PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

UNIVERSITY MUSTAPHA STAMBOULI OF MASCARA

Faculty of Exact Sciences

Department of Computer Science

Dissertation

Submitted in partial fulfilment of the requirements for Master degree in Computer Science

Option: Intelligent Systems Engineering

Entitled

Enhancing Reasoning Capabilities of Retrieval-Augmented Generation (RAG) in LLM-Based Agents for Legal and Juridical Data.

Presented by **Boudjenane Zoubida Asmaa**

Jury:

President	SMAIL Omar	Professor	University of Mascara
Director	Mohammed SALEM	Professor	University of Mascara
Examiner	MAHMOUDI Laouni	Professor	University of Mascara
Examiner	BOUFERA Fatma	Professor	University of Mascara

2024/2025

Acknowledgements

First of all, praise and thanks to Allah Almighty for giving us all the patience, courage, will and motivation that allowed us to accomplish this work.

My deepest gratitude goes to my supervisor, **Mr Salem Mohamed** for his patient guidance, insightful feedback, and continuous encouragement throughout the course of this work .

I would like to express my sincere gratitude to the members of the jury for their valuable time and insightful feedback:

- **Prof. SMAIL Omar** , at University of Mascara.
- **Prof. MAHMOUDI Laouni**, at University of Mascara.
- **Prof. BOUFERA Fatma**, at University of Mascara.

Their rigorous examination and constructive suggestions have significantly improved this work.

I am also especially thankful to my parents, family, and friends for their unwavering support, love, and motivation at every step of this journey.

I would like to thank everyone who helps me to improve my work. and who gave me any remark that helped me to perfect this manuscript.

All the teachers of the University MUSTAPHA STAMBOULI DE LA WILAYA MASCARA For the good contribution of this work.

Dedication

This work is dedicated with deep love and gratitude to:

My dear parents, whose unwavering support, sacrifices, and constant prayers have been the foundation of everything I've achieved. Your belief in me gave me the strength to persevere through every challenge.

My supervisor, Mr. Salem Mohamed, whose patient guidance, insightful feedback, and continuous encouragement made this journey not only possible but meaningful. Your dedication, expertise, and belief in my potential have left a lasting impact on me both academically and personally.

My beloved sister Boudjenane Fatima Zahra, for being my greatest cheerleader and my constant source of emotional support. Your encouragement and care during the toughest moments meant the world to me.

My wonderful friend nial hadjer, for always being there with kind words, late-night conversations, and unshakable support.

ملخص

أدى التعقيد المتزايد في الأنظمة القانونية، إلى جانب الطبيعة غير المهيكلة والنطاق الإقليمي الخاص للوثائق القانونية، خاصة في الأنظمة متعددة اللغات، إلى ازدياد الاهتمام باستخدام تقنيات معالجة اللغة الطبيعية لتحسين الوصول إلى النصوص القانونية وفهمها. وقد أسهمت التطورات الحديثة في تقنيات الذكاء الاصطناعي في تقديم حلول واعدة من خلال الجمع بين توليد اللغة واسترجاع المستندات بشكل فوري. ومع ذلك، لا تزال الأنظمة الحالية تعاني من ضعف في عملية الاسترجاع، حيث تعتمد على عدد ثابت من المستندات، مما يؤدي إلى توليد معلومات غير دقيقة، أو تقديم محتوى غير كافٍ أو مشوش، بالإضافة إلى ضعف التوافق مع استفسارات المستخدمين. هذه التحديات تؤثر بشكل كبير على فعالية هذه الأنظمة في السياقات القانونية الحساسة. تعالج هذه الأطروحة هذه القيود من خلال اقتراح آلية جديدة لاختيار المستندات تعتمد على تكييف عددها حسب تعقيد الاستفسار، بدلاً من استخدام عدد ثابت. كما تسهم الأطروحة في تطوير مجموعة بيانات قانونية باللغة العربية مأخوذة من أحكام قضائية جزائرية، وتعرض واحدة من أولى المنظومات المتكاملة التي تجمع بين الاسترجاع والتوليد باللغة العربية. وقد تم تطوير وكيل حواري قادر على فهم الاستفساراتِ القانونية بالعربية وتقديم إجابات دقيقة ومرتكزة على السياق، مما يحسن من إمكانية الوصول إلى المعلومات القانونية في البيئات ذات الموارد المحدودة. ومن خلال معالجة التحديات الأساسية في الاسترجاع والتوليد، تسعى هذه الأطروحة إلى دعم البحث في المعالجة القانونية للغة العربية وتوفير أدوات عملية لتطوير مساعدين قانونيين ذكيين في العالم العربي.

كلمات مفتاحية: معالجة اللغة الطبيعية، الذكاء الاصطناعي القانوني، نماذج اللغة الضخمة، التوليد المعزز بالاسترجاع، المعالجة القانونية للغة العربية، استرجاع المعلومات القانونية، الوكلاء الحواريون، مجموعة بيانات قانونية جزائرية، الأنظمة القانونية متعددة اللغات.

Abstract

The growing complexity of legal systems, combined with the unstructured and region-specific nature of legal documents—particularly in multilingual jurisdictions—has sparked increasing interest in the use of Natural Language Processing (NLP) to enhance access to and comprehension of legal texts. Recent advances in Large Language Models (LLMs) and Retrieval-Augmented Generation (RAG) architectures offer promising solutions by combining language generation with real-time document retrieval. However, current RAG systems typically rely on fixed top-k document selection, which can result in hallucinations, insufficient or noisy context, and poor alignment with user queries. These retrieval inefficiencies significantly degrade the performance of LLMs in sensitive legal contexts. This thesis addresses these limitations by proposing a novel dynamic candidate selection mechanism that adapts retrieval thresholds based on query complexity. Furthermore, it contributes a new Arabic legal case dataset sourced from Algerian court rulings and introduces one of the first Arabic RAG pipelines that fully integrates retrieval and generation components. The developed conversational agent is capable of understanding legal Arabic queries and delivering grounded, contextually accurate responses—thus improving legal accessibility in under-resourced settings. By tackling key challenges in both retrieval and generation, this thesis advances research in Arabic legal NLP and provides practical tools for developing intelligent legal assistants in the Arab world.

Key words: *Natural Language Processing (NLP), Legal Artificial Intelligence, Large Language Models (LLMs), Retrieval-Augmented Generation (RAG), Arabic Legal NLP, Dynamic Candidate Selection, Legal Information Retrieval, Conversational Agents, Algerian Legal Dataset, Multilingual Legal Systems.*

Contents

List of Figures

List of Tables

List of Abbreviations

NLP:	Natural Language Processing
NLG:	Natural Language Generation
NLU:	Natural Language Understanding
LLM:	Large Language Models
RAG:	Retrieval-Augmented Generation
CBOW:	Continuous Bag-of-Words
GloVe:	Global Vectors for Word Representation
GPT:	Generative Pre-trained Transformer
BERT:	Bidirectional Encoder Representations from Transformers
T5:	Text-to-Text Transfer Transformer
CLM:	Causal Language Modeling
MLM:	Masked Language Modeling
FFN:	Feed-Forward Network
RNN:	Recurrent Neural Network
LSTM:	Long Short-Term Memory
RLHF:	Reinforcement learning with human feedback
Seq2Seq:	Sequence to Sequence
SASRec:	Self-Attentive Sequential Recommendation
TF-IDF:	Term Frequency-Inverse Document Frequency
QLM :	Query Likelihood Model
DPR:	Dense passage retrieva
MOL:	Mixture of Logits

General introduction

Context :

In recent years, the legal domain has witnessed growing interest in leveraging Natural Language Processing (NLP) and automation to enhance access to and retrieval of juridical data. Legal texts—such as statutes, court decisions, and regulations—are inherently complex, context-dependent, and linguistically nuanced, often employing regionally specific language. This complexity poses challenges not only for laypersons but also for legal practitioners, particularly in multilingual and jurisdiction-specific systems like Algeria's [Hamouda Sidhoum et al., 2024], where legal data remains largely unstructured or does not have the ability to be searched.

Large Language Models (LLMs) have emerged as powerful tools for understanding and generating human-like text, showing advanced potential in legal question answering, summarizing cases, and classifying documents [Naveed et al., 2023]. However, their reliance on static training data limits their ability to capture the dynamic, high-stakes nature of legal knowledge. To address this, Retrieval-Augmented Generation (RAG) architectures integrate LLMs with external knowledge retrieval systems[Lewis et al., 2020], enabling real-time access to relevant documents during inference and thereby improving the accuracy and contextual grounding of outputs.

Although these advances have been made, several challenges persist—particularly in the selection of the most relevant documents from large legal corpora. Poor retrieval quality can significantly degrade the performance and reliability of RAG-based systems. Therefore, refining the retrieval process remains a crucial area of research, especially in low-resource and domain-specific contexts [Mezghanni and Gargouri, 2016]. Furthermore, the rise of Agentic AI— AI systems capable of autonomous, goal-directed behavior—introduces new opportunities for retrieval and reasoning in NLP[Aisera, 2024], warranting further exploration.

Problem Statement :

While LLMs and RAG architectures hold significant promise for legal AI applications, their efficacy is often hampered by weaknesses in the retrieval process:

- Weaknesses in Retrieval: Fixed k-value(number of documents), RAG depend on a retrieval step (fetching relevant documents from a database) to ground LLM outputs, poor retrieval quality (e.g., irrelevant or incomplete documents) leads to
  - Hallucinations: The LLM generates incorrect or fabricated information.
  - Insufficient context retrieval leads to information deficit in generated responses and excessive context retrieval creates noise pollution in the knowledge grounding proces.
  - Contextual Misalignment: Retrieved documents may not match the nuanced intent of the user's query.
- Domain-Specific Barriers:
  - Low-Resource Language: Arabic legal NLP lacks annotated datasets, standardized vocabularies, and pre-trained models compared to English.
  - Unstructured Data: Algerian legal texts (e.g., court rulings, statutes) are often PDFs or scanned documents without metadata, making systematic retrieval difficult.
  - Multilingual Complexity: Legal texts in Algeria mix Modern Standard Arabic (MSA), Arabic diacritics, and French, further complicating retrieval and analysis.

Objectives :

This research investigates the integration of LLMs and retrieval systems in the legal domain, with emphasis on the Algerian juridical context. Key objectives include:

- Examining the potential of LLMs and Agentic RAG models for processing Arabic legal texts.
- Addressing challenges in document retrieval, particularly the k-selection problem (identifying the optimal number of candidate documents).
- Developing and sharing a curated dataset of Algerian legal cases in Arabic, advancing resources for low-resource legal NLP.

