches of a size specified by the developer, preventing the duplication of training data. The fine-tuning is performed entirely in one epoch. Also in this case, we integrate AMP strategy and Adam as optimizer. Due to the computational demand of full-model fine-tuning, this process is executed on an AWS EC2 instance specifically configured for this purpose, with a training duration of approximately three days.

In addition to MNRL, another experiment explores the use of **Cosine Similarity Loss** (CSL) for training. In this case, data are still considered in pairs but with a label associated, indicating the similarity between queries and answers. Because of the absence of pre-defined similarity scores within our training set, labels are assigned based on the relative ranking of answers, descending from the most (first) to the least (last) relevant. However, this approach results in model overfitting, leading to a lower capacity for generalization and a tendency to recommend predominantly popular films from the training set, discovered mainly by a manual evaluation.

4.3.3. LoRA fine-tuning

Low-Rank Adaptation (LoRA) is the third methodology adopted for fine-tuning our models, which deviates from traditional fine-tuning practices. LoRA introduces a more sophisticated approach by embedding trainable low-rank decomposition matrices into specific or all layers of the Transformer architecture, thus maintaining the original pre-trained model weights intact. This approach stands out for its efficiency, which can be attributed to these matrices having fewer parameters than the entire model. A thorough explanation of LoRA can be found in Section 2.4.

For the practical implementation of this approach, we utilized the `peft` library, which makes it easy to incorporate these extra trainable matrices and produce a modified model encapsulated within the `PeftModel` class. The configuration of LoRA is conducted via the `LoraConfig` class from the same library, as demonstrated below:

```
peft_config = LoraConfig(
    r=8,
    lora_alpha=8,
    bias="none",
    task_type=TaskType.FEATURE_EXTRACTION,
    target_modules=["key", "query", "value"],
)
```

The `r` parameter specifies the rank of the introduced low-rank matrices. This parameter is crucial in balancing the trade-off between the model's expressiveness and its computational efficiency. A lower `r` value signifies a model with fewer parameters, enhancing efficiency but potentially limiting the model's adaptability to new tasks. Conversely, a higher `r` value augments the model's flexibility but increases computational demands. This parameter is evaluated in configurations of 4 and 8.

The `alpha` parameter, or scaling factor, controls how much the low-rank matrices influence the pre-trained model weights. A higher `alpha` value amplifies the impact of these matrices on the model's behavior, whereas a lower value ensures a more nuanced integration of changes. This parameter is also subjected to experimental validation with values set at 8 and 16, leading to the exploration of three distinct configurations from different combinations of `r` and `alpha`: 4-8, 8-8, and 8-16.

The `bias` parameter addresses the treatment of bias terms within the adapter layers, offering strategies such as "none", "all", or "lora_only" for updating bias terms during the training phase. Opting for "none" precludes the addition of any bias, which, while reducing the model's expressiveness and parameter count, also diminishes its computational

demands.

The `task_type` parameter clarifies the LoRA adaptation's intended application.

"`TaskType.FEATURE_EXTRACTION`" indicates that the adapted model is intended for extracting features from inputs, which is common in tasks like embedding generation or intermediate representations in transfer learning scenarios.

Lastly, the `target_modules` parameter lists the Transformer model components to be adapted using LoRA. Common targets include the "key", "query", and "value" components of attention mechanisms in transformer models. This selective adaptation enables control of the changes in the model behavior and targeted enhancements in model performance for specific tasks.

In this scenario, we designed a custom function to emulate the behavior of a `DataLoader`. This function is designed to load the training dataset and systematically divide it into batches, ensuring that no duplicates exist within a batch in terms of queries and their corresponding answers. The batch size for this method is set at 5 elements, echoing our strategy from previous methodologies to balance computational efficiency with the effectiveness of contrastive learning. Moreover, we developed a custom class responsible for implementing the Multiple Negative Ranking Loss (MNRL). Given that the available libraries, such as `peft`, do not furnish a method for our specific training demands, we undertook the development of the entire training process.

A single epoch of the fine-tuning process with LoRA needs about 5 hours to complete on the same AWS EC2 instance used for the previous methodologies. This effectiveness highlights the suitability of LoRA as a fine-tuning method, particularly in situations where computational resources are a limiting factor.

5 | Implementation

In this chapter, we explain the actual implementation of our system. The previously exposed models serve as the backbone for constructing a vector database capable of performing efficient query and retrieval operations, targeting the top k elements, within the video streaming context. After careful consideration of various technologies, we determined that Weaviate, an AI-native and open-source vector database, stands out as the optimal choice for our requirements.

Initially, the system assimilates the entirety of the dataset's catalog, with raw data and their corresponding embeddings, generated through our integrated embedding model. At the inference stage, the user submits a query, and our model translates it into a vector representation. This vectorized query subsequently serves as the basis for retrieval operations within the vector database. The final phase of the workflow involves an aggregation of the retrieval scores, from semantic and keyword searches, which culminates in the generation of a ranked list of results. This ranked output is then presented to the user. The workflow schema is illustrated in the Figure 5.1 below:

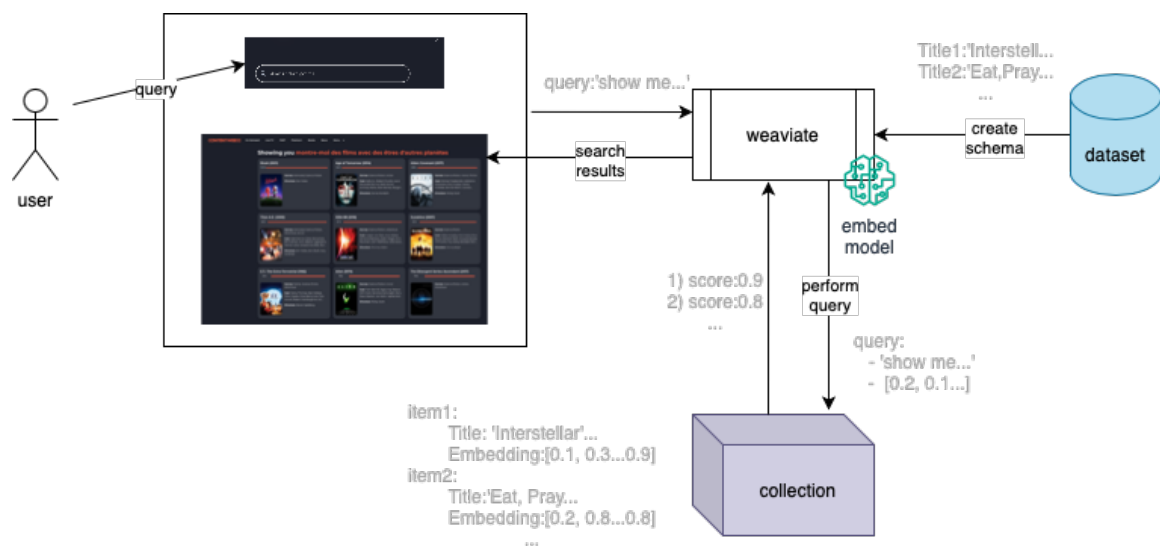


Figure 5.1: Schema of our information retrieval system: from data integration, to query processing and result ranking

5.1. Weaviate

Weaviate is a vector database designed to be AI-native and open-source that assists developers in building robust and intuitive applications that use AI. It allows for hybrid search over unstructured data. Several search methods are used in hybrid search to increase the precision and applicability of search results. The most efficient aspects of vector search techniques and keyword-based search algorithms are combined. It gives customers a more productive search experience by utilizing the advantages of several algorithms. Different algorithms calculate sparse and dense vectors. Dense vectors mostly contain non-zero values, whereas sparse vectors generally contain zero values with just a small number of non-zero values. Machine learning models, like GloVe and Transformers, provide dense embeddings. Sparse embeddings are produced via algorithms such as SPLADE and BM25.

The version of Weaviate we use to deploy our system uses the BM25 (BM is an abbreviation for best matching) algorithm for keyword searches. By using the binary independence model from the Inverse Document Frequency (IDF) calculation and adding a normalization penalty that measures a document's length in relation to the average length of all the documents in the database, BM25 improves upon the keyword scoring technique Term-Frequency Inverse-Document Frequency (TF-IDF). Given a query Q , containing keywords q_1, \dots, q_n the BM25 score of a document D is:

$$\text{score}(D, Q) = \sum_{i=1}^n \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{\text{avgdl}})} \quad (5.1)$$

where $f(q_i, D)$ is the number of times that the keyword q_i occurs in the document D , $|D|$ is the length of the document D in words, and avgdl is the average document length in the text collection from which documents are drawn. Additional static parameters, k_1 and b , are included in BM25 and may be used to tune performance to specific datasets. Weighing each query keyword's uniqueness in relation to the collection of texts yields the document-query pair score. $\text{IDF}(q_i)$ is the IDF weight of the query term q_i . It is computed as:

$$\text{IDF}(q_i) = \ln \left(\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} + 1 \right) \quad (5.2)$$

where N is the total number of documents in the collection and $n(q_i)$ is the number of documents containing q_i . (Wikipedia [28])

Weaviate enables the usage of locally saved customized embedding models, open-source models, and models obtained via OpenAI. For the latter, OpenAI's price for each embedding determines the expenses. The embeddings of every element in the dataset during the

setup phase are kept in this vector database, which also calculates the distance between the query embedding and the stored embeddings at inference time. Distance metrics indicate the degree of similarity or dissimilarity between two embeddings.

The features of both semantic and keyword search techniques are merged in a hybrid search through the combination of dense and sparse vectors. Sparse vectors are better at matching keywords, whereas dense vectors are better at comprehending the context of the query. There are several strategies to merge the outcomes of BM25 and dense vector search into a single ranked list, as explained in Section 1.5. Two Reciprocal Rank Fusion (RRF) algorithms are supported by Weaviate: `rankedFusion` and `relativeScoreFusion`. Reranking is an essential stage in the hybrid search implementation process. Alpha is a parameter that controls how each method is weighted and how the results are reranked. Its value ranges from 0 (a keyword-only search) to 1 (a semantic-only search). Below, Figure 5.2 presents a conceptual visualization aimed at elucidating the mechanics of hybrid search with Weaviate.

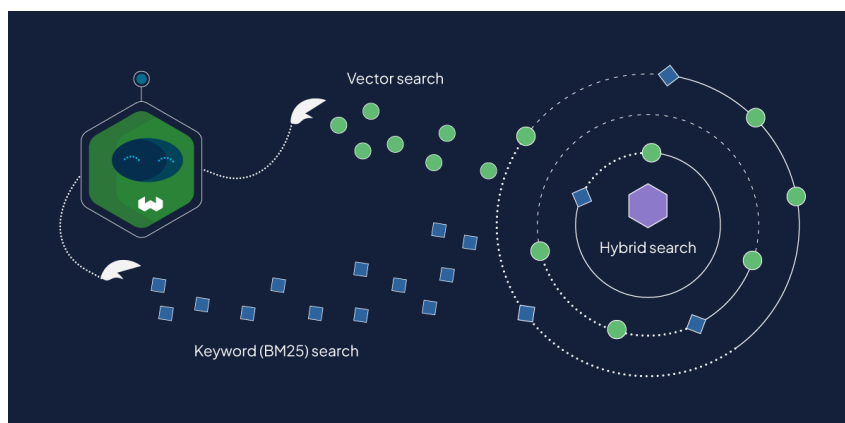


Figure 5.2: Illustration of hybrid search mechanism, from Weaviate blog post

5.2. Schema

In the integration of a vector database such as Weaviate into a system, it is crucial to define the schema and its components. The schema essentially manages the creation of collections, which are named entities facilitating the referencing and interaction with aggregated data. The first step in the Weaviate schema definition is to designate a "class", an essential component that specifies the name of the collection created by the schema. Classes play a pivotal role in organizing the data within the database, allowing for more efficient data retrieval and management. The structure of the Weaviate schema is illustrated by the following example:

```

schema= {
  "class": "Item",
  "vectorizer": "text2vec-transformers",
  "moduleConfig": {
    "text2vec-transformers": {
      "model": "e5b_mnrl",
      "modelVersion": "v2",
      "type": "text",
      "vectorizeClassName": True
    }
  },
  "vectorIndexType": "flat", #hnsw or flat
  "vectorIndexConfig": {
    "distance": "cosine" #cosine, dot, l2-squared, hamming, manhattan
  },
  "properties": [
    {
      "name": "item_id",
      "dataType": ["int"],
      "indexFilterable": True,
      "indexSearchable": False,
      "moduleConfig": {
        "text2vec-transformers": {
          "skip": True,
          "vectorizePropertyName": True
        }
      }
    },
    {
      "name": "n_actors",
      "dataType": ["text[]"],
      "tokenization": "word",
      "indexFilterable": True,
      "indexSearchable": True,
      "moduleConfig": {
        "text2vec-transformers": {
          "skip": False,
          "vectorizePropertyName": True
        }
      }
    },
    ...
  ]
}

```


Concerning the search algorithms, Weaviate uses the BM25 algorithm by default for keyword searches and offers customization for two of its parameters: `'b'` and `'k1'`. This flexibility enables enhanced search accuracy based on specific dataset characteristics. On the other hand, the dense vector search dimension is handled by configurations like `"vectorizer"`, which identifies the embedding model responsible for generating data embeddings, and `"moduleConfig"` for additional parameter settings. The `"vectorIndexConfig"` defines the distance metric for vector comparison. The options it includes are `'cosine'` for cosine similarity, `'dot'` for dot product, and others including `'l2-squared'`, `'hamming'`, and `'manhattan'`.

The selection of the vector indexing approach is incorporated in the `"vectorIndexType"`, which separates the `'flat'` index from `'hnsw'` (Hierarchical Navigable Small World). The `'flat'` index is distinguished by its simple, minimalist construction, which is fast to generate and memory-efficient, but does not support scaling with large data volumes because of its linear search complexity. The `'hnsw'` index, on the other hand, is an advanced implementation of the HNSW algorithm, optimizing for Approximate Nearest Neighbors (ANN) searches through a multi-layered graph structure. By utilizing short-range connections in lower levels for precision and long-range connections in top layers for quick navigation, this method dramatically improves search speed and accuracy, with a logarithmic search complexity.

As a variant of Approximate Nearest Neighbors (ANN) algorithms, the Hierarchical Navigable Small World (HNSW) belongs to the `"graphs"` category, which together with `"trees"` and `"hashes"` constitutes the tripartite categorization of ANN techniques. The layered structure of skip lists serves as inspiration for the HNSW design, which combines shorter edges in lower levels to improve search precision and longer edges in upper layers to enable rapid search traversal. This dual-edged approach optimizes the search process, enabling efficient navigation through the data space. A visual illustration of the HNSW structure can be found below in Figure 5.3:

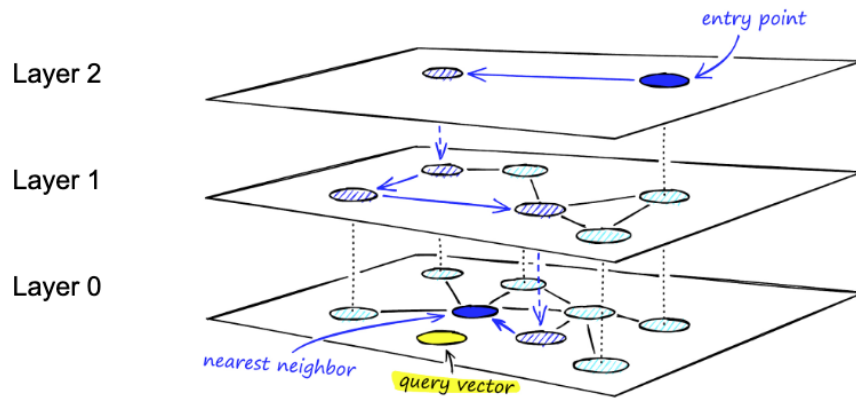


Figure 5.3: Hierarchical Navigable Small World (HNSW) structure

HNSW builds upon the idea of Navigable Small World (NSW) graphs, which rely on the construction of a proximity graph that incorporates both long-range and short-range connections. Compared to more conventional search algorithms, such a design significantly improves search speeds, reducing them to logarithmic complexity. In an NSW graph, each vertex is linked to multiple other vertices, creating a dense network of connections. The search process starts from a predefined entry point. The search progresses towards the closest vertices, with the algorithm terminating when no closer vertices can be found, potentially culminating in a local minimum. This mechanism reflects a sophisticated approach to navigating the graph, ensuring efficient retrieval of nearest neighbors.

HNSW improves the NSW model by incorporating a hierarchical structure, where links are distributed across different layers. This hierarchical arrangement allows for longer links at the top layer, facilitating rapid initial search steps, and shorter links at the bottom layer allow for more precise search results. Even when a local minimum is reached, unlike in the NSW model, the search in HNSW shifts to a lower level, continuing until the local minimum on the lowest layer is identified. The design of HNSW involves storing every node at the bottom layer, with its presence in higher layers determined by a probabilistic distribution similar to that used in skip lists. Reducing the overlap of common neighbors between layers improves HNSW performance and increases search accuracy and efficiency. However, there are trade-offs associated with this complex structure, including a noticeable increase in memory consumption and the possibility of longer search times in some situations. This compromise between resource efficiency and performance improvement highlights the intricacy and inventiveness of the HNSW method in the context of vector similarity searches.

In the subsequent phase of schema definition within Weaviate, the articulation of the ob-

ject data structure is defined. This structure is delineated through a set of "properties", each specifying attributes of the elements within a collection. These properties are defined by a name and a corresponding data type. The schema includes vectorizer settings in the "moduleConfig" to customize the vectorization procedure to certain attributes. It includes the parameter "skip", a boolean setting that, when enabled (set to True), excludes property from vectorization during the indexing phase. This selective vectorization approach allows for control over which attributes contribute to the semantic representation of the data. To further improve the data representation, the design also provides the option to incorporate certain property names throughout the vectorization process.

The "indexFilterable" and "indexSearchable" parameters are optional and by default set to True. They indicate whether a property is indexed for filtering and if it is indexed for access through keyword or hybrid search methods. With the help of these parameters, the search capability may be optimized, allowing for accurate and effective data retrieval based on predetermined criteria.

Another decision in schema configuration involves determining which properties to embed with the vectorizer, which to reserve for keyword search, and which are designated for both functions. Because of the constraints set by the embedding model - the maximum token input limit - this decision is particularly important. Thus, the strategic selection and ordering of metadata for input into the embedding model can maximize semantic relevance and model efficiency. The Table 5.1 provided illustrates this schema configuration strategy, focusing on the catalog from industrial dataset1 (Section 3.1.1) due to its extensive size, a similar strategy is applied to the other catalogs.

