Universidad Autónoma de Madrid

Escuela Politécnica Superior

23 | 24

Bachelor thesis

Exploration of Neural Embeddings and Feature Generation for Enhanced Real Estate Search

Fernando Ónega Rodrigo

Escuela Politécnica Superior
Universidad Autónoma de Madrid
C\Francisco Tomás y Valiente nº 11

Campus Internacional excelencia UAM CSIC+

www.uam.es

Universidad Autónoma de Madrid

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Bachelor as Ingeniería Informática Bilingüe

BACHELOR THESIS

Exploration of Neural Embeddings and Feature Generation for Enhanced Real Estate Search

Author: Fernando Ónega Rodrigo
Advisor: Alejandro Bellogín Kouki

mayo 2024

Fernando Ónega Rodrigo
Exploration of Neural Embeddings and Feature Generation for Enhanced Real Estate Search

Fernando Ónega Rodrigo
C\ Francisco Tomás y Valiente Nº 11

Agradecimientos

En primer lugar, quiero expresar mi agradecimiento a mis compañeros, por vuestro apoyo y amistad durante estos años. He aprendido mucho de cada uno de vosotros, tanto a nivel académico como personal.

A todos los profesores que me han guiado y formado durante mi carrera, gracias por vuestra paciencia y pasión por la enseñanza. Habéis sido pilares esenciales para mi formación inspirándome a seguir aprendiendo y creciendo cada día. Quiero expresar mi especial agradecimiento a Alejandro Bellogín por su dedicación como tutor y durante el desarrollo de este trabajo.

A mis padres, por su amor incondicional y apoyo constante.

A Judit, por estar siempre ahí para hacerlo todo más fácil.

A todos los que me han acompañado en este camino, les agradezco por hacer de él una experiencia inolvidable.

Resumen

La disponibilidad limitada de datos etiquetados puede dificultar la aplicación de modelos de redes neuronales para la Recuperación de Información (Neu-IR) en ámbitos como el sector inmobiliario. Este trabajo investiga el uso de Modelos de Lenguaje Grandes (LLMs) abiertos (*open-weights*) para la extracción automática de características de propiedades inmobiliarias de anuncios en portales web como Idealista. Esto nos permite desarrollar un proceso de creación de conjuntos de datos para *fine-tuning* de modelos usando representación vectorial de texto. Entrenamos un modelo basado en BERT, real-bert, en este conjunto de datos y evaluamos su efectividad en la búsqueda de propiedades inmobiliarias utilizando técnicas de recuperación por similitud vectorial.

Los resultados demuestran que real-bert supera significativamente a los modelos de referencia. Esto demuestra el potencial de utilizar LLMs locales para la extracción no supervisada de características, y permite la aplicación de Neu-IR en dominios con datos y recursos computacionales limitados.

Este trabajo investiga además distintas configuraciones de entrenamiento y demuestra la aplicación práctica de nuestro sistema de búsqueda a través del desarrollo y despliegue de una aplicación web. Esta investigación contribuye al avance de técnicas avanzadas de Recuperación de la Información en dominios con escasez de datos, abriendo nuevas vías de exploración con LLMs cada vez más sofisticados y el uso de datos multimodales.

Palabras clave

Recuperación de Información por Redes Neuronales (Neu-IR), Búsqueda Inmobiliaria, Fine-Tuning, Aprendizaje No Supervisado, Procesamiento de Lenguaje Natural (NLP), BERT, Aplicaciones Web, Modelo de Lenguaje Grande (LLM)

Abstract

The limited availability of labeled datasets often hinders the application of advanced Neural Information Retrieval (Neu-IR) techniques in specialized domains like real estate. This thesis explores the use of open-weights Large Language Models (LLMs) for automated feature extraction from property listings, enabling the creation of a comprehensive dataset for fine-tuning text embedding models.

A BERT-based model, real-bert, is trained on this dataset and evaluated for its effectiveness in real estate search using vector similarity retrieval techniques. The results demonstrate that real-bert significantly outperforms baseline models, highlighting the potential of local LLMs for unsupervised feature augmentation and enabling the application of Neu-IR techniques in domains with limited data and computational resources.

The thesis further investigates optimal training configurations and showcases the practical application of the approach through a demo web application. This research contributes to the advancement of Neu-IR in data-scarce domains, opening avenues for further exploration with increasingly sophisticated LLMs and multimodal data integration.

Keywords

Neural Information Retrieval (Neu-IR), Real Estate Search, Fine-Tuning, Unsupervised Learning, Natural Language Processing (NLP), BERT, Web Applications, Large Language Model (LLM).

Table of Contents

Lists

List of codes

List of figures

List of tables

1 Introduction

1.1. Motivation

In the last two decades, the real estate industry has found an indispensable ally in digital platforms, which have become a central hub for property listings [1]. With Europe's real estate market projected to grow annually by 3.27 % [1], and the rise in online leasing platforms and services [2], it can be inferred that the digital segment of the industry could see an even bigger growth.

Despite the increasing reliance on real estate portals, current property search methods pose challenges, even though it attracted researchers from the very beginning of the field [2], including industry [3]. Traditional search methods, which often rely on predefined filters or keywords, can be inefficient, inflexible, and unintuitive for both customers and agents [4]. These methods can also result in 'filter bubbles' [5], limiting the diversity of search results by confining users to properties within their usual search parameters or neighborhoods. Furthermore, information gaps in property ads can complicate the process [5], potentially leading to overlooked properties or missed opportunities.

At the same time, Neu-IR [6] has advanced significantly with the incorporation of deep learning techniques, particularly in Natural Language Processing (NLP) tasks [7], with the introduction of Transformer models [8]. These advancements have been applied to web search engines [9], e-commerce [10], and social networks [11]. Despite their success in these domains, the application of these techniques remains largely unexplored in real estate, presenting unique opportunities.

The common issue of limited datasets for model training in new domains is addressed in our work by leveraging the capabilities of LLMs for feature extraction, utilizing a small, open-weights model (llama2 13B) for its computational efficiency and accessibility in experimental contexts. We investigate effective prompting strategies to unlock their potential while acknowledging potential limitations in reasoning and multilingual understanding. Through this exploration, we aim to demonstrate the feasibility of applying small LLMs for high-quality, effective dataset creation in this domain.

[1] `https://www.statista.com/outlook/fmo/real-estate/europe` (last accessed on 2024-04-10)
[2] `https://www.grandviewresearch.com/industry-analysis/real-estate-market` (last accessed on 2024-04-10)

1.2. Goals

The primary objective of this study is to introduce a novel approach in real estate search by applying Neu-IR techniques. We propose an Embedding-based Retrieval (EBR) method, leveraging a BERT-based model that has been fine-tuned for our specific task, which we refer to as ***real-bert***.

Our aim is to utilize a small open-weights Large Language Model (LLM) (Llama2 13B) to obtain more than 20 structured features for each property ad from a textual corpus of hundreds of real estate properties, primarily sourced from Idealista [3], known for its large, high-quality, and diverse listings. By prompting Llama2 13B with this corpus, we enable the automated creation of a fine-tuning dataset.

This dataset, alongside a dedicated evaluation corpus, allows us to rigorously assess the performance of our fine-tuned model compared to other baseline models, utilizing traditional and novel metrics. To further validate the practicality and effectiveness of our approach in a real-world scenario, we develop a web application that allows users to search for properties using natural language queries and directly compare the results obtained with different models. This user-centered evaluation, conducted in a production environment, complements our quantitative analysis and offers valuable insights into the user experience. It showcases our system's potential for a more intuitive and efficient real estate search process, emphasizing practicality alongside theoretical advancements.

1.3. Work Structure

This thesis is structured into five chapters, each addressing a crucial aspect of the research and development process:

Chapter 1. Introduction: Outlines the limitations of current real estate search methods and motivates the exploration of Neu-IR and LLMs as potential solutions.

Chapter 2. State of the Art: Review of relevant background information, covering NLP techniques, LLMs, and Neu-IR, with a specific focus in real estate search.

Chapter 3. Design and Implementation: Details the development process of the proposed search system, including data preparation, feature extraction with LLMs, dataset creation, model fine-tuning, and demo web application implementation.

Chapter 4. Experiments and Results: Describes the evaluation methodology, compares the performance of the fine-tuned model with various baseline models, and analyzes the impact of different training configurations.

Chapter 5. Conclusions and Future Work: Overview of our key findings, implications of the results, and potential directions for future research and development.

[3] `https://www.idealista.com/` (last accessed on 2024-04-10)

2 State of the Art

In this chapter, we take a closer look at the current state of the art in several key areas related to our study. We will explore the existing technologies and challenges in property search, delve into the foundational aspects of Natural Language Processing, and discuss the most relevant and successful applications of Neural Information Retrieval. We will also provide a concise overview of the latest research around Large Language Models (LLMs) with special emphasis on their evolution, characteristics, training methodologies, and practical applications. Each of these areas plays a crucial role in our work, and understanding the current landscape will provide valuable context for later chapters.

2.1. Natural Language Processing

NLP is a subfield of Artificial Intelligence (AI) which aims to bridge the gap between computers and humans through natural language understanding (comprehending human language input) and generation (creating human-like language output). In today's technological environment, where data is abundant and mostly unstructured, NLP plays a crucial role in deriving understanding and insights from such data.

Building upon the foundation of NLP, early systems were predominantly rule-based [12]. These systems relied on a pipeline architecture where syntax, semantics, and pragmatics were treated as separate modules. For instance, a syntactic parser would take raw text as input and output a parse tree, which would then be used by a semantic analyzer to determine the sentence's meaning. An example of a rule-based system is MYCIN [13], an early expert system developed for diagnosing bacterial infections.

The advent of neural networks marked a significant transition in NLP. Unlike rule-based systems, which rely on predefined linguistic rules and structures, neural network-based NLP systems learn directly from data. This learning is achieved through a process known as *training*. This involves feeding the model a large dataset of sentences and optimizing the model's parameters to minimize the difference between its predictions and the actual data. This way, the model learns to recognize patterns and structures in the language, guided not by explicit rules, but rather by the statistical properties of the data.

Once the patterns in the language are learned, neural NLP systems can use them to understand and generate language. This is called *inference*, where the pre-trained model is used to make predictions on some data.

2.1.1. Word Embeddings

Techniques such as word embeddings have been instrumental in this transition. A word embedding is a representation of a word in a high-dimensional space. In this space, semantically similar words appear closer together. This is clearly seen in Figure 2.1, where clusters of related words are formed.

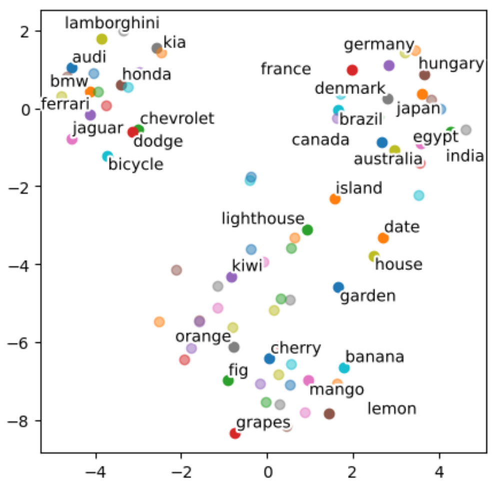


Figure 2.1: 2D visualization of word embeddings

The Word2Vec model [14] was a breakthrough in learning word embeddings. It utilized a neural network which learnt to predict a word based on its context (the words around it), thereby capturing semantic and syntactic relationships between words. However, Word2Vec generates a single embedding for each word, regardless of context. This means that it cannot generate embeddings for larger pieces of text like sentences or documents, which we refer to as *sentence embeddings*.

2.1.2. Transformer Models and Sentence Embeddings

To address these challenges, the field explored new architectures that could capture the intricate interplay of words in their specific contexts. This led to the development of Recurrent Neural Networks (RNNs) [15], including their variant Long Short-Term Memory (LSTM) networks [16], which were designed to handle sequential data and context in sentences. However, they processed sentences sequentially, leading to computational inefficiencies and struggles with long-range dependencies in the text.

The *Transformer* model addressed these issues by introducing self-attention [17], allowing each word to compute its context independently and in parallel. Rather than processing sentences in a linear sequence, the model dynamically identifies and prioritizes relevant parts of the input during inference to generate contextually appropriate segments of the output.

Figure 2.2 shows a minimal representation of the attention mechanism in the task of machine translation, where the intensity of the boxes

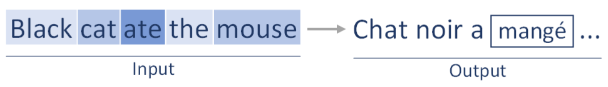


Figure 2.2: Attention Mechanism in Transformer Models

around each word in the input sentence represents the attention weight. The output token being generated is highlighted.

The efficiency and accuracy of Transformer models have revolutionized the field of NLP. Their versatility has enabled the creation of numerous cutting-edge applications. These include machine translation services (Google Translate [1]), instruction-following chatbots (ChatGPT [2]), high-accuracy speech recognition (Whisper [18]), and sentiment analysis, among many others.

We now delve into the specifics of BERT (Bidirectional Encoder Representations from Transformers), a particular implementation of the Transformer architecture that we have utilized in our project.

BERT

The Bidirectional Encoder Representations from Transformers (BERT) model has been a significant milestone in the application of Transformer models [19].

The initial input for BERT, akin to other language models, consists of *tokens*, which are numerical representations of words or characters in a text. In the realm of NLP, sentences are segmented into these tokens. BERT is designed to support a maximum of 512 tokens per input. This limit is a practical decision made during the design of BERT to balance the ability to handle reasonably long sequences of text with computational efficiency (since the attention score computation grows quadratically with the number of tokens). These tokens are then transformed into embeddings through an embedding layer, serving as the intermediate input for the subsequent layers of the model.

BERT's primary innovation is its bidirectional training, which allows the model to understand the context of a word based on all its surroundings (left and right of the word). This bidirectional approach enables BERT to generate a new set of word-level embeddings that carry richer contextual information. BERT then aggregates these embeddings to capture the overall semantic meaning of a given text, making it highly useful for many NLP tasks. However, to harness its full potential, fine-tuning BERT with domain-specific data is a common practice. This process allows the model to better understand and generate higher quality embeddings for the specific task at hand. The intricacies of our approach to this fine-tuning process will be unveiled in the subsequent sections.

2.2. Large Language Models

Large Language Models (LLMs) have significantly broadened the reach of NLP in everyday applications. While BERT, discussed in section 2.1.2, is indeed a type of LLM, the term 'LLMs' often refers to a newer generation of models that are larger and more capable. These include models such as the

[1] `https://blog.research.google/2017/08/transformer-novel-neural-network.html`
[2] `https://openai.com/research/instruction-following`

GPT series [20–22], Llama 2 [23], and PaLM 2 [24]. These advanced models, built upon the foundation laid by BERT and other transformer-based models, have introduced additional advancements and capabilities that have democratized NLP, propelling it into a wide array of everyday applications.

2.2.1. Characteristics and Training of LLMs

LLMs are Transformer-based models that predict the next token in a document. They are trained on a massive corpus of text data from various sources, such as Wikipedia, GitHub, Reddit, books, etc. These models have been found to be particularly effective for text generation tasks, setting them apart from other transformer-based models.

The size of these models, as shown in Table 2.3, varies significantly, ranging from millions to trillions of parameters. This variation in size is indicative of the diverse capabilities and complexities of different LLMs. For instance, larger models with trillions of parameters are typically capable of more sophisticated tasks and exhibit better performance on a range of NLP tasks. However, they also require more computational resources for training and inference, which can be a limiting factor in their use and deployment.

Figure 2.3: Summary of the most important LLMs and their sizes.

Model	Size (parameters)
BERT [19]	110M
GPT-3 [20]	175B
LLAMA-2 [23]	7B/13B/70B
PALM [24]	540B
PALM-2 [24]	Unknown (smaller than PALM)
GPT-4 (speculative)	1.76T (eight 220B models)

The performance of LLMs hinges on the quality and volume of their training data. They are trained on a corpus of high-quality, expertly curated data and further refined on a significantly larger internet dataset, adhering to the principle of scaling laws [25].

Additional fine-tuning methods are used for optimizing model performance. One such strategy is chat-finetuning, which is specifically designed to enhance the model's conversational abilities, including instruction-following [21]. Another key strategy is Reinforcement Learning from Human Feedback (RLHF) [26], a process where human helpers handpick the best answer from a set of model-generated responses, or through a system of community voting. Lastly, post-training alignment is utilized as a strategy to further refine the model's performance, factuality, and adherence to desired behavior, ensuring the model's output aligns closely with human expectations [22].

2.2.2. Practical Approaches to LLMs

Prompting Techniques

In this subsection, we aim to introduce some recent *prompt engineering* techniques, used to leverage the power of LLMs and handle their limitations, for a wide range of tasks.

Zero-shot prompting, introduced in [27], is a method of using natural language instructions to make an instruction-tuned language model perform a specific task without any training examples.

Following the concept of zero-shot prompting, another technique known as few-shot prompting has emerged. Introduced in [28], few-shot prompting involves providing the model with a small number of examples of the task at hand, as seen in Table 2.1. This allows the model to ”learn” the task from these examples and apply that learning to generate an appropriate response. This technique is particularly effective when the task is complex or when the desired output format is not easily described through a prompt alone.

Limitations of these methods include, for instance, zero-shot prompting might not always yield the desired results due to the lack of training examples, and few-shot prompting requires careful selection of high-quality relevant examples. A strategy of trial and error, active oversight, and critical assessment of language model outputs allows for iterative refinement of the prompts and examples used, leading to improved performance over time.

	Zero-shot [27]	**Few-shot [20]**

Input	`Premise: At my age you will probably have learnt one lesson.`
	`Hypothesis: It’s not certain how many lessons you’ll learn by your thirties.`
	`Does the premise entail the hypothesis?`
	`OPTIONS:`
	`-yes`
	`-it is not possible to tell`
	`-no`
		`A ”whatpu” is a small, furry animal native to Tanzania. An example of a sentence that uses the word whatpu is:`
		`We were traveling in Africa and we saw these very cute whatpus.`
		`To do a ”farduddle” means to jump up and down really fast. An example of a sentence that uses the word farduddle is:`

Output	`It is not possible to tell`	`When we won the game, we all started to farduddle in celebration.`

Table 2.1: Examples of zero-shot and few-shot prompting.

Chain-of-Thought (CoT) prompting [29] is a technique designed to enhance the reasoning capabilities of LLMs. It encourages them to break down complex problems into a series of intermediate reasoning steps. Instead of directly providing the LLM with a question and expecting a single-step answer, CoT prompting involves guiding the model through the reasoning process. This can be done by providing explicit steps or by prompting the model to “think step-by-step” and perform the reasoning by itself. Each logical step in the thought process helps the model arrive at the final answer through a chain of interconnected deductions. This approach has been shown to significantly improve the accuracy and logical coherence of LLM responses. It is especially effective for tasks that require common sense knowledge, as it encourages the model to generate more human-like responses. By explicitly modeling the reasoning process, we aim to utilize this prompting technique during our feature extraction step. This will aid our understanding of the reasoning behind LLM answers and reveal areas where its understanding breaks down, including due to logical connections or language limitations.

Knowledge Distillation

In addition to the aforementioned prompt engineering techniques, *knowledge distillation* has been explored by many researchers [30–32] due to the challenges associated with the deployment of large models. Many of these models are either proprietary, making them expensive to use, or they are too large and slow for practical use in a production or research environment.

Knowledge distillation is a process that relies on expertly crafted prompts, which guide the *teacher* model (a larger model) to generate high-quality training data. This data is then used to train the smaller *student* model. The objective of this process is to transfer the knowledge from the larger model to the smaller one, thereby enhancing the capabilities of the smaller model.

The goal of knowledge distillation is to improve the zero-shot capabilities of the smaller model. With the help of another expertly crafted prompt, the smaller model can generate responses that are similar in quality to those of the larger model, even without any additional training examples.

A clear example of the application of this technique is RankVicuna [30], an open-source LLM that performs listwise document reranking (the reordering of items in a list to improve output quality). It is based on Vicuna (a 7B parameter fine-tuned Llama 2 model) and is further trained on data generated by a larger LLM using knowledge distillation. Despite being 30 times smaller than other proprietary LLMs such as GPT-4, it achieves comparable performance in reranking tasks. Additionally, other recent studies [32, 33] have showcased the potential of LLMs in generating high-quality, contextually relevant synthetic data for training.

However, our research takes a different path. We explore the task of feature extraction, leveraging the ability of these models to capture complex patterns and relationships in textual data, specifically, from property ads. We hypothesize that such an approach would provide greater control, flexibility, and a lower error likelihood, in comparison to directly generating text using an LLM to build the dataset.

Quantization

Quantization is a straight-forward approach that has been utilized to deploy these large models on devices with limited computational resources. It involves reducing the precision of the weights in neural network models. In standard models, weights are typically represented as 32-bit floating-point numbers. However, through quantization, these can be represented as lower-precision formats, such as 16-bit integers or even lower.