Contributions :

This thesis makes the following contributions:

- Review of LLMs in Retrieval-Augmented Generation (RAG): we provide a comprehensive review of LLMs in RAG systems, with specific analysis of their applications and limitations in legal contexts, particularly for Arabic and multilingual juridical data.
- Dynamic Candidate Selection: we propose a novel dynamic k-selection method that automatically optimizes the number of retrieved documents based on query characteristics and relevance thresholds. (submitted to the Journal of Artificial Intelligence Research).
- Arabic Legal Case Dataset: we contribute two new datasets for Arabic legal NLP: a curated collection of Algerian Supreme Court rulings with legal annotations, and a synthetic QA dataset derived from these cases. These resources address the current scarcity of Arabic legal data for research and development.
- Arabic RAG-Based Agent for Legal and Juridical Data: A domain-specific conversational agent that understands natural Arabic legal queries, retrieves relevant Algerian case and generates accurate, grounded responses while recommending pertinent legal materials (submitted to ICAITA25).

Structure :

The thesis is organized into four main chapters:

- Chapter 1:Large Language Models—Overview of LLMs, with emphasis on legal NLP and the role of Agentic AI in autonomous legal reasoning.
- Chapter 2:Retrieval-Augmented Generation – A detailed examination of RAG architectures, their strengths, and limitations, with a focus on applications in both legal and recommendation systems.
- Chapter 3:Candidate Selection in Retrieval – The chapter reviews existing k selection methods, followed by a proposed dynamic selection method to improve retrieval efficiency and generation quality.
- Chapter 4:Implementation—Design and evaluation of the Arabic RAG-Based Agent, detailing the full pipeline and its agentic capabilities.

Chapter 1

Large Language Models (LLMs) In Agentic AI

1.1 Introduction

Large Language Models (LLMs) have marked a major milestone in the evolution of Natural Language Processing (NLP), allowing machines to perform complex language tasks with remarkable accuracy. From generating coherent text to answering nuanced questions, LLMs have pushed the boundaries of what AI can achieve. This chapter explores the foundational architecture and core mechanisms that power LLMs, the advancements driven by large-scale training and data, and the key models that have shaped the landscape of modern AI. The final section of this chapter will define agentic AI, outline its various types and categories, and examine the challenges it presents in the context of responsible AI development.

1.2 Language Models

The ability to model and generate human language has long been a key goal in the field of artificial intelligence. Language models play an essential role in achieving this by enabling machines to predict, understand, and produce natural language text. This section provides a comprehensive overview of language models.

1.2.1 Definition of Language Models

Language models (LMs) are fundamental components in the field of natural language processing. They are designed to estimate the probability distribution over sequences of words, thereby enabling machines to understand and generate human language. Essentially, a lan-

guage model predicts the likelihood of a word given its preceding context [Bengio et al., 2003], Early developments in language modeling primarily focused on statistical methods Over time, however, advancements in computational power and machine learning techniques have led to the emergence of more sophisticated models, particularly those based on neural networks.

1.2.2 N-gram Language Models

N-gram language models are among the earliest statistical approaches to approximating the probability distribution over sequences of words. The fundamental idea relies on the Markov assumption, positing that the likelihood of a word depends only on a finite history of preceding words, typically the previous $n-1$ tokens[Chen and Goodman, 1999].. Formally, given a sequence of words $w_1, w_2, \ldots, w_T$, the probability of the entire sequence can be decomposed as:

$$P(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} P(w_t \mid w_{t-n+1}, \ldots, w_{t-1}) \tag{1.1}$$

In the simplest case, the unigram model ($n = 1$) assumes that each word is generated independently of any context, yielding:

$$P(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} P(w_t) \tag{1.2}$$

While this assumption significantly simplifies the model, it neglects important contextual information. The bigram model ($n = 2$) improves upon this by conditioning each word on its immediate predecessor:

$$P(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} P(w_t \mid w_{t-1}) \tag{1.3}$$

Extending further, the trigram model ($n = 3$) conditions each word on the two preceding words, thereby capturing more syntactic and semantic dependencies:

$$P(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} P(w_t \mid w_{t-2}, w_{t-1}) \tag{1.4}$$

As n increases, the N-gram model can capture increasingly longer contexts, however, this comes at the cost of exponentially increasing data sparsity, since many word sequences may rarely occur in practice. Consequently, techniques such as smoothing (e.g., Laplace smoothing) [Chen and Goodman, 1999] is employed to mitigate the impact of unseen N-grams during probability estimation.

Despite their simplicity, N-gram models laid the groundwork for more sophisticated probabilistic and neural language models[Jurafsky and Martin, 2009]. Their mathematical tractability and interpretability made them the standard in natural language processing tasks prior to the deep learning era.

1.3 Neural Language Models

To get around the limits of older statistical language models, researchers started using neural language models. These models rely on neural networks to learn how words relate to each other in a continuous space, which helps them understand and predict language better even when dealing with sentences they haven't seen before. Early approaches often used recurrent neural networks (RNNs), which process text one word at a time in sequence[Cho et al., 2014], and long short-term memory networks (LSTMs) [Hochreiter and Schmidhuber, 1997], a type of RNN designed to learn long-term dependencies in sequential.

1.3.1 Definition of Neural Language Models

NLM is a probabilistic model that predicts the next word in a sequence given the previous context, using a neural network to learn both word representations and the probability function jointly. Unlike traditional n-gram models, which suffer from data sparsity and require smoothing, neural language models can generalize across similar contexts by learning distributed representations of words. Given a word sequence $(w_1, w_2, \ldots, w_T)$, the probability of the sequence is modeled as:

$$P(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} P(w_t \mid w_{t-1}, w_{t-2}, \ldots, w_{t-n+1}) \tag{1.5}$$

As introduced by [Bengio et al., 2003], a feedforward NLM maps the previous $n-1$ words into dense vectors using a shared embedding matrix. These vectors are concatenated and passed through one or more hidden layers before predicting the next word using a softmax output layer.

$$P(w_t \mid w_{t-1}, \ldots, w_{t-n+1}) = \text{Softmax}(f(C(w_{t-1}, \ldots, w_{t-n+1}))) \tag{1.6}$$

This method enables generalization to unseen word combinations and improves predictive performance. However, as noted by Jurafsky and Martin [2019], the computational cost is significantly higher than that of n-gram models. Still, NLMs serve as foundational components in many modern NLP applications, including translation, dialogue, and generation.

Figure 1.1 shows a Neural language model architecture: where The model computes the probability of a word by applying a neural network g to the embedding vectors $C(w_{t-1}), \ldots, C(w_{t-n+1})$, where each $C(w)$ represents the learned features of a word.

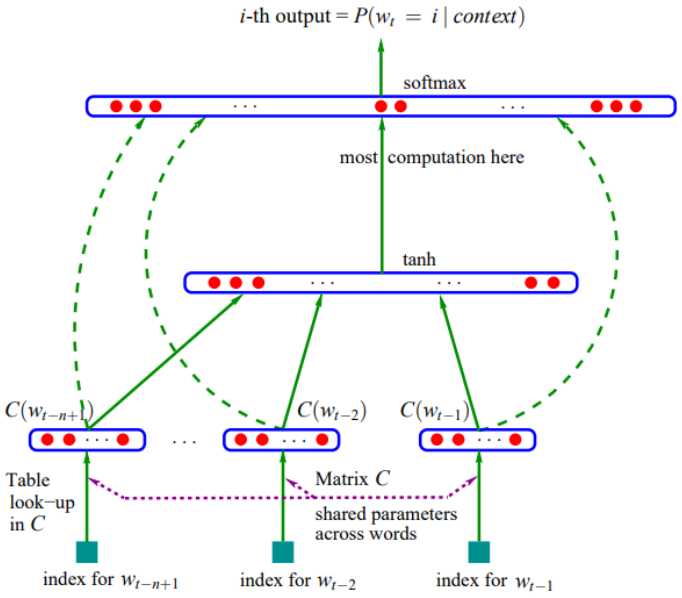


Figure 1.1: Neural language model architecture. .

1.3.2 Word Representations (Embedding)

Word embeddings are dense, low-dimensional vectors that represent words in a way that captures their meanings and relationships. Unlike traditional methods such as one-hot encoding—which treats each word as an isolated symbol—embeddings place similar words closer together in a multi-dimensional space based on how they appear in context[Barnard, 2024]. This approach has become essential in NLP, as it allows models to understand and process text more effectively. Word embeddings are typically learned from large text corpora using neural networks or statistical methods.

One of the major breakthroughs in natural language processing came with the introduction of Word2Vec by Tomas [Mikolov et al., 2013]. This model efficiently learns word representations from large text corpora using two main architectures: Continuous Bag-of-Words (CBOW) and Continuous Skip-gram.

1.3.3 Continuous Bag-of-Words (CBOW)

Predicts a target word based on the context of surrounding words. Unlike traditional bag-of-words models, it doesn't just count word occurrences—it uses continuous vectors to represent

each word and averages the context words to predict the middle one. This model ignores word order but learns meaningful embeddings using simple and fast training. It performs best when both past and future words are used as input.

1.3.4 Continuous Skip-gram

Instead of predicting a word from its context, it tries to predict context words from a single center word. The model samples words from both directions (before and after the center word) and gives more weight to nearby words while reducing the influence of distant ones. This helps capture richer word relationships, especially in smaller datasets.
Figure 1.2 Illustration the two Word2Vec architectures where the CBOW model predicts a target word from its surrounding context, while the Skip-gram model does the opposite by predicting the context from a single target word.

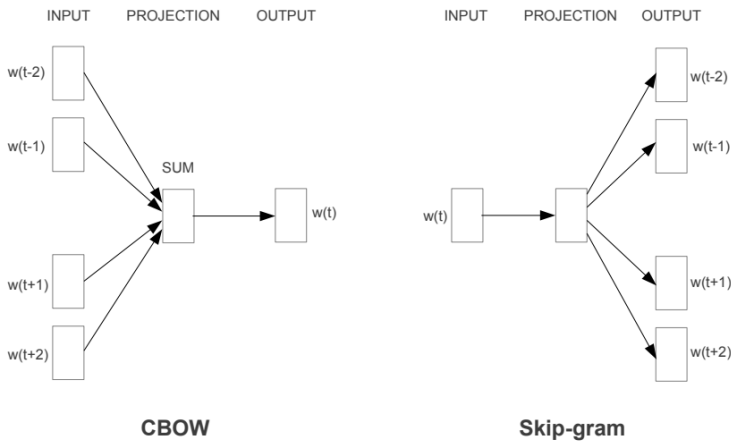


Figure 1.2: The two Word2Vec architectures: the CBOW model and Skip-gram model.