Prefix	Properties	Semantic
aa_	TITLEFULL	True
ab_	EPISODENAME	True
b_	SUMMARYLONG	True
c_	COUNTRYOFORIGIN	True
d_	GENRESCRXARRAY	True
e_	SCENARIOARRAY	True
f_	SETTINGSARRAY	True
g_	MOODSARRAY	True
h_	THEMEARRAY	True
i_	CHARACTERARRAY	True
l_	SUBJECTARRAY	True
ma_	YEAR	True
mb_	TIMEPERIODARRAY	True
n_	ACTORSLASTNAMEFIRSTARRAY	True
o_	DIRECTORSLASTNAMEFIRSTARRAY	True
p_	SEMANTICCATEGORIESARRAY	True
q_	SUBGENRESARRAY	True
r_	KEYWORDSARRAY	True
s_	LEADACTOR	True
	AWARDSARRAY	False
	CONCEPTSOURCEARRAY	False
	SHOWTYPE	False
	RUNTIME	False
	SEASONNAME	False

Table 5.1: Division of attributes for industrial dataset 1

Because of the alphabetic storage convention of Weaviate, a 'Prefix' is added atop the name of the attributes to ensure the inclusion of semantically significant properties in the embedding process. This priority is particularly important for datasets with a large number of tokens since the model's performance and the precision of semantic searches can be greatly affected by the features chosen for embedding. It is noteworthy to note that all properties are considered in the BM25 because this does not produce performance degradation or increased latency. This approach mitigates the risk of missing relevant terms that fall outside the embedding model's input token window, thereby enhancing the robustness of the search functionality.

5.3. Search configurations

Once the schema is settled, it is possible to index the whole catalog and populate the collection inside the Weaviate vector database. Subsequently the indexing phase, Weaviate allows for search queries through API calls. It is possible because Weaviate operates as a service encapsulated within a Docker container, featuring a built-in functionality for hybrid search. The evaluation of the system performance is made through a series of experiments employing diverse configurations.

These configurations include the use of best-performing models fine-tuned with LoRA or entirely. The objective is to determine which of these two models has the final best performance in hybrid search tasks for video streaming platforms. Furthermore, the experiments comprehend analysis of different values of the parameter alpha set to 1, 0.7, and 0.5. Alongside tests are made with the encapsulation of metadata from the different datasets available, to test our service on catalogs in various languages. The evaluation methodology comprised two distinct approaches: an automated retrieval of the top twenty items using a hybrid search for a set of test queries from the QA synthetic test set (Section 3.2), and a manual investigation of queries sourced directly from a form (Section 3.3). The latter approach allows for direct observation of how the system's results can vary based on different configurations available in Weaviate. This includes the ability to apply filters within a query, allowing for a more refined search process. Additionally, Weaviate enables the weighting of specific attributes, thereby granting the flexibility to emphasize or de-emphasize particular aspects as more relevant in the keyword search. Through this dual approach, we aimed to capture a comprehensive view of the system's efficacy across a spectrum of search scenarios. The results are reported in detail in the Chapter 6.

It is crucial to underscore that the selection of query configurations does not stick to a binary paradigm of 'right' or 'wrong'. Instead, the best setup depends on the desired outcomes and the specific requirements of the end-users. By experimenting with various settings and scenarios, we demonstrate the system's versatility and capacity to be customized in alignment with the unique preferences and expectations of the final clients.

6 | Results

In this section, we present an exploration of the results obtained from our work. First, we present our findings, highlighting the achievements of our final solution through visual demonstrations from a demo environment and specific examples. Following, we delve into a comprehensive evaluation, beginning with practical applications within the Weaviate system and extending to theoretical metrics assessment. Our investigation covers a broad spectrum, from the performances of different embedding models and fine-tuning techniques to configurations of hybrid search. Through iterative testing and refinement, we investigate these models' efficacy across various linguistic contexts and query types. This approach allows us to initially mark the advancements our solution brings to the field, inviting readers to subsequently explore the depth of our research and the rationale behind our decisions. This exploration aimed to contribute to the ongoing development of sophisticated, efficient Information Retrieval (IR) technologies.

6.1. Metrics

Evaluation metrics for information retrieval (IR) systems define how effectively an index, search engine, or database retrieves results from a resource collection in response to a user's query. Numerous factors are useful to evaluate an IR system's performance, such as relevance, speed, user satisfaction, usability, efficiency, and reliability [30].

Evaluation metrics can be categorized along multiple dimensions: offline versus online, order-aware versus order-unaware, and user-based versus system-based. Online experimentation evaluates user interactions with the search system in a live environment, and offline evaluation measures how well an IR system performs in a controlled setting against a static collection.

Offline metrics are divided into order-aware and order-unaware categories, based on whether the ranking of results influences the metric's score. Order-aware metrics consider the position of relevant results in the list, highlighting the importance of retrieving relevant information and presenting it in an order that reflects its relevance to the user's

query. Otherwise, are order-unaware metrics.

Due to the unavailability of a live demonstration of our system and subsequent online metric collection, we primarily leverage offline metrics for performance evaluation. To simulate user interactions and their search queries within our system, we utilize queries collected through a form (Section 3.3). This approach, while innovative, inherently lacks the direct measurement of user satisfaction and appreciation levels that online metrics can provide.

6.1.1. Offline metrics

First, it is crucial to categorize the retrieval outcomes of our models in comparison to a predefined test set. A ranked list of movies and series generated by a GPT model in response to specific queries serves as a benchmark and constitutes the test set to evaluate the effectiveness of our system (Section 3.2). The concepts worth explaining are:

- True Positive (TP): An item is considered a True Positive if it is retrieved by our IR system and is also present in the GPT-generated test set.
- False Positive (FP): An item is classified as a False Positive if it is retrieved by our system but does not appear in the test set.
- False Negative (FN): A False Negative occurs when an item, despite being included in the test set, is not retrieved by our system.
- True Negative (TN): True Negatives are items not retrieved by our system and absent from the test set. However, in the context of information retrieval, TNs are less relevant since their absence in both system results and the test set does not definitively indicate irrelevance. Thus, TNs are not utilized as an evaluation metric in this study.

Order-unaware metrics

Recall, also known as sensitivity, quantifies the proportion of accurately identified relevant items out of all relevant items in the test set. It is mathematically expressed as:

$$Recall = \frac{TP}{TP + FN} = \frac{|relevantitems \cap retrieveditems|}{|relevantitems|} \quad (6.1)$$

An extension of this concept, Recall@K focuses on the retrieval efficiency within the top K outcomes of an information retrieval system. A perfect Recall@K score means a system that retrieved all pertinent items within the top K results, in our case the same outcome

returned by GPT.

However, Recall@K is not without limitations. One significant drawback is its potential for manipulation; by adjusting K to cover the entire dataset (or close to it), an artificially perfect score can be achieved. Furthermore, Recall@K does not account for the order in which the relevant items are presented, it is order-unaware metrics.

Precision is a metric that measures the accuracy of the retrieval process by estimating the proportion of retrieved instances that are relevant to the user's query. It is also referred to as the positive predictive value in some contexts. It is mathematically represented as:

$$Precision = \frac{TP}{TP + FP} = \frac{|relevantitems \cap retrieveditems|}{|retrieveditems|} \quad (6.2)$$

Recall becomes less useful in modern information retrieval systems, when a single query may return thousands of potentially relevant items. In contrast, Precision at K (Precision@K) remains a useful metric.

Average Precision at K (AP@K) is a metric that averages the precision scores obtained for each of the top K positions, weighted by the relevance of the items at each position. The formula for AP@K is:

$$AP@K = \frac{\sum_{k=1}^K precision@k * rel_k}{numberofrelevantresults} \quad (6.3)$$

rel_k is a binary relevance indicator, which equals 1 if the item at position k is relevant, and 0 otherwise. AP@K offers a single-query evaluation metric, denoted as $AP@K_q$, which represents the average precision calculated for a specific query q at the cut-off K.

The F1-score is a comprehensive metric that synergizes precision and recall into a single measure, employing the harmonic mean to balance the two. This metric, also known as the balanced F-score or traditional F-measure, is calculated as:

$$F1 - score = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad (6.4)$$

Order-aware metric

Mean Average Precision at K (MAP@K) is a metric sensitive to the order in which items are presented, focusing on the aggregated performance across multiple queries. It integrates the precision of retrieved items at every point up to K and their relevance by quantifying the mean of the Average Precision at K (AP@K) scores for each query. For-

mally, MAP@K is defined as:

$$MAP@K = \frac{1}{Q} \sum_{q=1}^Q AP@K_q \quad (6.5)$$

Q represents the total number of queries evaluated, and $AP@K_q$ denotes the Average Precision at K for a single query q . The binary relevance parameter rel_K represents its limitation, precluding the possibility of gradational relevance of the items.

Moving forward to order-aware metrics within the realm of information retrieval evaluation, we introduce Mean Reciprocal Rank (MRR), a metric specifically designed to assess the position of the first relevant result returned by a search system in response to a query. MRR is formulated as:

$$MRR = \frac{1}{Q} \sum_{q=1}^Q \frac{1}{rank_q} \quad (6.6)$$

Q is the total number of queries evaluated, and $rank_q$ indicates the rank position of the first relevant item identified for query q .

However, MRR has its limitations. Primarily, it focuses solely on the rank of the first relevant item, ignoring the overall distribution of relevant items within the retrieved set. Furthermore, because MRR places more emphasis on ranking than on the presence of relevant items, it might still be difficult to fully understand when compared to more intuitive metrics like Recall@K, even though it has a simpler interpretation than some assessment metrics.

To evaluate information retrieval systems with an emphasis on the relevance and ordering of results, we introduce a series of metrics culminating in the Normalized Discounted Cumulative Gain (NDCG). Beginning with Cumulative Gain at position K (CG@K), which aggregates the relevance scores of items up to the K th position:

$$CG@K = \sum_{k=1}^K relevance_k \quad (6.7)$$

It's essential to note that CG@K does not account for the order in which items appear, treating all positions equally. To address this and introduce a penalty for items appearing later in the list, the concept of Discounted Cumulative Gain (DCG) incorporates a logarithmic discounting factor:

$$DCG@K = \sum_{k=1}^K \frac{relevance_k}{\log_2(k+1)} \quad (6.8)$$

where k is the position of the item in the ranked list, and the relevance score of the item at position k is denoted by $relevance_k$. Here, $relevance_k$ is a grading scale from k (first, most relevant) to 0, rather than just a binary value indicating relevant or non-relevant.

The Ideal DCG (IDCG) is then introduced as the optimal DCG value that could be achieved with a perfect ranking of items according to their relevance. The Normalized DCG (NDCG) emerges as a solution to DCG’s interpretability challenge. For a specific cut-off at K , the formula is:

$$NDCG@k = \frac{DCG@k}{IDCG@k} \quad (6.9)$$

Because it is optimized for high relevance and sensitivity to result ordering, NDCG@K is popular in offline evaluations, especially for web search engines, making it powerful and easily interpretable. Nevertheless, it requires a comprehensive understanding of item relevances within a query set, demanding more complex data preparation to discern different levels of relevance across items.

6.2. Visual results

In this section, we first highlight outcomes from our final solution, integrated within a specially configured virtual environment designed to simulate the actual demo environment of Contentwise. This demonstration is designed to offer customers a tangible experience of the service, enabling them to visualize its functionality and directly interact with it. Presented below are three queries (Figure 6.1, Figure 6.2, and Figure 6.3) with their first 9 items retrieved from our system in various languages. It’s important to note that the demo environment contains metadata from industrial dataset 1, all in English. This choice of test queries in languages other than English is driven by our intention to highlight the model’s proficiency in interpreting queries across different languages, showcasing its versatile linguistic capabilities. Consequently, the `multilingual-e5-base` model, fine-tuned using the traditional technique, has been selected and integrated as the final solution for deployment in production within the system. This decision is rooted in its demonstrated ability to effectively understand, generalize, and process multilingual queries, aligning with our objective of offering a universally accessible service.

Showing you **montre-moi des films avec des êtres d'autres planètes**









<p>Titan A.E. (2000) 100%</p>  <p>Genres: Animated, Science fiction, Adventure, Action Cast: Matt Damon, Drew Barrymore, Bill Pullman, John Alberto Leguizamo, Nathan Lane, Janeane Garofalo, Ron Perlman, Alex D... Directors: Art Vitello, Don Bluth, Gary Goldman</p>	<p>Galaxy Quest (1999) 97%</p>  <p>Genres: Comedy, Science fiction, Adventure, Action Cast: Tim Allen, Sigourney Weaver, Alan Rickman, Tony Shalhoub, Sam Rockwell, Daryl Mitchell, Enrico Colanoni, Robin Sachs, Patrick... Directors: Dean Parisot</p>	<p>Lost in Space (1998) 84%</p>  <p>Genres: Science fiction Cast: William Hurt, Mimi Rogers, Heather Graham, Lacey Chabert, Jack Johnson, Gary Oldman, Matt LeBlanc, Jared Harris, Mark... Directors: Stephen Hopkins</p>
<p>Predators (2010) 80%</p>  <p>Genres: Science fiction, Action, Adventure, Thriller Cast: Adrien Brody, Topher Grace, Alice Braga, Walton Goggins, Oleg Taktarov, Laurence Fishburne, Danny Trejo, Louis Ozawa... Directors: Nimród Antal</p>	<p>Alien: Covenant (2017) 74%</p>  <p>Genres: Science fiction, Horror, Thriller Cast: Michael Fassbender, Katherine Waterston, Billy Crudup, Danny McBride, Demian Bichir, Carmen Ejogo, Jussie Smollett, Callie... Directors: Ridley Scott</p>	<p>Star Trek Beyond (2016) 72%</p>  <p>Genres: Science fiction, Adventure, Action, Fantasy Cast: Chris Pine, Zachary Quinto, Karl Urban, Zoe Saldana, Simon Pegg, Anton Yelchin, John Cho, Idris Elba, Sofia Boutella, Lydia Wilson Directors: Justin Lin</p>
<p>Planet of the Apes (2001) 70%</p>  <p>Genres: Science fiction, Action, Adventure Cast: Mark Wahlberg, Tim Roth, Helena Bonham Carter, Michael Clarke Duncan, Paul Giamatti, Estella Warren, Cary-Hiroyuki Tagawa... Directors: Tim Burton</p>	<p>Blush (2021) 69%</p>  <p>Genres: Animated, Science fiction Directors: Joe Mateo</p>	<p>Stargate (1994) 65%</p>  <p>Genres: Science fiction, Adventure, Action Cast: Kurt Russell, James Spader, Jaye Davidson, Viveca Lindfors, Alexis Cruz, Mili Avital, Leon Rippy, John Diehl, Erick Avari Directors: Roland Emmerich</p>

Figure 6.1: query: 'montre-moi des films avec des êtres d'autres planètes' (translation: show me movies with beings from other planets)

Showing you **un film con una donna forte come protagonista**


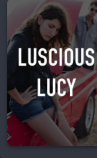

<p>Evita (1996) 100%</p>  <p>Genres: Musical Cast: Madonna, Antonio Banderas, Jonathan Pryce, Jimmy Nani, Julian Littman, Olga Merediz, Andrea Corr, Peter Polycarpou Directors: Alan Parker</p>	<p>I Feel Pretty (2018) 95%</p>  <p>Genres: Comedy Cast: Amy Schumer, Michelle Williams, Rosy Swovel, Emily Ratajkowski, Busy Philipps, Aidy Bryant, Naomi Campbell, Lauren Hutton, Tom... Directors: Abby Kohn, Marc Silverstein</p>	<p>The Help (2011) 92%</p>  <p>Genres: Drama Cast: Viola Davis, Emma Stone, Bryce Dallas Howard, Octavia Spencer, Jessica Chastain, Allison Janney, Sissy Spacek, Ahna O'Reilly... Directors: Tate Taylor</p>
<p>Luscious Lucy (2015) 91%</p>  <p>Genres: Drama Cast: Ngozi Ezeonu, Queen Nwokoye, Eddie Watson Directors: [Not specified]</p>	<p>Suffragette (2015) 74%</p>  <p>Genres: Biography, Historical drama Cast: Carey Mulligan, Helena Bonham Carter, Brendan Gleeson, Anne-Marie Duff, Ben Whishaw, Meryl Streep, Natalie Press, Romola... Directors: Sarah Gavron</p>	<p>Carol (2015) 66%</p>  <p>Genres: Romance, Drama, LGBTQ Cast: Cate Blanchett, Rooney Mara, Sarah Paulson, Jake Lacy, John Magaro, Cory Michael Smith, Carrie Brownstein, Kevin Crowley, Nik... Directors: Todd Haynes</p>
<p>The Danish Girl (2015) 59%</p>  <p>Genres: Biography, Historical drama Cast: Eddie Redmayne, Alicia Vikander, Ben Whishaw, Sebastian Koch, Amber Heard, Matthias Schoenaerts, Pip Torrens, Emerald... Directors: Tom Hooper</p>	<p>The Founders (2016) 52%</p>  <p>Genres: Documentary Cast: Ella Cyr, Bella Lotz, Bryn Vale, Marilyn Smith, Shirley Spork, Louise Suggs, Marlene Bauer Vossler Directors: Charlene Flisk, Carrie Schrader</p>	<p>Bad Moms (2016) 51%</p>  <p>Genres: Comedy Cast: Mila Kunis, Kristen Bell, Kathryn Hahn, Christina Applegate, Oona Laurence, David Walton, Annie Mumolo, Jay Hernandez, Jada... Directors: Jon Lucas, Scott Moore</p>

Figure 6.2: query: 'film con una donna forte come protagonista' (translation: Movies with a strong female as main character)

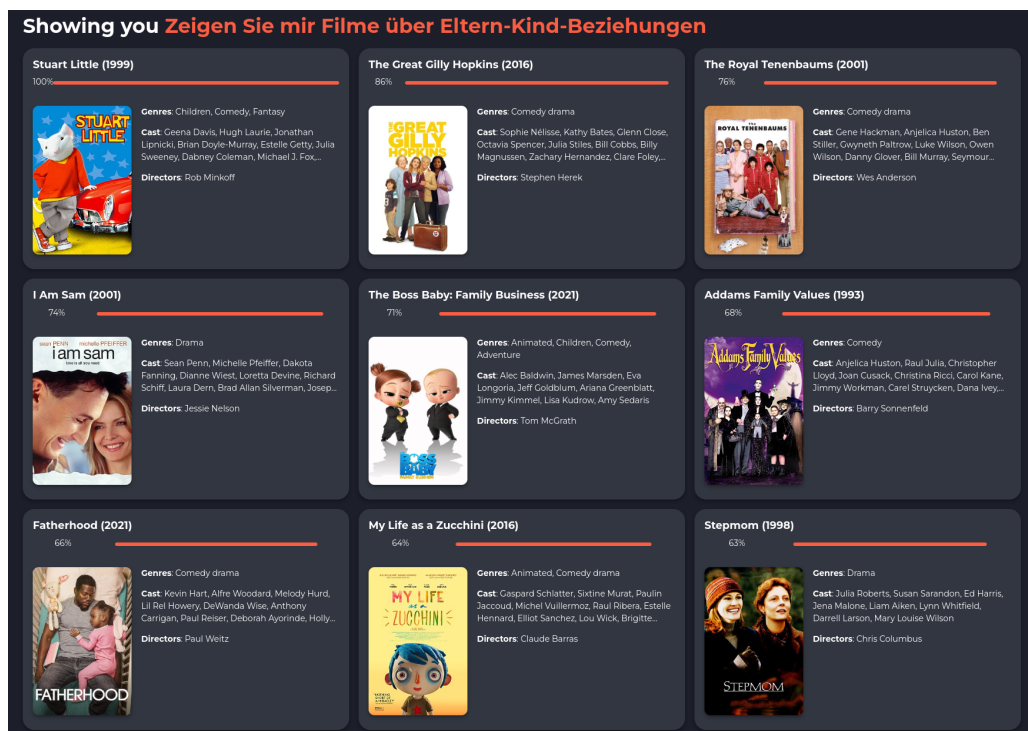


Figure 6.3: query: 'Zeigen Sie mir Filme über Eltern-Kind-Beziehungen' (translation: Show me movies about parent-child relationships)

The decision to opt for the `multilingual-e5-base` model over the `bge-base-en-v1.5` model is driven by several considerations. The selection process focused on these two models due to their respective performances and capabilities. A significant limitation of the `bge-base-en-v1.5` model is its pre-training focused exclusively on English text. Consequently, despite fine-tuning with queries in various languages, it exhibits a notable deficiency in comprehending queries formulated in languages other than English. Instead of generating results pertinent to the query's actual intent, it tends to associate the query with elements related to the language itself. For instance, an Italian query might yield a list of movies set in Italy, have Italian titles, or are otherwise connected to Italy, rather than addressing the query's substantive content.