The process of quantization, while beneficial in reducing memory requirements and computational cost, does potentially degrade the performance of the model due to the reduced precision of the weights. According to [34], an 8-bit quantization should be preferred over smaller full precision models. For instance, an 8-bit 30B quantized model outperforms a 13B model of similar size and should have lower latency and higher throughput in practice. This also holds for an 8-bit 13B model compared with a 16-bit 7B model.

One of the latest advancements in this field is GPTQ [35], a method for quantizing LLMs to 3 or 4 bits per weight, without retraining or significant accuracy loss.

Quantization enables the deployment of powerful LLMs in resource-constrained environments without a substantial compromise in performance and enabling the use and research of LLMs on more affordable consumer-level hardware.

2.3. Neural Information Retrieval

Neu-IR represents a significant advancement in the field of information retrieval [36]. It applies neural network models to enhance the understanding of query semantics, providing a more nuanced approach compared to traditional keyword-based search systems.

One common approach in Neu-IR is the use of embedding-based methods. These methods are particularly effective due to their ability to capture semantic similarities in text, a crucial aspect for understanding and processing natural language data.

2.3.1. Embedding-based Retrieval

Building on the concept discussed in the previous section, where words of similar meaning are closer in the embedding space, embedding-based methods in Neu-IR extend this principle to documents and queries. They map these entities into a shared vector space, enabling the computation of relevance based on proximity.

In embedding-based retrieval, documents are transformed into embeddings using a pre-trained model and stored in a *vector database*. Upon receiving a query (e.g., question, description, criteria), it is also converted into an embedding. The system retrieves documents by comparing embeddings to the query embedding using a measure such as *Cosine Similarity*, resulting in a ranked list of documents by similarity. This system is generally considered very efficient and can handle large document corpora.

Recent research has shown that Transformer models, when pre-trained with carefully designed tasks, can significantly enhance the effectiveness of embedding-based retrieval, outperforming traditional methods and other embedding models [37]. Furthermore, several practical cases have recently demonstrated the utility of these methods [38–40]. For instance, Facebook has applied embedding-based retrieval to their search system [41]. They developed a unified embedding framework to model semantic embeddings for personalized search.

2.3.2. SentenceTransformers

The SentenceTransformers [3] library is a Python framework that simplifies the generation and use of sentence and document embeddings using transformer models such as BERT. It is built on top of the Hugging Face's Transformers library and PyTorch, providing an accessible interface for working with these models.

This is enabled by the Sentence-BERT (SBERT) paper [42], which introduces a modification of the BERT network that significantly enhances the efficiency of embedding-based retrieval, making it much more suitable for tasks like semantic similarity search.

The SentenceTransformers library provides an API for *encoding* (embeddings generation) and model fine-tuning, making it an invaluable resource for our work. Furthermore, a variety of pre-trained models based on the BERT architecture, or its variants are offered in the library, such as *all-MiniLM-L6-v2* known for its efficiency and compact size, *multi-qa-mpnet-base-dot-v1* trained on a large question-answering dataset, and *paraphrase-multilingual-MiniLM-L12-v2* designed for cross-lingual tasks. These models, fine-tuned using unique datasets for different input lengths and NLP tasks, provide high-quality embeddings and will be evaluated in Chapter 4.

2.3.3. Evaluation

The evaluation of text embeddings, including retrieval, is a challenging task due to the complexity and subjectivity of language, the diversity of domains and tasks, the lack of standardized metrics and the scalability issues associated with the increasing size of language models and the volume of data they are trained on.

One way of addressing these issues is through comprehensive benchmarks like the Massive Text Embedding Benchmark (MTEB) [43]. MTEB spans 8 embedding tasks, covering several datasets and languages, providing a more robust and diverse basis for evaluation. For the retrieval task, which is the focus of our project, the main metric used in MTEB is normalized discounted cumulative gain at 10 (nDCG@10).

[3] `https://www.sbert.net/`

nDCG is a widely used metric for measuring the effectiveness of a ranking (retrieval) system [44]. Each item in the ranked list is assigned a relevance score. The gain of each item is then discounted based on its position in the list. This is calculated as the Discounted Cumulative Gain (DCG):

$$DCG_p = \sum_{i=1}^{p} \frac{rel_i}{log_2(i+1)}$$

where p is the position in the ranked list and rel_i is the relevance of the result at position i.

However, DCG does not consider the varying lengths of the ranked lists. Therefore, DCG is normalized by the Ideal Discounted Cumulative Gain (IDCG), which is the DCG value for the ideal ranking of the items. The nDCG is then calculated as:

$$nDCG_p = \frac{DCG_p}{IDCG_p}$$

nDCG values range from 0 to 1, where 1 indicates the perfect retrieval and ranking of the items.

Additionally, F1 score is another commonly used metric in the evaluation of text embeddings. The F1 score is the harmonic mean of precision and recall, providing a balance between these two metrics. It ranges from 0 to 1, where 1 indicates perfect precision and recall.

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

where precision is the proportion of true positive results among the total number of cases classified as positive, and recall is the proportion of true positive results among the total number of positive cases.

In addition to these traditional metrics, task-specific metrics are often used to measure system performance against specific objectives, such as user satisfaction, increased engagement, or higher conversion rates. A/B testing is a useful for assessing the impact of system configurations or algorithms on user behavior and goals.

In the context of real estate search, A/B testing can be used to compare different ranking algorithms, interface designs, or embedding models. It measures their effect on metrics like click-through rates, time spent on listings, or explicit user votes. This approach provides insights into the practical effectiveness of the system and guides further development and optimization efforts. Evaluation data can be used to improve the performance of existing models through RLHF, as introduced in Section 2.2.1.

2.4. Real Estate Search

The advancements in Neu-IR techniques have demonstrated significant potential in various domains, among which real estate stood out as an area ready for disruption and improvement. Existing research in this domain has primarily focused on recommendation systems, often utilizing structured data and collaborative filtering techniques, overlooking the potential of EBR for enhancing search and retrieval tasks [45].

Early efforts, such as RentMe [46], relied on knowledge-based systems, employing structured information and predefined rules to match users with properties. Both [47] and [48] propose the integration of embeddings with collaborative filtering models for generating property recommendations. However, these methods primarily focused on user-item interactions and did not explicitly leverage any property data. Other works proposed alternative techniques for incorporating textual information. For instance, [49] explored the application of Neural Tensor Networks for ranking property recommendations, utilizing word2vec representations of structured property features. [50] employed TF-IDF based on property titles, descriptions, and addresses to estimate user interest. However, this approach relies on bag-of-words representations, which fail to capture the contextual relationships and semantic nuances present in natural language descriptions. This limitation reflects the current state of online real estate listings, where, aside from some platforms that offer keyword-based search [4], the complexity of their search engine is primarily determined by the quantity and precision of the filters provided.

In contrast, our work proposes utilizing EBR, representing both the document (full real estate ad with textual description) and the user query as dense vectors capturing semantic meaning, as detailed in Section 2.3.1. This allows users to directly search using natural language queries, potentially improving the accuracy, flexibility, and accessibility of the search experience.

The embrace of other cutting-edge technologies showcases the readiness of this industry for innovation. These technologies include Virtual Reality (VR) and Augmented Reality (AR) for tours and renovations [5], Big Data for price estimation [6] and identifying market trends, and advanced machine learning (diffusion models) for automatic room decoration [7]. This strengthens our argument for a potential reinvention of the search engine paradigm focused on the real estate domain.

[4] `https://www.zoopla.co.uk/` (last accessed on 2024-04-10)
[5] `https://www.schwoererhaus.com/es-es/casas/visitas-virtuales/` (last accessed on 2024-04-11)
[6] `https://www.idealista.com/valoracion-de-inmuebles/` (last accessed on 2024-04-11)
[7] `https://stageinhome.com/` (last accessed on 2024-04-11)

3 Design and Implementation

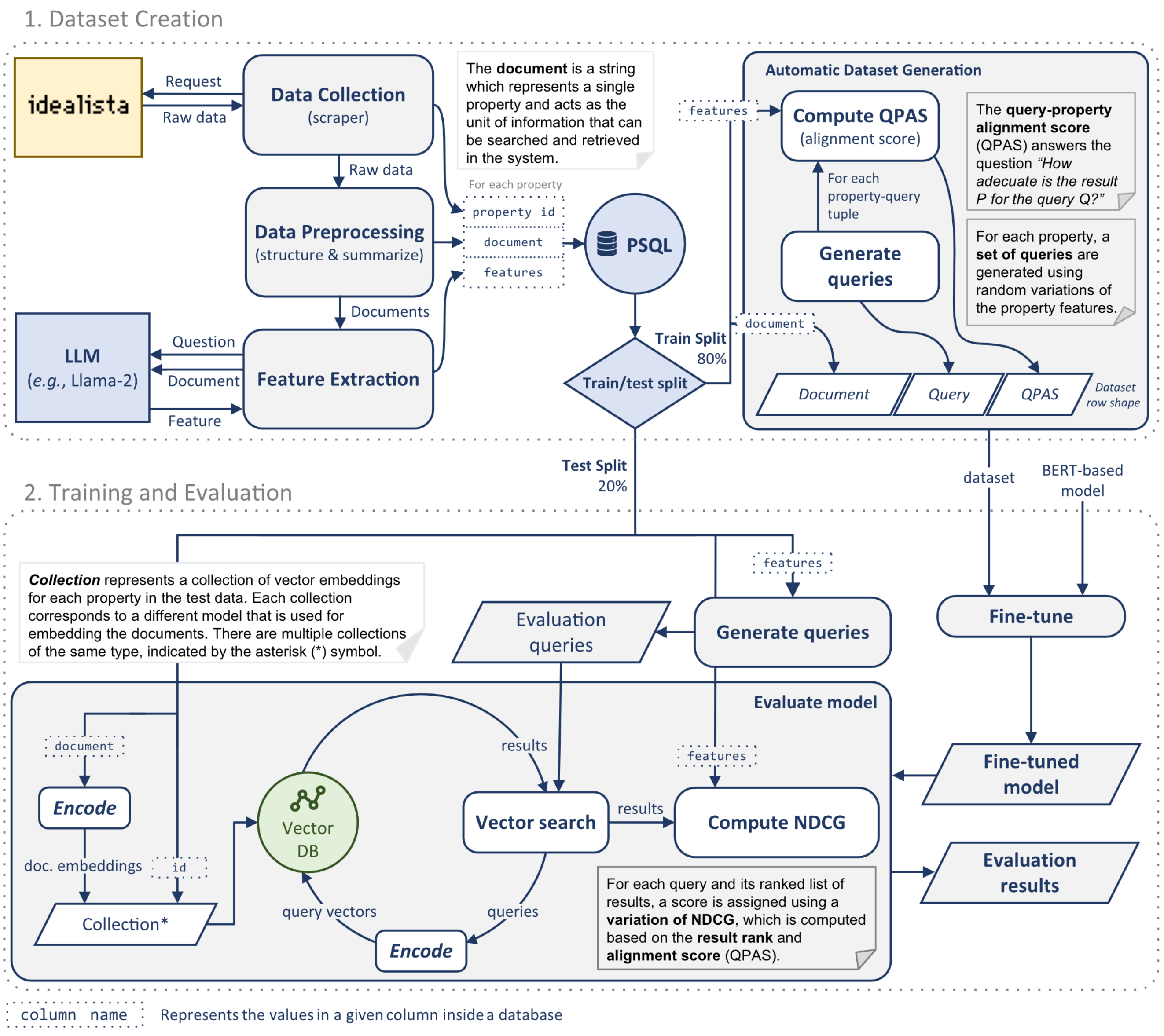
In this chapter, we aim to provide a clear and comprehensive view of our system. We delve into its design, detail the implementation process, and discuss the decisions made throughout its development.

3.1. Project Structure

The project is structured into three interconnected modules, each addressing a crucial stage in developing an enhanced real estate search system based on embedding-based retrieval and language models.

1. **Dataset Creation.** Lays the groundwork by collecting property listings, extracting structured features, generating synthetic queries, and creating training and evaluation datasets. This module leverages web scraping techniques and the capabilities of large language models to transform unstructured data into a format suitable for model training.
2. **Model Training and Evaluation.** Focuses on optimizing an embedding model for the real estate domain. It encompasses the selection of a pre-trained model, fine-tuning using the generated datasets, and empirical evaluation against baseline models to assess performance improvements.
3. **Web Application and Deployment.** Bridges the gap between the underlying technology and the user. It involves developing a backend API for processing user queries and a frontend user interface for interacting with the system. This module demonstrates the practical application of the project and facilitates real-time comparison of the performance of different models.

Figure 3.1.1 provides a visual representation of the data flow and interactions between the various modules within our project, offering a comprehensive overview of its architecture. The diagram outlines the journey of data, starting from the initial web scraping of property listings and culminating in the evaluation of our fine-tuned embedding model.



The **document** is a string which represents a single property and acts as the unit of information that can be searched and retrieved in the system.

The **query-property alignment score** (QPAS) answers the question *"How adecuate is the result P for the query Q?"*

For each property, a **set of queries** are generated using random variations of the property features.

Collection represents a collection of vector embeddings for each property in the test data. Each collection corresponds to a different model that is used for embedding the documents. There are multiple collections of the same type, indicated by the asterisk (*) symbol.

For each query and its ranked list of results, a score is assigned using a **variation of NDCG**, which is computed based on the **result rank** and **alignment score** (QPAS).

Figure 3.1.1: Data flow of the project, organized by modules, from acquisition to evaluation, highlighting key processes within the system.

3.2. Requirements Analysis

3.2.1. Dataset Creation

Functional Requirements

FR-DC-1.– The dataset must be created from real estate property listings.

FR-DC-1.1.– Property listings should be sourced from a reliable and diverse online platform (e.g., Idealista).

FR-DC-1.2.– The dataset should include a representative sample of property types, locations, and features.

FR-DC-2.– Each property listing in the dataset must have structured features extracted from its textual description.

FR-DC-2.1.– Feature extraction should be automated using a large language model (LLM) like Llama2.

FR-DC-2.2.– The extracted features should encompass key property attributes such as property type, size, number of rooms, amenities, and location information.

FR-DC-3.– The dataset must include synthetically generated queries that align with the extracted property features.

- **FR-DC-3.1.–** Query generation should incorporate variations in phrasing, terminology, and complexity to reflect real-world user search behavior.
- **FR-DC-3.2.–** Queries should cover a range of property features and search criteria.

FR-DC-4.– Each query-property pair in the dataset must have a corresponding relevance score.

- **FR-DC-4.1.–** The relevance score should quantify the degree of match between the query and the property features using the Query-Property Alignment Score (QPAS).
- **FR-DC-4.2.–** The QPAS calculation should consider partial matches and attribute-specific scoring criteria to reflect the flexibility of real estate search.

FR-DC-5.– The dataset must be divided into training and evaluation sets.

- **FR-DC-5.1.–** The split should ensure no overlap to prevent data leakage during model training and evaluation.
- **FR-DC-5.2.–** The evaluation set should include a diverse and representative sample of properties and queries.

Non-functional Requirements

NFR-DC-1.– The dataset creation process must be automated and efficient to handle a large volume of property listings.

NFR-DC-2.– The data collection and scraping process must respect website limitations and ethical considerations.

NFR-DC-3.– The feature extraction process using LLMs should maintain a low error rate to ensure data quality.

NFR-DC-4.– The dataset size and balance should be configurable to allow for experimentation and optimization.

3.2.2. Model Training and Evaluation

Functional Requirements

FR-MTE-1.– The system must allow evaluating different pre-trained embedding models for their suitability for fine-tuning.

FR-MTE-2.– The system must be able to fine-tune a pre-trained embedding model using the generated dataset.

FR-MTE-3.– The fine-tuning process should adapt the model for semantic similarity in real estate search.

FR-MTE-4.– The system should allow for experimentation with different hyperparameters during fine-tuning, including:

- **FR-MTE-4.1.–** Batch size
- **FR-MTE-4.2.–** Learning rate
- **FR-MTE-4.3.–** Number of epochs
- **FR-MTE-4.4.–** Loss function (e.g., Cosine Similarity, Softmax)

FR-MTE-5.– The fine-tuning process must be reproducible with consistent results.

FR-MTE-6.– The system should allow for saving and loading the fine-tuned model for further use and evaluation.

FR-MTE-7.– The system must allow comprehensive evaluation of fine-tuned and baseline models, including the metrics:

- **FR-MTE-7.1.–** Normalized Discounted Cumulative Gain (NDCG)
- **FR-MTE-7.2.–** F1 Score
- **FR-MTE-7.3.–** Adaptive NDCG (aNDCG)

FR-MTE-8.– The evaluation should be conducted using the evaluation dataset and a defined evaluation pipeline.

FR-MTE-9.– The system should provide structured reporting of evaluation results.

Non-functional Requirements

NFR-MTE-1.– The fine-tuning process must be computationally efficient and utilize available hardware resources optimally.

NFR-MTE-2.– The system should provide clear logging and monitoring of the training process.

NFR-MTE-3.– The evaluation process should be automated and efficient.

3.2.3. Web Application and Deployment

Functional Requirements

FR-WAD-1.– The system must include a web application that allows users to interact with the real estate search system.

FR-WAD-2.– The web application must provide a user-friendly interface for inputting natural language queries.

FR-WAD-3.– The application must display search results ranked by their similarity score, ensuring the most relevant properties are shown first.

FR-WAD-4.– The results must be displayed including relevant property details and images.

FR-WAD-5.– The application should allow users to compare search results obtained from different embedding models (*real-bert* and baseline models).

Non-functional Requirements

NFR-WAD-1.– The web application must have a responsive and intuitive user interface.

NFR-WAD-2.– The system must be deployable to a cloud platform or server infrastructure.

NFR-WAD-3.– The application must be secure and protect user data.

NFR-WAD-4.– The application must be maintainable and allow for easy updates and future development.

NFR-WAD-5.– The deployment should be cost-effective and utilize resources efficiently.

3.3. Design

This section outlines our real estate search system's architecture, detailing the design and interaction of its essential components, and providing a foundation for understanding the system's comprehensive functionality and workflow.

3.3.1. Dataset Creation

Figure 3.3.1 illustrates the sequence of interactions involved in our feature extraction process utilizing the Llama2 language model and storing the results in a relational database. The user interacts with a Command-Line Interface (CLI) to select the dataset and specify the number of properties to parse. The main program handles data loading, prompt construction, communication with the Llama2 server, answer parsing, and database operations. The diagram highlights key steps such as prompt building, question processing with dependency checks, answer validation, and data storage.

After our structured corpus is constructed, as will be further detailed in Section 3.4, we shift our focus to the generation of synthetic queries, the dynamic counterpart to our static corpus. Figure 3.3.2 illustrates our different query generation classes within this pipeline, employing a modular design. The `RandomQueryGenerator` simulates diverse, real-world search behaviors. The `InverseQueryGenerator` generates queries matching some given property attributes, mostly utilized for model training. Lastly, the `AttributeQueryGenerator` focuses on specific features, ensuring comprehensive attribute coverage and natural language variations.