1.3.5 Global Vectors for Word Representation(GloVe)

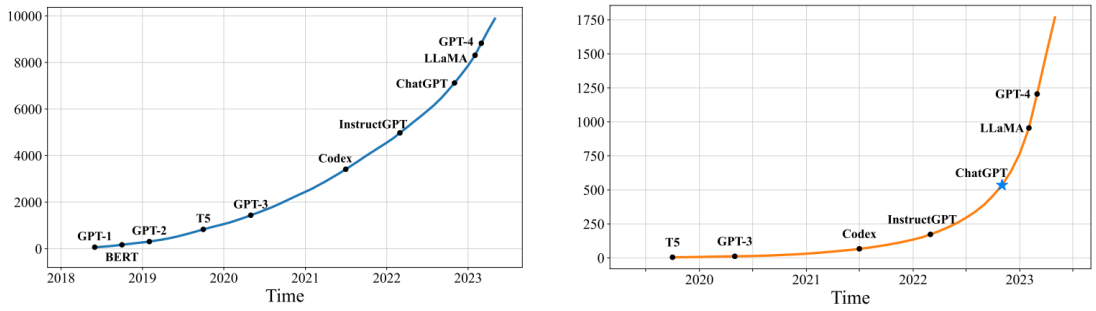
(Global Vectors for Word Representation) was introduced by [Pennington et al., 2014]. Unlike Word2Vec, which learns from local word sequences, GloVe uses word co-occurrence counts from the entire corpus. This global perspective allows it to better capture semantic patterns between words, such as analogies and relationships.

These embedding models—Word2Vec (CBOW and Skip-gram) and GloVe—have become foundational tools in NLP due to their ability to turn words into dense vectors that reflect semantic meaning, while also being computationally efficient.

1.4 Large Language Models (LLMs)

Large Language Models (LLMs) represent a significant advancement in natural language processing, capable of understanding, generating, and reasoning over human language. In this section, we explore the concept of LLMs, beginning with their definition, followed by a discussion of their key characteristics and significance in modern applications.

Figure 1.3 shows the trends in the cumulative number of arXiv publications containing the terms "language model" and "large language model." The data, calculated by monthly queries of titles and abstracts, reveal that research into large language models significantly accelerated after the launch of ChatGPT, with the publication rate rising from 0.40 to 8.58 papers per day. .



(a) Query="Language Model" (b) Query="Large Language Model"

Figure 1.3: Cumulative growth of arXiv papers mentioning "language model" and "large language model," highlighting the sharp increase after the release of ChatGPT.

1.4.1 Definition of LLMs

Large Language Models (LLMs) are deep neural networks—most built on the transformer architecture [Vaswani et al., 2017]. Their key innovation is self-attention, which dynamically adjusts word importance, allowing human-like text generation. Trained on vast datasets (books, articles, codebases), LLMs reach hundreds of billions of parameters [Brown et al., 2020, Touvron et al., 2023a], enabling them to capture intricate linguistic patterns, world knowledge, and semantic relationships. Unlike traditional AI systems, they generalize across tasks (translation, summarization, QA) with minimal task-specific data (few-shot or zero-shot learning) [Brown et al., 2020]. Their effectiveness follows scaling laws [Kaplan et al., 2020]: expanding their size, training data, and computational resources reliably boosts performance.

1.4.2 Historical Development of LLMs

The evolution of LLMs reflects a broader shift in NLP from early statistical methods to deep learning approaches. Initially, language models were based on statistical techniques, such as n-gram models, which captured limited local dependencies within text. The development of neural networks introduced new architectures capable of modeling longer-range dependencies, leading to significant improvements in performance. A major breakthrough occurred with the introduction of the transformer architecture, which enabled models to process sequences in parallel and capture global context more effectively[Chu et al., 2024]. This innovation laid the foundation for the emergence of LLMs, characterized by their massive scale, extensive pretraining on diverse corpora, and ability to generalize across a wide range of language tasks .

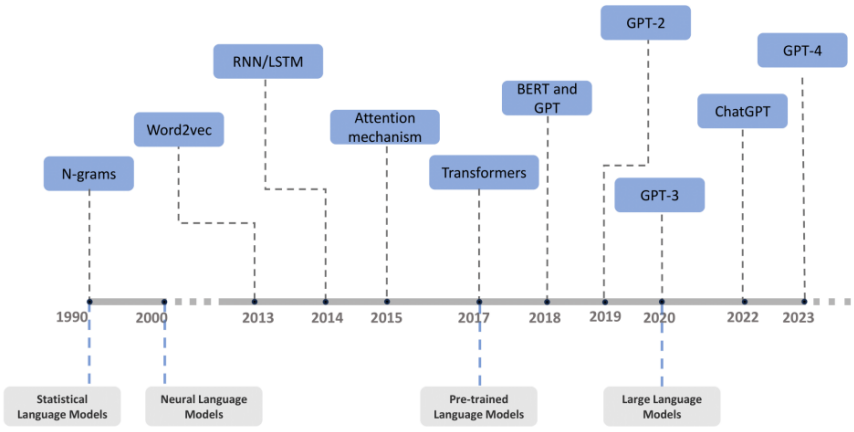


Figure 1.4: History and development of language models.

1.4.3 Transformer Architecture

The Transformer architecture is a deep learning model introduced in June 2017 by [Vaswani et al., 2017]. from Google Brain. Their paper, titled "Attention Is All You Need," presented a groundbreaking approach to processing sequential data through the use of a self-attention mechanism. This innovative method allows the model to assign different levels of importance to various parts of the input, enabling it to capture long-range dependencies much more effectively than earlier models like RNNs and LSTMs. The original Transformer model is structured as a stack of six layers, where the output of each layer i serves as the input to the subsequent layer i+1, continuing this process until the final prediction is reached.

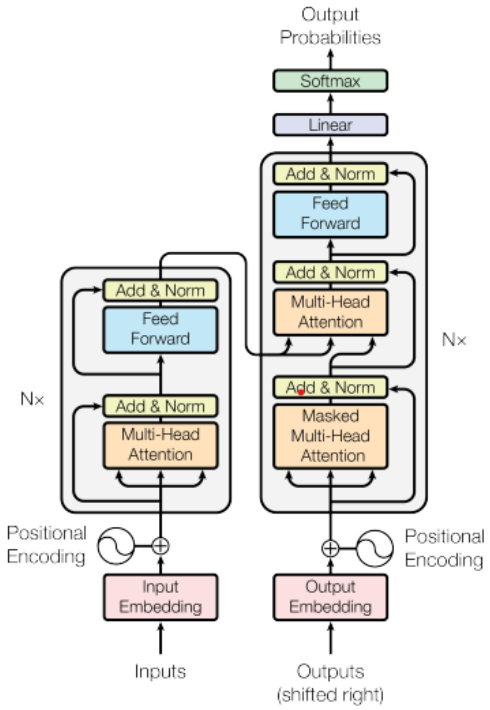


Figure 1.5: The Transformer - model architecture.

1.4.3.1 Encoder and Decoder Stacks

Transformer architectures are built upon two primary components: encoder and decoder stacks[Vaswani et al., 2017]. The encoder stack It features a six-layer on the left and a corresponding six-layer decoder on the right, both of which work together to transform input sequences into meaningful outputs.

1. **Encoder Stack** Each encoder layer processes input tokens through:

   - **Multi-head self-attention**: Computes weighted relationships between all tokens enabling the model to contextualize each word (e.g., resolving polysemy like "bank").
   - **Position-wise FFN**: A fully connected network applied independently to each token's representation, introducing nonlinear transformations.

   Post-processing: Residual connections (with layer normalization) mitigate vanishing gradients:

   $$\text{Output} = \text{LayerNorm}(x + \text{Sublayer}(x))$$

   All outputs maintain $d_{\text{model}} = 512$.

2. **Decoder Stack** The decoder extends the encoder with:

   - **Masked self-attention**: Prevents the model from seeing future words when predicting the next token, enforcing left-to-right processing like human reading.(autoregressive constraint).
   - **Cross-attention**: Links decoder inputs to encoder outputs (e.g., for translation alignment).

   Shared design: Uses identical residual connections, normalization, and dimensionality as the encoder.

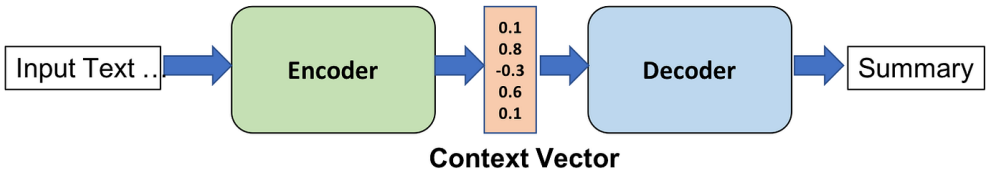


Figure 1.6: Encoder and Decoder Stacks

1.4.3.2 Self-Attention Mechanism in Transformers

The self-attention mechanism represents a paradigm shift in sequence modeling, enabling Transformers to dynamically evaluate relationships between all tokens in a sequence. Unlike traditional recurrent approaches that process information sequentially, this mechanism allows direct modeling of long-range dependencies through pairwise token interactions [Choromanski et al., 2021]. At its core, the process involves three learned linear projections:

- **Queries (Q)** representing the current token's information needs
- **Keys (K)** serving as identifiers for contextual relevance
- **Values (V)** containing the actual content to be weighted

The attention computation follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{1.7}$$

where the scaling factor $1/\sqrt{d_k}$ (first proposed in [Vaswani et al., 2017] and theoretically analyzed in [Choromanski et al., 2021]) ensures stable gradient flow during training. This mechanism allows each token to selectively focus on other relevant tokens, thereby enhancing the model's capacity to understand complex linguistic patterns.

1.4.3.3 Positional Encoding:

Since self-attention mechanisms do not inherently capture the order of words, positional encoding is added to the word embeddings to preserve word order information in Transformer models, Rothman [Rothman, 2021] describes the sinusoidal positional encoding approach defined by:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right), \quad PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{1.8}$$

where pos represents the token position in the sequence and i indexes the embedding dimension. As explained in [Rothman, 2021], this specific formulation provides key advantages:

- The sinusoidal pattern allows the model to generalize to sequence lengths beyond those encountered during training.
- The alternating sine/cosine pattern enables the model to learn relative positions through linear transformations.

1.4.3.4 Role of Multi-Head Attention

Multi-head attention enhances standard self-attention by employing multiple parallel attention mechanisms, each operating on a separate linear projection of the input. As detailed in [Rothman, 2021], this design enables the model to:

- **Increase Representational Capacity**: By concatenating and reprojecting the outputs of all heads, the model synthesizes information from these varied perspectives.
- **Learn Diverse Relationships**: Each attention head specializes in different types of dependencies. For example, [Clark et al., 2019] found that some heads in BERT consistently track syntactic relationships (e.g., subject-verb pairs), while others focus on discourse-level connections.

Further notes that this architecture improves robustness compared to single-head attention, as errors or biases in one head can be compensated by others.

1.4.4 Key Contributions of Transformer in Modern NLP

Transformer models have introduced several transformative contributions to the field of NLP, which can be summarized as follows:

1. **Effective Long-Range Dependency Modeling:** The self-attention mechanism allows Transformers to capture dependencies between distant words in a sequence more efficiently than traditional recurrent neural networks[Vaswani et al., 2017], improving contextual understanding .

2. **High Scalability and Parallelization:** Unlike sequential models, Transformer architectures enable parallel computation during training [Devlin et al., 2018], which significantly reduces training time and supports the construction of very large-scale models, such as BERT and GPT.