In contrast, the `multilingual-e5-base` model demonstrates a superior ability to accurately grasp and respond to the intended request in the query, regardless of the language. Because of its built-in multilingualism, this model can interpret and process queries across a diverse linguistic spectrum effectively. Therefore, the `multilingual-e5-base` model with traditional fine-tuning, is our final choice. This decision demonstrates our commitment to providing a solution that is not only versatile across different languages but also accurately responsive to the users' needs, making it the ideal candidate for integration

into the production environment.

	multilingual-e5-base	multilingual-e5-b (MNRL)	multilingual-e5-b (LoRA)
1	'ISRA 88'	'ISRA 88'	'Aliens'
2	'400 Days'	'Star Trek Into Darkness'	'2010'
3	'Battlestar Galactica'	'2010'	'ISRA 88'
4	'Battlestar Galactica'	'Gravity'	'Mars Attacks!'
5	'Battlestar Galactica'	'Passengers'	'Starman'
6	'Blush'	'2001: A Space Odyssey'	'Passengers'
7	'Lost in Space'	'The Space Between Us'	'Dark Matter'
8	'Above and Beyond: NASA's Journey to Tomorrow'	'Above and Beyond: NASA's Journey to Tomorrow'	'Alien'
9	'Battlestar Galactica'	'Lost in Space'	'Arrival'
10	'Battlestar Galactica'	'Arrival'	'Blush'

Table 6.1: Retrieval results of **multilingual-e5-base** and its versions fine-tuned for the query '*viaggi nello spazio*' (Space travel)

	bge-base	bge-base (MNRL)	bge-base (LoRA)
1	'Pompeii'	'Under the Tuscan Sun'	'Malena'
2	'Bicycle Thieves'	'Eat Pray Love'	'Bicycle Thieves'
3	'Under the Tuscan Sun'	'Midnight in Paris'	'The Getaway'
4	'Suburra'	'The Hundred-Foot Journey'	'Under the Tuscan Sun'
5	'Mean Streets'	'La Strada'	'Something in the Air'
6	'Diners, Drive-Ins and Dives'	'Malena'	'Mean Streets'
7	'Night on Earth'	'Bicycle Thieves'	'Le Violon Rouge'
8	'Tini: The New Life of Violetta'	'Suburra'	'The Italian Job'
9	'Eat Pray Love'	'The Italian Job'	'Cannibal Holocaust'
10	'Turistas'	'Lost in Translation'	'The Italian Job'

Table 6.2: Retrieval results of **bge-base-en-v1.5** and its versions fine-tuned for the query '*viaggi nello spazio*' (Space travel)

An illustration of the disparities between the two models discussed is detailed in Table

6.1 and Table 6.2 above, showcasing the results for the query "*viaggi nello spazio*" which translates to "space travel."

Furthermore, we present an example to illustrate the performance of the `multilingual-e5-base` model fully fine-tuned (denoted with 'MNRL') and its version fine-tuned with the LoRA technique. Additionally, we examine the outcomes of these two models when integrated within the Weavite platform, executing hybrid searches with an alpha parameter set at 0.7 (denoted with 'Hybrid'). This comparison aims to showcase the nuanced differences in model effectiveness in a hybrid search environment.

In Table 6.3, the results are displayed for the two fine-tuned versions of the model and their corresponding outcomes when employed in a hybrid search context. The query formulated is "*mostre-me documentários sobre a sustentabilidade do alimento*" (show me documentaries about food sustainability).

	MNRL	MNRL Hybrid	LoRA	(LoRA Hybrid
1	Before the Flood	Parched	Hover	Hover
2	The Founder	Parched	Shark Tank	Shark Tank
3	Parched	Last Call at the Oasis	Diners, Drive-Ins and Dives	Chopped
4	Parched	Parched	Guy's Grocery Games	Before the Flood
5	Pandora's Promise	Hover	Guy's Grocery Games	Guy's Grocery Games
6	Chef	Call the Council	Bob's Burgers	Ice on Fire
7	Last Call at the Oasis	Man v. Food Nation	Guy's Grocery Games	Parched
8	Where to Invade Next	Ice on Fire	Guy's Grocery Games	Guy's Grocery Games
9	Parched	Shark Tank	Chopped	Shark Tank
10	The Hundred-Foot Journey	Diners, Drive-Ins and Dives	Guy's Grocery Games	Parched

Table 6.3: Retrieval results of `multilingual-e5-base` and its versions fine-tuned inside weavite, performing an hybrid search with $\alpha = 0.7$, for the query "*mostre-me documentários sobre a sustentabilidade do alimento*" (show me documentaries about food sustainability)

These results provide insight into the different capacities of each model in understanding

and addressing the specificity of the query. The variety of suggestions, involving broader environmental documentaries alongside those with a culinary focus, highlights a fundamental challenge in semantic search: the capability of models to accurately interpret and reflect the core intent of queries that traverse overlapping thematic domains.

The 'MNRL' model, along with its hybrid version, demonstrates its ability to combine themes of environmental sustainability with culinary content. This synergy reflects the model's capacity to grasp and integrate the multifaceted nature of the query. In contrast, the LoRA model, despite its strengths, shows a predisposition towards emphasizing culinary themes, somewhat at the expense of the sustainability aspect, suggesting a possible overgeneralization.

This performance distinction underscores the 'MNRL' model's ability to interpret complex and theme-interconnected queries, highlighting its suitability for applications that demand nuanced understanding, thereby ensuring search results closely align with the user's intent.

Below, Table 6.4 is presented to illustrate how variations in the alpha parameter influence the outcomes. The query submitted is: 'I would like to watch a sci-fi movie with time travels, possibly with paradoxes, it must include a love story.'

	alpha=0	alpha=0.5	alpha = 0.7	alpha = 1
1	Galaxy Quest	The Map of Tiny Perfect Things	The Map of Tiny Perfect Things	The Map of Tiny Perfect Things
2	The Map of Tiny Perfect Things	I Origins	MyFutureBoyfriend	Predestination
3	The 100	Galaxy Quest	Moonshot	The Time Machine
4	The 100	MyFutureBoyfriend	Time Lapse	12 Monkeys
5	Outlander	Moonshot	12 Monkeys	Time Lapse
6	Before I Fall	The 100	12 Monkeys	MyFutureBoyfriend
7	Before I Fall	The 100	The Time Machine	Blush
8	Power	Time Lapse	12 Monkeys	Passengers
9	Power	Outlander	11.22.63	Age of Tomorrow
10	Black Sails	12 Monkeys	12 Monkeys	Titan A.E.

Table 6.4: Retrieval results of multilingual-e5-base fine-tuned with LoRA technique on dataset 1, with the query *I would like to watch a sci-fi movie with time travels, possibly with paradoxes, it must include a love story.*

The hybrid search mechanism, designed to blend keyword and semantic search capabilities, introduces a strategic approach to query interpretation. By setting the alpha parameter to 0.7, the aim was to enhance the relevance of search results through a nuanced understanding of content.

The results from our evaluations show good performance; however, depending on the query, there may be instances of a scarce or limited selection of items matching the query's intent precisely. To address this variability and mitigate the risk of overfitting to the metadata of dataset 1, on which the model is trained, our approach includes conducting identical tests across different datasets. Specifically, we applied the same queries but sourced the retrieval elements from datasets 2 and 3. This methodology not only aims to assess the impact of having items more closely aligned with the queries in these additional datasets on the model's overall effectiveness and applicability but also verifies its robustness and adaptability to data it has never seen before, ensuring it can cater to a broader range of informational needs.

The query previously mentioned in Table 6.3 is also evaluated using dataset 3. The outcomes derived from employing the `multilingual-e5-base` model, fine-tuned with the traditional approach in a hybrid search context, are documented in Table 6.5 provided below.

	Industrial Dataset 1	Industrial Dataset 3
1	Parched	Fernando Gabeira
2	Parched	Diário de uma Vegana T01 Ep06
3	Last Call at the Oasis	Eating Our Way To Extinction
4	Parched	Eating Our Way To Extinction
5	Hover	Diário de uma Vegana T01 Ep09
6	Call the Council	Diário de uma Vegana T01 Ep10
7	Man v. Food Nation	Rotten
8	Ice on Fire	10 Bilhões - O que tem para comer?
9	Shark Tank	10 Bilhões - O que tem para comer?
10	Diners, Drive-Ins and Dives	Hambúrguer de Feijão com Cheddar Vegano

Table 6.5: Retrieval results of `multilingual-e5-base` fine-tuned, on dataset 1 and dataset 2, with the query "*mostre-me documentários sobre a sustentabilidade do alimento*" (show me documentaries about food sustainability) and $\alpha = 0.7$

However, due to the extensive time and resources required to upload entire datasets into Weaviate and to configure the schemas, a comprehensive exploration of all possible

combinations and queries was beyond our scope. Nonetheless, the variety of comparisons conducted has given us a general idea of the interplay between customer demands, Weaviate configuration, and model selection. This knowledge serves as a crucial guide in customizing the environment to best meet the specific needs of clients, ensuring that the selected model and configuration of Weaviate optimally support their objectives.

6.3. Weaviate hybris search results

This section aims to evaluate also in an automatic way our models integrated into Weaviate, a system where it is possible to perform hybrid search mechanisms, with many possible search customizations.

We conduct evaluation with standard tests, under three distinct settings of Weaviate hybrid search: $\alpha = 1$ (exclusively semantic search), 0.7, and 0.5 (where semantic and keyword searches are weighted equally). These tests were performed on the second version of the synthetic QA test set 1 (with additional queries translated into other languages different from English), embedding the metadata from industrial dataset 1 and a Top K setting of 20.

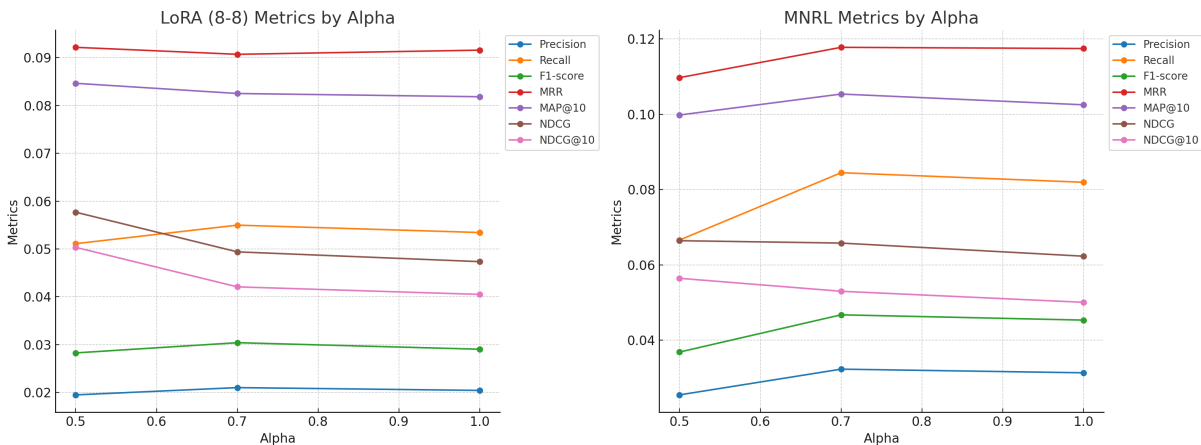


Figure 6.4: Metrics evaluation on test set v2 with catalog 1 of fine-tuned model 'multilingual-e5-base' integrated in Weaviate tested with three alpha configurations (1, 0.7 and 0.5)

Figure 6.4 illustrates the following results: LoRA (8-8) shows slight performance variability with changes in the alpha parameter, lacking a consistent trend tied to any specific alpha value. Traditional fine-tuning (denoted with MNRL) demonstrates a positive correlation with increasing alpha values, indicating a benefit from relying on a higher emphasis on semantic search.

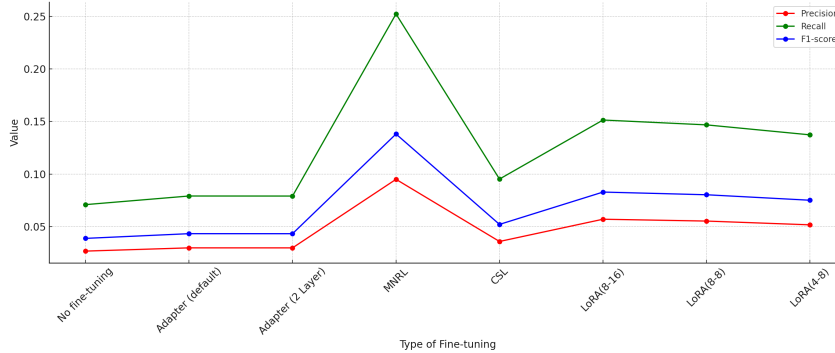
These results serve as a valuable indicator of performance across different hybrid search configurations; however, they do not provide a comprehensive evaluation of how outcomes vary under diverse hybrid search settings. Consequently, the inspections in the previous Section 6.2 conducted manually with several human queries retrieved from the form, as detailed in Section 3.3, hold greater significance. The applicability in real-world scenarios is one of the main factors guiding our selection of the final model for our system.

6.4. Fine tuned model results

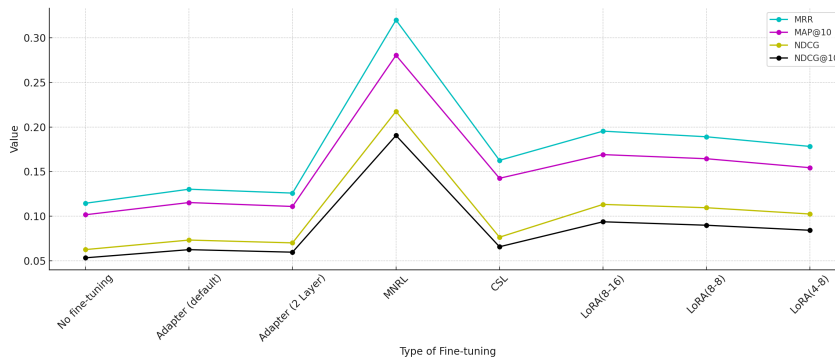
This section delves into the examination of the efficacy of advanced fine-tuning techniques adopted, providing deeper insights that further demonstrate the rationale behind our final choice. Specifically, we explore three distinct strategies: the utilization of Adapters, the comprehensive model updating employing two divergent loss functions—Modified Normalized Rank Loss (MNRL) and Contrastive Semantic Loss (CSL), and the application of Low-Rank Adaptation (LoRA) as delineated in Section 4.3. Here, we evaluate their impact on model performance, aiming to reveal the most effective strategy for refining semantic search outcomes within the constraints of computational efficiency and model scalability, which as previously anticipated are LoRA(8-8) and traditional fine-tuning with MNRL.

For the fine-tuning process, various tests are conducted. The `bge-base-en-v1.5` is the one on which more configurations of this training process are made, not because it stands as the absolute best in terms of performance, nor because it is the smallest in terms of model size and embedding dimensions but for its optimal trade-off. Given that fine-tuning is a resource-intensive process, demanding significant time, memory, and computational power, the `bge-base-en-v1.5` emerged as the preferred choice. For these same considerations, we later opted to fine-tune the `multilingual-e5-base` model rather than its larger counterpart, with only the most effective strategy.

The tests reported below, in Figure 6.5, evaluate the fine-tuning approaches, performed on data of the industrial dataset 1 and the corresponding QA test set. It is important to note that the metrics reported in the table correspond to evaluations where the Top K value is set to 20, even though tests were also carried out with K set to 10 (reported in Appendix A).



(a) Precision, Recall and F1-score on different fine-tuning techniques on `bge-base-en-v1.5` model, dataset 1 and $K = 20$



(b) MRR, MAP@10, NDCG and NDCG@10 on different fine-tuning techniques on `bge-base-en-v1.5` model, dataset 1 and $K = 20$

Figure 6.5: Comparison of order-aware and order-unaware metrics on `bge-base-en-v1.5` model fine-tuned with different techniques, performing query on QA dataset 1 and metadata from industrial dataset 1, retrieving the Top $K = 20$ items per query

In Figure 6.5 the label 'No Fine-tuning' represents the performance of `bge-base-en-v1.5` without fine-tuning. The term 'Adapter(default)' refers to the default configuration of the llama-index library, incorporating a single additional layer, whereas 'Adapter(2Layer)' denotes two supplementary layers atop the model. The acronym MNRL signifies the fine-tuning of the entire model using Multiple Negative Ranking Loss, while CSL indicates a similar full-model update using Cosine Similarity Loss. The graphs also detail three distinct configurations of LoRA (Low-Rank Adaptation), differentiated by the parameters `r` and `alpha`. Specifically, 'LoRA(4-8)' corresponds to a configuration with `r=4` and `alpha=8`, and similarly, 'LoRA(8-8)' and 'LoRA(8-16)'.

From Figure 6.5a and Figure 6.5b several considerations can be made: Adapter (default) exhibits a modest rise across all metrics, with improvements ranging approximately from 11.5% to 17.2%. This indicates a uniform enhancement across all evaluated metrics.

Adapter (2 Layer) is similar to the default adapter setup, with a slightly more limited range of increase, about 9.1% to 12.0%. MNRL marks the most pronounced performance elevation, with increases spanning from 176% to 257.5% across all metrics. CSL also achieves notable enhancements in model performance, with increases observed between 22.2% and 42.0% across all metrics. LoRA demonstrates a significant uplift in performance, with improvements ranging from 60% to 80% across all metrics.

The MNRL fine-tuning method distinctly stands out for its surprising improvements across all metrics, further validated by the manual tests illustrated in the previous Section. While the Adapter-based approaches and CSL yield positive results, LoRA configurations offer more significant and balanced improvements.

In the subsequent Figure 6.6, various graphs are presented to illustrate the performance of the `multilingual-e5-base` model in three distinct conditions: without fine-tuning, fine-tuned entirely (labeled with MNRL), and fine-tuned using LoRA. These results are compared across the three available datasets, offering insights into the impact of different fine-tuning approaches in diverse data environments.