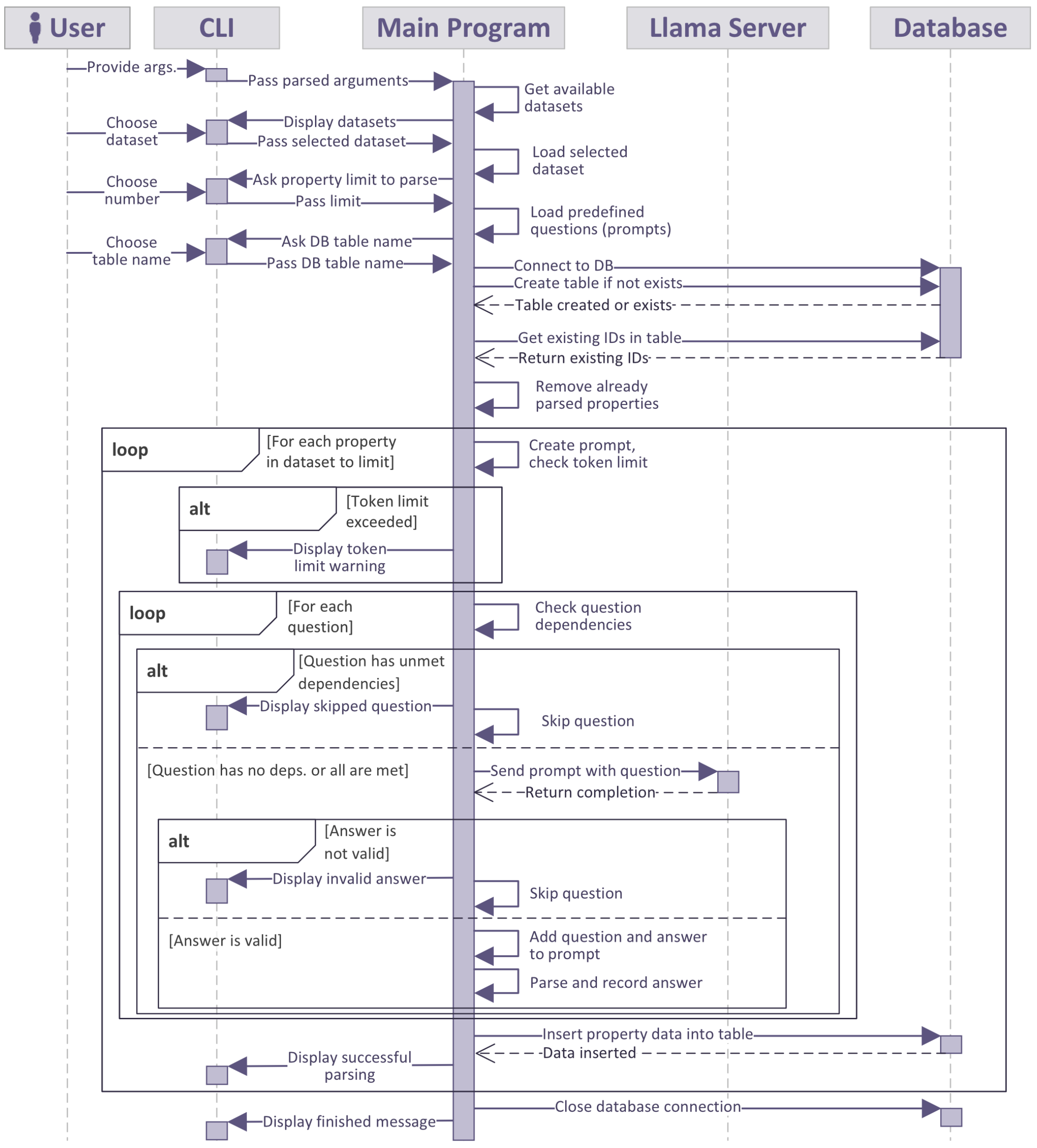


Figure 3.3.1: Sequence Diagram: Real Estate Property Feature Extraction Process.

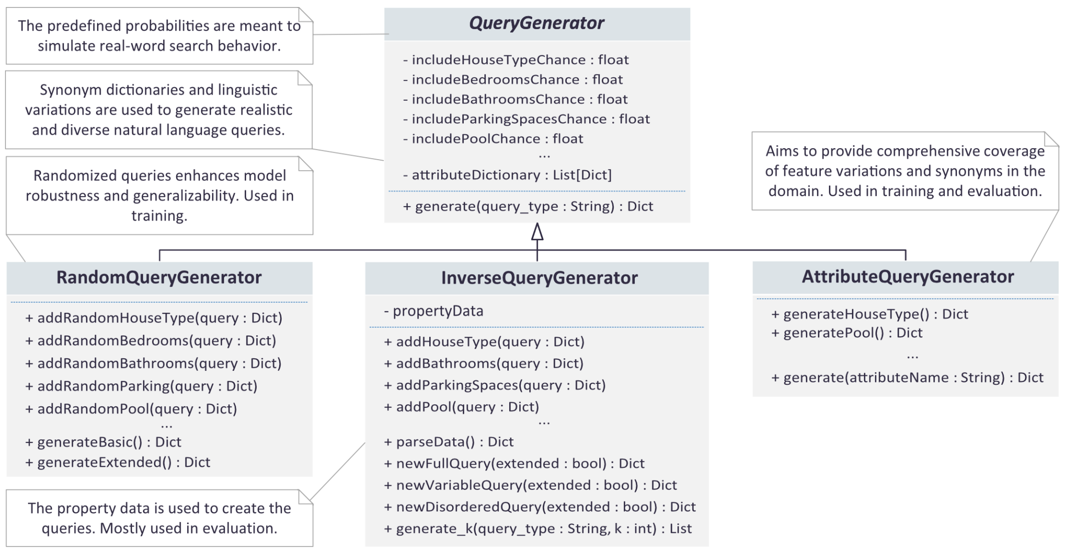

The predefined probabilities are meant to simulate real-word search behavior.
Synonym dictionaries and linguistic variations are used to generate realistic and diverse natural language queries.
Randomized queries enhances model robustness and generalizability. Used in training.
QueryGenerator
- includeHouseTypeChance : float
- includeBedroomsChance : float
- includeBathroomsChance : float
- includeParkingSpacesChance : float
- includePoolChance : float
...
- attributeDictionary : List[Dict]
+ generate(query_type : String) : Dict
Aims to provide comprehensive coverage of feature variations and synonyms in the domain. Used in training and evaluation.
RandomQueryGenerator
+ addRandomHouseType(query : Dict)
+ addRandomBedrooms(query : Dict)
+ addRandomBathrooms(query : Dict)
+ addRandomParking(query : Dict)
+ addRandomPool(query : Dict)
...
+ generateBasic() : Dict
+ generateExtended() : Dict
InverseQueryGenerator
- propertyData
+ addHouseType(query : Dict)
+ addBathrooms(query : Dict)
+ addParkingSpaces(query : Dict)
+ addPool(query : Dict)
...
+ parseData() : Dict
+ newFullQuery(extended : bool) : Dict
+ newVariableQuery(extended : bool) : Dict
+ newDisorderedQuery(extended : bool) : Dict
+ generate_k(query_type : String, k : int) : List
AttributeQueryGenerator
+ generateHouseType() : Dict
+ generatePool() : Dict
...
+ generate(attributeName : String) : Dict
The property data is used to create the queries. Mostly used in evaluation.

Figure 3.3.2: Class Diagram: Query Generation Strategies.

3.3.2. Model Training and Evaluation

To evaluate our selection of fine-tuned and baseline embedding models, a structured system for managing and utilizing these models is essential. This system must efficiently handle the storage, loading, and diverse generation APIs of the models within the evaluation pipeline. Figure 3.3.3 presents a class diagram that illustrates the core components and relationships essential for this task. It emphasizes the modularity and flexibility of the design, achieved using abstract classes and inheritance, which enables the seamless integration of various embedding models and their generation methods.

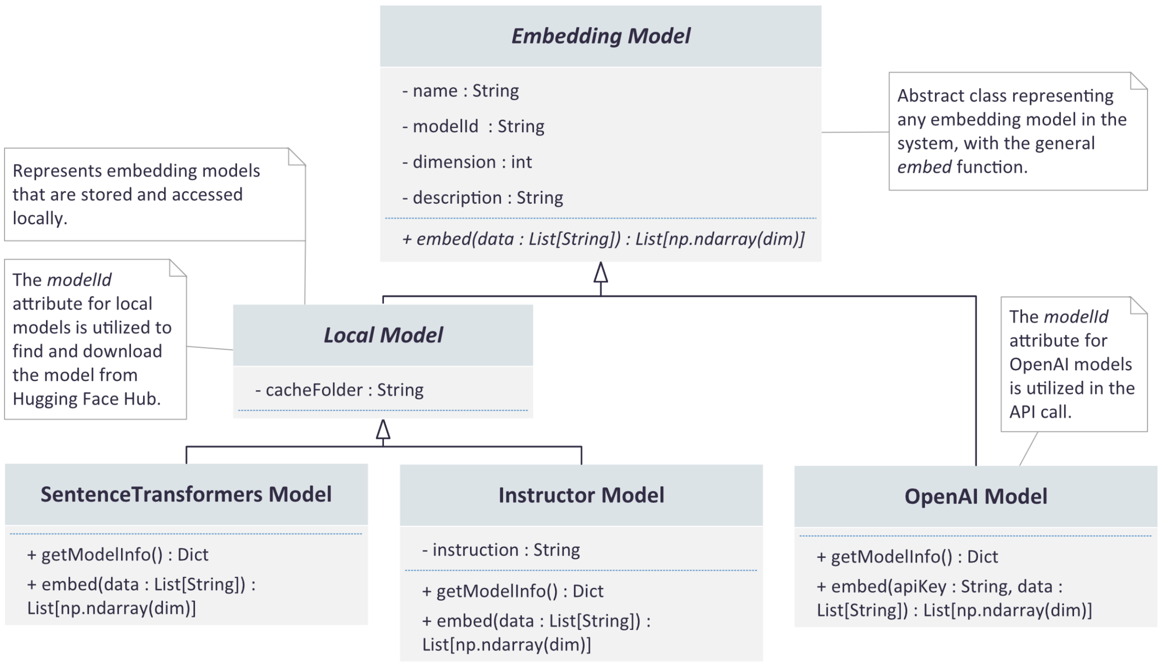

Embedding Model
- name : String
- modelId : String
- dimension : int
- description : String
+ embed(data : List[String]) : List[np.ndarray(dim)]
Abstract class representing any embedding model in the system, with the general embed function.
Represents embedding models that are stored and accessed locally.
The modelId attribute for local models is utilized to find and download the model from Hugging Face Hub.
Local Model
- cacheFolder : String
The modelId attribute for OpenAI models is utilized in the API call.
SentenceTransformers Model
+ getModelInfo() : Dict
+ embed(data : List[String]) : List[np.ndarray(dim)]
Instructor Model
- instruction : String
+ getModelInfo() : Dict
+ embed(data : List[String]) : List[np.ndarray(dim)]
OpenAI Model
+ getModelInfo() : Dict
+ embed(apiKey : String, data : List[String]) : List[np.ndarray(dim)]

Figure 3.3.3: Class Diagram: Embedding Model Management Structure

3.3.3. Web Application and Deployment

Bridging the gap between the underlying technology and the user experience, this section details the design of our demo web application. Figure 3.3.4 presents a sequence diagram that delineates the interactions between the components of our deployment during a typical user search and subsequent model comparison. It depicts the data and control flow, starting from the initial user query through to the display of ranked search results and the option to compare rankings from different embedding models. Furthermore, it highlights the utilization of several frameworks: Vite for the build tooling, Vue.js for constructing the frontend user interface, Gradio for facilitating the backend API, and Milvus as the vector database.

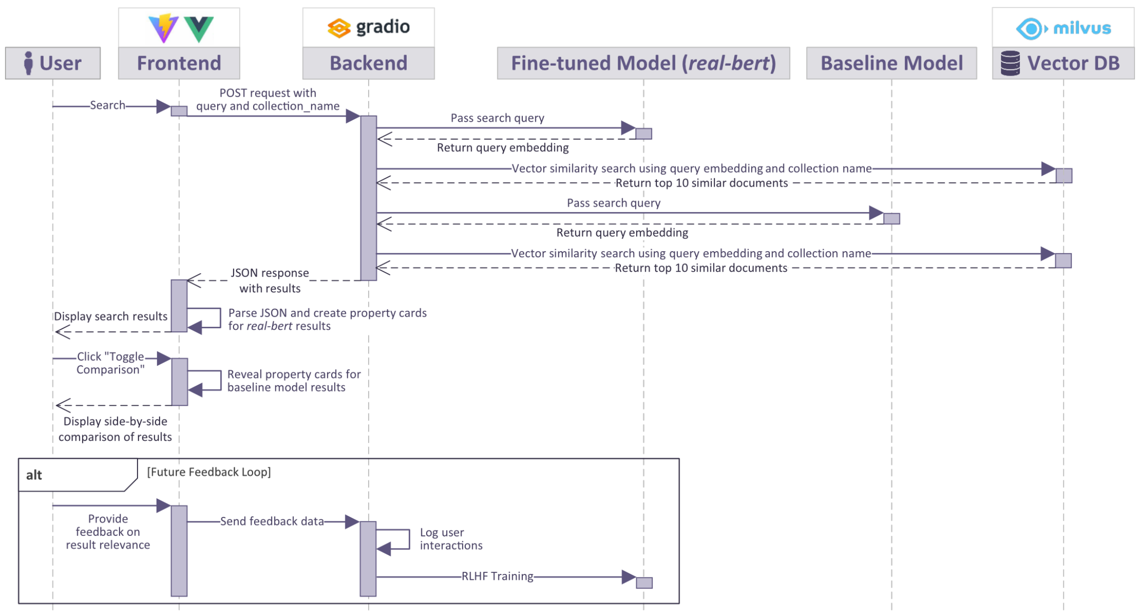

gradio
milvus
User
Frontend
Backend
Fine-tuned Model (*real-bert*)
Baseline Model
Vector DB
Search
POST request with query and collection_name
Pass search query
Return query embedding
Vector similarity search using query embedding and collection name
Return top 10 similar documents
Pass search query
Return query embedding
Vector similarity search using query embedding and collection name
Return top 10 similar documents
JSON response with results
Display search results
Parse JSON and create property cards for *real-bert* results
Click "Toggle Comparison"
Reveal property cards for baseline model results
Display side-by-side comparison of results
alt
[Future Feedback Loop]
Provide feedback on result relevance
Send feedback data
Log user interactions
RLHF Training

Figure 3.3.4: Sequence Diagram: Demo Web Application Component Interaction

3.4. Implementation

In this section, we will examine in detail the specific implementation of all the modules that collaborate in our system. To achieve this, we aim to present a narrative that follows the flow of data from its initial collection to its ultimate application in property search. This approach enables us to align the discussion with the actual data pipeline, offering a transparent and thorough understanding of our methodology.

3.4.1. Data Preparation

This section outlines our initial data preparation phase with the creation of our *unstructured corpus*. We focus on collecting property listings and transforming them into documents suitable for retrieval and feature extraction.

Data Collection

To obtain high-quality and authentic data, we turned to web scraping the Idealista online property listing. A basic web crawler was developed using Scrapy [1], a fast and powerful open-source web crawling framework. The crawler operates in two steps to ensure efficient data collection. In the first step of our pipeline, the crawler navigates through a given area listing to identify links to individual property ads. Our goal is to collect data from any given number of properties in several distinct areas, creating a sufficiently large data source for robust model training and evaluation. In the second step, the crawler

[1] `https://scrapy.org/` (last accessed on 2024-04-12)

extracts several text attributes using the HTML tag and class of all identified property ad fields. This gathering forms the initial *raw data* of our research, which maintains the same level of structure as the original website layout. A sample of this data is displayed in Figure A.1. To ensure an uninterrupted data collection process, we implemented a simple 50ms delay adhering to the website request rate limits.

Data Preprocessing

The property ad comment was of particular interest as it often contained additional, unstructured information about the house attributes and amenities. This information is crucial as it provides a more detailed and nuanced understanding of each property, beyond what is captured in the structured data fields, utilized in traditional search methods.

The next step in our pipeline is to generate the *document*. In information retrieval, this term refers to a single unit of retrievable information. Here, our document is a string representing a single property, containing all its relevant details.

To manage the token limit of BERT-based models (see Section 2.1.2), we summarize the comment text before creating the document. We employ an extractive summarization method by defining a set of labels for categorizing each paragraph within the text. This approach allows us to systematically assign categories to paragraphs using a zero-shot classification model, specifically the *Zero-shot Spanish ELECTRA* [51] (fine-tuned on the XNLI dataset [52]). Paragraphs labeled with categories deemed irrelevant–such as paragraphs containing real estate company data–are then excluded from the text.

These categories were designed utilizing a form of few-shot prompting where keywords are included in the category definition to aid the model and improve performance. They were tested and refined multiple times to ensure their effectiveness. A sample of these categories can be found in Table A.1. The 'Relevance' column indicates if a category is included in the summary ('Relevant') or excluded ('Not relevant'). After this process, most comments were found to be well under 512 tokens, ensuring that our data is manageable by most embedding models and ready for the subsequent document creation step.

Unstructured Corpus

The document, as shown in Figure A.2, contains the property ad values from the raw data corpus (title, location, price, list of key characteristics, and energy efficiency labels) along with the summarized comment. It follows a rigid, predefined format, where the relevant attributes are added in this general pattern. This implied text structure will serve as the training foundation. By fine-tuning on a large dataset of similarly formatted property ads, the model learns to recognize this pattern, enhancing its ability to extract information from new ads in the same format and potentially improving its overall performance on properties using this consistent document structure. The collection of these documents, each representing a single property ad, forms our *unstructured corpus*. The detail and diversity inherent in this corpus are crucial for the feature extraction, fine-tuning, and evaluation processes that follow.

3.4.2. Feature Extraction

While the documents from our unstructured corpus could indeed be embedded using a pre-trained model to establish a basic embeddings-based property search system, we aim to push the boundaries of this initial setup. As it will be evidenced in Section 4.2.1, existing models, despite their capabilities, still leave a considerable margin for improvement.

A common obstacle in many domains, including real estate, is the lack of publicly available datasets for model training, making the fine-tuning process unfeasible. To overcome this challenge, we employ a novel *feature extraction* approach which aims to effectively transform the diverse and complex content of property descriptions into a structured set of features.

The novelty of our approach lies in its utilization of Large Language Models (LLMs) for semantic interpretation and subsequent automatic labeling based on the extracted features, ultimately fine-tuning our model. These LLMs, introduced in Section 2.2.1, are a key component of our work.

Utilizing LLMs

The latest advancements in open-weights LLMs (Llama2 [23], phi-1 [53], Falcon [54], etc.) provide an unprecedented opportunity to automate language-related tasks in our feature extraction process. One key improvement is the reduction in hallucinations [55], instances where the model generates false information that is not present in the input data. Figure A.3 illustrates our feature extraction process using the 4-bit Llama2-Chat 13B model. As shown, this model can effectively address potential misconceptions in property ads, making it suitable for question-answering tasks in this domain.

The Llama2-Chat model is an open-weights chat aligned LLM, trained on a massive corpus of diverse text data [23]. It comes in three different sizes: 7B, 13B, and 70B, where the number indicates the number of parameters in the model. Due to hardware limitations, even with the smaller-sized models, running LLMs locally posed a challenge. To address this issue, we opted to use a 4-bit GPTQ quantized 13B Llama2. This model reduces the memory requirement by 2.6x compared to the original 32-bit 13B model and can run on consumer hardware requiring less than 11GB of VRAM. We deployed this model using the *llama.cpp* library [2], which provides an efficient inference server accessible via HTTP requests. Despite its smaller size, this compressed model performs well on our domain-specific task, with a detailed discussion on its performance and error rate to follow (Section 3.4.2).

Question-Answer Mechanism

By leveraging the language understanding capabilities of Llama2, we implement a question-answer mechanism to automatically extract property features from our unstructured corpus. These features, such as number of rooms, floors, parking spaces, and the presence of amenities like pools, align with the

[2] `https://github.com/ggerganov/llama.cpp` (last accessed on 2024-05-04)

diverse and specific queries that potential property seekers may have. This ensures a comprehensive *structured corpus* of property ads and features.

To achieve better accuracy and alignment in multi-turn question answering with Llama2, we use a chat template to dynamically include the history of answered questions for a given property, post-instruction delimiter. The model integrates this as a starting point into its internal representation of the conversation, enhancing its ability to produce contextually relevant subsequent responses. Essentially, the model continues the conversation during inference as if the appended answers were its own. This few-shot prompt engineering technique, together with custom stop sequences, ensure that only a minimal number of completion tokens need to be generated.

Figure A.4 illustrates the dynamically constructed prompt string, including the document, instruction, history, question, and the model's completion, halfway through the feature extraction process for a sample document. This dynamic prompt construction process is fully automated for every property ad in our corpus. Completions from the model are parsed to extract structured data, accounting for potential variations in phrasing and terminology. The parsed data is stored in a relational database for further analysis and serves as the foundation for our fine-tuning dataset. By automating this, we establish a robust and efficient process applicable to large volumes of data.

Error Analysis and Mitigation

To assess the accuracy of Llama2's feature extraction capabilities, we manually annotated a set of properties and compared the results to the model's output. Initial attempts to extract features using arbitrary questions often yielded inaccurate results, necessitating an iterative refinement process. For each targeted feature, we formulated a question and meticulously refined it through manual inspection of the model's output across numerous examples from our dataset. This iterative human-in-the-loop approach ensured that the questions elicited the most accurate responses possible from Llama2 for the maximum number of properties within the corpus.

The system demonstrated high accuracy for basic features like the number of bedrooms and bathrooms, with error rates consistently below $1\,\%$. Questions about the presence of specific amenities presented a greater challenge for the model, with missed detections ranging from $4\,\%$ to $7\,\%$. These error rates are calculated as the ratio of missed detections (false negatives) to the total instances of properties with the feature in question. Reported error rates are included in Table A.2.

Our main approach for question refinement was directly querying the LLM to understand its reasoning. By asking the LLM "why" questions after it provides incorrect answers, we can delve into its reasoning processes. This allows us to uncover comprehension gaps, assess the quality of its logic, and understand limitations in handling multilingual input. Explanations from the model can reveal areas where its understanding breaks down, including difficulties in making logical connections or aligning its thought processes with human intuition. This developer-LLM dialogue often highlights either a lack of sufficient evidence in the provided property ad text or challenges related to synonyms and translations.