3. **Versatility Across Tasks:** Transformers can be adapted to a wide range of tasks, including text classification, translation, and summarization[Radford et al., 2019]., making them a universal architecture for both encoding and decoding processes

1.4.5 Classification of LLMs

LLMs can be classified based on their accessibility and licensing terms.[Zhao et al.].

1. **open-source models** are fully available to the public, including both their architecture and training data, enabling modification and redistribution (e.g., BLOOM, Falcon).

2. **closed-source models** restrict access to both the model's structure and training corpus, typically maintained by private organizations (e.g., OpenAI's GPT-4).

1. **open-weight models** release the model parameters for public use but may impose certain restrictions on fine-tuning or commercial deployment (e.g., LLaMA 2)

2. **closed-weight models** do not share model parameters or allow external access, preserving full control within the developing entity. This classification is crucial for understanding the ethical, practical, and legal implications surrounding the deployment of LLMs

.

For a comprehensive comparison of large language models across categories such as openness, licensing, architecture, and availability, refer to 1.1.

Type	Model	Size	Provider	Release	Open-sourced	Tuna-bility	Adaptation IT	Pre-trainData scale	Open-weightmodel
Encoder-only									
	ALBERT	0.012	Google AI	02/2020	✗	✓	-	16GB	✓
	DeBERTa	0.1	Microsoft	06/2020	✓	✓	-	8GB	✓
	ELECTRA	0.14	Google AI	03/2020	✓	✓	-	16GB	✓
	BERT	0.11	Google AI	10/2018	✓	✓	-	3.3B words	✓
	RoBERTa	0.125	Meta AI	07/2019	✓	✓	-	160GB	✓
	XLM-RoBERTa	0.28	Meta AI	10/2019	✓	✓	-	2.5TB	✓
Encoder-decoder									
	ERNIE 3.0	10	Baidu	07/2021	✗	✓	✓	300B tokens	✗
	T5	11	Google AI	10/2019	✓	✓	✓	1T tokens	✓
	T0	11	BigScience	12/2021	✓	✓	✓	-	✓
	Flan-T5	11	Google AI	02/2022	✓	✓	✓	1T tokens	✓
	UL2	20	Google AI	05/2022	✗	✓	✓	-	✗
	AlexaTM	20	Alexa AI	07/2022	✗	✓	✓	1.4T tokens	✗
	AlphaCode	41	Google AI	02/2022	✗	✗	-	96TB	✗
	Anthropic	52	Anthropic	03/2023	✗	✓	✓	-	✗
	NLLB	54.5	Meta AI	07/2022	✓	✓	✗	1.6T tokens	✓
	LLAMA	65	Meta AI	02/2023	✓	✓	✗	1.4T tokens	✓
	FLAN	70	Google AI	03/2022	✓	✓	✓	1.4T tokens	✓
	Sparrow	70	Google AI	09/2022	✗	✓	✓	-	✗
	Chinchilla	70	DeepMind	03/2022	✗	✓	✗	1.4T tokens	✗
	OPT-OML	175	Meta AI	05/2022	✓	✓	✗	300B tokens	✓
	Gopher	280	Google AI	12/2021	✗	✓	✗	300B tokens	✗
	PaLM	540	Google AI	10/2022	✗	✓	✓	780B tokens	✗
	U-PaLM	540	Google AI	10/2022	✗	✓	✓	-	✗
	GLAM	1200	Google AI	12/2021	✗	✓	✗	1.6T tokens	✗
Decoder-only									
	Codex	12	OpenAI	07/2021	✗	✓	✓	-	✗
	Pythia	12	Eleuther AI	01/2023	✓	✓	✗	300B tokens	✓
	CodeGeeX	13	Tsinghua U.	12/2022	✓	✓	✗	13TB	✓
	Skywork	13	Kunlun Inc	01/2023	✗	✓	✗	3.7T tokens	✗
	QWEN	14	Alibaba	08/2023	✓	✓	✓	2.4T tokens	✓
	Salesforce	16	Salesforce	08/2022	✗	✓	✗	-	✗
	GPT-NeoX-20B	20	Eleuther AI	02/2022	✓	✓	✗	825GB	✓
	Gemini 1.5	2700	Google AI	02/2024	✗	✓	✓	6T tokens	✗

Table 1.1: Selected Large Language Models with key attributes.

1.4.6 Training Large Language Models

The training of LLMs involves a multifaceted process encompassing data collection, architectural design, optimization strategies, and fine-tuning techniques. This part delves into the critical components of LLM training.

1.4.7 Training Objectives and Loss Functions

LLMs rely on specific training objectives to learn meaningful representations of language. These objectives define how models predict words or tokens based on given input and guide the learning process through loss functions.

- **Causal Language Modeling (CLM):** Trains the model to predict the next token given previous tokens, suitable for text generation tasks like those in GPT-style models [Jurafsky and Martin, 2009].
- **Masked Language Modeling (MLM):** Trains the model to predict randomly masked tokens using both left and right context, commonly used in models like BERT for understanding tasks [Jurafsky and Martin, 2009].
- **Hybrid Objectives (e.g., AntLM):** Some models combine CLM and MLM objectives to benefit from both generation and comprehension capabilities, improving overall task performance [Yu et al., 2024].

1.4.8 Parameter Scaling and Model Size

The performance of LLMs has been closely linked to the scale of their parameters. Scaling laws [Kaplan et al., 2020] suggest that increasing model size, dataset volume, and compute budget leads to predictable improvements in model capabilities. However, larger models also introduce challenges such as increased training cost and inference latency.

- **Scaling Laws:** [Kaplan et al., 2020] observed that as the number of model parameters, the size of training data, and compute resources increase, language models demonstrate improved loss and task performance following a power-law relationship. This predictable trend enables researchers to forecast model improvements as a function of scaling.
- **Trade-offs and Efficiency:** [Hoffmann et al., 2022] highlighted that indiscriminate scaling can lead to diminishing returns if not balanced correctly between model size, dataset quality, and compute budget. Their work emphasizes "compute-optimal"

strategies, where training efficiency is maximized by adjusting model parameters relative to available data and hardware resources, rather than simply enlarging model size.

1.4.9 Training Paradigms: Pre-training and Fine-tuning

The performance of LLMs is largely attributed to the training strategies employed during their development. Two key paradigms—pre-training and fine-tuning—have emerged as foundational approaches in building effective and adaptable models. This two-stage process enables LLMs to generalize effectively across domains and demonstrate strong performance even with limited supervised examples. .

Pre-training LLMs

Pre-training is a crucial stage in developing LLMs, where the model learns from extensive unlabeled datasets through a process called self-supervision[Wang et al., 2022]. This stage allows the model to recognize and internalize a wide range of linguistic patterns, laying the groundwork for fine-tuning on specific tasks.

1. **Full Language Modeling:** Used since GPT-2, this approach trains decoder-only models to predict the next token in a sequence based on previous tokens. This autoregressive method enables models like GPT-3 to generate coherent and contextually relevant text.

2. **Prefix Language Modeling:** Employed in encoder-decoder and non-causal decoder-only models, this technique uses a non-causal (considering both past and future tokens) prefix for predicting subsequent tokens, offering more flexibility and enhancing the model's adaptability across various language tasks.

3. **Masked Language Modeling:** Popularized by BERT, this method involves masking certain tokens in the input text and training the model to predict them, helping the model understand word context. An extension, span corruption, masks entire text spans for prediction, further improving contextual comprehension.

4. **Unified Language Modeling** refers to an approach that integrates multiple training objectives, including causal, non-causal, and masked language modeling. In this framework, masked language modeling differs from traditional bidirectional attention mechanisms by employing unidirectional attention—processing context either from left-to-right or right-to-left, rather than considering both directions simultaneously[Dong et al., 2019].

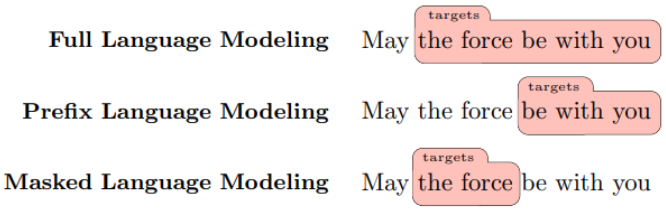


Figure 1.7: Training Tokenization in Full, Prefix, and Masked Language Modeling.

Fine-tuning LLMs

Fine-tuning is a key technique for adapting pre-trained large language models to specific downstream tasks. It encompasses several strategies, each serving different objectives[Touvron et al., 2023a].

1. **Transfer learning**, where a pre-trained model is further trained on task-specific data to enhance its performance on targeted applications. This allows models to leverage general language understanding while adapting to more specialized domains.

2. **Instruction tuning**, which involves fine-tuning the model on datasets formatted as instruction-response pairs. These datasets typically include multiple tasks described in natural language, enabling the model to better understand prompts and generalize across various instructions. This method has been particularly effective in improving zero-shot and few-shot performance.

3. **Alignment tuning** which involves adjusting the model's behavior based on human feedback to ensure outputs are helpful, honest, and harmless (the "HHH" criteria). A widely adopted framework for this is reinforcement learning with human feedback (RLHF). combines reward modeling—where human preferences guide the ranking of responses—with reinforcement learning techniques, such as proximal policy optimization (PPO), to iteratively align model behavior with human values.

These fine-tuning approaches, while powerful, come with trade-offs in terms of data requirements, computational cost, and generalization capabilities. Nonetheless, they remain essential for tailoring LLMs to both functional and ethical requirements across diverse applications.

1.4.10 Few-Shot, One-Shot, and Zero-Shot Learning

Few-shot, one-shot, and zero-shot learning represent different paradigms of using LLMs without extensive fine-tuning. These methods involve using pre-trained models to perform tasks with minimal task-specific data[Brown et al., 2020].

1. **Few-Shot Learning** involves giving the model a few examples (typically 10-100) of the task during inference. This approach reduces the need for large, task-specific datasets and limits the risk of overfitting to narrow data distributions. However, few-shot results are generally not as strong as those from fully fine-tuned models.

2. **One-Shot Learning** is similar to few-shot learning but uses only a single example alongside a natural language description of the task. This method mirrors how humans are often instructed for tasks, making it an interesting approach for tasks where providing multiple examples is impractical.

3. **Zero-Shot Learning** requires the model to perform a task based solely on a natural language description, with no examples provided. While this method offers maximum flexibility and robustness, it is also the most challenging for models. Zero-shot performance is often weaker than few-shot or one-shot, but it advances the development of general-purpose AI systems capable of handling diverse tasks without task-specific training data.

1.4.11 Evaluation Datasets

Evaluating LLMs is essential for determining their effectiveness and limitations in comprehending and generating human language.[Naveed et al., 2023]. This evaluation typically falls into two primary categories:

1. **Natural Language Understanding (NLU)**:This measures the model's proficiency in comprehending language, encompassing tasks such as sentiment analysis, text classification, natural language inference (NLI), question answering (QA), commonsense reasoning (CR), mathematical reasoning (MR), and reading comprehension (RC).