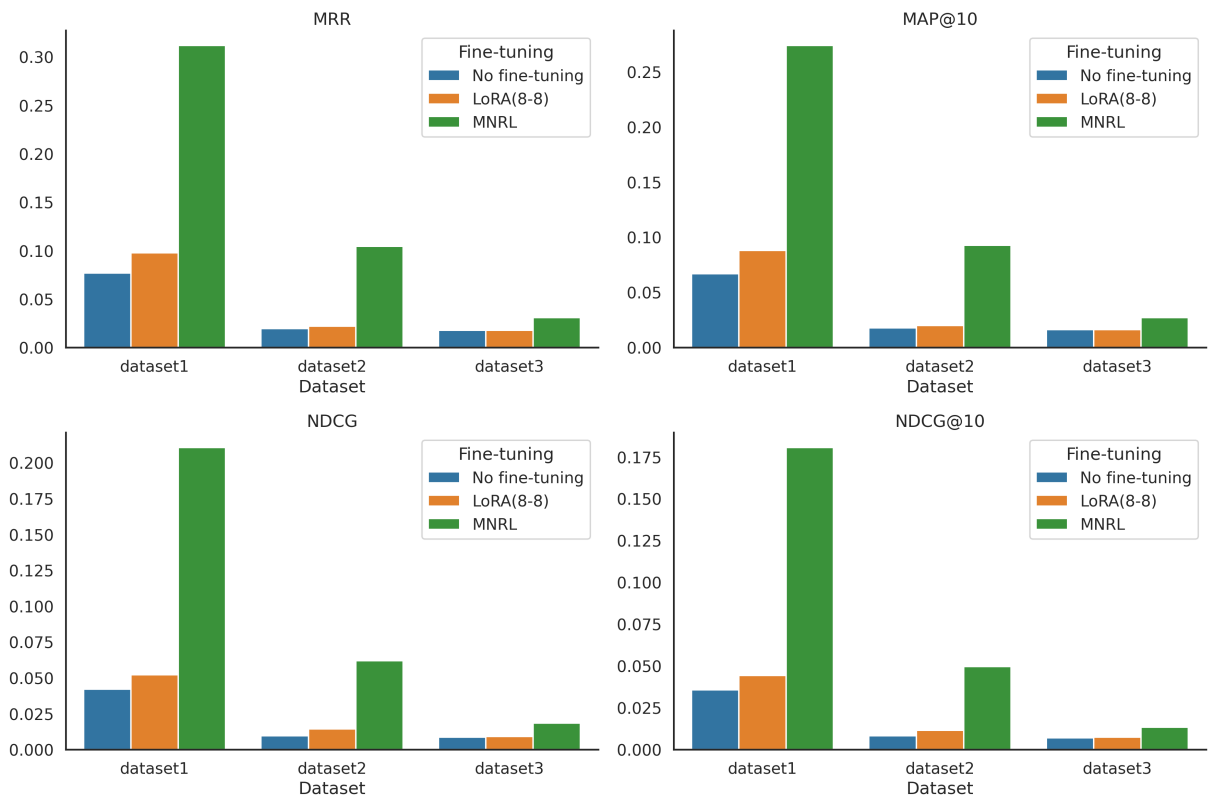


Figure 6.6: Performance comparison of the `multilingual-e5-base` model across different Fine-Tuning approaches and datasets

The comprehensive evaluation of the `multilingual-e5-base` model, when subjected to various fine-tuning approaches across three distinct datasets, highlights the potential of fine-tuning on model performance. The traditional fine-tuning method (updating the whole model), in particular, demonstrates a significant improvement in performance metrics. Across datasets 1 and 2, this method produced enormous performance enhancements, with increases exceeding 300% in several metrics such as MRR, MAP@10, NDCG, and NDCG@10. Importantly, this trend persisted in tests across all three datasets, reinforcing the method’s efficacy and its superiority over the LoRA fine-tuning approach, which showed moderate improvements. The consistent results across datasets 1, 2, and 3 — the latter containing data never seen by the model during training — help exclude the possibility of overfitting or misjudged evaluation, affirming the robustness and generalizability of the traditional fine-tuning method.

The decision to conduct further evaluations exclusively on LoRA (8-8) and traditional fine-tuning is driven by these findings. On the other hand, we opt not to further examine the adapter technique, as it demonstrated no significant improvements. Regarding traditional fine-tuning with CSL loss, while initially promising (yet still less performant than LoRA), a manual evaluation of the actual results revealed that this approach is not adequate. We hypothesize that assigning high labels to the true positives in the training set results in an excessive closeness of the embeddings within the vector space, leading to a tendency to return often similar results and a reduced exploration of the space.

6.5. Model selection results

In this section, we present our evaluations on the model selected, as listed in Section 4.1, without any modifications. These results explain the initial evaluation process applied to the entire list of models, which leads to a smaller subset. This process reduced the number of models to the last two subjected to final evaluations, guiding our decision for the final model selection. Our analysis starts with the industrial dataset 1 and an evaluation based on the respective QA synthetic test set. We examine the model’s performance in scenarios where the retrieved list encloses the first 20 or 10 elements. Subsequently, we delve into an additional evaluation that considers only the availability of the title and overview from industrial dataset 1, aiming to understand the implications of information reduction on model efficacy. We subsequently apply the model to additional QA test sets and metadata from industrial datasets 2 and 3. This step is crucial for assessing the model’s robustness across different languages, thereby providing a comprehensive understanding of its versatility and effectiveness in multilingual contexts.

6.5.1. Tests on dataset 1

After retrieving the number of TP, FP, and FN we can calculate Precision, Recall, and F1-score, treating each query as an individual classification problem. It is important to remember that these metrics might not fully capture the nuances of ranking tasks. These results are calculated on the first version of QA test set 1 (with fewer queries translated) and the industrial dataset 1, as a catalog of the item to retrieve, with top K as 20.

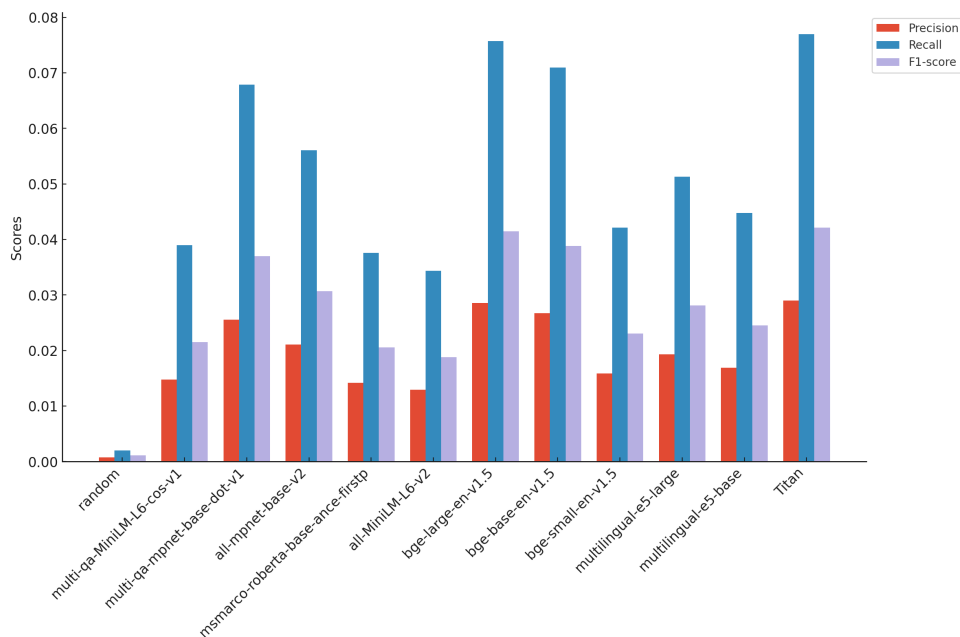


Figure 6.7: Precision Recall and F1-score with industrial dataset 1 and K= 20

The bar graph (Figure 6.7) presents a comparative analysis of various embedding models, elucidating their performance across precision, recall, and F1-score metrics. By establishing Titan as our benchmark for comparison, we observe that `bge-large-en-v1.5` closely aligns with the benchmark’s performance. The distinction in performance between `bge-large-en-v1.5` and `bge-base-en-v1.5` is marginal. Conversely, a substantial performance gap is observed with `bge-small-en-v1.5`.

A further observation from the analysis reveals that models generating smaller embeddings (384 dimensions), namely `multi-qa-MiniLM-L6-cos-v1`, `all-MiniLM-L6-v2`, and `bge-small-en-v1.5`, belong in the lower performance category. This correlation between embedding size and model performance suggests that reduced dimensionality may compromise the model’s ability to capture and utilize semantic relationships effectively.

Interestingly, despite having a larger embedding size (768 dimensions), `msmarco-roberta-base-ance-firstp` performs comparably to the models with smaller

embeddings mentioned above. This similarity in performance may indicate that it can not be adequate for our final purpose.

The graph also features the performance of a random algorithm as a baseline to emphasize the effectiveness of the models. Notably, all models outperform this random baseline, with an approximate increase of F1-score of 4000%.

After the initial evaluation, the analysis considers a scenario wherein the evaluation metric considers the top 10 retrieved items ($k=10$). This additional test is based on the realistic application scenario where the retrieval of a smaller set of items is preferred, assuming that users are most interested in the first results returned by a query. The results are illustrated in the bar plot Figure 6.8:

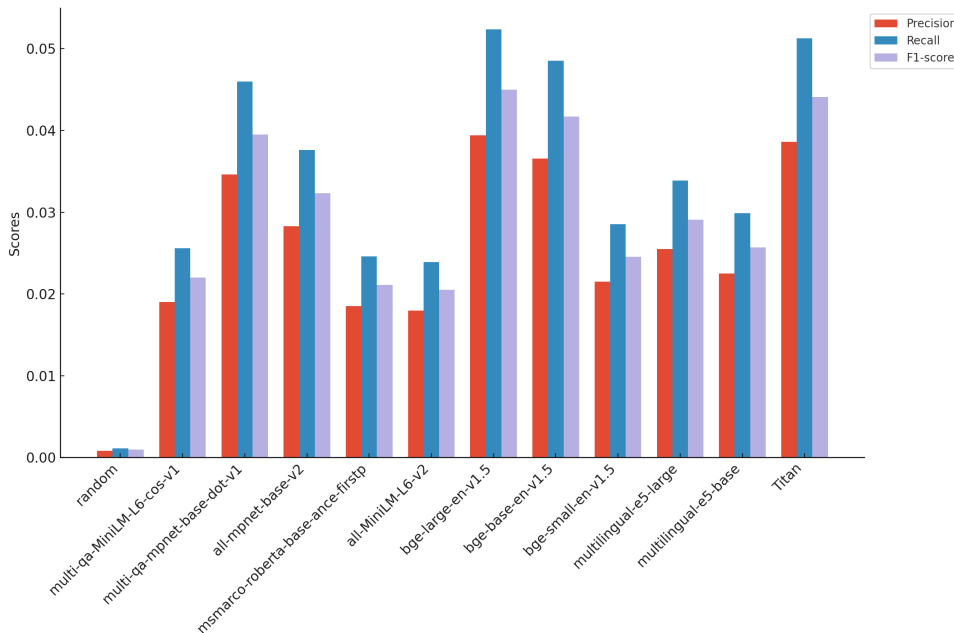


Figure 6.8: Precision Recall and F1-score with industrial dataset 1 and $K=10$

In contrast to the previous results, this evaluation reveals a more uniform distribution across the precision, recall, and F1-score metrics among the models tested. Despite this overall homogeneity, a distinct subset of models still manages to outperform the others, although the dissimilarities in their performance metrics are less pronounced than in the initial evaluation. This observation suggests a leveling effect of the Top $k=10$ criteria on model performance distinctions.

Within this context, the performance of `bge-large-en-v1.5` beats that of the benchmark model `Titan`. The margin of this outperformance is not substantial, from a recall metric of 0.052335, slightly higher than the one of `Titan`, which is 0.051278. It is nevertheless sig-

nificant, underscoring `bge-large-en-v1.5`'s superior efficacy in retrieving relevant items within the top 10 results, further affirming its utility in scenarios where precision in early retrieval ranks is essential.

The next phase of our analysis evaluates the performance of the embedding models using Recall@K for K values ranging from 1 to 10. This test aims to demonstrate two key points: first, as previously anticipated, recall increases as the value of K increases, reflecting a model's improved capability to retrieve a larger set of relevant items. Secondly, it looks for differences in how increasing K affects the performance of the models. This systematic methodology clarifies how scalable the model performance is concerning retrieval depth, providing information on how well each model performs in expanding search fields.

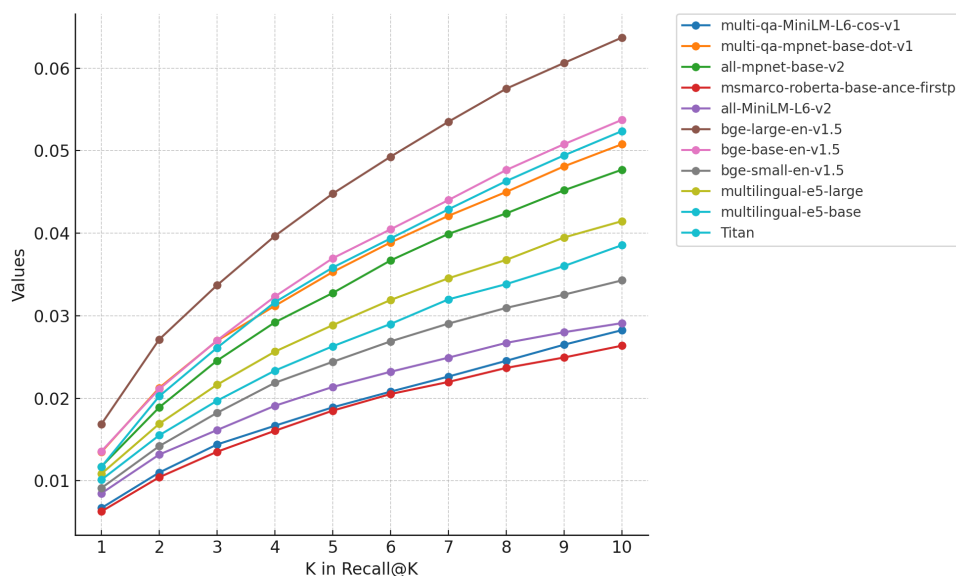


Figure 6.9: Recall@K with K from 1 to 10 in the QA test set 1

In Figure 6.9 we demonstrate the trend across the models where recall increases as K increases. It is immediately clear that `bge-large-en-v1.5` stands out for its superior performance across the entire range of K, starting from Recall@1 through Recall@10. This model not only starts from a higher baseline but also maintains a steeper increase in recall as K grows, highlighting its effectiveness in capturing relevant information even in the top ranks.

In contrast, models like `multi-qa-MiniLM-L6-cos-v1` and `msmarco-roberta-base-ance-firstp` show lower recall values overall, suggesting these models might have difficulties with precision in the early retrieval stages or possibly lack the same degree of semantic understanding and retrieval effectiveness as their counterparts.

Continuing with our comprehensive evaluation of embedding models, we extend our analysis to incorporate order-aware metrics, specifically Mean Reciprocal Rank (MRR), Mean Average Precision (MAP), and Normalized Discounted Cumulative Gain (NDCG). These metrics provide deeper insights into the models' ability to retrieve relevant items while ranking them in an order that reflects their relevance more accurately. The following results reflect the scenario with 20 as Top K retrieval.

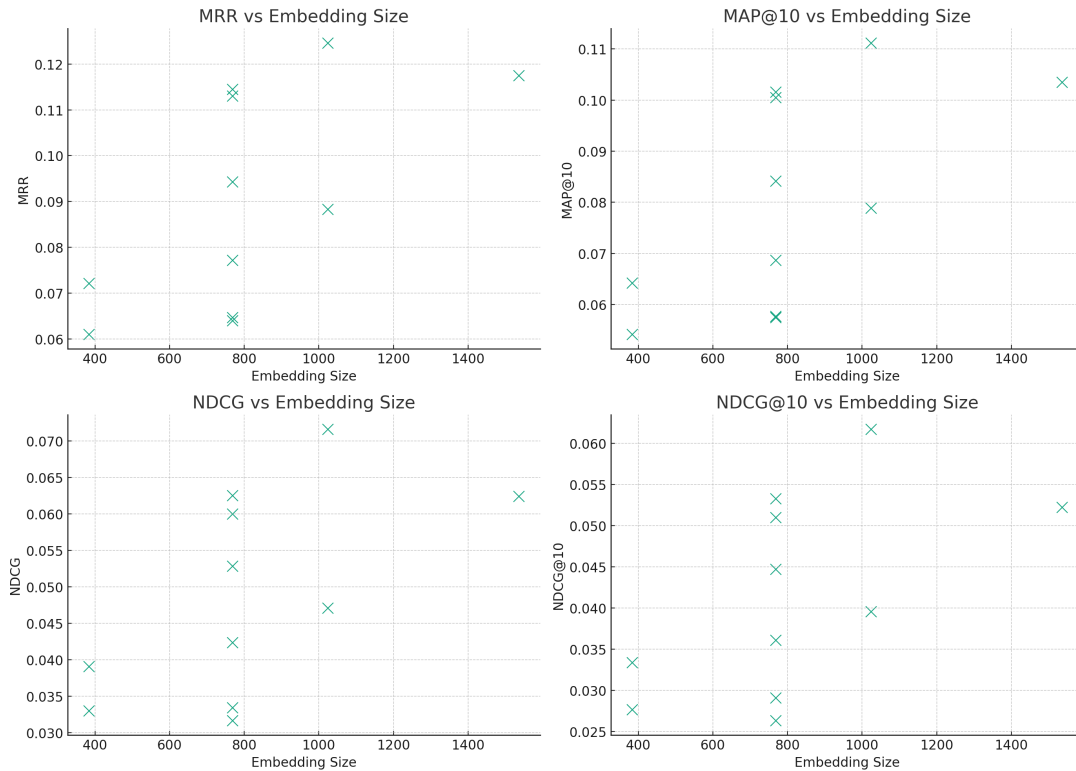


Figure 6.10: Comparative analysis of embedding size impact on model performance with order-aware metrics

The scatter plots for each performance metric (MRR, MAP@10, NDCG, NDCG@10) against embedding size in Figure 6.10 illustrate the relationship between the embedding size of the models and their respective performance metrics. The correlation coefficients between embedding size and each metric are as follows:

- MRR: 0.629
- MAP@10: 0.624
- NDCG: 0.583
- NDCG@10: 0.567

A correlation coefficient can range from -1 to 1, where 1 indicates a perfect positive correlation, -1 indicates a perfect negative correlation, and 0 indicates no correlation. These positive correlation coefficients indicate a moderately strong positive relationship between embedding size and model performance across the metrics evaluated. As the embedding size increases, the performance metrics tend to improve as well, suggesting that models with larger embedding sizes might perform better on the tasks measured by these metrics. However, it's also important to note that while there is a positive trend, the correlation is not perfect, indicating that other factors beyond embedding size also impact these metrics.

Figure 6.11 Figure 6.12 and Figure 6.13 illustrates a bar plot to a more intuitive comparison of the performance of the different embeddings models, displaying their scores in MRR, MAP@10, NDCG, and NDCG@10. The results are again compared with those obtained from a random algorithm for comparative analysis.