Analysis revealed that precise questions and the few-shot prompting technique significantly improved the accuracy. Our approach was to incorporate relevant examples and synonyms into the question itself (e.g., “Is it a house (chalet) or apartment (including duplex, studio, flat, etc.)?”). Interestingly, Spanish questions often yielded false positives, while purely English questions resulted in false negatives. This was addressed by using English questions with key Spanish keywords, such as “Is it a penthouse top-floor apartment (ático)?”. Samples of the revised questions utilized can be found in Table A.2.

3.4.3. Query Generation System

With the ability to represent property ads as structured features, we can now explore synthetic query generation to create a diverse set of search queries that directly correspond to these features. This approach offers several advantages for effectively fine-tuning our embedding model. First, it allows us to easily create a large and diverse set of queries, eliminating the need for specialized datasets or user search history data. Second, it leverages the duality of query representation: we generate both natural language query strings and their corresponding feature dictionaries, enabling the model to learn from both the explicit features and the implicit linguistic nuances, as further elaborated in Section 3.4.5. However, we acknowledge that synthetic queries may not fully capture the implicit context and unpredictable nature of real user search behavior.

Natural Language Attribute Dictionaries

Each query is generated by randomly selecting both the property features themselves (e.g., number of bedrooms, parking availability) and their values (e.g., 3 bedrooms, 2 parking spaces), except for boolean features which are simply present or absent (e.g., terrace, elevator).

Then, the natural language query string is constructed using predefined dictionaries containing synonyms, hypernyms, and other linguistic variations such as different phrasings, connectors, grammatical elements and word orders. These dictionaries are designed with both linguistic and domain knowledge in mind, allowing us to generate diverse and realistic queries that reflect the various ways users might express their search criteria.

Random and Semi-Random Query Generation

Building upon the foundation of attribute dictionaries, we employ random and semi-random generation techniques to further tailor the diversity and variability of our synthetic queries, ensuring their relevance to actual property listings.

Random query generation exposes the model to a wide range of search criteria, including those queries that may not have a perfect match within the available properties. This is crucial, as partially-correct queries are arguably the most prevalent query type encountered in real-world search scenarios.

In contrast, semi-random generation uses a specific property ad as input and generates perfectly matching queries. Despite perfect feature matches, the linguistic variations within the query remain random. This improves fine-tuning by boosting recognition of similar properties despite query variations.

Query-Property Alignment Score (QPAS)

To measure the relevance of generated queries to specific property ads, we introduce the Query-Property Alignment Score (QPAS). This score quantifies how good of a match a property ad is for a specific search, playing a crucial role in training and evaluation.

The QPAS considers all features present in the query (e.g., house type, number of bedrooms, amenities) and compares them to the corresponding features extracted from the property ad (Section 3.4.2). Each attribute is assigned a weight reflecting its perceived importance in the search process. The final score is a weighted average of the individual attribute matches.

To capture the inherent flexibility of real-world search queries, our QPAS implementation supports partial scoring and incorporates attribute-specific scoring techniques. Taking ’3 bedrooms’ as an example query, a property with 4 bedrooms could still be considered a match, thus contributing positively to the overall score. Similarly, “swimming pool” would match both urbanization and private pools unless “private pool” is explicitly specified. Figure A.5 displays scored property-query tuples from our dataset.

3.4.4. Dataset Creation

This step in our pipeline transforms our real estate ad corpus into a format suitable for model training, with the aid of our robust query generation and scoring system. The dataset shapes the learning process and ultimately determines the effectiveness of our embedding model for real estate search. In this section, we detail the variables and settings that guide the automated dataset generation process. The impact of these configurations is analyzed in Chapter 4, where we experiment with various parameters to identify the optimal combination for dataset size, query type, balance, and loss function to enhance model performance. A dataset preview sample can be found in Figure A.5.

Dataset Shape and Balance

To ensure no overlap between training and evaluation, we set aside a selection of properties to be used exclusively for evaluation, as will be further discussed in Section 4.1.2.

Our training dataset structure draws upon the format of the Stanford Natural Language Inference (SNLI) corpus [56], widely used in natural language inference tasks. Each entry comprises a pair of texts: a query and a property ad. However, instead of using categorical labels as in SNLI, we employ a continuous value, Query-Property Alignment Score (QPAS), to represent the degree of relevance between the two texts. Score representation in the dataset varies with the loss function, as will be detailed in Section 3.4.5. The size of the dataset is determined by the number of queries generated per property.

The dataset is created by utilizing the random and semi-random query generation process detailed above (Section 3.4.3), computing QPAS for each resulting query-property pair, with the property ad text sourced from our unstructured corpus (Section 3.4.1). The score distribution, both overall and per property, significantly influences learning. To manage this, we implement several balancing strategies that control the frequency of each score. For example, a ”flat” balance ensures an equal number of queries for each score, while a ”increasing ramp” balance results in a dataset in which queries with higher scores are more prevalent. Figure 3.1 illustrates this concept. The x-axis represents scores from lower to higher, and the y-axis represents the frequency of queries with those scores.

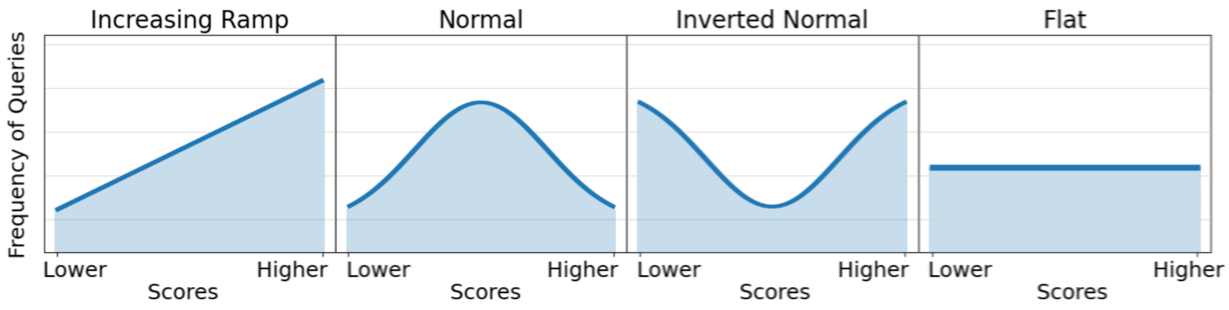


Figure 3.1: Illustration of different score distribution strategies.

Query Types

To enhance the model’s ability to learn various property features, our dataset is composed of three distinct query types. Each type includes queries that focus on a specific subset of the extracted property features. *Basic queries* concentrate on essential features such as house type, number of bedrooms and bathrooms, and parking availability, providing the model with a foundational understanding of core property attributes. *Extended queries* build upon the basic queries by also requesting specific amenities or equipment, such as a pool, garden, or terrace. Their goal is to ensure comprehensive coverage of the real estate domain and enhance the model’s generalizability. Lastly, *attribute-specific queries* are designed to focus on a single feature, aiming to enhance the model’s understanding of textual variations within that specific attribute.

We adjust the proportion of each query type in our dataset to reflect both potential user search patterns and the distribution of amenities in the actual data. This approach creates a more realistic training environment and helps prevent overfitting to less common property features, crucial for fine-tuning an effective and generalizable model.

3.4.5. Fine-Tuning

This section will delve into the specifics of fine-tuning our embedding model for real estate search. Building upon the dataset created in the previous section (Section 3.4.4), we will discuss the resources, techniques, and considerations involved in optimizing the model's performance for this specific task.

Computational Resources

We employed the SentenceTransformers framework for fine-tuning (see Section 2.3.2), leveraging its high-level API including the `fit()` method, which handles batch processing, loss calculation, back-propagation, and model optimization. Additionally, the Adam optimizer [57] was utilized, a popular choice for training large deep learning models due to its efficient memory usage and adaptive learning rate (adjusted individually for each weight).

Large transformer models require significant memory to store parameters and intermediate values during training. GPUs are commonly employed to accelerate this process by parallelizing the matrix operations involved. However, they can be limited by memory capacity. In our experiments, we utilized an RTX 2080Ti GPU, which has a VRAM capacity of 11GB–the largest available to us. We also considered using the free T4 GPU provided by Google Colab, which offers 15GB of VRAM. However, we found that training on this platform was approximately four times slower. We utilized the full 11GB of VRAM available on the 2080Ti. However, the actual VRAM usage can vary depending on the specific model being trained and the batch size, which we will discuss now.

Hyperparameters

Determining the optimal hyperparameters for training is typically an iterative process guided by both theoretical understanding and empirical results, rather than a deterministic process.

The batch size hyperparameter refers to the number of training examples used per iteration of gradient descent during the training process. While larger sizes can accelerate training, smaller ones can mitigate overfitting. In our experiments, hardware limitations dictated a batch size of 14.

Another hyperparameter, the loss function, defines the objective the model aims to minimize during training. We explored both Cosine Similarity Loss and Softmax Loss (see Chapter 4). Cosine Similarity encourages representations where similar items are closer together in the embedding space, often used for semantic similarity. Softmax Loss, typically used for classification tasks, assigns inputs to categories using a probability distribution. In our context, it is used for classifying properties into five discrete relevance categories (from 'highly irrelevant' to 'highly relevant'). We anticipate Cosine Similarity Loss to be more effective due to its ability to achieve nuanced and flexible matching compared to discrete classification.

The number of epochs, another crucial hyperparameter, determines how many times the learning algorithm will work through the entire training dataset. Balancing the epochs is key to avoid underfitting and overfitting. We found that 2 epochs was optimal for our dataset, as detailed in Chapter 4.

We retained the default settings for other hyperparameters in SentenceTransformers, including the learning rate of $0{,}001$, learning rate decay, weight decay, among others, under the assumption that these defaults would be well-suited for our task.

BERT-based Model Selection

Rather than training a model from scratch, we opted to fine-tune an existing embeddings model. This allows us to leverage pre-existing knowledge captured by the model from large-scale datasets, increasing the potential generalization capabilities and saving computing time. Resources utilized for this task include the SBERT model evaluation [3] and the MTEB Leaderboard [4].

Table 3.1 presents a comparison of several pre-trained models considered for fine-tuning. The table highlights key attributes of each model, including the dimensionality of the embeddings, the maximum sequence length (in tokens) the model can handle, the language of the training data, and the typical text granularity (sentence or paragraph level) the model is designed for. Additionally, we also consider the model's size and performance score on semantic search benchmarks, as reported in [3].

Model	Dim.	Seq. Len.	Purpose	Language	Granularity	Performance	Size
all-MiniLM-L6-v2 [58]	384	256	General	English	Sentence, Short Paragraph	49.54	80MB
all-MiniLM-L12-v2 [58]	384	256	General	English	Sentence, Short Paragraph	50.82	120MB
multi-qa-distilbert-cos-v1 [59]	768	512	Semantic Search	English	Sentence to Paragraph	52.83	250MB
sentence-similarity-spanish-es [5]	768	512	Textual	Spanish	Sentence Similarity	N/A	450MB
all-mpnet-base-v2 [60]	768	384	General	English	Sentence	57.02	420MB
all-distilroberta-v1 [61]	768	512	General	English	Sentence	50.94	290MB
instructor-xl [62]	768	512	Multi-task	English	Flexible	N/A	5GB

Table 3.1: Comparison of Selected Pretrained Models for Fine-Tuning and Evaluation.

Both MiniLM models [58] (*all-MiniLM-L6-v2* and *all-MiniLM-L12-v2*) offer an attractive size-performance balance. However, their limitation lies in the shorter maximum sequence length (256 tokens). The *multi-qa-distilbert-cos-v1* model [59], pre-trained on a large dataset of question-answer pairs for the semantic search task, aligns well with our fine-tuning dataset, comprised of shorter queries and longer documents. Conversely, the *sentence-similarity-spanish-es* model [5] is trained for the sentence-level textual similarity task. However, it is based on BETO (spanish-BERT) [63] and further fine-tuned on the spanish subset of the multilingual STS Benchmark (STSb) [64], which could provide a solid foundation given

[3] `https://www.sbert.net/docs/pretrained_models.html#model-overview` (last accessed on 2024-04-14)
[4] `https://huggingface.co/spaces/mteb/leaderboard` (last accessed on 2024-04-14)
[5] `https://huggingface.co/hiiamsid/sentence_similarity_spanish_es` (last accessed on 2024-04-15)

our Spanish-only data. The *all-mpnet-base-v2* [60] and *all-distilroberta-v1* [61] models, both trained for general sentence similarity tasks, present a trade-off between performance and sequence length. The former, delivering superior accuracy, is limited by a shorter (384-token) input capacity. The latter offers the ability to handle longer sequences at the cost of average performance. Lastly, the *instructor-xl* model [62] stands out for its versatility and high performance across a range of NLP tasks[4], uniquely leveraging task instructions for embedding.

The *sentence-similarity-spanish-es* model was chosen as the foundation for our *real-bert* model due to its strong performance in preliminary evaluations, which will be detailed and discussed in Chapter 4.

3.4.6. Embedding-based Retrieval

Having established the foundation for our real estate search system through the meticulous creation of a fine-tuning dataset (as explored in Section 3.4.4) and the overview of suitable pre-trained embedding models for our task (detailed in Section 3.4.5), we now delve into an essential component of our work: the implementation of the EBR system.

This section outlines the storage, indexing, and utilization of document embeddings for efficient property retrieval and ranking in response to user queries. The system introduced here is pivotal, forming the basis for our demo web application (Section 3.4.7) and systematic evaluation (Chapter 4).

Vector Database and Indexing

Introduced in Section 2.3.1, the vector database handles the storage and retrieval of document embeddings and texts. We sought a scalable, efficient, developer-friendly, and robust database, choosing Milvus [6], an open-source solution optimized for vector and traditional queries. Its Python SDK integrates well with our data pipelines, simplifying the implementation process. The database is set up in a standalone Docker container. However, Milvus also supports Kubernetes, which aligns with our mindset of implementing a realistic and almost production ready EBR system.

Each Milvus collection uses a *schema* to define entry structure. Our schema includes document text (from the unstructured corpus in Section 3.4.1), a unique property ID, and document embeddings generated with the `encode` function from SentenceTransformers. Additionally, property metadata, including extracted features and property descriptors (e.g., price, location), stored in the relational database from Section 3.4.2, is linked to the vector database via property ID.

Similar to relational databases, indexes are used to improve the speed of similarity searches. We chose the `IVF_FLAT` index, which partitions the embedding space into clusters. Initially for each search, only the distances between the input vector and the center of each cluster are computed.

[6] `https://milvus.io/` (last accessed on 2024-04-15)

To eliminate potential index-related recall errors and ensure every element is factored in, we set the `nlist` parameter to a small value for finer partitioning and the `nprobe` parameter to a high value to query as many clusters as possible. Other index types based on `IVF` quantize the vector embeddings to reduce CPU and disk resource usage, such as `IVF_SQ8`.

Ranking and Retrieval

Having established how to populate our vector database, we now delve into the actual search functionality, which details how the real estate properties are selected and ranked given a search query.

We employ a two-step retrieval with the `IVF_FLAT` index. First, the query is transformed into a vector by an embedding model, guiding the selection of relevant clusters in the collection index. Then, the embedded query is compared to the individual elements within these clusters for reranking, returning a list of results sorted by similarity. It is important to note that both the relative values among these distances and their magnitudes can vary significantly, depending on the embedding model used. Each result is a dictionary which includes the fields defined in our collection schema, providing the necessary information to evaluate or present the results to the user.

The `collection.search()` method in Milvus abstracts the search process, accepting arguments like embeddings, similarity metric (we use cosine similarity for relevance), `k` for the maximum number of results, offset, and an optional expression to filter the results based on traditional queries.

For evaluating retrieval and embedding quality, we utilize well-established metrics such as F1 and Normalized Discounted Cumulative Gain (NDCG), to measure precision and ranking accuracy. Additionally, we introduce a novel metric: **adaptive-NDCG (aNDCG)**, which aims to address the shortcomings of considering a fixed number of relevant results in sparse corpora, leveraging our structured corpus. aNDCG adjusts the number of considered results (`k`) in NDCG@k based on the actual perfect matches in the corpus, identified via structured queries in our database. This count is doubled to include perfect and near-matches, with `k` set between predefined limits of 4 and 10 to highlight the most relevant results. These metrics and their impact will be discussed in Chapter 4.

3.4.7. Demo Web Application

Building upon the solid foundation of our comprehensive EBR system (Section 3.4.6) and embedding model fine-tuning implementation (Section 3.4.5), we now shift our focus to developing a tangible solution for real estate search. This section delves into the development of a demo web application, comprised of a back-end REST API and a front-end public static site. Together, they bridge the connection between our vector database, embedding model and the user, with the goal of demonstrating the practical application of our work and providing a user-friendly interface for intuitive real estate property search.

In addition to the standard search functionality and visualization of ranked real estate ads, our web app aims to compare model performance in real-time by showing results from both our fine-tuned model and a baseline model, potentially highlighting the ranking improvements from fine-tuning.

Back-end Design with Gradio

Gradio, introduced in [65], is an open-source Python library for quickly creating demos or web applications for machine learning models. It is employed to build the REST API for our application, offering easy deployment, rapid prototyping, and flexible integration with machine learning frameworks.

The API exposes a single endpoint, `/search`, designed to handle user queries and return relevant property listings. This is defined using a Gradio Interface class, which encapsulates a `search` function that is invoked for each request. The `/search` interface requires the `query` and `collection_name` parameters as inputs, which are strings containing the natural language user query and the name of the vector database collection to search within, respectively. The code snippet A.1 illustrates the instantiation of this Interface class.

The `search` function embeds the user query, conducts a similarity search in Milvus, and returns a JSON response with ranked results and relevant details (see Section 3.4.6). To avoid deploying a relational database and ensure each API response includes comprehensive property information, we expanded the Milvus collection schemas to include additional property descriptors such as the listing ad title, description, price, location, and image URLs. This way, each element in the response encompasses all the necessary details from the original property ad, ready to be displayed to the user.

In this implementation, two distinct embedding models are employed for the real-time comparison of search results. We avoid the significant latency of initializing the embedding models by pre-loading them at startup. The models are downloaded from the Hugging Face Hub [7] using the SentenceTransformers library (see Section 2.3.2). Additionally, the backend establishes a connection to our Milvus server, utilizing the *pymilvus* library [8]. Credentials and host information are securely stored and accessed via environment variables.

The deployment of our Milvus database and Gradio API is detailed in the conclusion of this section.

Front-end Design with Vite

The end-user interacts with our backend API via a static web application built with Vite [9] and Vue.js [10]. We utilize Vite as our front-end build tool due to its support for rapid iterative development with Hot Module Replacement (HMR), faster startup with ES Modules (ESM), and efficient build optimiza-

[7] `https://huggingface.co/` (last accessed on 2024-04-16)

[8] `https://github.com/milvus-io/pymilvus` (last accessed on 2024-04-16)

[9] `https://vitejs.dev/` (last accessed on 2024-04-17)

[10] `https://vuejs.org/` (last accessed on 2024-04-17)

tions. The User Interface (UI) is constructed using Vue.js, leveraging its component-based architecture and reactivity system, which are well-suited for the dynamic and interactive visualization of the search results. Instead of relying on external UI frameworks, we opt for a custom CSS approach for simplicity.

Figure 3.2 displays the home screen UI. It features a distinct header, which contains a search bar for user input. Below the search bar, there are pre-defined search suggestions that users can select to populate the search input. These suggestions serve as examples to inspire users and showcase the system's ability to understand and respond to natural language queries.

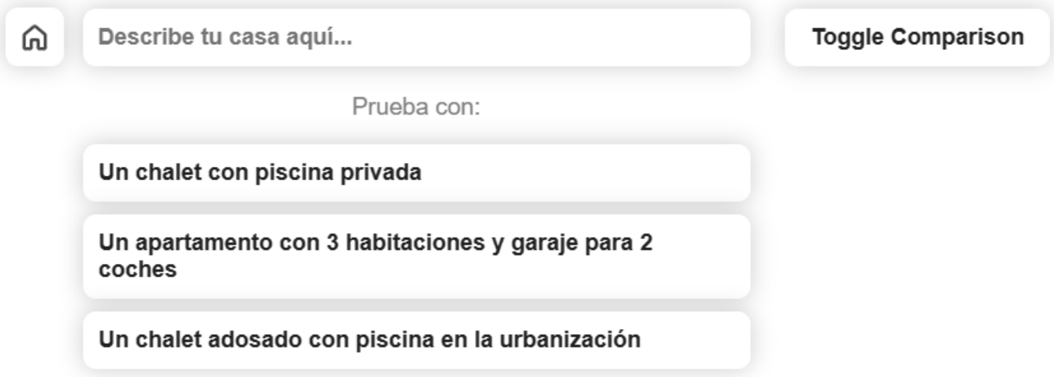

Describe tu casa aquí...
Toggle Comparison
Prueba con:
Un chalet con piscina privada
Un apartamento con 3 habitaciones y garaje para 2 coches
Un chalet adosado con piscina en la urbanización

Figure 3.2: Web application user interface for natural language real estate search.