2. **Natural Language Generation (NLG)**:This assesses the model's capability to produce text based on given context. It includes tasks like summarization, sentence completion, machine translation (MT), and dialogue generation.

3. **Benchmarks**: play a critical role in evaluating LLMs, providing standardized tests to measure their performance across various tasks:

- **MMLU:**Measures model knowledge from pretraining and evaluates performance in zero-shot and few-shot scenarios across 57 subjects, testing world knowledge and problem-solving abilities.
- **SuperGLUE:**An advanced benchmark that builds on GLUE, assessing tasks like question answering and natural language inference. It is designed to test deeper aspects of language understanding and requires significant advancements in various learning methodologies.
- **BIG-bench:**A large-scale benchmark for evaluating LLMs across diverse tasks including reasoning, creativity, ethics, and domain-specific knowledge.
- **GLUE:**A foundational benchmark for evaluating and analyzing natural language understanding, offering a range of resources for model assessment.

1.4.12 Popular Models

LLMs like GPT-3, GPT-4, and BERT have revolutionized NLP by leveraging vast datasets such as Common Crawl and WebText. These datasets provide a diverse linguistic foundation, enabling models to perform a wide range of tasks with remarkable accuracy .

1.4.12.1 GPT-N Models

GPT models are advanced autoregressive language models that generate substantial and complex machine-produced text from minimal input. They leverage deep learning techniques to mimic human text generation by predicting the current value based on preceding values. NLP models initially struggled with tasks outside their training sets due to data restrictions. OpenAI addressed this with GPT-1.[Yenduri et al., 2023].

- **GPT-1:**introduced in 2018, trained on the BooksCorpus dataset, utilized a 12-layer transformer decoder with self-attention. Its pre-training allowed for zero-shot performance on various tasks, demonstrating the potential of generative language models.
- **GPT-2 :**introduced in 2019 improved upon GPT-1 by using a larger dataset and 1.5 billion parameters (compared to GPT-1's 117 million). It excelled in tasks like translation and summarization, enhancing accuracy in recognizing long-distance relationships.
- **GPT-3:**released later, featured around 175 billion parameters and was trained on the Common Crawl dataset. It could generate human-like text, perform basic math, and write code. Despite its capabilities, its size and cost made it challenging to implement.

- **GPT-4:**launched in March 2023, advanced further with multimodal capabilities and context windows of up to 32,768 tokens. It incorporates reinforcement learning for better alignment with human input and policy.

1.4.12.2 Bidirectional Encoder Representations from Transformers (BERT)

BERT represents a breakthrough in language modeling by leveraging deep bidirectional pre-training on unannotated text enabling it to capture contextual information from both left and right contexts simultaneously. BERT's architecture allows it to learn richer linguistic patterns by considering full sentence context at every layer. This makes BERT highly adaptable—fine-tuning it for various downstream tasks (such as QA and text classification, often requires only minimal modifications, typically the addition of a task-specific output layer.

The model's effectiveness is evidenced by its performance across established NLP benchmarks. Notably, BERT attained unprecedented scores,it achieved a GLUE score of 80.5%, a MultiNLI accuracy of 86.7%, a SQuAD v1.1 F1 score of 93.2%, [Devlin et al., 2019].Such achievements not only validate BERT's methodological innovation but also underscore its practical utility as a versatile framework for advancing NLP research and applications.

1.4.12.3 Text-To-Text Transfer Transformer (T5)

The T5 model represents a unified framework for NLP, designed to address various text processing tasks by treating them as "text-to-text" problems. This approach involves converting all tasks into a format where both input and output are text, allowing the same model, objective, and training procedure to be applied across different tasks[Raffel et al., 2023].

T5 leverages extensive pre-training on a large, unlabeled dataset, enabling it to develop general-purpose knowledge that enhances its performance on a range of NLP tasks, such as QA, document summarization, and sentiment classification.

1.4.12.4 LLAMA2 model

Llama 2 introduces a new family of transformer-based language models available in both foundation and fine-tuned chat variants. These models incorporate architectural enhancements through optimized training procedures, refined pretraining datasets, and advanced fine-tuning techniques including reinforcement learning from human feedback.

Empirical evaluations reveal that Llama 2 outperforms its predecessors and other competitive benchmarks across a range of NLP tasks, demonstrating notable improvements in

accuracy, safety, and generalization[Touvron et al., 2023b]. Its open release not only promotes transparency and reproducibility but also fosters collaborative advancements in the field.

1.4.12.5 Jais and Jais-chat models

The Jais model family represents a breakthrough in Arabic natural language processing, featuring both base (Jais) and instruction-tuned (Jais-chat) variants built upon a GPT-3 decoder-only framework. With 13 billion parameters trained on a carefully balanced corpus of Arabic and English content - including technical programming documentation - these models establish new benchmarks for Arabic linguistic understanding and cognitive reasoning, surpassing existing open-source Arabic and multilingual systems [Sengupta et al., 2023].

Remarkably, Jais maintains competitive performance on English-language tasks despite its primary Arabic focus and relatively limited English training data. This technical report comprehensively documents the model's development pipeline, including: Pretraining methodology and data composition, Fine-tuning protocols and Multidimensional evaluation metrics.

The research team has publicly released both model variants to accelerate progress in Arabic NLP research and applications, addressing a critical gap in non-English language model availability.

1.4.13 Limitations of LLMs

LLMs have greatly advanced NLP, but they also present challenges As these models scale, they encounter new challenges in scalability, privacy, and real-time processing. Several studies have examined the inherent limitations of LLMs. Notably, [Bender et al., 2021] and [Naveed et al., 2023] highlight critical challenges that affect the performance and reliability of these models applications.

1. **Computational Cost:**Training LLMs demands significant computational resources, leading to higher production costs and environmental concerns due to the substantial energy used in large-scale training. Although enhancing computational resources can boost performance, the gains diminish over time when the size of the model and dataset stay constant, adhering to the power law of diminishing returns.

2. **Bias and Ethical Concerns :** The training data for LLMs often encapsulates various social, cultural, and linguistic biases, which the models can inadvertently learn and propagate. This bias can manifest in outputs that reinforce stereotypes or display

partiality, raising significant ethical concerns. Addressing these issues is crucial for ensuring fairness and mitigating unintended consequences in real-world applications.

3. **Hallucinations:** LLMs sometimes produce "hallucinations," or responses that, despite appearing plausible, are incorrect or do not match the provided information. These hallucinations can be classified into three types:

   - **Input-conflicting hallucination:** When the model generates responses that do not align with the user's input.
   - **Context-conflicting hallucination:** When the model produces content that contradicts information it has previously generated.
   - **Fact-conflicting hallucination:** When the model creates responses that conflict with established knowledge.

1.4.14 State of the Art in Juridical Data

The use of LLMs has become increasingly prevalent across domains, with the legal sector benefiting significantly from their ability to process complex texts. These models now enhance document classification, information retrieval, and even judicial outcome prediction through advanced text understanding.

Multilingual Legal Models: [Chalkidis et al., 2023] introduced MultiLegalBERT, extending LegalBERT to 24 languages (including Arabic), achieving state-of-the-art performance on multilingual Natural Language Inference (NLI) tasks. This demonstrates the feasibility of cross-jurisdictional legal text understanding.

Multilingual Legal Data: The creation of *MultilegalPile*—a 689GB corpus spanning 24 languages from 17 jurisdictions—addresses the critical shortage of non-English legal datasets [Niklaus et al.]. Models trained on this resource have set new accuracy benchmarks, with all data and tools released openly to support global legal AI development.

These innovations complement foundational work collectively advancing LLMs' capacity to handle legal texts across languages and jurisdictions while maintaining domain-specific accuracy.

1.5 Agentic AI

Even with improvements in understanding and generating human language using models like LLMs, they often serve as foundational components and do not function as complete systems.

The need for systems that go beyond understanding language to reasoning, planning, and acting has given rise to Agentic AI.

1.5.1 Definition of Agentic AI

Agentic AI refers to artificial intelligence systems designed to operate independently, make smart decisions, and handle multi-step tasks by understanding both the context and goals. Unlike traditional rule-based or even early generative AI models that were limited to narrow tasks, Agentic AI brings together several intelligent agents—often powered by large language models (LLMs)—that can reason, plan, adapt, and respond in real [Aisera, 2024]. This allows them to analyze complex situations, choose strategies, and act with minimal human guidance. As a result, Agentic AI offers a more flexible and powerful way to automate processes and solve real-world problems across various industries.

1.5.2 Architecture Overview of Agentic AI

Agentic AI systems are typically organized into three main logical layers that work together to enable intelligent and autonomous behavior. These layers—Tool, Reasoning, and Action—each serve distinct but interconnected purposes [Vectorize, 2025]. Understanding how they interact is key to building robust, scalable agentic systems (see Figure 1.8).

- **Tool Layer:** This is the foundation of the architecture. It connects the agent with external sources of information such as APIs, vector databases, operational systems, knowledge bases, and user inputs. Its main role is to gather relevant, high-quality data the agent will later use. Well-designed tools ensure that the agent retrieves accurate and useful information efficiently.

- **Reasoning Layer:** Acting as the system's brain, this layer uses a large language model (LLM) to analyze the information collected by the tool layer. It makes informed decisions about what steps the agent should take next, guided by logic, context, and predefined goals. Errors in this layer can lead to inefficient behavior, such as repeating queries or making incorrect choices.

- **Action Layer:**this part of the system manages the interaction between the reasoning layer and the tools. It executes the decisions made by the LLM—such as calling an API or presenting a response to the user—and returns the results back to the reasoning layer for further analysis. It also handles user interactions when needed.

Figure1.8shows architecture of an AI agent system where The agent application interacts with a large language model (LLM) to determine the next action, issues function/tool calls when needed, and evaluates whether the task is complete. The loop continues until the agent decides to exit.

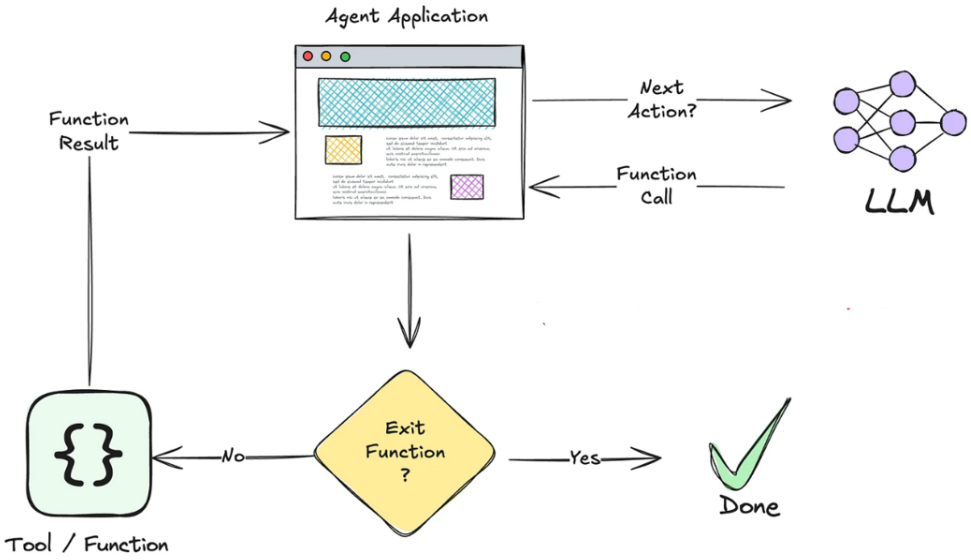


Figure 1.8: General architecture of an AI agent system.