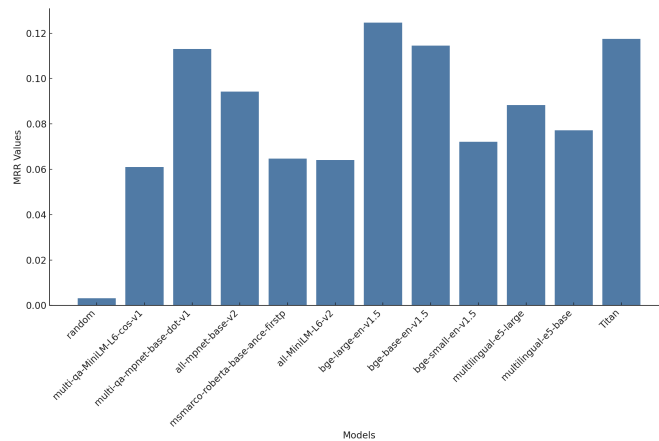


Figure 6.11: Bar plots of MRR on dataset 1 with all embeddings models and K=20

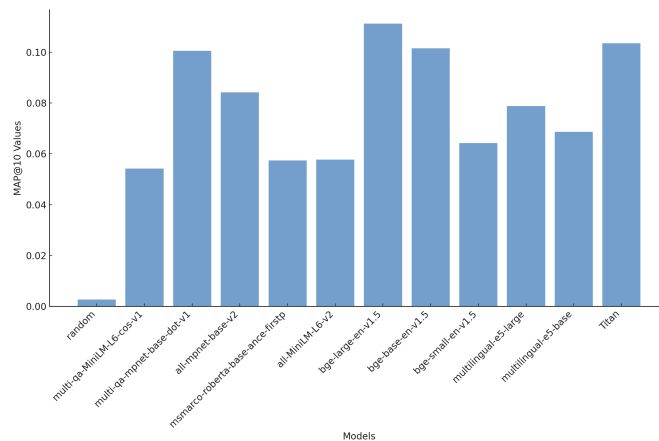


Figure 6.12: Bar plots of MAP@10 on dataset 1 with all embeddings models and K=20

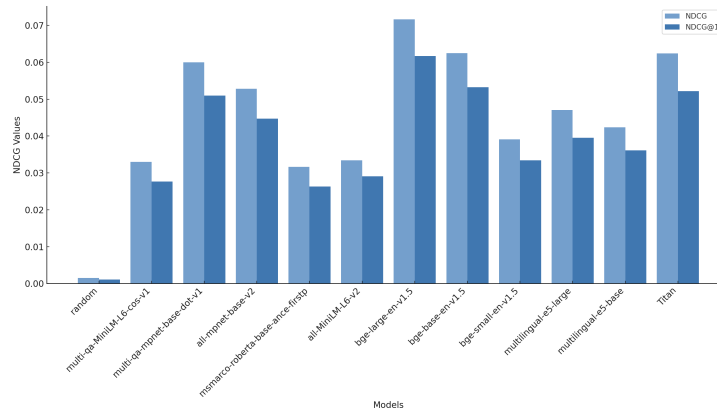


Figure 6.13: Bar plots of NDCG and NDCG@10 on dataset 1 with all embeddings models and $K=20$

The model that performs best in this scenario is **bge-large-en-v1.5**. This is because it is not only the most effective at placing the first relevant document in a higher position within the ranked list (higher MRR) but also maintaining consistent performance in ensuring the top 10 ranked documents are relevant (MAP@10 and NDCG@10), even outperforming Titan. It is noteworthy that **bge-base-en-v1.5** also demonstrates good performances, making it a strong contender for applications requiring precision and depth in search results. **multi-qa-mpnet-base-dot-v1** stands out for its significant performance improvement compared to some smaller or similarly sized models, even if far from top results as **bge-base-en-v1.5**. On the lower end, **msmarco-roberta-base-ance-firstp** and **multi-qa-MiniLM-L6-cos-v1** show more modest results, confirming their difficulties in this task.

Across all models, there is an observable increase in NDCG when moving from the NDCG@10 to the general NDCG score, which considers a broader set of returned items (in this case @20). This pattern underscores a general capability to identify relevant documents beyond the top 10 rankings.

Restricted information scenario

An additional test is conducted on industrial dataset 1, focusing on a more realistic semantic search scenario where only the Title and Overview of catalog elements are embedded, contrasting with previous tests that utilized a more complete dataset. This test aimed to isolate and understand the impact of the quantity and type of information available on the performance of various semantic search models. Also in this scenario, we explore the effects of different retrieval depths, with both the top 10 and top 20 elements. However,

to understand how the modification in data granularity affects model effectiveness, the analysis primarily emphasizes the results for the top 20 elements. The outcome of this focused test is directly compared with those from the initial, more information-rich catalog using the same metrics.

The subsequent graph (Figure 6.16) provides a visual comparison of model performance in these two distinct scenarios:

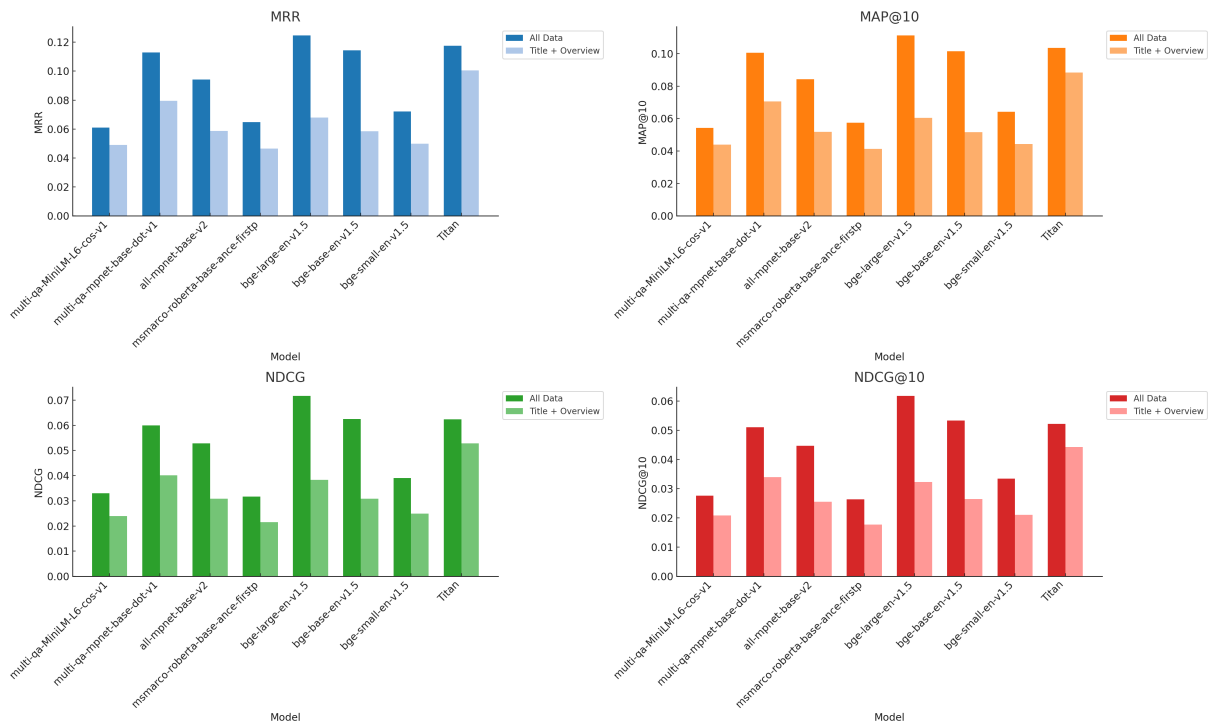


Figure 6.14: Comparison of order-aware metrics in full metadata availability and restricted information scenario across different embedding models with $K=20$

The results reveal varying degrees of robustness among the models when subjected to a change in available metadata. Here, robustness refers to the model's ability to maintain performance despite reduced information. **Titan** emerges as the most robust model, exhibiting the smallest performance degradation across all metrics, around 14 and 15%. **Multi-qa-MiniLM-L6-cos-v1** also demonstrates notable robustness with a moderate decrease in metrics: between 18% for MRR and MAP, and 24 and 27% for NDCG and NDCG@10. In contrast, models like **all-mpnet-base-v2**, **bge-base-en-v1.5**, **bge-large-en-v1.5**, and **bge-small-en-v1.5** exhibited more substantial decreases in performance metrics, indicating a higher sensitivity to the reduction in available metadata. Specifically, **bge-base-en-v1.5** faced the most significant declines, with around a 50% decrease in the metrics.

The evaluation of the order unaware metrics—Precision, Recall, and F1-score—provides an additional perspective on the models’ performance under conditions of reduced meta-data availability. Despite these metrics not being the optimal choice for this specific task, their inclusion offers a more comprehensive view of each model’s robustness. **Titan** again shows the most solidity among the models evaluated, with a decrease in Precision, Recall, and F1-score all around 17%. **bge-base-en-v1.5** exhibits the most significant decrease across these metrics, with Precision, Recall, and F1-score all dropping by approximately 49.9%. The other models experience varying degrees of performance decline, ranging from approximately 28% to nearly 35% in Precision, Recall, and F1-score.

An extensive documentation of all the results presented so far can be found in Appendix A.

6.5.2. Tests on dataset 2 and 3

The next part of our analysis consists of the evaluation of embedding models with two additional datasets, specifically industrial dataset 2 and industrial dataset 3, which differ in contents and language composition from industrial dataset 1. This evaluation aims to evaluate the adaptability of model performance across varied linguistic contexts, offering insights into the generalizability of the models.

Based on the performance in previous tests conducted on industrial dataset 1, a first selection of a subset of the model is made, resulting in the following group: **Titan** also in this scenario as a benchmark model or standard comparison. Alongside, **bge-large-en-v1.5** and **bge-base-en-v1.5** are included due to their superior efficacy and nuanced capabilities in processing and retrieving relevant information. **multi-qa-mpnet-base-dot-v1** is chosen for its significant performance, potentially indicative of its adaptability to various datasets. Lastly, **multilingual-e5-base** is selected to specifically assess the performance of a model designed with multilingual capabilities in mind.

In the following images are illustrated the results of just **bge-large-en-v1.5**, because its superior performance on previous tests, on different datasets compared with **Titan** performances. The other model’s results are reported in Appendix A.

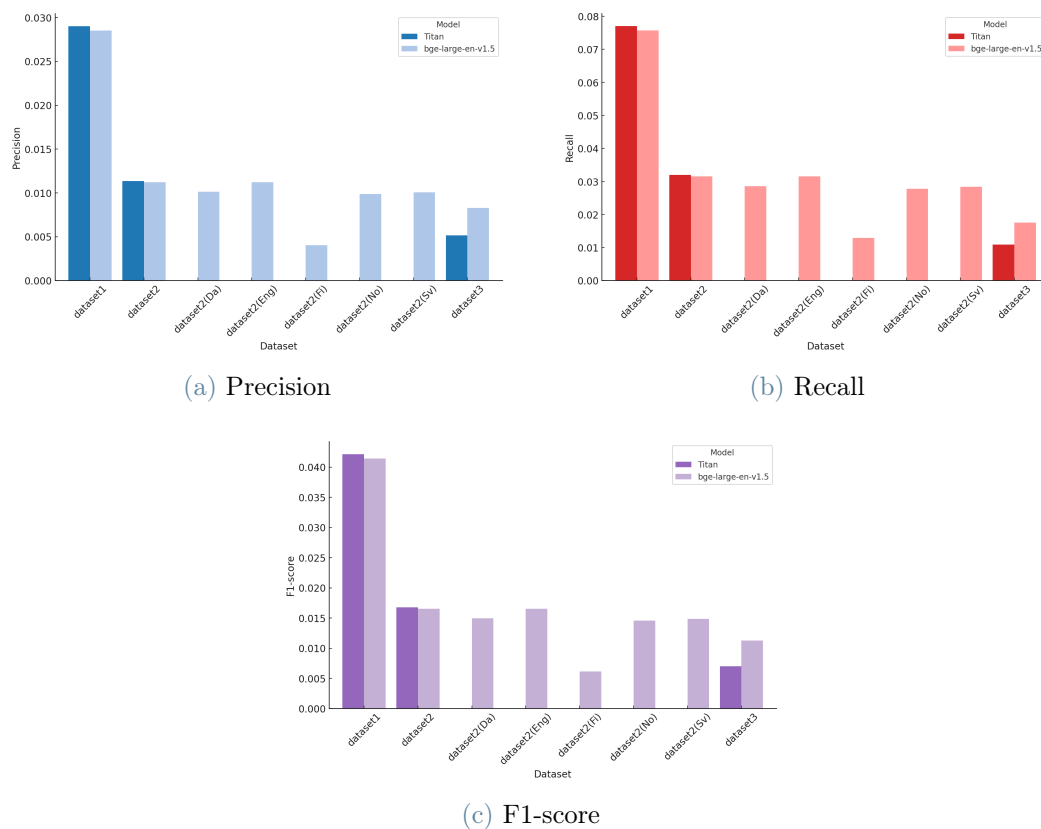


Figure 6.15: The graphs represent the Precision, Recall, and F1-score results of `bge-large-en-v1.5` with different dataset metadata to see how the metadata in different languages influence the model’s results, compared with `Titan` in the most representative examples, and $K=20$

In the illustration presented in Figure 6.15, we delineate and compare the performances of two models, `bge-large-en-v1.5` and `Titan`. The distinction among the datasets, as described in Chapter 3.1.3, aims to measure the influence of metadata language on model performance. Industrial `dataset1`, comprising entirely English metadata, serves as a baseline for model evaluation. In contrast, `dataset2` is segmented into versions based on the language of the metadata, facilitating a direct comparison of how linguistic variations influence model performance. Within `dataset2`, metadata is provided in English (Eng), Danish (Da), Finnish (Fi), Norwegian (No), and Swedish (Sv). In the graphs, with the label ‘`dataset2`’ we refer to the version of the dataset wherein, for each item, the available language versions of the overview (e.g., ‘`overview_eng:...`’, ‘`overview_da:...`’) are included. `Dataset3` introduces metadata in Portuguese, thus expanding the linguistic scope of the evaluation. The test utilizes all metadata from the different catalogs, with a retrieval cutoff of the top $K=20$ results.

To get an overview of the models' performances, we first evaluate standard metrics such as Precision, Recall, and F1-score. When `bge-large-en-v1.5` is applied on multilingual dataset 2, there is a noticeable decrease in performance metrics compared to its application on monolingual dataset 1. Interestingly, the metrics for the English version of dataset2 closely mirror those of the aggregated dataset2. This reflects the fact, as highlighted in Section 3.1.3, that this dataset contains less metadata, particularly for the overview field, which is extremely relevant for accomplishing semantic search.

Further granular analysis of dataset2, delineated by language (Danish-Da, Finnish-Fi, Norwegian-No, Swedish-Sv), reveals disparities in model performance that are not solely due to the linguistic characteristics of the data. Danish and Swedish metadata show performances barely distinct from that of English, suggesting minimal linguistic divergence. Norwegian data records a marginal decline, while Finnish data's performance is noticeably inferior. These observations, however, should be considered in light of the fact, as highlighted previously, that this dataset's versions with fewer items containing the overview metadata exhibit lower performances. This indicates that the disparities in model performance may also reflect the reduced availability of critical metadata in certain language versions, rather than just the model's capacity to understand or process specific languages.

When evaluating dataset3, composed of Portuguese metadata, a significant decrease in performance is observed for both models. `Titan`, despite its design for multilingual comprehension, does not avoid this trend and registers lower performance metrics in comparison to `bge-large-en-v1.5`. It is noteworthy that this third dataset contains no overviews for the items and exhibits a general scarcity of attributes, which also significantly influences the model's performance.

In Figure 6.16, alternative metrics are reported, as more aligned with the objectives of the current study as they offer a nuanced perspective on the models' ability to retrieve and rank relevant items within a dataset. The outcomes support the conclusions drawn from the primary metrics (Precision, Recall, and F1-score). Across the alternative metrics considered, there is a noticeable decrease in the performance of the models when applied to these two other datasets.

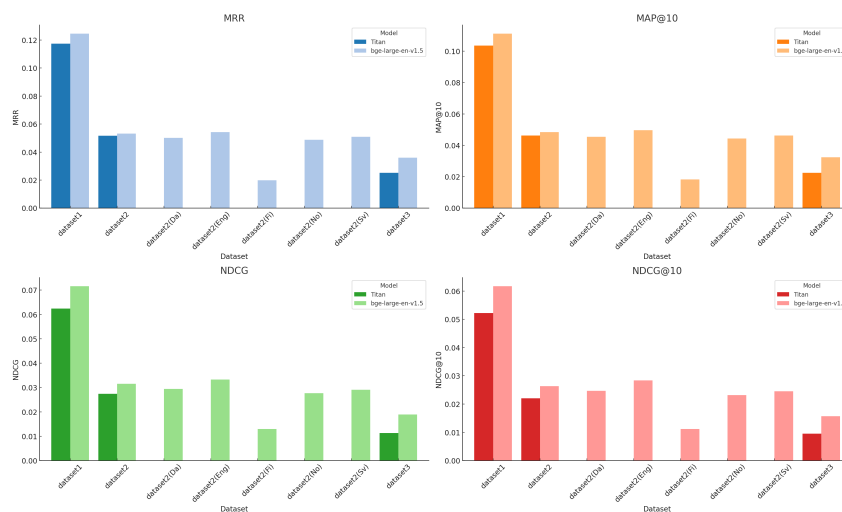


Figure 6.16: The graphs represent the MRR, MAP@10, NDCG, and NDCG@10 results of `bge-large-en-v1.5` with different dataset metadata to see how the metadata in different languages influence the model’s results, compared with `Titan` in the most representative examples, and $K=20$

The diminution in performance is most acute for Finnish language data within `dataset2(Fi)`, with reductions ranging from approximately 81% to 84% across the assessed metrics. `Dataset3` also exhibits a considerable decline in performance, with reductions of approximately 71% to 75% across the metrics, which could reflect the inherent challenges in handling Portuguese data. Conversely, while `dataset2(Eng)` also experiences a decrease in performance, the extent of this reduction is less severe.

This consideration highlights the role of metadata volume and detail in influencing model performance. As previously evidenced, the models exhibit performance degradation when supplied with metadata that is sparse in information content, such as when only titles and overviews are provided. The additional datasets examined not only contain limited information but also suffer from less precise or poorly articulated metadata elements. Such conditions simulate more realistic scenarios where data are unstructured, multilingual, and shallow explicable.

These tests are essential for assessing model robustness in real-world scenarios, highlighting the need for models that excel not only under ideal conditions but also demonstrate strength across varied, possibly challenging, data environments. Moreover, the results underline the significance of well-structured metadata in maximizing model performance, affirming that even the most advanced models benefit from access to comprehensive datasets.

7 | Conclusions and future developments

This chapter delineates and discusses the principal findings, contributions, and implications of this thesis. With the proliferation of digital streaming services, users are often overwhelmed by the vast amount of content options available, making the discovery of relevant and desired content challenging. This difficulty is further aggravated by traditional search mechanisms that rely on precise keyword matches, often failing to understand the complex meanings underlying user queries, further complicated by the potential for typographical errors within queries.

In response to this issue, this thesis introduces a novel approach to content discovery that leverages advancements in Natural Language Processing (NLP) and Information Retrieval (IR) technologies. By developing a hybrid search system that integrates semantic search capabilities with traditional search methods, we aim to transform how users explore and engage with streaming content, enabling a more intuitive and human-like search experience.

7.1. Contributions

Our research has made significant improvements in the realm of streaming platforms, primarily through the development of innovative technologies and methodologies to improve user experience in content discoverability. We have successfully developed a novel hybrid search system that integrates semantic search capabilities with traditional keyword search methodologies, enabling a more human-like interaction with digital content catalogs.

One of our main research contributions is the development and integration of a custom embedding model, specifically designed for the streaming platform context. By fine-tuning an open-source model, we designed a solution that is optimized for the challenges and requirements inherent in streaming content discovery. This approach increased the model's performance and adaptability, ensuring its alignment with the requirements of

our task.