The client-side `search` function, triggered via the suggested queries or by pressing the Enter key within the input field, transmits a POST request to the designated `/search` API endpoint. This request contains both the user query and the collection name and awaits a JSON-formatted response dictionary. For an API response sample, refer to Figure A.6 in the appendix.

A list of UI property cards is dynamically created with the response, displaying the property data, as shown in Figure A.7. The results appear ordered by relevance to the user query. The Swiper [11] library is utilized for image slideshows. To optimize performance, we employ lazy loading for images, ensuring that they are only loaded when they become visible within the user's viewport. During this process, the UI provides visual indicators to signal loading states and potential errors.

To enable model performance comparison, we utilize a predefined keyword "all" for the collection name in the API request and create a specific UI button labeled "Toggle Comparison". This button triggers the display of a new list of property cards, each created from the results of the baseline model being compared. The cards themselves are aligned between both lists to facilitate direct comparison and easily evaluate the behavior and quality of the embeddings of each model. Figure A.8 displays an example.

Additionally, each property card features 'thumbs up' and 'thumbs down' icons. While currently non-functional, these icons could potentially be used for user feedback to perform A/B testing, gather user appraisals on the relevance of search results, and ultimately enhance our training corpus.

[11] `https://swiperjs.com/` (last accessed on 2024-04-17)

Deployment

Upon completing the development of our web application, our primary objective was to deploy our site ensuring its broad availability. Although local deployments of Milvus, Gradio, and Vite were utilized during our research, evaluation, and development process, creating and managing our own high-availability, secure and reliable server infrastructure is a task that falls outside the scope of this work. To overcome this, we opted for a distributed deployment strategy, leveraging the unique advantages of different platforms and capitalizing on the free-tier offerings of various platform-as-a-service companies. This approach, though it adds a layer of complexity in orchestrating service interactions and managing a distributed architecture, mirrors more closely a real-world deployment scenario, evaluating the scalability and resilience of our system.

We employ Milvus, a high-performance vector database discussed in Section 3.4.6, for storing and retrieving property listing data, including embeddings, via Zilliz Cloud's managed service [12]. The database is populated from our unstructured corpus of properties, which are embedded utilizing two pre-selected models and stored into different collections in our Zilliz serverless cluster, for later retrieval. Property listings and embeddings are uploaded to Zilliz through a *pymilvus* script executed within our development environment.

The back-end API is hosted on Hugging Face Spaces [13], a platform for sharing ML demo apps in the Hugging Face Hub ecosystem. It offers built-in support for Gradio, automatically providing the API endpoint and a web interface for testing from the platform itself. Its Git integration and automated builds upon code updates furthermore facilitated deployment. The selected, most minimal, hardware configuration (2 virtual CPUs, 16GB of RAM, no persistent storage), fulfills the operational requirements of our back-end system, at no cost.

The front-end Vue.js application, responsible for user interaction and presentation of search results, is hosted on Render.com [14] The platform offers a straightforward solution for deploying our front end, including resources for serving the static assets of our app.

To provide open access to the application while safeguarding sensitive code and credentials, we implement a public Hugging Face Space that acts as a gateway to the private API endpoint. This deployment strategy and the interactions between the distributed components are detailed in the sequence diagram shown in Figure 3.3.4.

[12] `https://zilliz.com/cloud` (last accessed on 2024-04-18)
[13] `https://huggingface.co/spaces` (last accessed on 2024-04-18)
[14] `https://render.com/` (last accessed on 2024-04-18)

4 Experiments and Results

This chapter delves into the empirical evaluation of our proposed approach for enhancing real estate search through EBR, domain-specific fine-tuning and a unique dataset created leveraging LLMs.

We present a comprehensive performance analysis of our fine-tuned embedding model, *real-bert*, comparing it with various baseline models to demonstrate the effectiveness of our methodology. Additionally, we investigate the impact of different training and dataset configurations, providing valuable insights into the factors that influence the model's accuracy and generalizability. Through a rigorous evaluation process, we aim to validate the hypothesis that our approach significantly improves the efficiency and intuitiveness of property search within the real estate domain.

4.1. Experimental Setup

This section delves into the experimental framework designed to evaluate the efficacy of our proposed approach for enhancing real estate search through fine-tuned embedding models. It builds upon the structured corpus and query generation processes detailed in Sections 3.4.2 and 3.4.4, respectively. It includes a detailed data preparation step for diversity, an EBR deployment for retrieval using synthetic evaluation queries, and a scoring system based on established metrics like NDCG and F1 score, alongside our novel aNDCG metric (see Section 3.4.6), tailored to address the unique characteristics of our structured corpus and query sets.

4.1.1. Data Preparation

To facilitate a robust evaluation, we partition the property listings from our unstructured corpus (see Section 3.4.1) into an $80\,\%$ training set and a $20\,\%$ test set. The selection of properties was manually revised to ensure a diverse and representative sample for evaluation. The diversity is reflected in the variety of locations, property types, and range of amenities. The $20\,\%$ test set, out of a total of 120 properties, comprises our *evaluation corpus* of 25 properties. It will be employed to assess the comparative performance of each embedding model by conducting EBR and comparing the features of the retrieved results with those specified in the query. This evaluation leverages the previously extracted features of each property in the test set (see Section 3.4.2) to assess the alignment between retrieved properties and the corresponding query, utilizing the QPAS as a measure of similarity. The extracted features were revised to ensure their accuracy and mitigate potential errors.

The queries utilized for evaluation are organized into distinct query sets, as outlined in Section 3.4.4, with each set being independently evaluated. This approach allows to report evaluation scores for each query set individually, providing insights into the strengths and weaknesses of each model in handling specific property features. Our dataset is divided into 307 *basic*, 391 *extended*, and 53 *attribute-specific* queries, among which 13 specify variations of pools and 40 request specific house types (e.g., duplex, penthouse, chalet).

The queries are automatically generated, utilizing the semi-random query generation methodology outlined in Section 3.4.3. We generate 20 queries for each property listing and query set to guarantee enough diversity, all of which are created to align perfectly with the property but keeping random linguistic nuances and feature quantities. Duplicate queries within each set are removed. Although this approach facilitates an accurate and controlled evaluation of the performance of each model, we acknowledge the limitations of procedurally generated queries, which may not fully encompass the variability and complexity inherent in real-world user queries.

Refer to Table B.1 for a concise summary of our evaluation dataset.

4.1.2. Evaluation Pipeline

Upon populating the evaluation corpus and query sets for EBR, we establish an evaluation pipeline to assess the performance of various embedding models within this specific domain and corpus. The evaluation utilizes the same underlying tools and techniques as described in Section 3.4.6, with a local Milvus deployment serving as the vector database and the relational database holding our structured corpus of extracted property features.

For each embedding model under evaluation (see Section 3.4.5), we create a dedicated Milvus collection to store the property representations. Each collection entry comprises a property ID, linking it to the corresponding structured data within our relational database, and the property's embedding vector generated by the respective model.

The core evaluation step involves embedding each query and performing a similarity search within the corresponding Milvus collection for each model. This retrieves the top-10 most similar property ads based on cosine similarity. However, the exact number of retrieved results considered for evaluation varies depending on the specific metric used, as detailed later.

To assess the relevance of retrieved results, we employ QPAS as a measure of similarity between each property-query pair. As explained in Section 3.4.3, QPAS considers the overlap of features present in the query and the corresponding structured data for each retrieved property. These scores serve as the foundation for calculating various evaluation metrics, enabling a comprehensive assessment of both ranking accuracy and retrieval effectiveness.

For each query type and model, the individual query scores obtained from the evaluation metrics are aggregated to provide an overall performance measure. This allows for a comprehensive comparison of model effectiveness across varying levels of query specificity and property feature types. The evaluation results are presented in a table (see Section 4.2.1), highlighting the best-performing model for each query set and showcasing relative scores to demonstrate performance improvements compared to the baseline models.

4.1.3. Evaluation Metrics

This section outlines the evaluation metrics employed to quantify the performance of each embedding model. We combine traditional information retrieval metrics with a novel approach leveraging our unique structured corpus, with the goal of offering a clear framework for interpreting our evaluation results and assessing model strengths and limitations.

We employ two widely used metrics in Information Retrieval: Normalized Discounted Cumulative Gain (NDCG@10) [44] and F1@10 score, as previously introduced in Section 2.3.3. NDCG evaluates the ranking quality of the retrieved results, considering the position and relevance of each property. The F1 score balances precision and recall, offering a comprehensive measure of retrieval effectiveness by considering both result accuracy and the capture of all relevant properties. Both NDCG and F1 scores are calculated using only the top 10 retrieved results, denoted by ”@10”. This cutoff makes the evaluation focus on the most relevant items, aligning more closely to user behavior. In our evaluation, the relevance score for each retrieved property, determined by the QPAS, is a continuous value ranging from 0 to 1. While NDCG directly utilizes this continuous value for ranking assessment, F1 score calculation employs a $0{,}7$ threshold to determine binary relevance (relevant or irrelevant) for precision and recall.

Additionally, our novel aNDCG score leverages our structured corpus for a more nuanced evaluation. By considering the varying number of relevant results for each query, it provides a fairer assessment of retrieval performance compared to traditional NDCG, as detailed in Section 3.4.6.

4.2. Model Comparison and Analysis

Having established a robust evaluation framework, we now delve into the comparative analysis of various embedding models in our task and domain. This section meticulously examines the performance of our fine-tuned model, *real-bert*, against several baseline models, providing insights into the effectiveness of our proposed approach. Additionally, through empirical investigation, we explore the impact of various training and dataset configurations, aiming to identify the optimal settings that enhance accuracy and generalizability within the real estate domain.

4.2.1. Baseline vs. Fine-Tuned Model Performance

To validate the efficacy of our fine-tuning methodology, we compare the performance of *real-bert* against a selection of baseline models, previously discussed in Section 3.4.5. The choice to fine-tune *real-bert* on the *sentence-similarity-spanish-es* model was primarily influenced by its superior performance in this evaluation compared to other baseline models, possibly stemming from its initial advantage of being pre-trained on Spanish data. Table 4.1 presents the results obtained for each model across both basic and extended query types. The evaluation was conducted utilizing the pipeline, dataset, and metrics outlined in Section 4.1.

	Basic			**Extended**		
Model	**aNDCG**	**NDCG@10**	**F1@10**	**aNDCG**	**NDCG@10**	**F1@10**
all-MiniLM-L6-v2	0.54	0.51	0.41	0.55	0.52	0.42
all-MiniLM-L12-v2	0.55	0.52	0.41	0.54	0.51	0.40
multi-qa-distilbert-cos-v1	0.51	0.47	0.37	0.51	0.50	0.41
all-mpnet-base-v2	0.50	0.48	0.38	0.51	0.50	0.39
all-distilroberta-v1	0.52	0.49	0.38	0.57	0.54	0.44
instructor-xl	0.54	0.5	0.38	0.58	0.55	0.44
sentence-similarity-spanish-es	0.51	0.48	0.38	0.57	0.54	0.44
real-bert	**0.78**	**0.69**	**0.63**	**0.75**	**0.69**	**0.62**

Table 4.1: Model performance on *Basic* and *Extended* query types.

Our model, ***real-bert***, outperforms all the base models by a large margin on all metrics and query types. This indicates that our approach to leveraging an unsupervised, automatic feature augmentation process, for creating a fine-tuning dataset allows to greatly improve the quality of the embeddings for real estate search.

The superior performance of *real-bert* is not only reflected in its ability to rank perfect and near-perfect matches more accurately, as evidenced by the higher aNDCG and NDCG@10 scores (up to 41 % and 29 % relative improvement with respect to the best baseline for each query type in terms of aNDCG), but also in its capacity to retrieve relevant properties with a higher degree of precision, as indicated by the F1@10 score.

4.2.2. Fine-Tuning Experiments and Results

To explore the nuances of the fine-tuning process and its impact on model performance, we conducted a series of experiments investigating the influence of various training and dataset configurations.

Dataset Size and Balance

We fine-tuned and evaluated *real-bert* by varying both the number of queries per property and the score balance within the training dataset. Our goal is to identify the optimal dataset size and assess the impact of different score balancing strategies (introduced in Section 3.4.4). A training set of 95 properties and optimal hyperparameters were kept constant throughout this experiment. Figure 4.1 shows model performance according to aNDCG obtained by training on 20, 50, 75, 150, and 250 synthetic basic queries per property, using a normal distribution as *balance*. For testing the *balance*, we consider the following distributions: inverted normal distribution, uniform distribution (flat), increasing function (ramp), and normal distribution, all trained with 150 queries.

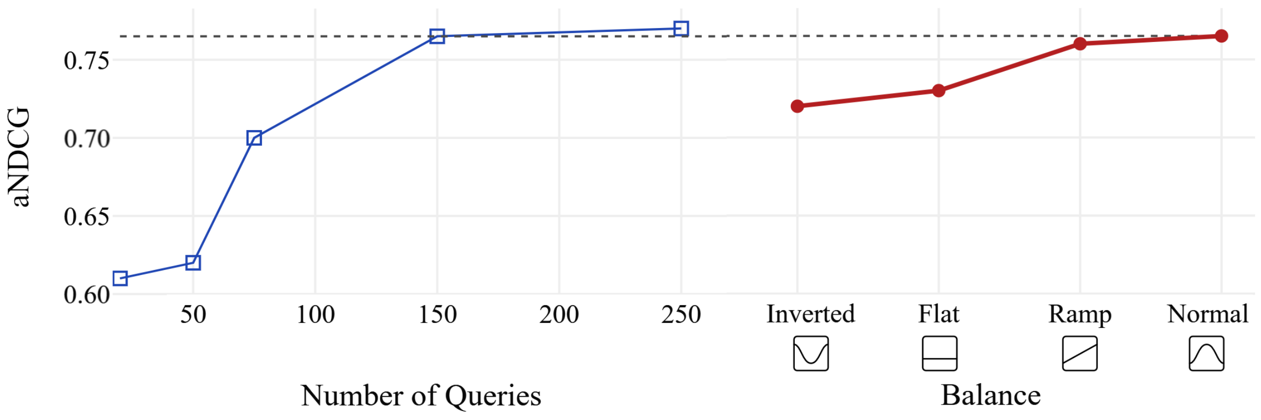


Figure 4.1: Line chart of model performance for varying query numbers (left) and balance types (right). The dotted horizontal line denotes the model using normal balance and 150 queries.

These results evidence that increasing the number of queries beyond 150 does not lead to significant improvement in the aNDCG score, potentially indicating overfitting. With respect to the balance types, the normal distribution balance yields the highest score, likely due to its alignment with real-world query score distributions. In contrast, an inverted balance may introduce biases, whereas a flat distribution assigns equal probability to all data points, failing to capture the natural variability in the data. This suggests that the model learns better from a dataset that matches the actual similarity of properties in the real estate domain, rather than from a dataset that is artificially balanced or skewed.

Epoch Number & Loss Function

To optimize training, we explored the effect of epoch number (1 or 2) and loss function (Cosine Similarity vs. Softmax) on model performance. The training dataset is constructed from 95 properties with normal score distribution, using 300 queries per property for 1 epoch and 150 queries for 2 epochs.

Table 4.2 reveals that training for 2 epochs with Cosine Similarity Loss, particularly with the inclusion of the house-type-query set, yielded the best performance in *basic* and *extended* query evaluations. This suggests that cosine similarity captures the semantic relationships between embeddings more effectively than the softmax function in this task. Regarding the epoch number, the results indicate that training for 2 epochs with cosine similarity yields mixed outcomes: it slightly reduces performance for *pool-specific* queries, maintains high performance for *basic* and *extended* queries, and notably enhances performance by $16\,\%$ in the *house-type* query evaluation.

Hyperparameters	Basic	Extended	Pools	House Type
1 epoch, Softmax Loss	0.68	0.67	**0.83**	N/A
2 epochs, Softmax Loss	0.68	0.65	0.81	N/A
1 epoch, Cosine Similarity	0.76	0.77	0.80	0.42
1 epoch, Cosine Similarity*	0.77	0.76	0.81	0.49
2 epochs, Cosine Similarity*	**0.77**	**0.77**	0.78	**0.58**

*With new house type attribute-specific query set

Table 4.2: aNDCG scores for training on different epoch and loss function combinations.

Attribute-Specific Query Types

We investigate how attribute-specific query types, introduced in Section 3.4.4, affect model understanding of features such as pool presence and variations (e.g., community, private), and house types (e.g., duplex, penthouse, chalet). The proportion of attribute-specific queries in the training dataset is $3{,}72\,\%$ and $11{,}45\,\%$, for pool and house type, respectively. Figure 4.2 illustrates the impact of incorporating these queries on model performance, evaluating the model both on the *extended* and *attribute-specific* query evaluation sets. Initial fine-tuning including pool-specific queries led to improved performance in both *extended* and *pools* queries, with scores up to $0{,}76$ and $0{,}8$, respectively. Adding house-type-specific queries and extending training to 2 epochs enhanced *house type* performance by $32\,\%$ and *extended* query performance by $1\,\%$. These results highlight the effectiveness of this targeted training approach to improve specific attribute recognition while maintaining a robust overall model understanding, as evidenced by sustained performance on the *extended* queries evaluation. However, the inclusion of house-type-specific queries slightly reduced *pools* performance, underscoring the importance of balanced training for uniform improvements across query types.

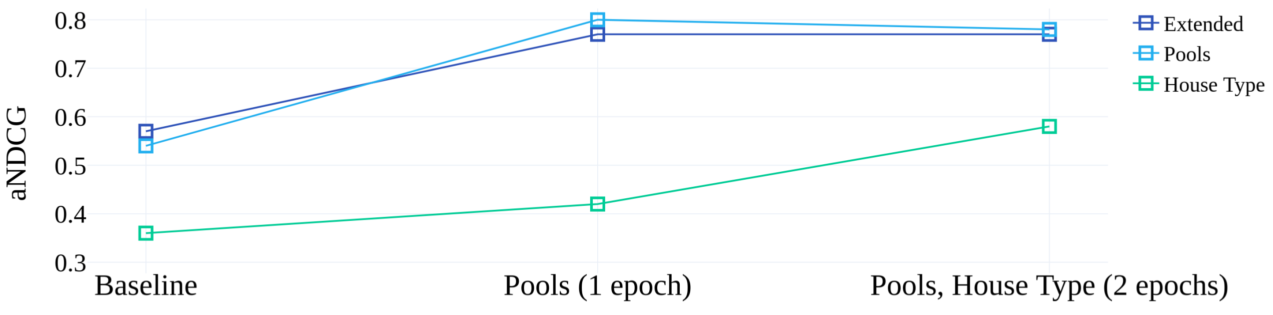


Figure 4.2: Impact of Fine-Tuning Configurations on aNDCG Scores.

5 Conclusions and Future Work

5.1. Conclusions

This thesis aimed to improve real estate search by applying neural information retrieval techniques, specifically embedding-based retrieval with a fine-tuned BERT model named *real-bert*. A novel dataset was created using the Llama2 large language model (LLM) to extract features from property listings and generate synthetic queries. The research involved data preparation, feature extraction, dataset creation, model fine-tuning, evaluation, and the development of a demo web application for user interaction.

The evaluation results demonstrably confirm the effectiveness of our approach. Our fine-tuned embedding model significantly outperformed all selected top-performing BERT models across all evaluation metrics and improved baseline model performance (by $41\,\%$, $35\,\%$, and $49\,\%$ in aNDCG, NDCG@10 and F1@10, respectively). Our model showcases its superior understanding of property features and user requests through its ability to both retrieve and rank relevant properties.

Our dataset generation approach combined organic real estate listings with synthetic queries based on extracted features. Our evaluation demonstrates the effectiveness of this approach, which harnesses the richness and diversity of real-world data while providing greater control, flexibility and lowering the likelihood of errors, in comparison to directly generating dataset text using an LLM. Llama2, even in its smaller, quantized version, proved to be a valuable tool for automated feature extraction, achieving high accuracy for most features (with error rates below $1\,\%$ for basic attributes and 4-$7\,\%$ for amenities) after several iterations of prompt refinement. Additionally, fine-tuning experiments revealed that a normal score distribution and 150 queries per property were ideal. Cosine similarity loss and the use of attribute-specific queries, strategically with a higher epoch number, further optimized performance.

Our demo public web application was deployed to tfg-embeddings-static.onrender.com, showcasing the intuitive nature of the proposed methodology for real estate search, utilizing document samples not used in training. It uses our API at huggingface.co/spaces/Onegafer/tfg-embeddings-public, featuring a fast and responsive user interface that allows users to directly compare search results.

Despite this significant progress, some limitations and challenges remain. The effectiveness of the proposed system relies on the quality of textual property descriptions. Incomplete or poorly written descriptions can hinder accurate retrieval and matching of properties, potentially leading to suboptimal search results. Additionally, these descriptions also limit the accuracy of the feature extraction process. Factual errors, colloquial language and regional dialects can lead to misinterpretations by the LLM.