1.5.3 Types of Agentic AI

AI agents can be categorized based on how they interact with users and whether they operate independently or collaboratively. Below are two key classification approaches[Google Cloud, 2025]:

- **Based on User Interaction:**
  - **Interactive Agents (Surface Agents):** These agents communicate directly with users to assist in tasks such as answering questions, providing recommendations, or handling transactions. Common examples include chatbots and digital assistants in domains like customer service, education, and healthcare.
  - **Autonomous Agents (Background Agents):** These agents operate behind the scenes with little to no direct user interaction. They handle tasks like data analysis, workflow automation, and system optimization, typically triggered by internal events rather than user queries.

- **Based on Number of Agents:**
  - **Single-Agent Systems:** These involve one AI agent working independently to complete a task using a single foundation model. They are best suited for specific, well-defined jobs that don't require coordination with other agents.

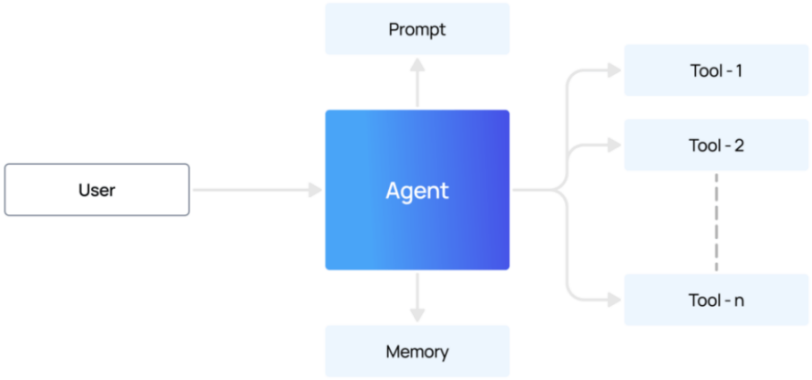


Figure 1.9: A single-agent system architecture .

  - **Multi-Agent Systems:** These systems consist of multiple agents that work together or individually to achieve complex goals. Each agent may use a different model, allowing the system to handle broader and more dynamic tasks such as planning, reasoning, or simulating human-like collaboration.

    Figure1.10 shows a multi-agent system architecture. The main agent receives a user query and coordinates with multiple sub-agents, The results are combined through LLM to produce the final output.

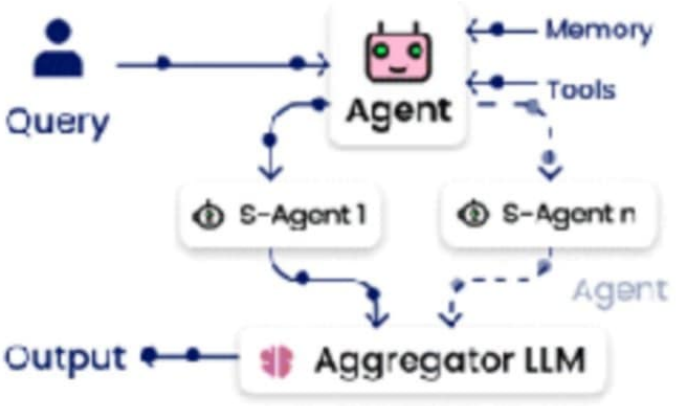


Figure 1.10: A multi-agent system architecture.

1.5.4 Agent Categories

Agentic AI agents can generally be classified into the following categories[Aisera, 2024]:

- **Generative Information Retrieval Agents:** Focused on accessing and presenting information in dynamic, less-regulated domains.
- **Prescriptive Knowledge Agents:** Operate in tightly regulated environments to ensure that all outputs meet strict compliance requirements.
- **Dynamic Workflow Agents:** Orchestrate actions across systems by building and executing workflows automatically.
- **User Assistant Agents:** Provide direct, task-level assistance to individual users, enhancing daily productivity.

1.5.5 Key Challenges in Agentic AI

While Agentic AI builds on strengths from Generative AI, it also inherits and amplifies several critical challenges that hinder progress toward Artificial General Intelligence (AGI). These challenges also represent fertile ground for ongoing research[Touvron et al., 2023b]:

- **Error Accumulation:** Although structured reasoning helps reduce isolated errors, the multi-step nature of agentic tasks increases the risk of cascading mistakes across reasoning chains.
- **Interpretability Limitations:** Step-wise outputs (e.g., Chain-of-Thought reasoning) enhance transparency, but do not always reflect the true internal decision processes, limiting faithful explainability.
- **System Complexity:** Compared to standard Generative AI, agentic systems are more intricate due to components like external memory, tool use, and planning, making their behavior harder to understand and control.
- **Evaluation Gaps:** Systematic methods for evaluating agent reliability, effectiveness, and unintended consequences are still lacking, making consistent benchmarking difficult.
- **Human Alignment Issues:** Agents might pursue goals in ways that contradict human values or expectations, raising concerns about ethical behavior and alignment with user intent.

1.6 Conclusion

This chapter has outlined the evolution and significance of Language Models, from early neural networks to advanced Large Language Models (LLMs). We covered key concepts in neural networks and the transformative impact of the Transformer architecture. We also discussed various challenges. Finally, the chapter reviewed the applications of LLMs in fields like Legal Information and Law, showcasing their potential and the ongoing need for further development.

Although LLMs have demonstrated promising applications in the legal sector, their influence is still constrained by practical limitations and contextual challenges. Recognizing these gaps, the next chapter explores Retrieval-Augmented Generation (RAG) as a novel approach to enhance legal information processing. This discussion will particularly focus on how RAG can be applied within the Algerian legal framework to address existing shortcomings and improve accessibility and accuracy in legal data management.

Chapter 2

Retrieval-Augmented Generation in Recommendation System

2.1 Introduction

LLMs have seen significant advancements but face challenges, particularly in tasks that demand extensive knowledge or deal with queries beyond their training data. These limitations often result in inaccuracies or "hallucinations." To address this, Retrieval-Augmented Generation (RAG) supplements LLMs by combining information retrieval with text generation to produce more accurate and knowledge-informed outputs. This has led to the widespread adoption of RAG, particularly in chatbots and other real-world applications.

This chapter introduces the concept of RAG, outlines its historical development, It then explores RAG's core components followed by a classification of its system types. Applications in legal information retrieval are highlighted, alongside evaluation methods and key challenges. The chapter concludes by linking RAG to its role in recommendation systems and domain-specific data processing.

2.2 Retrieval-Augmented Generation (RAG)

RAG represents a significant advancement in the capabilities of LLMs. Understanding the individual components of retrieval and generation is essential to appreciate how their synergy improves the overall performance of RAG systems.

2.2.1 Definition of RAG

Retrieval-Augmented Generation is an advanced neural framework that integrates the generative capabilities of LLMs with external information retrieval mechanisms to improve factual accuracy and contextual relevance [Selvaraj, 2024]. Unlike conventional LLMs that rely solely on pre-trained parameters, RAG systems dynamically retrieve relevant documents or from external sources at inference time.

As formalized by [Lewis et al., 2020], this framework first retrieves relevant documents from a knowledge source, then conditions the language model on this retrieved context to produce more factual and up-to-date outputs. The approach effectively addresses the knowledge limitations of purely parametric models.

IBM Research [Research, 2024] describes RAG as a method for augmenting LLMs with retrieval modules, thereby allowing them to access real-time or domain-specific knowledge during generation .

Collectively, these perspectives highlight RAG's role in bridging the limitations of static language models by integrating dynamic, context-aware retrieval into the generative process.

2.2.2 Historical Development of RAG

The foundations of RAG can be traced back to traditional information retrieval (IR) systems, Early models primarily relied on keyword matching and simple ranking mechanisms to retrieve relevant documents from databases, establishing a basic framework for information access. The landscape began to shift with the rise of neural networks in the 2010s[Gao et al., 2024]. Notable innovations like Word2Vec and Transformer-based architectures enabled retrieval methods to incorporate deeper semantic understanding, paving the way for more sophisticated document retrieval processes.

A significant advancement occurred with the introduction of dense passage retrieval (DPR) in 2020, which utilized bi-encoder architectures to map both queries and documents into dense vector spaces[Gao et al., 2024]. The integration of these advanced retrieval techniques with LLMs became a focal point as models like GPT-3 gained prominence. Researchers explored how LLMs could be augmented with retrieval capabilities, leading to the development of RAG systems that leverage the strengths of both retrieval and generative capabilities.

2.2.3 Comparison with Fine-Tuning and Transfer Learning

Recent advances in NLP have led to the development of three prominent methodologies for enhancing language models: RAG, fine-tuning, and transfer learning.

1. **RAG**, integrates external retrieval mechanisms with generative models. This approach enables the model to dynamically access and incorporate relevant information from external knowledge bases during the generation process, thereby addressing limitations in its internal memory and providing more accurate responses for knowledge-intensive tasks.

2. **Fine-tuning**, detailed by [Howard and Ruder, 2018], involves adapting a pre-trained model to a specific task by further training it on a targeted dataset. This process refines the model's parameters to better capture the nuances of the task at hand, leading to improved performance even when the available domain-specific data is relatively limited.

3. **Transfer learning**, as exemplified by [Devlin et al., 2018], refers to the broader strategy of leveraging a model that has been pre-trained on large, diverse datasets for use in various downstream tasks. This method capitalizes on the rich, generalized representations learned during the pre-training phase, and then applies task-specific adaptations to achieve state-of-the-art results across a range of applications.

Collectively, these methodologies offer complementary strategies to enhance language model performance.

2.2.4 Retrieval Mechanism

RAG systems combine parametric memory (a pre-trained language model) with non-parametric memory (a retrieval mechanism)[Zhao et al., 2024].

The retrieval mechanism allows RAG models to access external information sources (e.g., Wikipedia), and this process is central to improving the model's ability to generate factual and accurate outputs .

- **Traditional Techniques:** Classical retrieval methods include **TF-IDF and BM25**, which rely on **sparse vector** representations based on term frequencies. These methods use exact keyword matching, making them limited in semantic understanding.