Our work comprehends an exploration and application of advanced fine-tuning techniques, including Adapters, traditional fine-tuning, and Low-Rank Adaptation (LoRA). By effectively applying these techniques, we have demonstrated their potential in refining the semantic understanding and retrieval capabilities of our embedding models.

Another significant contribution to our study is the creation of an exhaustive evaluation framework, allowing for a detailed analysis of embedding models and fine-tuning techniques across languages and datasets. This evaluation approach has been crucial in validating the efficacy of our contributions.

Finally, our work includes the successful integration of the optimized models into the Weaviate system. This achievement not only demonstrates that implementing sophisticated IR models in real-world search systems is feasible but also indicates the potential of our research to revolutionize the field of content discoverability on streaming platforms.

7.2. Limitations

Our research encounters certain limitations that merit discussion. The efficacy and precision of the task are intrinsically subjective, varying according to the final results expected by different users or clients. Our model's design handles this variability by allowing customizable weighting between semantic and keyword-based queries. However, this flexibility also introduces complexity in determining a universally optimal balance and is difficult to determine.

A significant challenge encountered in our research is the availability and quality of data. We observed that a lack of metadata leads to a degradation in model performance. Real-world scenarios, as demonstrated by datasets from ContentWise's partner companies, often feature sparse data availability. This limitation not only impacts the effectiveness of our system but also reflects a common constraint in similar scenarios across the industry.

The generation of our QA datasets, synthesized with `gpt-3.5-turbo`, presents two primary limitations. Firstly, the use of such technology has a cost. Secondly, as GPT is a Large Language Model (LLM) with knowledge up to 2021, it can make errors or miss possible solutions in the synthesized datasets.

Developing a system that operates effectively in real time imposes stringent requirements on latency and computational efficiency. Despite the demonstrated robustness of larger model versions, we opted for smaller, slightly less performant models due to their lower

response times and reduced memory requirements. This trade-off between performance and operational efficiency highlights a critical limitation in the practical deployment of advanced IR models.

Our investigation into multilingual support revealed a trade-off between specialized performance in a single language versus broader linguistic comprehension. Models specifically optimized for IR tasks in English showed superior performance on English queries but struggled with other languages. Conversely, multilingual models, while offering broader linguistic support, tended to perform slightly worse in IR-specific tasks. This limitation points to the inherent challenge of developing a model that excels both in specific language tasks and across multiple languages.

7.3. Future Work

From the work of Lu et al. [14], the integration of a reranking model, trained and fine-tuned for reorganizing retrieval outputs, emerges as a promising direction. However, a critical examination is required to verify whether the computational overhead introduced by such a model increases the response latency, thus maintaining a balance between performance and efficiency.

Furthermore, inspired by the research of Penha et al. [17], a preliminary model dedicated to refining user queries presents a possible improvement. Given the common occurrence of typographical errors, the use of acronyms, or the formulation of queries that lack clarity, the deployment of a model capable of modifying user queries could substantially increase the effectiveness of an information retrieval system.

In the evolving domain of Natural Language Processing (NLP), the advent of new models and technologies is a constant. The introduction of new embedding models, both multilingual and highly performant on IR tasks, could offer the possibility for future enhancements.

The potential applications of our system extend beyond the realm of streaming media. The flexible and adaptive nature of our approach makes it suitable for a wide array of other domains. For example, sectors such as e-commerce, cosmetics, and pharmacy, which feature extensive product catalogs, would benefit from a more intuitive search experience.

Bibliography

- [1] A. Aghajanyan, L. Zettlemoyer, and S. Gupta. Intrinsic dimensionality explains the effectiveness of language model fine-tuning, 2020.
- [2] R. E. Bellman. *Dynamic Programming*. Princeton University Press, 1957.
- [3] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, 2003. URL <https://api.semanticscholar.org/CorpusID:221275765>.
- [4] C. Channarong, C. Paosirikul, S. Maneeroj, and A. Takasu. Hybridbert4rec: a hybrid (content-based filtering and collaborative filtering) recommender system based on bert. *IEEE Access*, 10:56193–56206, 2022.
- [5] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.
- [6] T. Cover. Estimation by the nearest neighbor rule. *IEEE Transactions on Information Theory*, 14(1):50–55, 1968.
- [7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Open sourcing bert: State-of-the-art pre-training for natural language processing. *Google AI Blog*, 2018. URL <https://blog.research.google/2018/11/open-sourcing-bert-state-of-art-pre.html>.
- [8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.
- [9] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.
- [10] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models, 2021.
- [11] W. Li, Y. Zhang, Y. Sun, W. Wang, W. Zhang, and X. Lin. Approximate nearest

- neighbor search on high dimensional data — experiments, analyses, and improvement (v1.0), 2016.
- [12] Y. Li and T. Yang. Word embedding for understanding natural language: A survey. 2018. URL <https://api.semanticscholar.org/CorpusID:65057131>.
- [13] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.
- [14] J. Lu, K. B. Hall, J. Ma, and J. Ni. Hyrr: Hybrid infused reranking for passage retrieval. *ArXiv*, abs/2212.10528, 2022. URL <https://api.semanticscholar.org/CorpusID:254877343>.
- [15] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space, 2013.
- [16] S. I. Nikolenko. Synthetic data for deep learning, 2019.
- [17] G. Penha, E. Palumbo, M. Aziz, A. Wang, and H. Bouchard. Improving content retrievability in search with controllable query generation, 2023.
- [18] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing*, 2014. URL <https://api.semanticscholar.org/CorpusID:1957433>.
- [19] L. E. Peterson. K-nearest neighbor. *Scholarpedia*, 4(2):1883, 2009.
- [20] R. Puri, R. Spring, M. Patwary, M. Shoeybi, and B. Catanzaro. Training question answering models from synthetic data. *arXiv preprint arXiv:2002.09599*, 2020.
- [21] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.
- [22] S. E. Robertson and H. Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3:333–389, 2009. URL <https://api.semanticscholar.org/CorpusID:207178704>.
- [23] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *The Web Conference*, 2001. URL <https://api.semanticscholar.org/CorpusID:8047550>.
- [24] R. Shaikh. A comprehensive guide to understanding bert from beginners to advanced, 2023. URL <https://medium.com/@shaikhrayyan123/a-comprehensive-guide-to-understanding-bert-from-beginners-to-advanced-2379699e2b51>

- [25] C. Y. Suen. n-gram statistics for natural language understanding and text processing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):164–172, 1979. doi: 10.1109/TPAMI.1979.4766902.
- [26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2023.
- [27] L. Wang, N. Yang, X. Huang, L. Yang, R. Majumder, and F. Wei. Multilingual e5 text embeddings: A technical report, 2024.
- [28] Wikipedia contributors. Okapi bm25 — Wikipedia, the free encyclopedia, 2024. URL https://en.wikipedia.org/w/index.php?title=Okapi_BM25&oldid=1194828429.
- [29] Wikipedia contributors. Word n-gram language model — Wikipedia, the free encyclopedia, 2024. URL https://en.wikipedia.org/w/index.php?title=Word_n-gram_language_model&oldid=1201932063.
- [30] Wikipedia contributors. Evaluation measures (information retrieval) — Wikipedia, the free encyclopedia, 2024. URL [https://en.wikipedia.org/w/index.php?title=Evaluation_measures_\(information_retrieval\)&oldid=1210654800](https://en.wikipedia.org/w/index.php?title=Evaluation_measures_(information_retrieval)&oldid=1210654800).
- [31] S. Xiao, Z. Liu, P. Zhang, and N. Muennighoff. C-pack: Packaged resources to advance general chinese embedding, 2023.
- [32] L. Xiong, C. Xiong, Y. Li, K.-F. Tang, J. Liu, P. Bennett, J. Ahmed, and A. Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval, 2020.
- [33] W. Xu, M. Maimaiti, Y. Zheng, X. Tang, and J. Zhang. Auto-mlm: Improved contrastive learning for self-supervised multi-lingual knowledge retrieval, 2022.

A | Appendix A

A.1. Model selection

Below are presented results for offline testing considering all the embedding model take into consideration.

A.1.1. dataset 1

Model	TP	FP	FN
random	181	235759	88688
multi-qa-MiniLM-L6-cos-v1	3497	232443	85372
multi-qa-mpnet-base-dot-v1	6033	229907	82836
all-mpnet-base-v2	4989	230951	83880
msmarco-roberta-base-ance-firstp	3340	232600	85529
all-MiniLM-L6-v2	3054	232886	85815
bge-large-en-v1.5	6732	229208	82137
bge-base-en-v1.5	6304	229636	82565
bge-small-en-v1.5	3746	232194	85123
multilingual-e5-large	4561	231279	84308
multilingual-e5-base	3980	231960	84889
Titan	6842	229098	82027

Table A.1: Calculous of True Positive (TP), False Positive (FP), and False Negatives (FN) by all selected models with K=20 on industrial dataset1

Model	K	Precision	Recall	F1-score
random	20	0.000767	0.002037	0.001114
random	10	0.000839	0.001114	0.000957
multi-qa-MiniLM-L6-cos-v1	20	0.014800	0.039000	0.021500
multi-qa-MiniLM-L6-cos-v1	10	0.019000	0.025600	0.022000
multi-qa-mpnet-base-dot-v1	20	0.025570	0.067886	0.037000
multi-qa-mpnet-base-dot-v1	10	0.034627	0.045967	0.039499
all-mpnet-base-v2	20	0.021100	0.056100	0.030710
all-mpnet-base-v2	10	0.028300	0.037600	0.032300
msmarco-roberta-base-ance-firstp	20	0.014156	0.037583	0.020566
msmarco-roberta-base-ance-firstp	10	0.018513	0.024575	0.021100
all-MiniLM-L6-v2	20	0.012940	0.034365	0.018805
all-MiniLM-L6-v2	10	0.017988	0.023878	0.020518
bge-large-en-v1.5	20	0.028530	0.075750	0.041452
bge-large-en-v1.5	10	0.039420	0.052335	0.044972
bge-base-en-v1.5	20	0.026719	0.070930	0.038817
bge-base-en-v1.5	10	0.036569	0.048540	0.041710
bge-small-en-v1.5	20	0.015877	0.042151	0.023066
bge-small-en-v1.5	10	0.021500	0.028540	0.024530
multilingual-e5-large	20	0.019330	0.051323	0.028084
multilingual-e5-large	10	0.025498	0.033848	0.029085
multilingual-e5-base	20	0.016869	0.044785	0.024507
multilingual-e5-base	10	0.022523	0.029898	0.025691
Titan	20	0.028999	0.076990	0.042129
Titan	10	0.038600	0.051278	0.044060

Table A.2: Performances evaluated with Precision, Recall, and F1-score of all selected models with dataset1 and K=20, 10

Model	Recall@1	Recall@2	Recall@3	Recall@4	Recall@5
multi-qa-MiniLM-L6-cos-v1	0.00670	11.00000	0.01440	0.01667	0.01890
multi-qa-mpnet-base-dot-v1	0.01348	0.02126	0.02697	0.03120	0.03530
all-mpnet-base-v2	0.01173	0.01890	0.02456	0.02920	0.03276
msmarco-roberta-base-ance-firstp	0.00629	0.01043	0.01352	0.01606	0.01850
all-MiniLM-L6-v2	0.00848	0.01318	0.01615	0.01909	0.02137
bge-large-en-v1.5	0.01687	0.02713	0.03370	0.03967	0.04479
bge-base-en-v1.5	0.01358	0.02111	0.02701	0.03232	0.03696
bge-small-en-v1.5	0.00912	0.01420	0.01824	0.02187	0.02442
multilingual-e5-large	0.01085	0.01691	0.02164	0.02565	0.02886
multilingual-e5-base	0.01012	0.01551	0.01970	0.02335	0.02630
Titan	0.01170	0.02025	0.02612	0.03164	0.03584

Table A.3: Recall@K with K from 1 to 5 of all selected models with K=20

Model	Recall@6	Recall@7	Recall@8	Recall@9	Recall@10
multi-qa-MiniLM-L6-cos-v1	0.020800	0.022626	0.024537	0.026489	0.028247
multi-qa-mpnet-base-dot-v1	0.038880	0.042100	45.000000	0.048100	0.050777
all-mpnet-base-v2	0.036700	0.039910	0.042400	0.045200	0.047690
msmarco-roberta-base-ance-firstp	0.020510	0.021974	0.023677	0.024939	0.026369
all-MiniLM-L6-v2	0.023207	0.024918	0.026710	0.028001	0.029100
bge-large-en-v1.5	0.049269	0.053500	0.057508	0.060630	0.063700
bge-base-en-v1.5	0.040467	0.044009	0.047653	0.050788	0.053727
bge-small-en-v1.5	0.026890	0.029044	0.030949	0.032555	0.034279
multilingual-e5-large	0.031905	0.034534	0.036770	0.039481	0.041457
multilingual-e5-base	0.029001	0.031980	0.033830	0.036021	0.038550
Titan	0.039360	0.042890	0.046309	0.049430	0.052370

Table A.4: Recall@K with K from 6 to 10 of all selected models with K=20

Model	MRR	MAP@10	NDCG	NDCG@10	embed size
random	0.00312	0.0026645	0.00152	0.00111	-
multi-qa-MiniLM-L6-cos-v1	0.06100	0.054156	0.033000	0.027639	384
multi-qa-mpnet-base-dot-v1	0.113000	0.100500	0.060000	0.051000	768
all-mpnet-base-v2	0.094300	0.084169	0.052860	0.044700	768
msmarco-roberta-base-ance-firstp	0.064700	0.057450	0.031655	0.026322	768
all-MiniLM-L6-v2	0.064030	0.057680	0.033406	0.029107	768
bge-large-en-v1.5	0.124610	0.111201	0.071632	0.061730	1024
bge-base-en-v1.5	0.114478	0.101580	0.062520	0.053287	768
bge-small-en-v1.5	0.072087	0.064195	0.039098	0.033397	384
multilingual-e5-large	0.088270	0.078843	0.047090	0.039557	1024
multilingual-e5-base	0.077212	0.068639	0.042360	0.036103	768
Titan	0.117470	0.103500	0.062400	0.052220	1536

Table A.5: Performances evaluated with MRR, MAP@10, NDCG and NDCG@10 of all selected models with dataset1 and K=10

Model	Precision	Recall	F1-score	Catalog
multi-qa-MiniLM-L6-cos-v1	0.014800	0.039000	0.021500	all data
multi-qa-MiniLM-L6-cos-v1	0.009888	0.026000	0.014000	title + overview
multi-qa-mpnet-base-dot-v1	0.025570	0.067886	0.037000	all data
multi-qa-mpnet-base-dot-v1	0.017700	0.047000	0.025700	title + overview
all-mpnet-base-v2	0.021100	0.056100	0.030710	all data
all-mpnet-base-v2	0.013750	0.036550	0.019980	title + overview
msmarco-roberta-base-ance-firstp	0.014156	0.037583	0.020566	all data
msmarco-roberta-base-ance-firstp	0.009740	0.025870	0.014156	title + overview
bge-large-en-v1.5	0.028530	0.075750	0.041452	all data
bge-large-en-v1.5	0.017195	0.045650	0.024981	title + overview
bge-base-en-v1.5	0.026719	0.070930	0.038817	all data
bge-base-en-v1.5	0.013380	0.035535	0.019445	title + overview
bge-small-en-v1.5	0.015877	0.042151	0.023066	all data
bge-small-en-v1.5	0.011405	0.030280	0.016500	title + overview
Titan	0.028999	0.076990	0.042129	all data
Titan	0.023955	0.063599	0.034800	title + overview

Table A.6: Precision, Recall, and F1-score comparison between full data and restricted data information (title and overview only) on dataset1 and K=20

Model	MRR	MAP@10	NDCG	NDCG@10	Catalog
multi-qa-MiniLM-L6-cos-v1	0.061000	0.054156	0.033000	0.027639	all data
multi-qa-MiniLM-L6-cos-v1	0.048979	0.043955	0.023900	0.020900	title + overview
multi-qa-mpnet-base-dot-v1	0.113000	0.100500	0.060000	0.051000	all data
multi-qa-mpnet-base-dot-v1	0.079490	0.070600	0.040180	0.034000	title + overview
all-mpnet-base-v2	0.094300	0.084169	0.052860	0.044700	all data
all-mpnet-base-v2	0.058600	0.051850	0.030880	0.025556	title + overview
msmarco-roberta-base-ance-firstp	0.064700	0.057450	0.031655	0.026322	all data
msmarco-roberta-base-ance-firstp	0.046480	0.041336	0.021556	0.017688	title + overview
bge-large-en-v1.5	0.124610	0.111201	0.071632	0.061730	all data
bge-large-en-v1.5	0.067979	0.060513	0.038278	0.032290	title + overview
bge-base-en-v1.5	0.114478	0.101580	0.062520	0.053287	all data
bge-base-en-v1.5	0.058581	0.051625	0.030828	0.026463	title + overview
bge-small-en-v1.5	0.072087	0.064195	0.039098	0.033397	all data
bge-small-en-v1.5	0.049949	0.044267	0.024930	0.021077	title + overview
Titan	0.117470	0.103500	0.062400	0.052220	all data
Titan	0.100480	0.088327	0.052880	0.044296	title + overview

Table A.7: MRR, MAP@10, NDCG, and NDCG@10 comparison between full data and restricted data information (title and overview only) on dataset1 and K=20

A.1.2. dataset 2 and 3

Dataset	K	Precision	Recall	F1-score	MRR	MAP@10	NDCG
dataset1	20	0.025570	0.067886	0.037000	0.060000	0.100500	0.0600
dataset1	10	0.034627	0.045967	0.039499	0.050210	0.100500	0.0502
dataset2	20	0.009392	0.026438	0.013860	0.024817	0.043202	0.0248
dataset2	10	0.013050	0.018360	0.015256	0.020800	0.043202	0.0208
dataset2(Eng)	20	0.007270	0.020467	0.010730	0.018108	0.032320	0.0181
dataset2(Eng)	10	0.009565	0.013460	0.011184	0.014750	0.032320	0.0147
dataset2(Da)	20	0.009198	0.025891	0.013570	0.023349	0.039939	0.0233
dataset2(Da)	10	0.012529	0.017630	0.014649	0.019390	0.039939	0.0194
dataset2(Fi)	20	0.009177	0.025830	0.013540	0.022800	0.040600	0.0228
dataset2(Fi)	10	0.012000	0.017020	0.014144	0.018700	0.040600	0.0187
dataset2(No)	20	0.009177	0.025830	0.013542	0.022806	0.040604	0.0228
dataset2(No)	10	0.012097	0.017026	0.014144	0.018700	0.040604	0.0187
dataset2(Sv)	20	0.005987	0.017820	0.008960	0.013860	0.025003	0.0139
dataset2(Sv)	10	0.007529	0.011200	0.009007	0.010870	0.025003	0.0109
dataset3	20	0.003490	0.007422	0.004751	0.008078	0.014590	0.0081
dataset3	10	0.004357	0.004628	0.004489	0.006386	0.014590	0.0064

Table A.8: multi-qa-mpnet-base-dot-v1 performance with order-unaware and order-aware metrics on different datasets with K=20, 10

Dataset	K	Precision	Recall	F1-score	MRR	MAP@10	NDCG
dataset1	20	0.028999	0.076990	0.042129	0.117470	0.10350	0.062520
dataset1	10	0.038600	0.051278	0.044060	0.113850	0.10350	0.052416
dataset2	20	0.011358	0.031970	0.016760	0.051646	0.04618	0.030290
dataset2	10	0.014499	0.020406	0.016953	0.048665	0.04618	0.024500
dataset3	20	0.005140	0.010900	0.006990	0.025200	0.02252	0.017775
dataset3	10	0.006470	0.006870	0.006700	0.023500	0.02252	0.014270