Synthetic queries used for evaluation and training, though controlled, may not fully capture real user search complexities. Real-world queries often include implicit preferences and subjective language, hard to mimic synthetically. This impacts the ability of the model to interpret implicit preferences, such as stylistic choices or neighborhood characteristics, due to their absence in the fine-tuning dataset.

Our extensive fine-tuning experiments revealed that enhancing our training dataset with a single attribute-specific query set greatly increased performance for that attribute. However, when the model was trained on multiple query sets, attribute-specific evaluation scores slightly decreased. This suggests potential limitations in the capacity of the model architecture or the training process. Moreover, the generalizability of our model to other languages or regions remains an open question. The model potentially requires adaptation and retraining to ensure effectiveness in diverse real estate markets.

Furthermore, while we have addressed potential biases and ethical concerns in our data collection processes, we acknowledge that training data (for all *real-bert*, its baseline model, and Llama2) may contain inherent societal biases or reflect historical discrimination in the housing market.

5.2. Future Work

Future work will focus on enhancing the system's ability to further understand implicit user preferences, improve the dataset quality, and explore the use of latest-generation language and vision models.

Enhancing our training and evaluation datasets could involve using real user search data, A/B testing and RLHF [26] to incorporate explicit human feedback for more realistic and diverse data. Furthermore, expanding query generation to include subjective attributes and property descriptors will improve the understanding and generalizability of the model.

Future work could explore recent embedding models such as E5 and its multilingual variant [66, 67]. Similarly, other state-of-the-art LLMs, released after the research conducted in this study (e.g., Llama3 [68]) and latest generation Small Language Models (e.g., Phi-3 [69]) could enhance feature extraction performance. This could unlock more complex features (e.g., property condition, subjective descriptions, neighborhood ambience). Integrating visual feature extraction from property images using vision models (e.g., LLaVa [70]) could provide valuable complementary data. These advancements present promising future work avenues.

Potential web application enhancements include a hybrid retrieval with filter-based search. A recommendation system leveraging our pipeline could offer user-specific suggestions. To mitigate potential biases, the use of NLP (e.g., labeling, sentiment analysis), and fairness-aware ranking algorithms [71] could aim to enhance user experience to ensure unbiased and inclusive search results.

Bibliography

[1] A. S. Geoff Boeing, Max Besbris and J. Kuk, “Housing search in the age of big data: Smarter cities or the same old blind spots?,” *Housing Policy Debate*, vol. 31, no. 1, pp. 112–126, 2021.

[2] D. J. Simon and B. L. Bateman, “HOMLIST: a computerized real estate information retrieval system,” in *American Federation of Information Processing Societies: AFIPS Conference Proceedings: 1972 Spring Joint Computer Conference, Atlantic City, NJ, USA, May 16-18, 1972*, vol. 40 of *AFIPS Conference Proceedings*, pp. 541–544, AFIPS, 1972.

[3] D. K. Hartz Jr., M. T. Gorman, E. Rossum, and R. Barney, “Real-estate information search and retrieval system,” 2001.

[4] D. Grant and E. Cherif, “Using design science to improve web search innovation in real estate,” *J. Organ. Comput. Electron. Commer.*, vol. 26, no. 3, pp. 267–284, 2016.

[5] G. Boeing, M. Besbris, A. Schachter, and J. Kuk, “Housing search in the age of big data: Smarter cities or the same old blind spots?,” *Housing Policy Debate*, vol. 31, no. 1, pp. 112–126, 2021.

[6] K. D. Onal, Y. Zhang, I. S. Altingovde, M. M. Rahman, P. Karagoz, A. Braylan, B. Dang, H. Chang, H. Kim, Q. McNamara, A. Angert, E. Banner, V. Khetan, T. McDonnell, A. T. Nguyen, D. Xu, B. C. Wallace, M. de Rijke, and M. Lease, “Neural information retrieval: at the end of the early years,” *Inf. Retr. J.*, vol. 21, no. 2-3, pp. 111–182, 2018.

[7] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, “Neural architectures for named entity recognition,” in *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016* (K. Knight, A. Nenkova, and O. Rambow, eds.), pp. 260–270, The Association for Computational Linguistics, 2016.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA* (I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, eds.), pp. 5998–6008, 2017.

[9] B. Mitra, F. Diaz, and N. Craswell, “Learning to match using local and distributed representations of text for web search,” in *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017* (R. Barrett, R. Cummings, E. Agichtein, and E. Gabrilovich, eds.), pp. 1291–1299, ACM, 2017.

[10] H. Zhang, S. Wang, K. Zhang, Z. Tang, Y. Jiang, Y. Xiao, W. Yan, and W. Yang, “Towards personalized and semantic retrieval: An end-to-end solution for e-commerce search via embedding learning,” in *Proceedings of the 43rd International ACM SIGIR conference on research and deve-*

lopment in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020 (J. X. Huang, Y. Chang, X. Cheng, J. Kamps, V. Murdock, J. Wen, and Y. Liu, eds.), pp. 2407–2416, ACM, 2020.

[11] J. Huang, A. Sharma, S. Sun, L. Xia, D. Zhang, P. Pronin, J. Padmanabhan, G. Ottaviano, and L. Yang, "Embedding-based retrieval in facebook search," in *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020* (R. Gupta, Y. Liu, J. Tang, and B. A. Prakash, eds.), pp. 2553–2561, ACM, 2020.

[12] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations* (K. Bontcheva and J. Zhu, eds.), (Baltimore, Maryland), pp. 55–60, Association for Computational Linguistics, June 2014.

[13] E. H. Shortliffe, "Mycin: A knowledge-based computer program applied to infectious diseases," *Proceedings of the Annual Symposium on Computer Application in Medical Care*, pp. 66–9, Oct 1977.

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[15] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, (Red Hook, NY, USA), pp. 6000–6010, Curran Associates Inc., 2017.

[18] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," in *Proceedings of the 40th International Conference on Machine Learning*, ICML'23, JMLR.org, 2023.

[19] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)* (J. Burstein, C. Doran, and T. Solorio, eds.), pp. 4171–4186, Association for Computational Linguistics, 2019.

[20] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, (Red Hook, NY, USA), Curran Associates Inc., 2020.

[21] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Chris-

tiano, J. Leike, and R. Lowe, “Training language models to follow instructions with human feedback,” *arXiv e-prints*, p. arXiv:2203.02155, Mar. 2022.

[22] OpenAI, “GPT-4 Technical Report,” *arXiv e-prints*, p. arXiv:2303.08774, Mar. 2023.

[23] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom, “Llama 2: Open foundation and fine-tuned chat models,” 2023.

[24] R. Anil, A. M. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, S. Shakeri, E. Taropa, P. Bailey, Z. Chen, E. Chu, J. H. Clark, L. El Shafey, Y. Huang, K. Meier-Hellstern, G. Mishra, E. Moreira, M. Omernick, K. Robinson, S. Ruder, Y. Tay, K. Xiao, Y. Xu, Y. Zhang, G. Hernandez Abrego, J. Ahn, J. Austin, P. Barham, J. Botha, J. Bradbury, S. Brahma, K. Brooks, M. Catasta, Y. Cheng, C. Cherry, C. A. Choquette-Choo, A. Chowdhery, C. Crepy, S. Dave, M. Dehghani, S. Dev, J. Devlin, M. Díaz, N. Du, E. Dyer, V. Feinberg, F. Feng, V. Fienber, M. Freitag, X. Garcia, S. Gehrmann, L. Gonzalez, G. Gur-Ari, S. Hand, H. Hashemi, L. Hou, J. Howland, A. Hu, J. Hui, J. Hurwitz, M. Isard, A. Ittycheriah, M. Jagielski, W. Jia, K. Kenealy, M. Krikun, S. Kudugunta, C. Lan, K. Lee, B. Lee, E. Li, M. Li, W. Li, Y. Li, J. Li, H. Lim, H. Lin, Z. Liu, F. Liu, M. Maggioni, A. Mahendru, J. Maynez, V. Misra, M. Moussalem, Z. Nado, J. Nham, E. Ni, A. Nystrom, A. Parrish, M. Pellat, M. Polacek, A. Polozov, R. Pope, S. Qiao, E. Reif, B. Richter, P. Riley, A. Castro Ros, A. Roy, B. Saeta, R. Samuel, R. Shelby, A. Slone, D. Smilkov, D. R. So, D. Sohn, S. Tokumine, D. Valter, V. Vasudevan, K. Vodrahalli, X. Wang, P. Wang, Z. Wang, T. Wang, J. Wieting, Y. Wu, K. Xu, Y. Xu, L. Xue, P. Yin, J. Yu, Q. Zhang, S. Zheng, C. Zheng, W. Zhou, D. Zhou, S. Petrov, and Y. Wu, “PaLM 2 Technical Report,” *arXiv e-prints*, p. arXiv:2305.10403, May 2023.

[25] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, “Scaling Laws for Neural Language Models,” *arXiv e-prints*, p. arXiv:2001.08361, Jan. 2020.

[26] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, “Deep reinforcement learning from human preferences,” in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[27] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, “Finetuned Language Models Are Zero-Shot Learners,” *arXiv e-prints*, p. arXiv:2109.01652, Sept. 2021.

[28] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh,

D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, “Language models are few-shot learners,” in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS’20, (Red Hook, NY, USA), Curran Associates Inc., 2020.

[29] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models,” *arXiv e-prints*, p. arXiv:2201.11903, Jan. 2022.

[30] R. Pradeep, S. Sharifymoghaddam, and J. Lin, “RankVicuna: Zero-Shot Listwise Document Reranking with Open-Source Large Language Models,” *arXiv e-prints*, p. arXiv:2309.15088, Sept. 2023.

[31] J. Ye, J. Gao, Q. Li, H. Xu, J. Feng, Z. Wu, T. Yu, and L. Kong, “Zerogen: Efficient zero-shot learning via dataset generation,” in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022* (Y. Goldberg, Z. Kozareva, and Y. Zhang, eds.), pp. 11653–11669, Association for Computational Linguistics, 2022.

[32] Y. Yu, Y. Zhuang, J. Zhang, Y. Meng, A. Ratner, R. Krishna, J. Shen, and C. Zhang, “Large language model as attributed training data generator: A tale of diversity and bias,” in *Thirty-Seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

[33] J. Ye, J. Gao, Q. Li, H. Xu, J. Feng, Z. Wu, T. Yu, and L. Kong, “Zerogen: Efficient zero-shot learning via dataset generation,” in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022* (Y. Goldberg, Z. Kozareva, and Y. Zhang, eds.), pp. 11653–11669, Association for Computational Linguistics, 2022.

[34] Z. Liu, B. Oguz, C. Zhao, E. Chang, P. Stock, Y. Mehdad, Y. Shi, R. Krishnamoorthi, and V. Chandra, “LLM-QAT: Data-Free Quantization Aware Training for Large Language Models,” *arXiv e-prints*, p. arXiv:2305.17888, May 2023.

[35] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, “OPTQ: accurate quantization for generative pre-trained transformers,” in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, OpenReview.net, 2023.

[36] K. D. Onal, Y. Zhang, I. S. Altingovde, M. M. Rahman, P. Karagoz, A. Braylan, B. Dang, H.-L. Chang, H. Kim, Q. McNamara, A. Angert, E. Banner, V. Khetan, T. McDonnell, A. T. Nguyen, D. Xu, B. C. Wallace, M. de Rijke, and M. Lease, “Neural information retrieval: at the end of the early years,” *Information Retrieval Journal*, vol. 21, no. 2, pp. 111–182, 2018.

[37] W.-C. Chang, F. Yu, Y.-W. Chang, and S. Kumar, “Pre-training tasks for embedding-based large-scale retrieval,” in *ICLR*, 2020.

[38] R. Jha, S. Subramaniyam, E. Benjamin, and T. Taula, “Unified embedding based personalized retrieval in etsy search,” *ArXiv*, vol. abs/2306.04833, 2023.

[39] J. Shi, V. Chaurasiya, Y. Liu, S. Vij, Y. Wu, S. Kanduri, N. Shah, P. Yu, N. Srivastava, L. Shi, G. Ven-

kataraman, and J. Yu, “Embedding based retrieval in friend recommendation,” in *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’23, (New York, NY, USA), pp. 3330–3334, Association for Computing Machinery, 2023.

[40] Y. Gan, Y. Ge, C. Zhou, S. Su, Z. Xu, X. Xu, Q. Hui, X. Chen, Y. Wang, and Y. Shan, “Binary embedding-based retrieval at tencent,” in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD ’23, (New York, NY, USA), pp. 4056–4067, Association for Computing Machinery, 2023.

[41] J.-T. Huang, A. Sharma, S. Sun, L. Xia, D. Zhang, P. Pronin, J. Padmanabhan, G. Ottaviano, and L. Yang, “Embedding-based retrieval in facebook search,” in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD ’20, (New York, NY, USA), pp. 2553–2561, Association for Computing Machinery, 2020.

[42] N. Reimers and I. Gurevych, “Sentence-BERT: Sentence embeddings using Siamese BERT-networks,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (K. Inui, J. Jiang, V. Ng, and X. Wan, eds.), (Hong Kong, China), pp. 3982–3992, Association for Computational Linguistics, Nov. 2019.

[43] N. Muennighoff, N. Tazi, L. Magne, and N. Reimers, “Mteb: Massive text embedding benchmark,” *arXiv preprint arXiv:2210.07316*, 2022.

[44] K. Järvelin and J. Kekäläinen, “Ir evaluation methods for retrieving highly relevant documents,” in *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’00, (New York, NY, USA), pp. 41–48, Association for Computing Machinery, 2000.

[45] A. Gharahighehi, K. Pliakos, and C. Vens, “Recommender systems in the real estate marketâ€”a survey,” *Applied Sciences*, vol. 11, no. 16, 2021.

[46] R. Burke, K. Hammond, and B. Young, “Knowledge-based navigation of complex information spaces,” in *Proceedings of the National Conference on Artificial Intelligence* (Anon, ed.), vol. 1, pp. 462–468, AAAI, Dec. 1996.

[47] H. J. Jun, J. H. Kim, D. Y. Rhee, and S. W. Chang, ““seoulhouse2vec”: An embedding-based collaborative filtering housing recommender system for analyzing housing preference,” *Sustainability*, vol. 12, no. 17, 2020.

[48] K. Milkovich, S. Shirur, P. Desai, L. Manjunath, and W. Wu, “Zenden-a personalized house searching application,” in *2020 IEEE Sixth International Conference on Big Data Computing Service and Applications (BigDataService)*, (Oxford, UK), pp. 173–178, IEEE, 13–16 April 2020.

[49] R. Kabir, B. Pervaiz, T. Khan, A. Ul-Hasan, R. Nawaz, and F. Shafait, “Deeprank: Adapting neural tensor networks for ranking the recommendations,” in *Proceedings of the Mediterranean Conference on Pattern Recognition and Artificial Intelligence*, (Istanbul, Turkey), pp. 162–176, Springer: Berlin/Heidelberg, Germany, 2019.

[50] T. Badriyah, S. Azvy, W. Yuwono, and I. Syarif, "Recommendation system for property search using content based filtering method," in *2018 International Conference on Information and Communications Technology (ICOIACT)*, (Yogyakarta, Indonesia), pp. 25–29, IEEE, 6-7 March 2018.

[51] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "ELECTRA: Pre-training text encoders as discriminators rather than generators," in *ICLR*, 2020.

[52] A. Conneau, R. Rinott, G. Lample, A. Williams, S. R. Bowman, H. Schwenk, and V. Stoyanov, "Xnli: Evaluating cross-lingual sentence representations," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2018.

[53] S. Gunasekar, Y. Zhang, J. Aneja, C. C. T. Mendes, A. D. Giorno, S. Gopi, M. Javaheripi, P. Kauffmann, G. de Rosa, O. Saarikivi, A. Salim, S. Shah, H. S. Behl, X. Wang, S. Bubeck, R. Eldan, A. T. Kalai, Y. T. Lee, and Y. Li, "Textbooks are all you need," 2023.

[54] G. Penedo, Q. Malartic, D. Hesslow, R. Cojocaru, H. Alobeidli, A. Cappelli, B. Pannier, E. Almazrouei, and J. Launay, "The refinedweb dataset for falcon LLM: Outperforming curated corpora with web data only," in *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

[55] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung, "Survey of hallucination in natural language generation," *ACM Comput. Surv.*, vol. 55, mar 2023.

[56] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," *arXiv preprint arXiv:1508.05326*, 2015.

[57] D. Kingma and L. Ba, "Adam: A method for stochastic optimization," in *ICLR 2015: Accepted Papers - Main Conference - Poster Presentations*, (Ithaca, NY), p. 13, ArXiv, 2015. Contribution to International Conference on Learning Representations, May 7-9, 2015, San Diego. Also earlier and later text versions at arXiv.org.

[58] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, "Minilm: deep self-attention distillation for task-agnostic compression of pre-trained transformers," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, (Red Hook, NY, USA), Curran Associates Inc., 2020.

[59] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," *arXiv e-prints*, p. arXiv:1910.01108, Oct. 2019.

[60] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu, "MPNet: Masked and Permuted Pre-training for Language Understanding," *arXiv e-prints*, p. arXiv:2004.09297, Apr. 2020.

[61] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," *arXiv e-prints*, p. arXiv:1910.01108, Oct. 2019.

[62] H. Su, W. Shi, J. Kasai, Y. Wang, Y. Hu, M. Ostendorf, W.-t. Yih, N. A. Smith, L. Zettlemoyer, and T. Yu, "One Embedder, Any Task: Instruction-Finetuned Text Embeddings," *arXiv e-prints*, p. arXiv:2212.09741, Dec. 2022.

[63] J. Cañete, G. Chaperon, R. Fuentes, J.-H. Ho, H. Kang, and J. Pérez, "Spanish Pre-trained BERT

Model and Evaluation Data," *arXiv e-prints*, p. arXiv:2308.02976, Aug. 2023.

[64] P. May, "Machine translated multilingual sts benchmark dataset.," 2021.

[65] A. Abid, A. Abdalla, A. Abid, D. Khan, A. Alfozan, and J. Zou, "Gradio: Hassle-Free Sharing and Testing of ML Models in the Wild," *arXiv e-prints*, p. arXiv:1906.02569, June 2019.

[66] L. Wang, N. Yang, X. Huang, B. Jiao, L. Yang, D. Jiang, R. Majumder, and F. Wei, "Text embeddings by weakly-supervised contrastive pre-training," *arXiv preprint arXiv:2212.03533*, 2022.

[67] L. Wang, N. Yang, X. Huang, L. Yang, R. Majumder, and F. Wei, "Multilingual e5 text embeddings: A technical report," *arXiv preprint arXiv:2402.05672*, 2024.

[68] AI@Meta, "Llama 3 model card," 2024.

[69] M. Abdin, S. A. Jacobs, A. A. Awan, J. Aneja, A. Awadallah, H. Awadalla, N. Bach, A. Bahree, A. Bakhtiari, H. Behl, A. Benhaim, M. Bilenko, J. Bjorck, S. Bubeck, M. Cai, C. C. T. Mendes, W. Chen, V. Chaudhary, P. Chopra, A. D. Giorno, G. de Rosa, M. Dixon, R. Eldan, D. Iter, A. Garg, A. Goswami, S. Gunasekar, E. Haider, J. Hao, R. J. Hewett, J. Huynh, M. Javaheripi, X. Jin, P. Kauffmann, N. Karampatziakis, D. Kim, M. Khademi, L. Kurilenko, J. R. Lee, Y. T. Lee, Y. Li, C. Liang, W. Liu, E. Lin, Z. Lin, P. Madan, A. Mitra, H. Modi, A. Nguyen, B. Norick, B. Patra, D. Perez-Becker, T. Portet, R. Pryzant, H. Qin, M. Radmilac, C. Rosset, S. Roy, O. Ruwase, O. Saarikivi, A. Saied, A. Salim, M. Santacroce, S. Shah, N. Shang, H. Sharma, X. Song, M. Tanaka, X. Wang, R. Ward, G. Wang, P. Witte, M. Wyatt, C. Xu, J. Xu, S. Yadav, F. Yang, Z. Yang, D. Yu, C. Zhang, C. Zhang, J. Zhang, L. L. Zhang, Y. Zhang, Y. Zhang, Y. Zhang, and X. Zhou, "Phi-3 technical report: A highly capable language model locally on your phone," 2024.

[70] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," 2023.

[71] S. C. Geyik, S. Ambler, and K. Kenthapadi, "Fairness-aware ranking in search & recommendation systems with application to linkedin talent search," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019* (A. Teredesai, V. Kumar, Y. Li, R. Rosales, E. Terzi, and G. Karypis, eds.), pp. 2221–2231, ACM, 2019.

Acronyms

AI Artificial Intelligence.

aNDCG adaptive-NDCG.

CLI Command-Line Interface.

EBR Embedding-based Retrieval.