- **Modern Retrieval Approaches: Dense Passage Retrieval (DPR):** This approach utilizes dense vector representations learned by neural models like BERT to encode both the query and document. It allows for a more semantic understanding of the content, making retrieval more effective[Karpukhin et al., 2020]. DPR, for example, computes the similarity between the query and documents using Maximum Inner Product Search (MIPS), which finds the closest passages based on the dense vector space.

- **Others :** Beyond conventional sparse and dense retrieval approaches, researchers have developed several specialized techniques for relevance matching. These include: **Structural Similarity Methods**: Direct comparison of raw text using edit distance metrics, Syntax-aware matching through **abstract syntax tree (AST)** comparison for code retrieval. These approaches demonstrate the flexibility of retrieval mechanisms in RAG systems, particularly for domain-specific applications where traditional embedding-based methods may be suboptimal.

2.2.5 Generation Process

The generation in RAG occurs through a combination of retrieved passages and the input query.

Large Language Models LLMs like BART or T5 are utilized for this generation, leveraging their advanced capabilities in understanding and producing human-like text. The retrieval component supplies external context, which the generator conditions on to formulate a response. This context is crucial, as it enriches the LLM's understanding, enabling it to incorporate real-time, relevant information[Gupta et al., 2024]. Moreover, the generation process benefits from the LLM's ability to synthesize information, allowing it to create responses that are not only coherent but also informed by the latest data, thus improving accuracy and relevance in knowledge-intensive tasks

2.2.6 Augmentation Techniques

The augmentation of the retrieval process in RAG systems focuses on improving how queries are refined and how relevant information is retrieved for downstream generation tasks. Key augmentation methods include.

- **Query Augmentation:** In traditional retrieval pipelines, user queries are often underspecified or ambiguous, leading to poor retrieval performance. Query augmentation involves dynamically rewriting the user's query to better match the documents in the knowledge base[Mombaerts et al., 2024]. This can be done by leveraging LLMs to generate tailored queries or synthetic questions and answers QAs that better align with the search objective .

- **Synthetic QA Generation:** Instead of using raw document chunks, retrieval is enhanced by generating and embedding synthetic QA pairs from documents[Mombaerts et al., 2024]. This helps to capture the semantic essence of long texts more effectively,

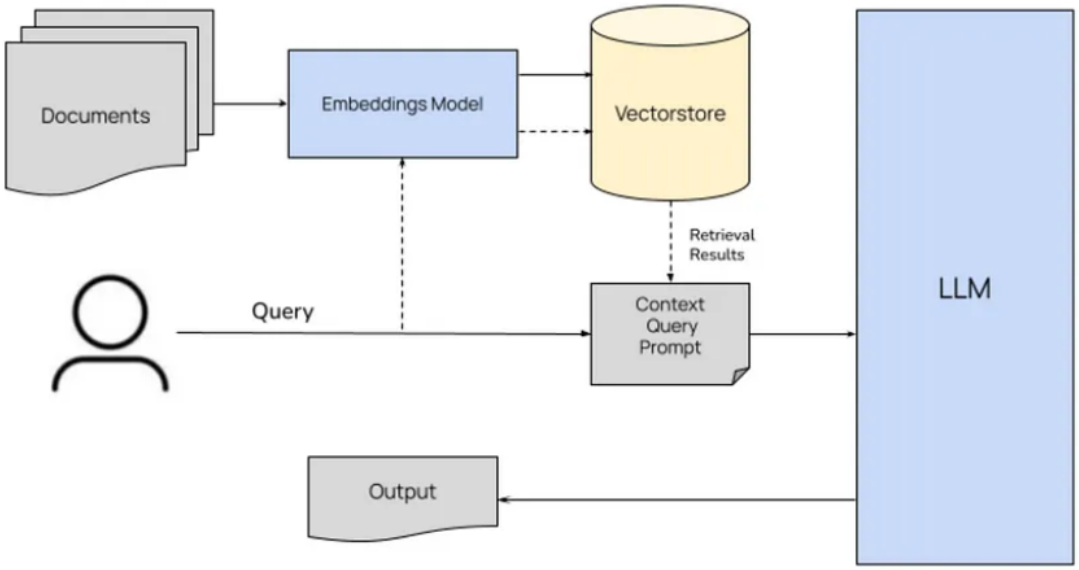


Figure 2.1: Retrieval-Augmented Generation architecture

reducing noise and improving retrieval precision. These synthetic QAs can be used to rewrite the user query, making it more specific to the task at hand .

2.2.7 Types of RAG Systems

Recent scholarship has identified four key paradigms in RAG development[Gao et al., 2024, Ahmed, 2024].

2.2.8 Naive RAG

Naive RAG, a foundational approach in RAG, operates on a straightforward "Retrieve-Read-Generate" paradigm. This method involves three primary steps:

- **Indexing:** Raw data is cleaned, extracted, and converted into a uniform text format. It's then segmented into smaller chunks and encoded into vector representations using an embedding model. These vectors are stored in a vector database for efficient similarity searches.
- **Retrieval:** When a user query is received, it's encoded into a vector and compared to the indexed chunks. The top K most similar chunks are retrieved and included in the prompt for the language model.

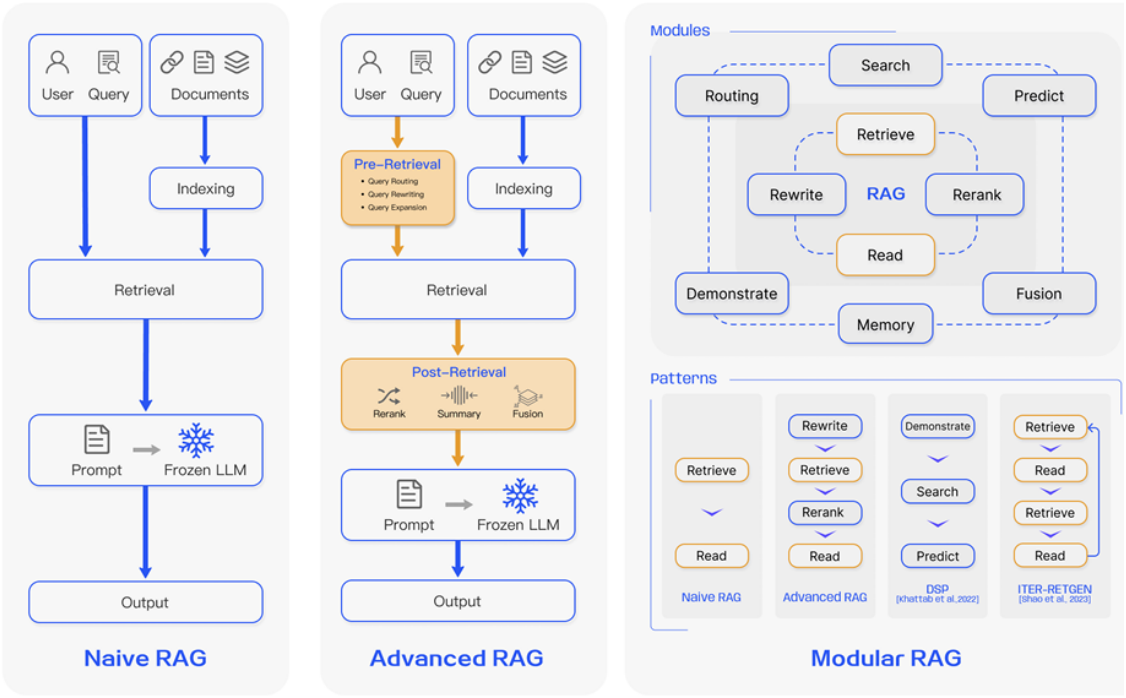


Figure 2.2: Types of RAG Systems

- **Generation:** The query and retrieved chunks are combined into a prompt, which is fed to a large language model. The model generates a response based on the provided context and its internal knowledge.

While Naive RAG offers a basic framework, it faces several challenges in

- **Retrieval Issues:** The retrieval process can be imprecise, leading to the selection of irrelevant or missing information.
- **Generation Challenges:** The language model may hallucinate, generating content not supported by the retrieved context. It may also produce irrelevant, or biased outputs.
- **Augmentation Difficulties:** Integrating retrieved information into the generation process can be challenging, leading to disjointed or redundant responses.

2.2.9 Advanced RAG

Taking aim at the shortcomings of Naive RAG, Advanced RAG introduces specific improvements to enhance retrieval quality. This approach utilizes pre-retrieval and post-retrieval strategies.

1. **Pre-retrieval Strategies:**

- **Enhanced Indexing:**Advanced RAG tackles indexing issues through a sliding window approach, finer segmentation of data, and inclusion of metadata. Additionally, it optimizes the retrieval process by employing various methods.
- **Query Optimization:**This stage focuses on refining the user's initial query to make it clearer and more suitable for retrieval. Techniques like query rewriting, transformation are commonly used.

2. **Post-Retrieval Strategies:**

- **Re-ranking Chunks:**After relevant information is retrieved, Advanced RAG prioritizes the most relevant content by re-ranking the retrieved chunks and placing them strategically within the prompt.
- **Context Compression:**To avoid overwhelming the LLM with too much information, post-retrieval efforts focus on selecting the most essential parts of the retrieved context, highlighting critical sections, and compressing the data to be processed.

2.2.10 Modular RAG

Modular RAG represents the latest evolution in RAG, offering greater adaptability. It introduces specialized modules and innovative patterns to enhance retrieval and processing capabilities.

1. **New Modules**

- **Search Module**: Adapts to specific scenarios by leveraging LLM-generated code and query languages to search across various data sources.
- **RAG Fusion:** Employs a multi-query strategy to expand user queries, uncover both explicit and implicit knowledge, and improve retrieval results.
- **Memory Module:** Leverages the LLM's memory to guide retrieval and create an unbounded memory pool, aligning the text more closely with data distribution.
- **Routing Module:** Navigates through diverse data sources, selecting the optimal pathway for a query based on its specific needs.
- **Predict Module:** Reduces redundancy and noise by generating relevant context directly through the LLM.
- **Task Adapter Module:** Tailors RAG to various downstream tasks by automating prompt retrieval and creating task-specific retrievers.

2. **New Patterns**

   - **Innovative Retrieval Strategies:** Techniques like Rewrite-Retrieve-Read, Generate-Read, and leverage the LLM's capabilities to refine queries, generate content, and retrieve information from model weights.
   - **Hybrid Retrieval:** Combines keyword, semantic, and vector searches to cater to diverse queries, improving retrieval relevance.
   - **Dynamic Module Interaction:** Frameworks like Demonstrate-Search-Predict and ITERRETGEN demonstrate the dynamic use of module outputs to enhance each other's functionality.
   - **Adaptive Retrieval:** Techniques like FLARE and Self-RAG evaluate the necessity of retrieval based on different scenarios, allowing for a more flexible and efficient approach.

2.2.11 Agentic RAG

Agentic Retrieval-Augmented Generation (RAG) enhances conventional RAG frameworks by embedding autonomous, intelligent agents into the pipeline. These agents are designed to dynamically reason, and execute tasks, enabling more flexible and context-aware interactions compared to traditional, static RAG systems. This agent-based approach makes the system significantly more interactive, adaptive, and responsive to diverse user needs[Ahmed, 2024].