Table A.9: Titan performance with order-unaware and order-aware metrics on different datasets with K=20, 10

Dataset	K	Precision	Recall	F1-score	MRR	MAP@10	NDCG
dataset1	20	0.028530	0.075750	0.041452	0.071632	0.111201	0.0716
dataset1	10	0.039420	0.052335	0.044972	0.060890	0.111201	0.0601
dataset2	20	0.011210	0.031570	0.016550	0.031499	0.048417	0.0315
dataset2	10	0.015150	0.021330	0.017720	0.026142	0.048417	0.0261
dataset2(Eng)	20	0.011216	0.031570	0.016551	0.033317	0.049680	0.0333
dataset2(Eng)	10	0.015510	0.021829	0.018135	0.028068	0.049680	0.0281
dataset2(Da)	20	0.010140	0.028567	0.014977	0.029447	0.045440	0.0294
dataset2(Da)	10	0.013470	0.018959	0.015750	0.024400	0.045440	0.0244
dataset2(Fi)	20	0.004031	0.012900	0.006140	0.012958	0.018248	0.0129
dataset2(Fi)	10	0.005499	0.008799	0.006768	0.010953	0.018248	0.0109
dataset2(No)	20	0.009868	0.027776	0.014560	0.027700	0.044379	0.0277
dataset2(No)	10	0.013246	0.018643	0.015488	0.022980	0.044379	0.0230
dataset2(Sv)	20	0.010080	0.028384	0.014880	0.029110	0.046260	0.0291
dataset2(Sv)	10	0.013687	0.019260	0.016003	0.024350	0.046260	0.0244
dataset3	20	0.008270	0.017570	0.011250	0.018890	0.032350	0.0189
dataset3	10	0.010756	0.011420	0.011080	0.015398	0.032350	0.0154

Table A.10: bge-lange-en-v1.5 performance with order-unaware and order-aware metrics on different datasets with K=20, 10

Dataset	Precision	Recall	F1-score	MRR	MAP@10	NDCG
dataset1	0.016869	0.044785	0.024507	0.077212	0.068639	0.042360
dataset2(Da)	0.003280	0.009242	0.004846	0.019650	0.017808	0.009529
dataset2(Eng)	0.009300	0.026180	0.013730	0.046300	0.042202	0.025850
dataset2(Fi)	0.002610	0.007345	0.003850	0.016360	0.014940	0.007615
dataset2(No)	0.004748	0.013365	0.007007	0.025130	0.022776	0.012530
dataset2(Fi)	0.004096	0.011529	0.006040	0.022270	0.020353	0.011288
dataset3	0.003816	0.008106	0.005189	0.017920	0.016230	0.008570

Table A.11: multilingual-e5-base performance with order-unaware and order-aware metrics on different datasets with K=20, 10

Dataset	K	Precision	Recall	F1-score	MRR	MAP@10	NDCG
dataset1	20	0.026719	0.070930	0.038817	0.114478	0.101580	0.062520
dataset1	10	0.036569	0.048540	0.041710	0.110653	0.101583	0.052416
dataset2	20	0.011670	0.032859	0.017227	0.048860	0.043900	0.030290
dataset2	10	0.015060	0.021197	0.017610	0.046167	0.043915	0.024500
dataset2(Eng)	20	0.015839	0.044583	0.023373	0.065530	0.058945	0.042509
dataset2(Eng)	10	0.020686	0.029114	0.024187	0.062374	0.058945	0.034660
dataset2(Da)	20	0.013588	0.038247	0.020050	0.055167	0.049600	0.035550
dataset2(Da)	10	0.017057	0.024006	0.019940	0.052080	0.049600	0.028000
dataset2(Fi)	20	0.013900	0.039135	0.020517	0.061815	0.056234	0.037574
dataset2(Fi)	10	0.018154	0.025550	0.021227	0.058900	0.056234	0.030860
dataset2(No)	20	0.014570	0.041020	0.021500	0.060950	0.055190	0.038315
dataset2(No)	10	0.018586	0.026159	0.021730	0.057660	0.055190	0.030860
dataset2(Sv)	20	0.013670	0.038478	0.020173	0.055599	0.050201	0.035090
dataset2(Sv)	10	0.017437	0.024541	0.020388	0.052500	0.050201	0.028020
dataset3	20	0.007886	0.016750	0.010720	0.033947	0.030868	0.017775
dataset3	10	0.010026	0.010649	0.010328	0.032022	0.030868	0.014270

Table A.12: bge-base-en-v1.5 performance with order-unaware and order-aware metrics on different datasets with K=20, 10

A.2. Fine-tuning

Below are presented results considering models fine-tuned with different approaches.

Fine-tuning	TP	FP	FN
None	6304	229636	82565
Adapter (default)	7032	228908	81837
Adapter (2 Layer)	7028	228912	81841
MNRL	22423	213517	66446
CSL	8465	227475	80404
LoRA(8-16)	13459	222481	75410
LoRA(8-8)	13045	222895	75824
LoRA(4-8)	12202	223738	76667

Table A.13: Calculous of TP, FP, and FN by bge-base-en-v1.5 fine-tuned with Adapter, traditional and LoRA fine-tuning techniques with K=20 on industrial dataset1

Fine-tuning	k	Precision	Recall	F1-score	MRR	MAP@10	NDCG
None	20	0.0256	0.0679	0.0370	0.113	0.1005	0.0600
Adapter(default)	20	0.0263	0.0691	0.0378	0.117	0.1047	0.0630
None	10	0.0346	0.0460	0.0395	0.109	0.1005	0.0502
Adapter(default)	10	0.0356	0.0472	0.0406	0.113	0.1047	0.0533

Table A.14: multi-qa-mpnet-base-dot-v1 finetuned with Adapter performances

Fine-tuning	Precision	Recall	F1-score	MRR	MAP@10	NDCG
None	0.0267	0.0709	0.0388	0.1145	0.1016	0.0625
Adapter (default)	0.0298	0.0791	0.0433	0.1302	0.1153	0.0732
Adapter (2Layer)	0.0298	0.0791	0.0433	0.1259	0.1109	0.0700
MNRL	0.0950	0.2523	0.1381	0.3200	0.2804	0.2175
CSL	0.0359	0.0952	0.0521	0.1626	0.1425	0.0764
LoRA (8-16)	0.0570	0.1514	0.0829	0.1953	0.1690	0.1132
LoRA (8-8)	0.0553	0.1468	0.0803	0.1890	0.1644	0.1094
LoRA (4-8)	0.0517	0.1373	0.0751	0.1782	0.1543	0.1024

Table A.15: bge-base-en-v1.5 finetuned with Adapters, hole update and MNRL or CSL loss, and LoRA performances

Fine-tuning	dataset	Precision	Recall	F1-score	MRR	MAP@10	NDCG
None	dataset1	0.0168	0.0439	0.0243	0.0768	0.0668	0.04200
LoRA(8-8)	dataset1	0.0227	0.0594	0.0328	0.0977	0.0880	0.05211
MNRL	dataset1	0.0969	0.2535	0.1402	0.3118	0.2740	0.21060
None	dataset2 (Da)	0.0033	0.0092	0.0048	0.0196	0.0178	0.00950
LoRA(8-8)	dataset2 (Da)	0.0257	0.0724	0.0380	0.1045	0.0927	0.06200
MNRL	dataset2 (Da)	0.0054	0.0153	0.0080	0.0221	0.0198	0.01440
None	dataset3	0.0038	0.0081	0.0052	0.0179	0.0162	0.00860
LoRA(8-8)	dataset3	0.0094	0.0199	0.0127	0.0307	0.0271	0.01850
MNRL	dataset3	0.0040	0.0084	0.0054	0.0178	0.0163	0.00900

Table A.16: multilingual-e5-base finetuned with LoRA and hole update performances

A.3. Hybrid search

Below are presented results considering models fine-tuned integrated into a hybrid search system.

Fine-tuning	Precision	Recall	F1-score	MRR	MAP@10	NDCG
MNRL	0.0471	0.1249	0.0684	0.1565	0.1411	0.0956
LoRA	0.0439	0.1166	0.0638	0.1508	0.1342	0.0890

Table A.17: bge-base-en-v1.5 performance in hybrid search environment with dataset1 and QA test set v2 alpha=0.7 and K=20

Alpha	Precision	Recall	F1-score	MRR	MAP@10	NDCG
1.0	0.0204	0.0534	0.0290	0.0916	0.0818	0.0474
0.7	0.0210	0.0550	0.0304	0.0907	0.0825	0.0494
0.5	0.0195	0.0511	0.0283	0.0921	0.0846	0.0577

Table A.18: multilingual-e5-base fine-tuned with LoRA performance in hybrid search environment with dataset1 and QA test set v2 and K=20

Alpha	Precision	Recall	F1-score	MRR	MAP@10	NDCG
1.0	0.0313	0.0819	0.0453	0.1174	0.1025	0.0623
0.7	0.0323	0.0845	0.0467	0.1178	0.1054	0.0658
0.5	0.0255	0.0666	0.0368	0.1097	0.0998	0.0664

Table A.19: multilingual-e5-base fine-tuned with hole update and MNRL loss performance in hybrid search environment with dataset1 and QA test set v2 and K=20

B | Appendix B

To evaluate the reachness of the outputs generated by different embedding models, we choose a diverse set of movies and series that cover various genres, themes, and styles.

Inception
 Year: 2010
 Country: USA
 Cast: Leonardo DiCaprio, Joseph Gordon-Levitt, Elliot Page, Tom Hardy, Ken Watanabe, Dileep Rao, Cillian Murphy, Tom Berenger, Marion Cotillard, Michael Caine
 Genre: Science Fiction, Thriller, Action
 Directors: Christopher Nolan
 Awards: Academy Award, Golden Globe, Screen Actors Guild Awards, British Academy of Film & Television Arts, Amnsteinprize
 Moods: Thrilling, Surreal, Dramatic, Psychological
 Characters: Con artist, Architect, Chemist, Rogue, Billionaire, Wife, Whales, Professors, Enemies, Chemists, Hiss, Businessmen, Thieves, Graduate Students, Partners
 Settings: Dreamworlds, Mountains, Hotels, Chess, Egg Cities, Fights, Dreams, The Subconscious Mind
 Themes: Manipulation, Risk, Guilt, Determination
 Time Periods: Early 21st Century
 Subjects: Dreams, Corporate espionage, Guilt, Dangerous assignment, Dreaming, Human Connection, Con Games
 Summary: Dom Cobb (Leonardo DiCaprio) is a thief with the rare ability to enter people's dreams and steal their secrets from their subconscious. His skill has made him a hot commodity in the world of corporate espionage but has also cost him everything he loves. Cobb gets a chance at redemption when he is offered a seemingly impossible task: Plant an idea in someone's mind. If he succeeds, it will be the perfect crime, but if dangerous enemy anticipates Cobb's every move.
 Duration: 148
 Scenario: Easy Situation, Unfinished Business, Banding Together, Chain of Events, Manipulation, Quest, Temporal Anomalies
 IMDb: 8.8
 TMDb: 8.4
 Critic Score: 8.8
 Available on: (US) Apple, Amazon, Google, YouTube, Vudu, Microsoft
 Based on: Original Screenplay
 IMDb: 8.8

Open AI
 Rendition, Interstellar, Ronin, Snowden, The Wolf of Wall Street, Catch Me if You Can, Transcendence, The Bourne Identity, The Matrix, Paycheck

Recommended by Titan
 Criminal, Unknown, Transcendence, Trance, The Prestige, Now You See Me, Dejavu, Inferno, The Infiltrator, Focus

Recommended by BGE Base
 The Wolf of Wall Street, Catch Me if You Can, The Aviator, Gangs of New York, The Score, Batman Begins, Trance, The Great Gatsby, The Prestige, Entrapment

Recommended by E5 Base
 Ex Machina, Criminal, Interstellar, Total Recall, Men in Black 3, Michael Clayton, Arrival, Despicable Me 2, Dejavu, Maps to the Stars

Figure B.1: Carousels of items similar to Inception

Title (original): The Godfather
Year: 1972
Country: USA
Cast: Marlon Brando, Al Pacino, James Caan, Richard Castellano, Robert Duvall, Sterling Hayden, John Marley, Richard Conte, Diane Keaton, Al Lettieri
Genres: Crime drama
Directors: Francis Ford Coppola
Awards: Academy Award, Golden Globe, British Academy of Film & Television Arts
Moods: Brooding, Brutal, Suspenseful
Characters: Mobster, Lawyer, Cop, Father, Son, Wife, Crooked Cops, Sons, Lawyers, Wives, Crime Families, Fathers, Italian-American Mafia
Settings: Criminal Events, The Coast, Weddings, Family Homes, Outdoor Spaces, Sicily, Big Cities, New York City
Themes: Power, Revenge, Family Bonds, Betrayal
Time Periods: 1940s, 1950s
Subjects: Organized crime, Mafia, Family, Betrayal, Murder, Feuds, Small Family Business, Family Relationships
Summary: Widely regarded as one of the greatest films of all time, this mob drama, based on Mario Puzo's novel of the same name, focuses on the powerful Italian-American crime family of Don Vito Corleone (Marlon Brando). When the don's youngest son, Michael (Al Pacino), reluctantly joins the Mafia, he becomes involved in the inevitable cycle of violence and betrayal. Although Michael tries to maintain a normal relationship with his wife, Kay (Diane Keaton), he is drawn deeper into the family business.
Duration: 177
Scenarios: Family Saga, Personal Relationships, Looming Threat, Rise & Fall, Power Struggle
TMDBid: 238
TMdb: /movie/2069000
Critics Score: 92
Available on: TuboTV, DIRECTV, Apple, Amazon, Google, YouTube, Vudu
Based on: Novels, Film Franchises
IMdbid: tt0068646

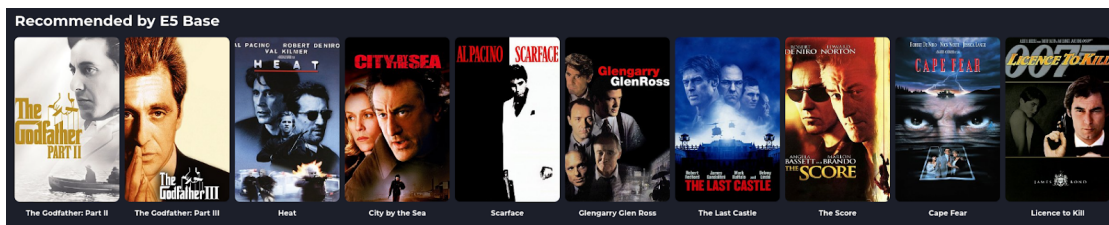
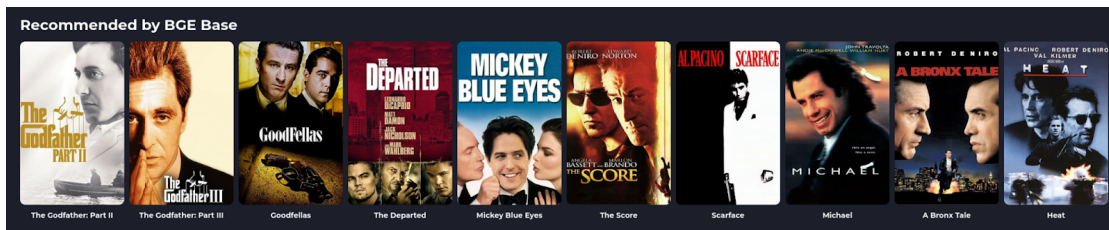
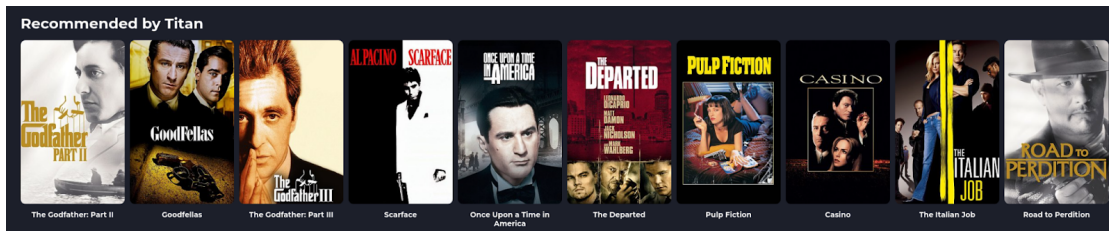
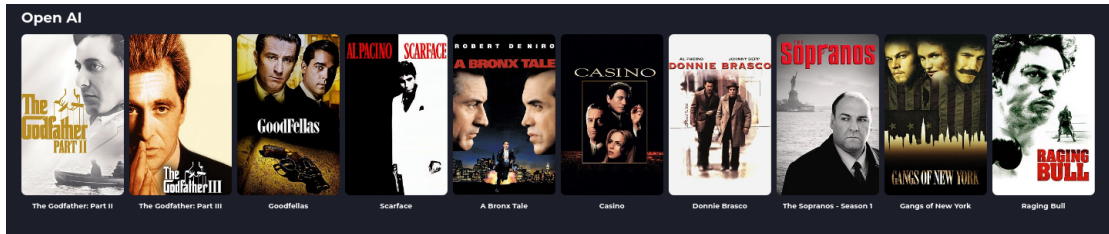


Figure B.2: Carousels of items similar to The Godfather

Pulp Fiction

Title (original): Pulp Fiction
Year: 1994
Country: USA
Cast: John Travolta, Samuel L. Jackson, Uma Thurman, Harvey Keitel, Tim Roth, Amanda Plummer, Maria de Medeiros, Ving Rhames, Eric Stoltz, Rosanna Arquette
Genres: Crime drama
Directors: Quentin Tarantino
Awards: Cannes Film Festival, Academy Award, Golden Globe, Screen Actors Guild Awards, British Academy of Film & Television Arts, Amandaprisen
Moods: Offbeat, Philosophical, Witty, Complex, Suspenseful
Characters: Gangster, Hitman, Actress, Boxer, Military officer, Drug dealer, Pawnbrokers, Friends, Giffhendi, Robbers, Drug Dealers, Military Officers, Security Guards, Gangsters, Hit Men, Boxers, Gangster's Wives
Settings: Accidental Killings, Diners, Vehicles, Los Angeles, Pawn Shops, Homes, Drug Overdoses, Murders
Themes: Consequences, Atonement
Time Periods: Mid 90s
Subjects: Crime, Faith, Drugs, Murder, Rape, Christianity
Summary: Vincent Vega (John Travolta) and Jules Winnfield (Samuel L. Jackson) are hitmen with a penchant for philosophical discussions. In this ultra-hip, multi-strand crime movie, their storyline is interwoven with those of their boss, gangster Marsellus Wallace (Ving Rhames), his actress wife, Mia (Uma Thurman), struggling boxer Butch Coolidge (Bruce Willis), master foxer Winston Wolfe (Harvey Keitel), and a nervous pair of armed robbers, Pumpkin (Tim Roth) and Honey Bunny (Amanda Plummer).
Duration: 153
Scenarios: Chain of Events, Intersecting Lives, Heist Gone Wrong, Cover-Up, Making An Escape
TMDBid: 690
Tmid: MV00039740000
Critics Score: 8.9
Available on: HBO, DIRECTV, TNT, TBS, Apple, Amazon, Google, YouTube
Based on: Original Screenplay
Imdbid: tt0110912