ESM ES Modules.

HMR Hot Module Replacement.

LLM Large Language Model.

LLMs Large Language Models.

LSTM Long Short-Term Memory.

MTEB Massive Text Embedding Benchmark.

NDCG Normalized Discounted Cumulative Gain.

Neu-IR Neural Information Retrieval.

NLP Natural Language Processing.

QPAS Query-Property Alignment Score.

RNNs Recurrent Neural Networks.

SBERT Sentence-BERT.

SNLI Stanford Natural Language Inference.

STSb STS Benchmark.

UI User Interface.

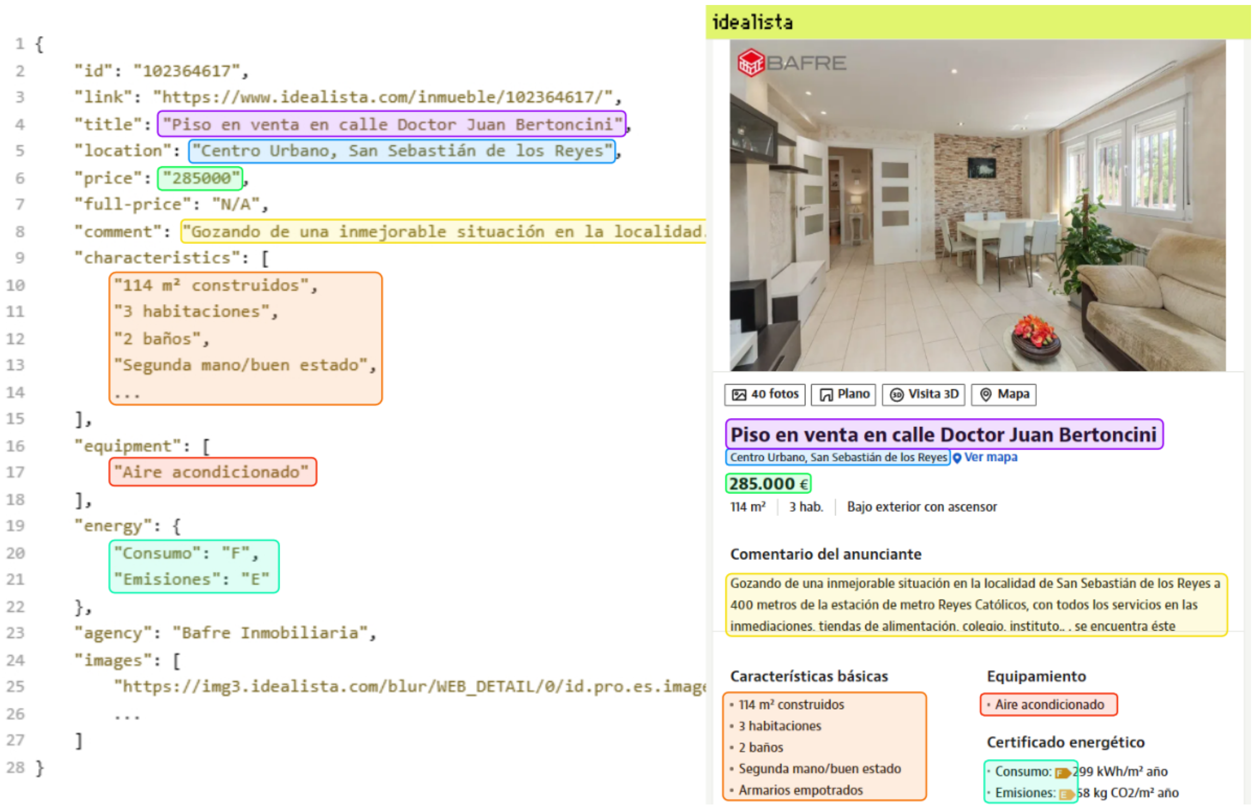
Appendices

A

Implementation Details

A.1. Dataset Creation

A.1.1. Raw Data Extraction



```
{
    "id": "102364617",
    "link": "https://www.idealista.com/inmueble/102364617/",
    "title": "Piso en venta en calle Doctor Juan Bertoncini",
    "location": "Centro Urbano, San Sebastián de los Reyes",
    "price": "285000",
    "full-price": "N/A",
    "comment": "Gozando de una inmejorable situación en la localidad
    "characteristics": [
        "114 m² construidos",
        "3 habitaciones",
        "2 baños",
        "Segunda mano/buen estado",
        ...
    ],
    "equipment": [
        "Aire acondicionado"
    ],
    "energy": {
        "Consumo": "F",
        "Emisiones": "E"
    },
    "agency": "Bafre Inmobiliaria",
    "images": [
        "https://img3.idealista.com/blur/WEB_DETAIL/0/id.pro.es.image
        ...
    ]
}
```

idealista

40 fotos | Plano | Visita 3D | Mapa

Piso en venta en calle Doctor Juan Bertoncini

Centro Urbano, San Sebastián de los Reyes Ver mapa

285.000 €

114 m² | 3 hab. | Bajo exterior con ascensor

Comentario del anunciante

Gozando de una inmejorable situación en la localidad de San Sebastián de los Reyes a 400 metros de la estación de metro Reyes Católicos, con todos los servicios en las inmediaciones. tiendas de alimentación. colegio. instituto.. . se encuentra éste

Características básicas

- 114 m² construidos
- 3 habitaciones
- 2 baños
- Segunda mano/buen estado
- Armarios empotrados

Equipamiento

- Aire acondicionado

Certificado energético

- Consumo: F 299 kWh/m² año
- Emisiones: E 58 kg CO2/m² año

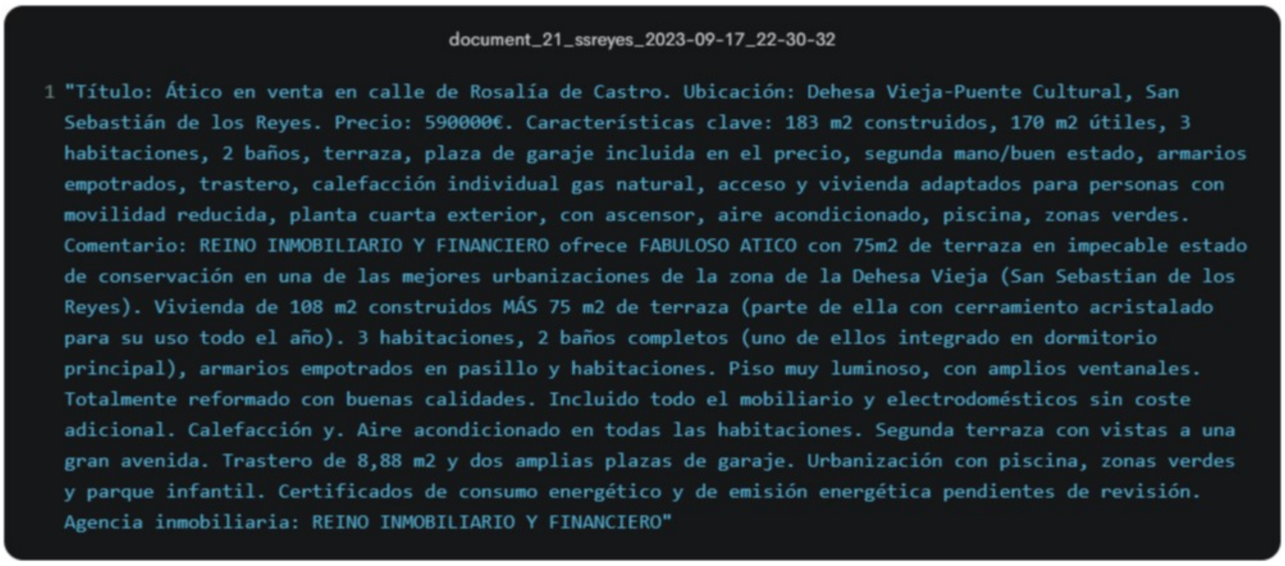
Figure A.1: Sample of JSON-formatted data collected from Idealista, compared directly with the original property listing.

A.1.2. Summarization Category Examples

Category	Relevance
Descripción de la propiedad: tipo, tamaño, metros cuadrados (m2), plantas, reforma, dormitorios, plantas, etc.	Relevant
Características de la urbanización: piscina, pista de pádel, zona infantil, zonas comunes, zonas verdes, seguridad, etc.	Relevant
Beneficios de la zona y servicios de alrededor	Not relevant
Servicios y ocio en la zona: centros comerciales, médicos, farmacia, deportivos, culturales, educación, universidad, restaurantes cercanos, etc.	Not relevant
Información de la inmobiliaria: web, ubicación, etc.	Not relevant

Table A.1: Sample categories for extractive text summarization.

A.1.3. Sample Document from Unstructured Corpus

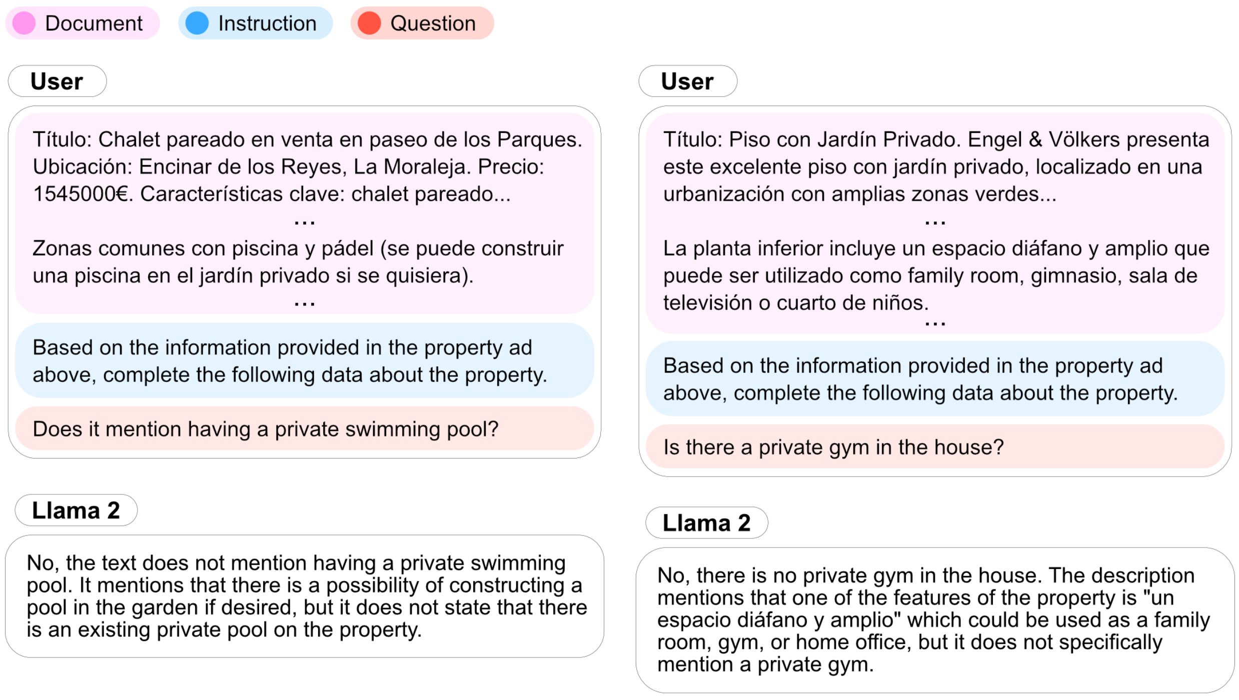


document_21_ssreyes_2023-09-17_22-30-32

```
"Título: Ático en venta en calle de Rosalía de Castro. Ubicación: Dehesa Vieja-Puente Cultural, San
Sebastián de los Reyes. Precio: 590000€. Características clave: 183 m2 construidos, 170 m2 útiles, 3
habitaciones, 2 baños, terraza, plaza de garaje incluida en el precio, segunda mano/buen estado, armarios
empotrados, trastero, calefacción individual gas natural, acceso y vivienda adaptados para personas con
movilidad reducida, planta cuarta exterior, con ascensor, aire acondicionado, piscina, zonas verdes.
Comentario: REINO INMOBILIARIO Y FINANCIERO ofrece FABULOSO ATICO con 75m2 de terraza en impecable estado
de conservación en una de las mejores urbanizaciones de la zona de la Dehesa Vieja (San Sebastian de los
Reyes). Vivienda de 108 m2 construidos MÁS 75 m2 de terraza (parte de ella con cerramiento acristalado
para su uso todo el año). 3 habitaciones, 2 baños completos (uno de ellos integrado en dormitorio
principal), armarios empotrados en pasillo y habitaciones. Piso muy luminoso, con amplios ventanales.
Totalmente reformado con buenas calidades. Incluido todo el mobiliario y electrodomésticos sin coste
adicional. Calefacción y. Aire acondicionado en todas las habitaciones. Segunda terraza con vistas a una
gran avenida. Trastero de 8,88 m2 y dos amplias plazas de garaje. Urbanización con piscina, zonas verdes
y parque infantil. Certificados de consumo energético y de emisión energética pendientes de revisión.
Agencia inmobiliaria: REINO INMOBILIARIO Y FINANCIERO"
```

Figure A.2: Document Sample.

A.1.4. Llama2 Potential Misconception Samples



● Document ● Instruction ● Question

User

Título: Chalet pareado en venta en paseo de los Parques. Ubicación: Encinar de los Reyes, La Moraleja. Precio: 1545000€. Características clave: chalet pareado...
...
Zonas comunes con piscina y pádel (se puede construir una piscina en el jardín privado si se quisiera).
...

Based on the information provided in the property ad above, complete the following data about the property.

Does it mention having a private swimming pool?

Llama 2

No, the text does not mention having a private swimming pool. It mentions that there is a possibility of constructing a pool in the garden if desired, but it does not state that there is an existing private pool on the property.

User

Título: Piso con Jardín Privado. Engel & Völkers presenta este excelente piso con jardín privado, localizado en una urbanización con amplias zonas verdes...
...
La planta inferior incluye un espacio diáfano y amplio que puede ser utilizado como family room, gimnasio, sala de televisión o cuarto de niños.
...

Based on the information provided in the property ad above, complete the following data about the property.

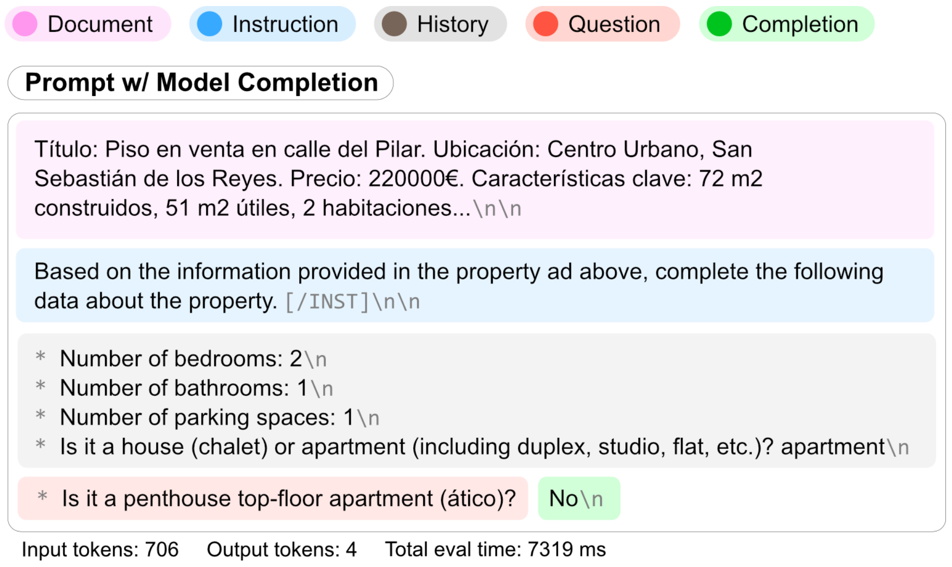
Is there a private gym in the house?

Llama 2

No, there is no private gym in the house. The description mentions that one of the features of the property is "un espacio diáfano y amplio" which could be used as a family room, gym, or home office, but it does not specifically mention a private gym.

Figure A.3: Llama2 addresses potential misconceptions effectively.

A.1.5. Llama2 Feature Extraction Prompt String Sample



● Document ● Instruction ● History ● Question ● Completion

Prompt w/ Model Completion

Título: Piso en venta en calle del Pilar. Ubicación: Centro Urbano, San Sebastián de los Reyes. Precio: 220000€. Características clave: 72 m2 construidos, 51 m2 útiles, 2 habitaciones...\n\n

Based on the information provided in the property ad above, complete the following data about the property. [/INST]\n\n

* Number of bedrooms: 2\n
* Number of bathrooms: 1\n
* Number of parking spaces: 1\n
* Is it a house (chalet) or apartment (including duplex, studio, flat, etc.)? apartment\n

* Is it a penthouse top-floor apartment (ático)? No\n

Input tokens: 706 Output tokens: 4 Total eval time: 7319 ms

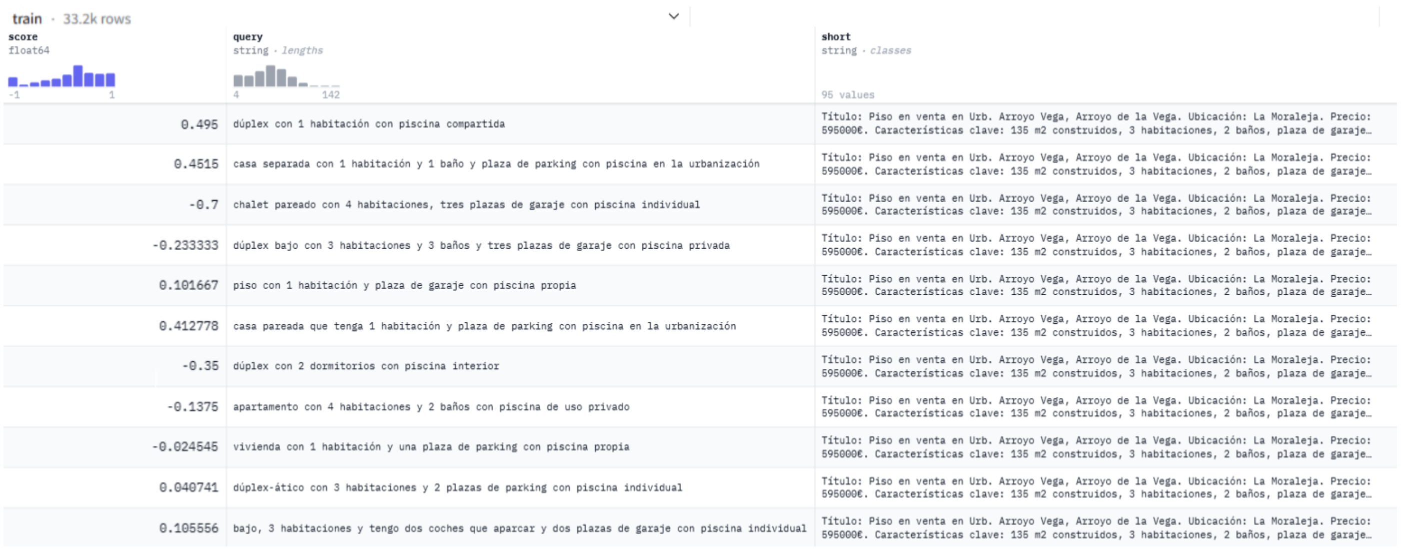
Figure A.4: Llama2 QA Prompt Example with Model Completion.

A.1.6. Revised Question Samples for Feature Extraction

ID	Type	Question	Dependencies	Error Rate
Q1	Integer	“Number of bedrooms:”	N/A	0-1 %
Q2	Integer	“Number of bathrooms:”	N/A	0-1 %
Q3	Multiple-choice	“Is it a house (chalet) or apartment (including duplex, studio, flat, etc.)?”	N/A	0-1 %
Q4	Yes/No	“Is it a semi-detached (adosado or pareado) house (yes/no)?”	Q3 = House	0-1 %
Q5	Yes/No	“Is it a penthouse top-floor apartment (ático)?”	Q3 = Apartment	5 %
Q6	Yes/No	“Is the apartment a duplex (2-stories)?”	Q3 = Apartment	5 %
Q7	Yes/No	“Is the apartment multi-story (more than 2 stories)?”	Q3 = Apartment, Q6 = No	5 %
Q8	Yes/No	“Does it have a garden, or a plot of land?”	N/A	3-4 %
Q9	Yes/No	“Does it mention having a private swimming pool?”	Q8 = Yes	6 %
Q10	Yes/No	“Does it have a pool in the zona comunitaria, urbanización or conjunto residencial?”	Q9 = No	0-1 %
Q11	Yes/No	“Does it have an elevator?”	Q3 = Apartment	0 %
Q12	Yes/No	“Does it have a terrace?”	Q3 = Apartment	0-1 %
Q13	Yes/No	“Is the terrace really large and spacious (yes/no)?”	Q3 = Apartment, Q12 = Yes	N/A

Table A.2: Samples of the final, revised questions utilized for feature extraction of property data prompting Llama2.