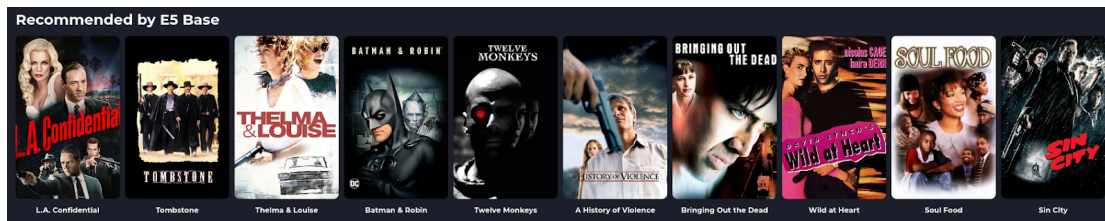
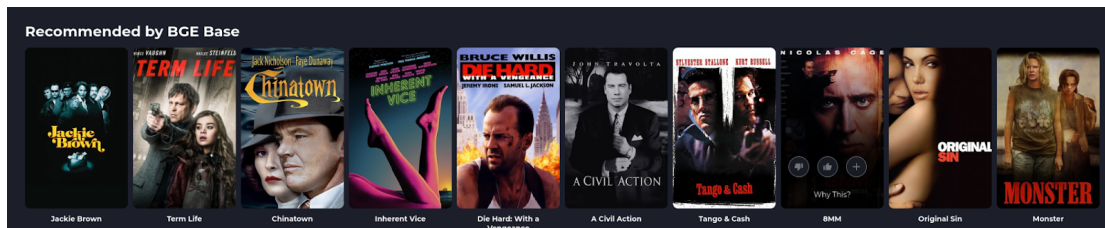
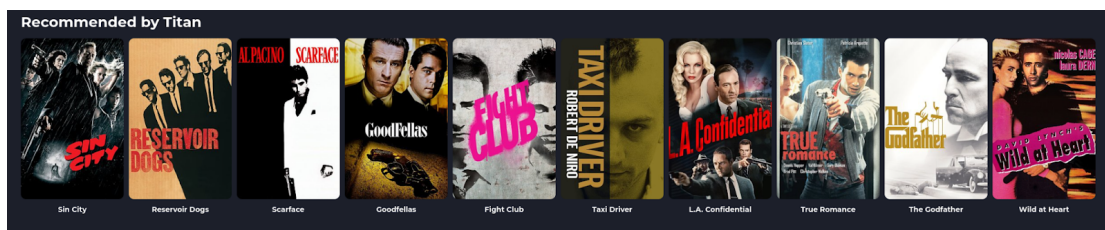
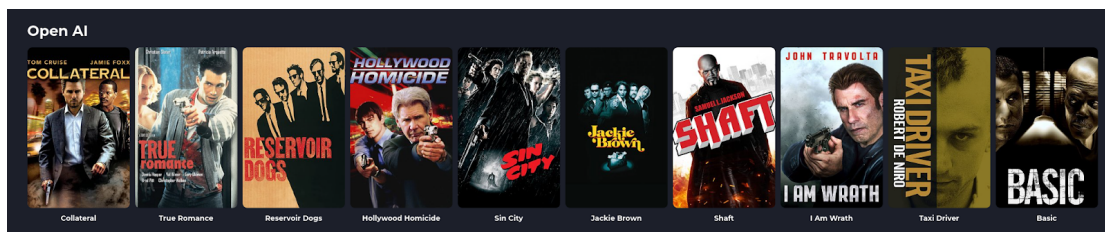


Figure B.3: Carousels of items similar to Pulp Fiction

Title (original): Toy Story

Year: 1995

Country: USA

Cast: Tom Hanks, Tim Allen, Don Rickles, Jim Varney, Wallace Shawn, John Ratzenberger, Annie Potts, John Morris, Erik von Detton, Laurie Metcalf

Genres: Animated, Children, Comedy, Adventure, Fantasy

Directors: John Lasseter

Awards: Academy Award, Golden Globe, British Academy of Film & Television Arts

Moods: Feel-good, Intense, Playful

Characters: Bully, Mother, Woody, Buzz Lightyear, Mr. Potato Head, Young boy, Living Toys, Little Boys, Bullies, Mothers, Neighbors

Settings: Household Moving, Backyards, Children's Rooms, Adventures, Family Homes, Restaurants, Neighborhoods, Cars

Themes: Zealousy, Reconciliation, Friendship, Rivalry

Time Periods: Mid 90s

Subjects: Toys, Friendship, Childhood, Interpersonal Rivalries, Friendship & Bonds, Bad Behavior

Summary: Woody (Tom Hanks), a good-hearted cowboy doll who belongs to a young boy named Andy (John Morris), sees his position as Andy's favorite toy jeopardized when his parents buy him a Buzz Lightyear (Tim Allen) action figure. Even worse, the arrogant Buzz thinks he's a real spaceman on a mission to return to his home planet. When Andy's family moves to a new house, Woody and Buzz must escape the clutches of maladjusted neighbor Sid Phillips (Erik von Detton) and reunite with their boy.

Duration: 80

Scenarios: Making Amends, Banding Together, Personality Clash, Rescue Effort, Misadventures, Saving The Day

TMDBid: 862

TmsId: MV00044190000

Critics Score: 83

Available on: Disney, Apple, Amazon, Google, YouTube, Vudu

Based on: Original Screenplay, Film Franchises

ImdbId: tt0114709

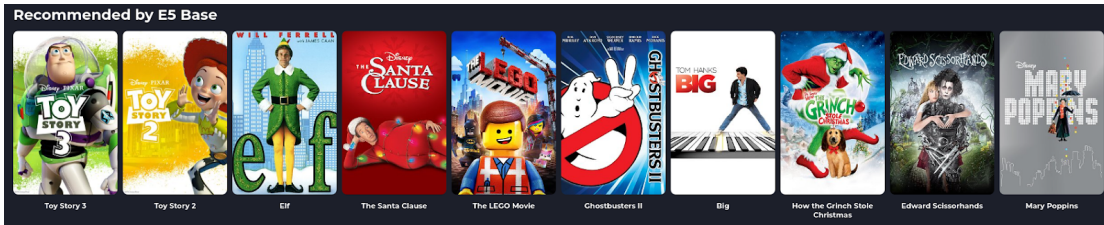
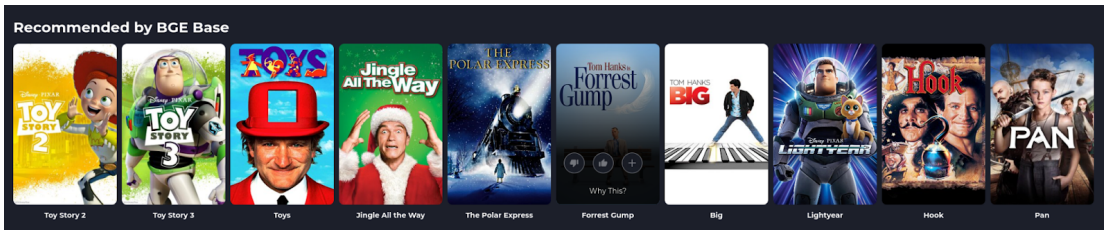
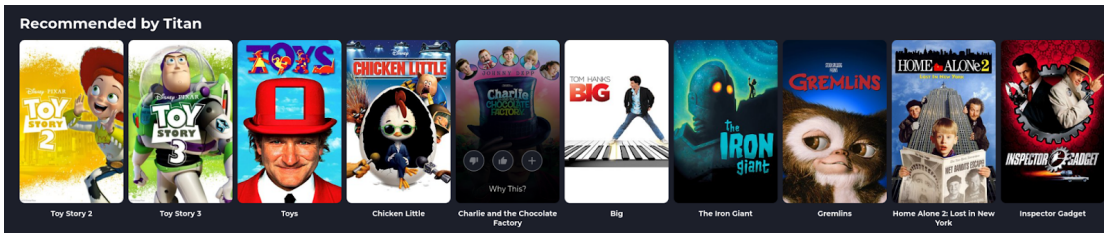
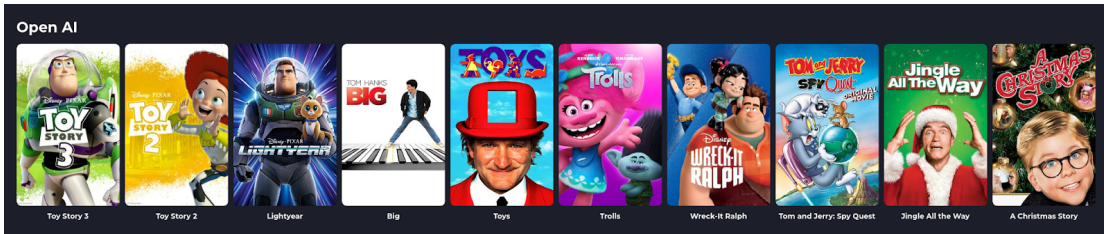


Figure B.4: Carousels of items similar to Toy Story



Title (original): Stranger Things

Series Name: Stranger Things

Series Name (original): Stranger Things

Year: 2016

Country: USA

Season Number: 1

Episode: 1

Cast: Winona Ryder, David Harbour, Matthew Modine, Finn Wolfhard, Natalia Dyer, Cara Buono, Charlie Heaton, Millie Bobby Brown, Caleb McLaughlin, Gaten Matarazzo

Genres: Drama, Science fiction, Mystery

Awards: Golden Globe, Screen Actors Guild Awards, Emmy (Primetime), British Academy of Film & Television Arts

Moods: Gripping, Suspenseful, Eerie, Engaging, Creepy, Playful

Settings: Small town, Alternate reality, Parallel universe, Middle school, Family home, United States, Suburbs, Suburb, Police station, School, Mythical land

Themes: Adventure, Discovery, Mystery, Pursuit, Rescue

Time Periods: 1980s

Summary: On his way home, Will sees something terrifying; a sinister secret lurks in the depths of a nearby government lab.

Duration: 48

TMDBid: 66732


TmsId: EP024519750001

Critics Score: 8.8

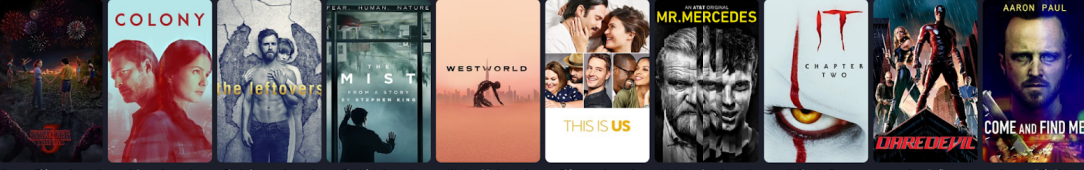
Episode Name: Chapter One: The Vanishing of Will Byers

ImdbId: tt4574334


Open AI



Recommended by Titan



Recommended by BGE Base



Recommended by ES Base

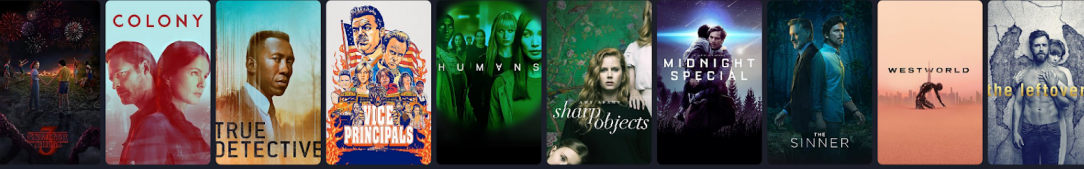


Figure B.5: Carousels of items similar to Stranger Things

List of Figures

2.1	Semantic analogies derived from subtraction between word embeddings. . .	6
2.2	Word2Vec models	7
2.3	Transformer model architecture.	9
2.4	(left) Single head attention (right) multiple-head attention	11
2.5	(left)BERT pre-training (right) BERT fine-tuning	12
2.6	BERT input representation	13
2.7	sBERT pre-training	15
2.8	The retrieving process where query and relevant items (red points) are close to each other in the vector embedding space.	17
2.9	Fine-tuning pipeline for Information Retrieval.	18
2.10	Visualization of triplet loss goal, reduce the distance of an anchor to the positive example and increase the distance to the negative one.	19
2.11	Visualization of low-rank decomposition.	21
2.12	Visual schema of fusion algorithm.	23
3.1	Item distribution based on TenantId	28
3.2	Comparison of items with overview and total items per dataset	30
3.3	Workflow diagram for synthetic Question-Answer dataset generation	31
3.4	Query distribution per type	33
4.1	Interface snapshot within demo environment, showing carousels of items similar to the selected one, based on similarity matrices developed with different embedding models	44
5.1	Schema of our information retrieval system: from data integration, to query processing and result ranking	51
5.2	Illustration of hybrid search mechanism, from Weaviate blog post	53
5.3	Hierarchical Navigable Small World (HNSW) structure	56
6.1	query: 'montre-moi des films avec des êtres d'autres planètes' (translation: show me movies with beings from other planets)	66

6.2	query: 'film con una donna forte come protagonista' (translation: Movies with a strong female as main character)	66
6.3	query: 'Zeigen Sie mir Filme über Eltern-Kind-Beziehungen' (translation: Show me movies about parent-child relationships)	67
6.4	Metrics evaluation on test set v2 with catalog 1 of fine-tuned model 'multilingual-e5-base' integrated in Weaviate tested with three alpha configurations (1, 0.7 and 0.5)	72
6.5	Comparison of different fine-tuning techniques tested	74
6.6	Performance comparison of the multilingual-e5-base model across different Fine-Tuning approaches and datasets	75
6.7	Precision Recall and F1-score with industrial dataset 1 and K= 20	77
6.8	Precision Recall and F1-score with industrial dataset 1 and K= 10	78
6.9	Recall@K with K from 1 to 10 in the QA test set 1	79
6.10	Comparative analysis of embedding size impact on model performance with order-aware metrics	80
6.11	Bar plots of MRR on dataset 1 with all embeddings models and K=20	81
6.12	Bar plots of MAP@10 on dataset 1 with all embeddings models and K=20	81
6.13	Bar plots of NDCG and NDCG@10 on dataset 1 with all embeddings models and K=20	82
6.14	Comparison of order-aware metrics in full metadata availability and restricted information scenario across different embedding models with K=20	83
6.15	Precision, Recall and F1-score on different languages	85
6.16	MRR, MAP@10, NDCG, and NDCG@10 on different languages	87
B.1	Carousels of items similar to Inception	107
B.2	Carousels of items similar to The Godfather	108
B.3	Carousels of items similar to Pulp Fiction	109
B.4	Carousels of items similar to Toy Story	110
B.5	Carousels of items similar to Stranger Things	111

List of Tables

3.1	Comparison of attributes in industrial datasets	30
3.2	Distribution of query-answer across Train, Validation, and Test sets	35
5.1	Division of attributes for industrial dataset 1	58
6.1	Retrieval results of multilingual-e5-base and its versions fine-tuned for the query ' <i>viaggi nello spazio</i> ' (Space travel)	68
6.2	Retrieval results of bge-base-en-v1.5 and its versions fine-tuned for the query ' <i>viaggi nello spazio</i> ' (Space travel)	68
6.3	Retrieval results of multilingual-e5-base and its versions fine-tuned inside weaviate, performing an hybrid search with $\alpha = 0.7$, for the query " <i>mostre-me documentários sobre a sustentabilidade do alimento</i> " (show me documentaries about food sustainability)	69
6.4	Retrieval results of multilingual-e5-base fine-tuned with LoRA technique on dataset 1, with the query <i>I would like to watch a sci-fi movie with time travels, possibly with paradoxes, it must include a love story.</i>	70
6.5	Retrieval results of multilingual-e5-base fine-tuned, on dataset 1 and dataset 2, with the query " <i>mostre-me documentários sobre a sustentabilidade do alimento</i> " (show me documentaries about food sustainability) and $\alpha = 0.7$	71
A.1	TP, FP, and FN of all models	97
A.2	Precision, Recall, and F1-score all models	98
A.3	Recall@K with K from 1 to 5	99
A.4	Recall@K with K from 6 to 10	99
A.5	MRR, MAP@10, NDCG and NDCG@10 all models	100
A.6	Precision, Recall, and F1-score comparison between full data and restricted data information	100
A.7	MRR, MAP@10, NDCG, and NDCG@10 comparison between full data and restricted data information	101
A.8	multi-qa-mpnet-base-dot-v1 performance on different datasets	102

A.9 Titan performance on different datasets	102
A.10 bge-lange-en-v1.5 performance on different datasets	103
A.11 multilingual-e5-base performance on different datasets	103
A.12 bge-base-en-v1.5 performance on different datasets	104
A.13 TP, FP, and FN of bge-base-en-v1.5 fine-tuned	104
A.14 multi-qa-mpnet-base-dot-v1 finetuned with Adapter performances . . .	105
A.15 bge-base-en-v1.5 finetuned with Adapters, hole update and MNRL or CSL loss, and LoRA performances	105
A.16 multilingual-e5-base finetuned with LoRA and hole update performances	105
A.17 bge-base-en-v1.5 performance in hybrid search environment with dataset1 and QA test set v2 alpha=0.7 and K=20	106
A.18 multilingual-e5-base fine-tuned with LoRA performance in hybrid search environment with dataset1 and QA test set v2 and K=20	106
A.19 multilingual-e5-base fine-tuned with hole update and MNRL loss per- formance in hybrid search environment with dataset1 and QA test set v2 and K=20	106

Acknowledgements

Desidero innanzitutto ringraziare il mio relatore, il Prof. Paolo Cremonesi, per il supporto e per avermi dato l'opportunità di intraprendere questo percorso. Inoltre vorrei ringraziare i miei correlatori, Federico e Riccardo, per la loro disponibilità e la loro fondamentale guida per la realizzazione di questa tesi. Vorrei inoltre ringraziare tutto il team di Contentwise, con cui ho avuto il piacere di lavorare durante questo periodo. Siete stati una fonte di ispirazione, con pazienza e professionalità mi avete insegnato molto ed avete contribuito a creare un bel clima in cui fa piacere andare a lavorare tutti i giorni.

Grazie alla mia famiglia, che mi ha consentito di perseguire i miei obiettivi supportandomi tutti questi anni. Grazie alla mia mamma Monica, che con il suo esempio di forza e tenacia mi ha insegnato a non mollare mai, anche di fronte alle difficoltà. Grazie al mio papà Carlo Alberto, per la sua calma e sincerità che mi hanno sempre aiutato a mantenere la lucidità e a prendere le decisioni migliori. Grazie a mio fratello Davide, per essere da sempre il mio esempio da seguire e per essere sempre stato al mio fianco, pronto a sostenermi e a difendermi.

Un ringraziamento speciale va anche a tutti i miei amici, quelli di una vita come quelli conosciuti più di recente. Con la vostra costante presenza e il vostro sostegno avete reso questo percorso più leggero e ricco.

Infine grazie a Giacomo, per il tuo costante supporto da sempre. Sei stato fondamentale in tutto il mio percorso incoraggiandomi e stando sempre al mio fianco, anche nei momenti più bui. La tua pazienza, il tuo altruismo e la tua fiducia in me mi hanno permesso di superare ostacoli che sembravano insormontabili.