A.1.7. Dataset Preview Sample



score	query	short
0.495	dúplex con 1 habitación con piscina compartida	Título: Piso en venta en Urb. Arroyo Vega, Arroyo de la Vega. Ubicación: La Moraleja. Precio: 595000€. Características clave: 135 m2 construidos, 3 habitaciones, 2 baños, plaza de garaje…
0.4515	casa separada con 1 habitación y 1 baño y plaza de parking con piscina en la urbanización	Título: Piso en venta en Urb. Arroyo Vega, Arroyo de la Vega. Ubicación: La Moraleja. Precio: 595000€. Características clave: 135 m2 construidos, 3 habitaciones, 2 baños, plaza de garaje…
-0.7	chalet pareado con 4 habitaciones, tres plazas de garaje con piscina individual	Título: Piso en venta en Urb. Arroyo Vega, Arroyo de la Vega. Ubicación: La Moraleja. Precio: 595000€. Características clave: 135 m2 construidos, 3 habitaciones, 2 baños, plaza de garaje…
-0.233333	dúplex bajo con 3 habitaciones y 3 baños y tres plazas de garaje con piscina privada	Título: Piso en venta en Urb. Arroyo Vega, Arroyo de la Vega. Ubicación: La Moraleja. Precio: 595000€. Características clave: 135 m2 construidos, 3 habitaciones, 2 baños, plaza de garaje…
0.101667	piso con 1 habitación y plaza de garaje con piscina propia	Título: Piso en venta en Urb. Arroyo Vega, Arroyo de la Vega. Ubicación: La Moraleja. Precio: 595000€. Características clave: 135 m2 construidos, 3 habitaciones, 2 baños, plaza de garaje…
0.412778	casa pareada que tenga 1 habitación y plaza de parking con piscina en la urbanización	Título: Piso en venta en Urb. Arroyo Vega, Arroyo de la Vega. Ubicación: La Moraleja. Precio: 595000€. Características clave: 135 m2 construidos, 3 habitaciones, 2 baños, plaza de garaje…
-0.35	dúplex con 2 dormitorios con piscina interior	Título: Piso en venta en Urb. Arroyo Vega, Arroyo de la Vega. Ubicación: La Moraleja. Precio: 595000€. Características clave: 135 m2 construidos, 3 habitaciones, 2 baños, plaza de garaje…
-0.1375	apartamento con 4 habitaciones y 2 baños con piscina de uso privado	Título: Piso en venta en Urb. Arroyo Vega, Arroyo de la Vega. Ubicación: La Moraleja. Precio: 595000€. Características clave: 135 m2 construidos, 3 habitaciones, 2 baños, plaza de garaje…
-0.024545	vivienda con 1 habitación y una plaza de parking con piscina propia	Título: Piso en venta en Urb. Arroyo Vega, Arroyo de la Vega. Ubicación: La Moraleja. Precio: 595000€. Características clave: 135 m2 construidos, 3 habitaciones, 2 baños, plaza de garaje…
0.040741	dúplex-ático con 3 habitaciones y 2 plazas de parking con piscina individual	Título: Piso en venta en Urb. Arroyo Vega, Arroyo de la Vega. Ubicación: La Moraleja. Precio: 595000€. Características clave: 135 m2 construidos, 3 habitaciones, 2 baños, plaza de garaje…
0.105556	bajo, 3 habitaciones y tengo dos coches que aparcar y dos plazas de garaje con piscina individual	Título: Piso en venta en Urb. Arroyo Vega, Arroyo de la Vega. Ubicación: La Moraleja. Precio: 595000€. Características clave: 135 m2 construidos, 3 habitaciones, 2 baños, plaza de garaje…

Figure A.5: Dataset preview sample from the Hugging Face Dataset Viewer tool. Scores have been normalized to range from -1 (lowest) to 1 (highest).

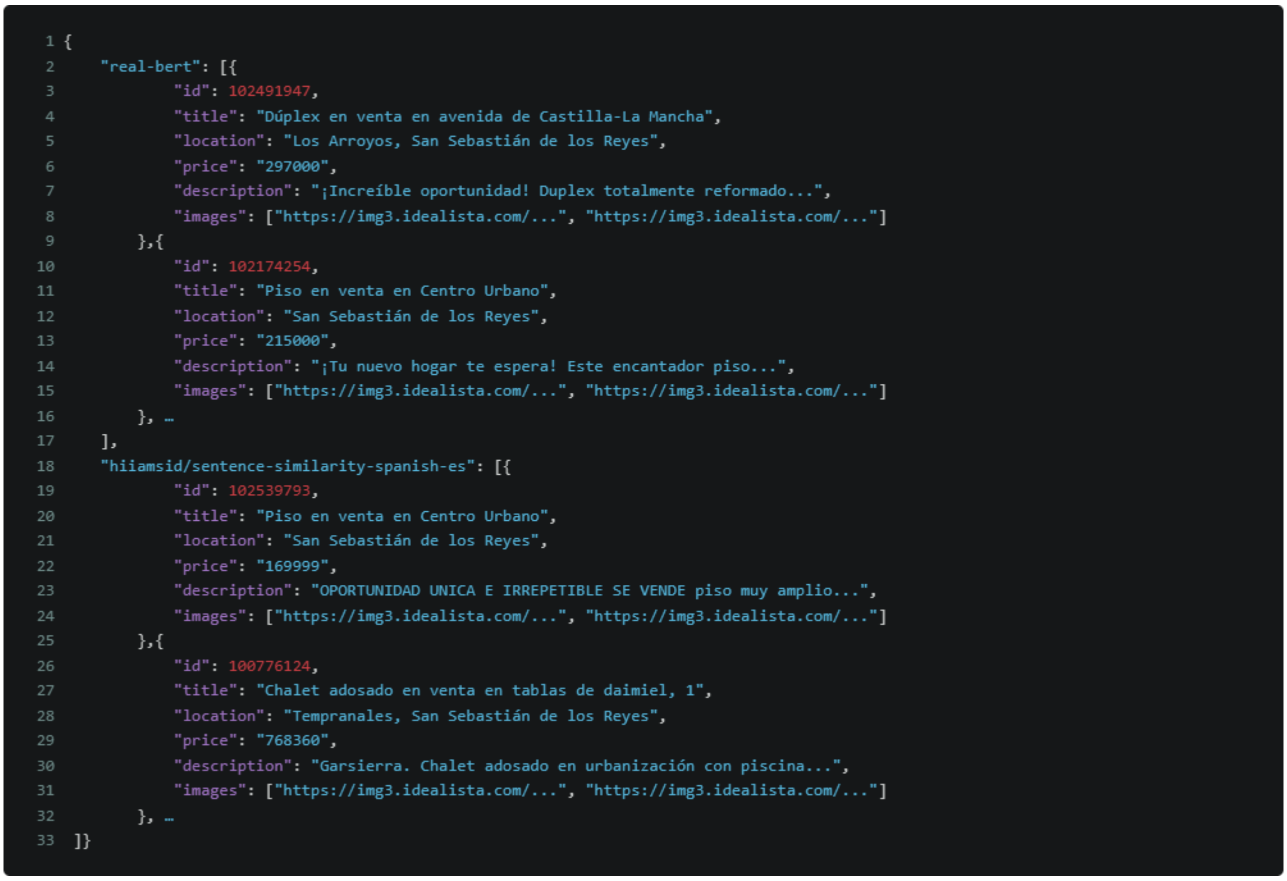
A.2. Web Application and Deployment

A.2.1. Gradio Interface Implementation

Code A.1: Definition of the Gradio Interface for our REST API, showcasing the setup of the search function, input parameters, and additional interface properties.

```
gradio_interface = gradio.Interface(
    fn=search,
    api_name="search",
    inputs=[
        "text", # user query
        "text" # collection name
        ],
    outputs="text", # JSON search results
    examples=[
        ["apartamento␣con␣3␣habitaciones", "real_bert"],
        ["chalet␣con␣piscina", "all"]
    ],
  title="TFG␣Web␣Demo␣REST␣API",
```

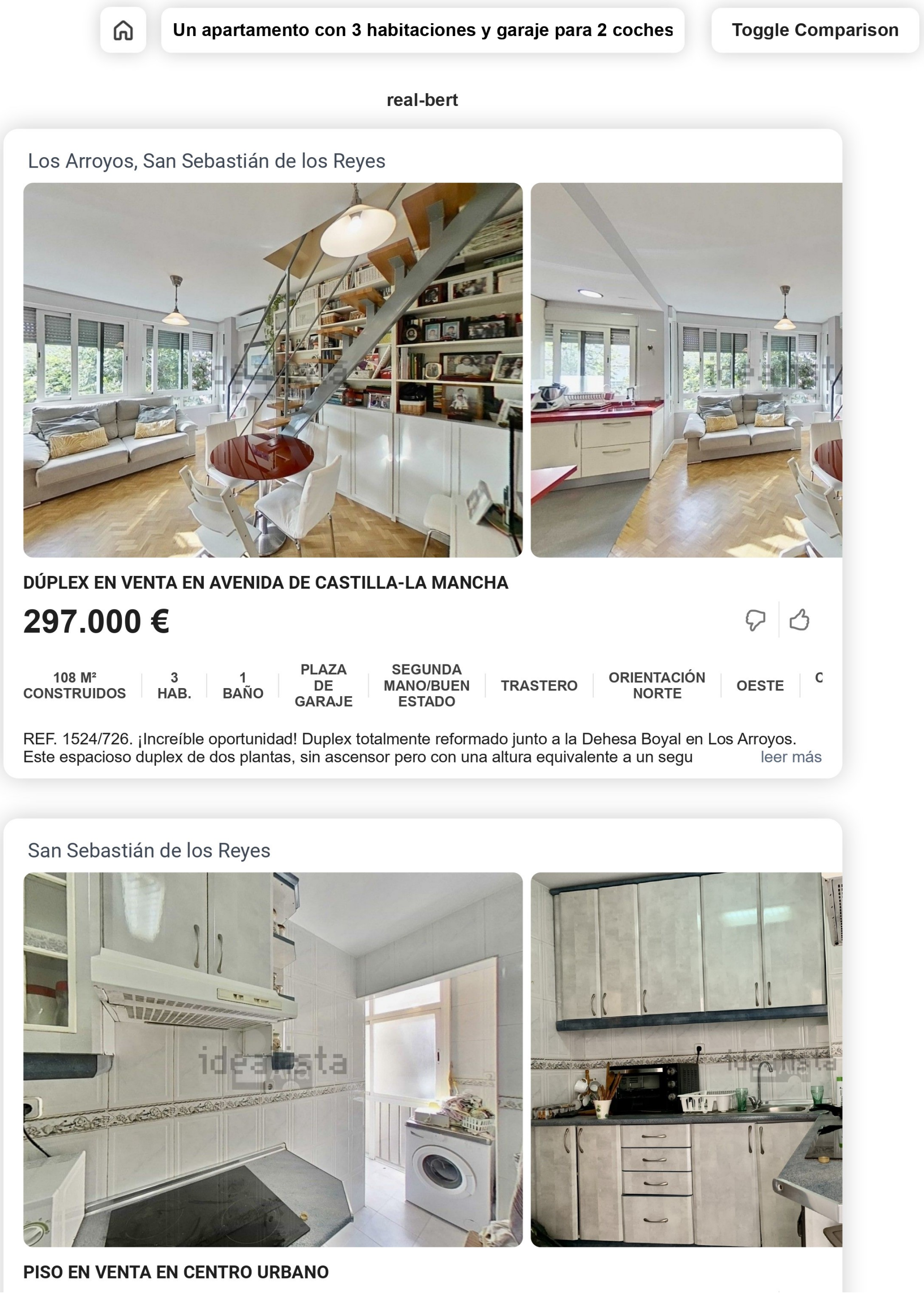
A.2.2. API Response



```
{
    "real-bert": [{
            "id": 102491947,
            "title": "Dúplex en venta en avenida de Castilla-La Mancha",
            "location": "Los Arroyos, San Sebastián de los Reyes",
            "price": "297000",
            "description": "¡Increíble oportunidad! Duplex totalmente reformado...",
            "images": ["https://img3.idealista.com/...", "https://img3.idealista.com/..."]
        },{
            "id": 102174254,
            "title": "Piso en venta en Centro Urbano",
            "location": "San Sebastián de los Reyes",
            "price": "215000",
            "description": "¡Tu nuevo hogar te espera! Este encantador piso...",
            "images": ["https://img3.idealista.com/...", "https://img3.idealista.com/..."]
        }, …
    ],
    "hiiamsid/sentence-similarity-spanish-es": [{
            "id": 102539793,
            "title": "Piso en venta en Centro Urbano",
            "location": "San Sebastián de los Reyes",
            "price": "169999",
            "description": "OPORTUNIDAD UNICA E IRREPETIBLE SE VENDE piso muy amplio...",
            "images": ["https://img3.idealista.com/...", "https://img3.idealista.com/..."]
        },{
            "id": 100776124,
            "title": "Chalet adosado en venta en tablas de daimiel, 1",
            "location": "Tempranales, San Sebastián de los Reyes",
            "price": "768360",
            "description": "Garsierra. Chalet adosado en urbanización con piscina...",
            "images": ["https://img3.idealista.com/...", "https://img3.idealista.com/..."]
        }, …
  ]}
```

Figure A.6: Sample API response for request with body: `query` 'apartamento con 3 habitaciones', `collection_name` 'all'.

A.2.3. Search Results UI



Un apartamento con 3 habitaciones y garaje para 2 coches

Toggle Comparison

real-bert

Los Arroyos, San Sebastián de los Reyes

DÚPLEX EN VENTA EN AVENIDA DE CASTILLA-LA MANCHA

297.000 €

108 M² CONSTRUIDOS | 3 HAB. | 1 BAÑO | PLAZA DE GARAJE | SEGUNDA MANO/BUEN ESTADO | TRASTERO | ORIENTACIÓN NORTE | OESTE | C

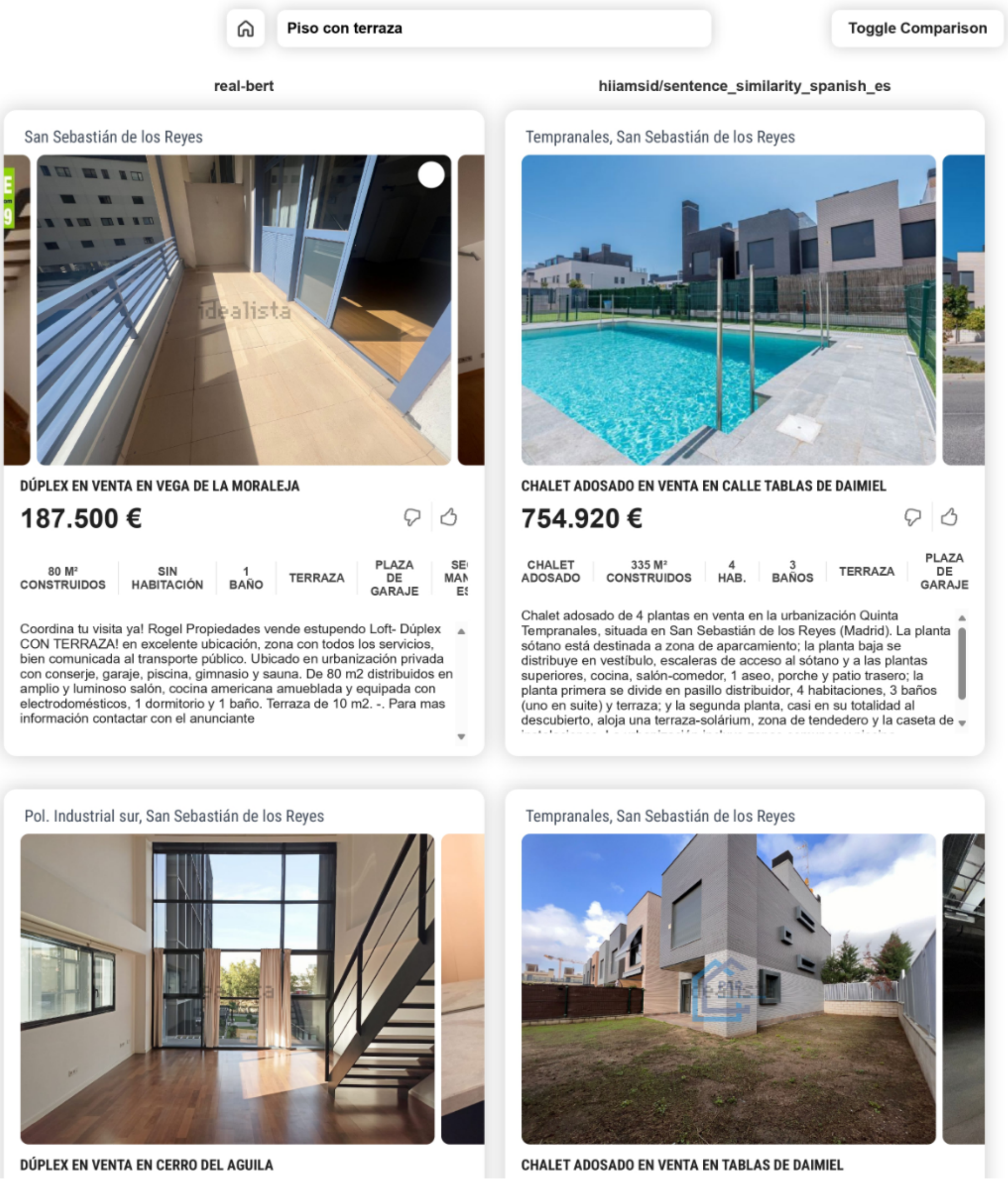
REF. 1524/726. ¡Increíble oportunidad! Duplex totalmente reformado junto a la Dehesa Boyal en Los Arroyos. Este espacioso duplex de dos plantas, sin ascensor pero con una altura equivalente a un segu leer más

San Sebastián de los Reyes

PISO EN VENTA EN CENTRO URBANO

Figure A.7: User interface for exploring property listing search results in our web app.

A.2.4. Comparison View UI


Piso con terraza

Toggle Comparison

real-bert

hiiamsid/sentence_similarity_spanish_es

San Sebastián de los Reyes

DÚPLEX EN VENTA EN VEGA DE LA MORALEJA

187.500 €

80 M² CONSTRUIDOS | SIN HABITACIÓN | 1 BAÑO | TERRAZA | PLAZA DE GARAJE | SE MAN ES

Coordina tu visita ya! Rogel Propiedades vende estupendo Loft- Dúplex CON TERRAZA! en excelente ubicación, zona con todos los servicios, bien comunicada al transporte público. Ubicado en urbanización privada con conserje, garaje, piscina, gimnasio y sauna. De 80 m2 distribuidos en amplio y luminoso salón, cocina americana amueblada y equipada con electrodomésticos, 1 dormitorio y 1 baño. Terraza de 10 m2. -. Para mas información contactar con el anunciante

Tempranales, San Sebastián de los Reyes

CHALET ADOSADO EN VENTA EN CALLE TABLAS DE DAIMIEL

754.920 €

CHALET ADOSADO | 335 M² CONSTRUIDOS | 4 HAB. | 3 BAÑOS | TERRAZA | PLAZA DE GARAJE

Chalet adosado de 4 plantas en venta en la urbanización Quinta Tempranales, situada en San Sebastián de los Reyes (Madrid). La planta sótano está destinada a zona de aparcamiento; la planta baja se distribuye en vestíbulo, escaleras de acceso al sótano y a las plantas superiores, cocina, salón-comedor, 1 aseo, porche y patio trasero; la planta primera se divide en pasillo distribuidor, 4 habitaciones, 3 baños (uno en suite) y terraza; y la segunda planta, casi en su totalidad al descubierto, aloja una terraza-solárium, zona de tendedero y la caseta de

Pol. Industrial sur, San Sebastián de los Reyes

DÚPLEX EN VENTA EN CERRO DEL AGUILA

Tempranales, San Sebastián de los Reyes

CHALET ADOSADO EN VENTA EN TABLAS DE DAIMIEL

Figure A.8: User interface for comparing the results from two different embedding models in our web app.

B

Experiments and Results

B.1. Experimental Setup

B.1.1. Evaluation Dataset Summary

Corpus Size	**Query Type**	**Query Count**	**Sample Query**	**Number of Perfect Matches**
25	Basic	307	*apartamento con 3 habitaciones*	10
25	Extended	391	*vivienda unifamiliar con pista de pádel*	6
25	Attribute: House Type	40	*dúplex ático*	4
25	Attribute: Pools	13	*piscina privada*	5

Table B.1: Query Types and Distribution of Perfect Matches. The ’Perfect Matches’ column indicates the number of properties in the corpus that perfectly match the features specified in the respective example query.

Universidad Autónoma de Madrid