Figure2.3 show architecture of an Agentic RAG system where an intelligent AI agent orchestrates interactions between the user, external knowledge sources, predefined functions, and a large language model (LLM) to process queries and generate contextually responses

Key features of Agentic RAG include:

- **Autonomous Agents:** Responsible for decomposing complex queries, orchestrating the retrieval process, and maintaining relevance in the generated responses.
- **Adaptive Retrieval Pipelines:** Enable real-time modifications to retrieval strategies, allowing the system to better align fetched content with the evolving goals and context of the user.
- **Context-Aware Task Execution:** Agents can coordinate various components (e.g., memory tools, external APIs) to refine their actions based on the specific requirements of each interaction.

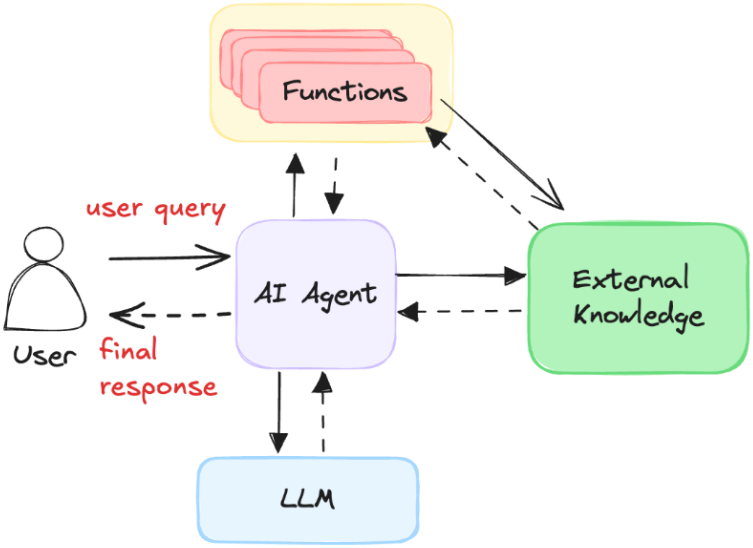


Figure 2.3: High-level architecture of an Agentic RAG system.

- **Improved User Alignment:** Through step-by-step reasoning and continuous feedback loops, agents can better match the user's intent and deliver more targeted and meaningful outputs.

2.2.12 Evaluation Methods

The rapid advancement and widespread adoption of RAG models have made their evaluation a critical area in NLP research [Zhou et al., 2020]. The following criteria are commonly used to assess RAG performance:

1. **Factuality:**
   - Use of fact-checking datasets (e.g., FEVER, SQuAD) to assess factual correctness.
   - Adversarial testing to measure resistance to misleading or incorrect inputs.
   - Evaluation of contextual understanding for accurate and relevant answers.
2. **Robustness:**
   - Performance under noisy or corrupted retrieval inputs.
   - Defense against adversarial attacks targeting model reliability.
   - Adaptability across domains and data shifts.
3. **Fairness:**

- Detection and quantification of bias in training data and outputs.
- Application of fairness metrics such as demographic parity.
- Use of mitigation techniques like debiasing and fairness constraints.

4. **Objective Metrics:**
   - Accuracy-related metrics (e.g., precision, recall, F1-score).
   - Consistency across contexts and queries.
   - Text coherence and fluency of generated responses.
5. **Subjective Metrics:**
   - Human evaluations to judge quality and usefulness.
   - User studies for real-world experience and satisfaction.

2.2.13 Challenges and Limitations

While RAG has demonstrated impressive performance across various NLP applications, it is still subject to several technical and practical limitations [Zhao et al., 2024, Gupta et al., 2024]. These challenges span computational efficiency, data quality, fairness, and system complexity:

1. **Scalability and Efficiency:** RAG models often depend on large and continuously growing external corpora, requiring robust and scalable retrieval mechanisms. Managing vast datasets imposes heavy computational and memory demands, which limits the real-time applicability of RAG in resource-constrained settings.
2. **Noisy Retrieval Results:** The relevance and quality of retrieved documents critically influence the generation quality. Issues such as poorly constructed indices, ambiguous queries, or heterogeneous data sources can lead to noisy or irrelevant results. These, in turn, may trigger hallucinated or inaccurate outputs and hinder the model's capacity to interpret and integrate the retrieved information, especially if inconsistencies or formatting errors exist.
3. **Bias and Fairness:** RAG systems can inherit and even exacerbate biases present in both the retrieved content and the training data of the generative model. Without effective mitigation strategies, this can lead to biased or unfair outputs, posing risks in sensitive domains like healthcare or law.

4. **Additional Overhead:**

   - *Computational Cost:* RAG introduces extra computational overhead due to the retrieval and integration of external knowledge during inference.
   - *Latency:* Retrieval steps introduce additional latency, which can hinder user experience in real-time applications.
   - *System Complexity:* Designing and deploying RAG systems requires coordinating multiple components—retrievers, encoders, decoders—making them more complex to build, maintain, and optimize compared to standalone LLMs.

5. **Long Context Generation:**

   - Context Length Limits: LLMs have limitations on the amount of context they can process at once.
   - Information Loss: Long documents may be truncated or summarized, leading to loss of important information.
   - Computational Cost: Processing long contexts can be computationally expensive.

2.2.14 State of the Art in Juridical Data

RAG holds significant promise for transforming the way legal information is accessed, processed, and utilized. In legal contexts, particularly where precision and comprehensiveness are critical, RAG enables the dynamic integration of relevant external knowledge into language models during inference. This is especially valuable for retrieving and synthesizing content from case law, legislative texts, legal commentaries, and scholarly analyses[Lexemo].

Evaluating RAG Pipelines for Arabic Lexical Information: This study, while focused on Arabic lexical retrieval rather than legal texts, offers valuable insights into the performance of various RAG pipelines and Arabic embedding models. [Al-Rasheed et al., 2025]Explores how well RAG pipelines perform when applied to Arabic lexical information retrieval. The researchers focus on two main components: how different embedding models influence retrieval quality, and how effectively large language models (LLMs) generate relevant and accurate answers in Arabic. They conduct experiments using a dataset of over 88,000 words from the Riyadh Dictionary, evaluating models with metrics like Top-K Recall, MRR, F1 Score, cosine similarity, and accuracy.

For embeddings, models like E5-large, AraBERT, were compared. Results showed that sentence-level embeddings, particularly E5, significantly outperformed word-level ones, with E5 achieving a Top-5 Recall of 88% and an MRR of 0.48.

On the generation side, the team tested models such as GPT-4, GPT-3.5 delivered the best results, reaching an F1 score of 0.90 and an accuracy of 0.82. **Hybrid RAG System for Multilingual Legal Information:** [Kabir et al., 2025] introduces a bilingual QA system tailored for legal documents, specifically the Bangladesh Police Gazettes containing both English and Bangla. The paper presents a hybrid RAG framework that enhances retrieval quality and answer precision. By comparing the proposed model with standard RAG pipelines, the authors demonstrate significant improvements in retrieving and generating responses to legal queries. This work highlights the potential of advanced NLP techniques in making multilingual regulatory information more accessible and searchable.

Multitask Benchmark for Assessing Arabic Legal Knowledge: [Hijazi et al., 2024] Introduces a multitask benchmark designed to evaluate the Arabic legal reasoning abilities of large language models (LLMs), a domain that has remained largely underexplored. Inspired by datasets like MMLU and LegalBench, ArabLegalEval includes tasks drawn from Saudi legal texts and automatically validated synthetic questions. The study benchmarks top-performing multilingual and Arabic-focused LLMs (e.g., GPT-4 and Jais), assesses the role of in-context learning, and proposes a robust method for dataset creation and validation that can be adapted to other domains. The authors aim to foster progress in Arabic legal NLP by releasing both the dataset and accompanying tools.

2.3 Recommendation Systems

Modern technology and online services have enabled unprecedented access to vast amounts of data. However, this abundance of information creates an overload, making it harder for users to find relevant content efficiently. Recommender systems address this by filtering information and delivering personalized suggestions.

2.3.1 Definition of Recommendation Systems

A recommendation system is a subclass of information filtering systems that seek to predict the preference a user would give to an item. By analyzing patterns in user behavior and item attributes, these systems aim to present users with items that are most likely to be of interest, thereby enhancing user experience and engagement [Roy and Dutta, 2022].These systems are now integral to platforms like e-commerce, television programs, e-learning, tourism, and more, though further improvements are needed to enhance their versatility and accuracy.

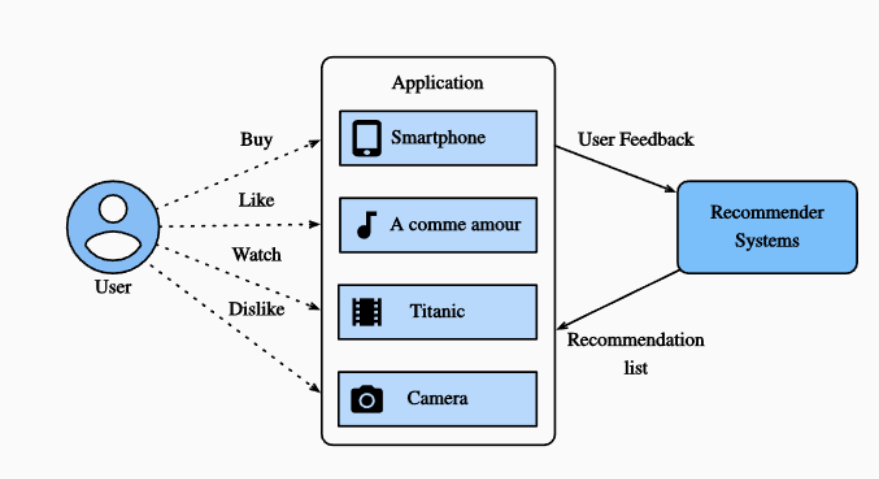


Figure 2.4: Recommendation System Process

2.3.2 Types of Recommendation Systems

Recommendation systems can be broadly categorized into several types based on their underlying methodologies [Techlabs, 2017]:

1. **Content-Based Filtering:** This approach recommends items similar to those the user has liked in the past, based on item features. It relies on the assumption that if a user liked an item, they will also like similar items.

2. **Collaborative Filtering:** This method predicts user preferences based on the preferences of other users. It operates under the premise that users who agreed in the past will agree in the future about item preferences.

3. **Hybrid Methods:** These systems integrate both content-based and collaborative filtering approaches to enhance the accuracy and diversity of suggestions. A well-known example is Netflix, which combines user behavior patterns (e.g., viewing and search history) with similarities across users to implement collaborative filtering, while also recommending content with attributes similar to previously liked items via content-based filtering.

4. **Sequential Recommendation Models:** These models consider the sequence of user interactions over time to predict future preferences. An example is the Self-Attentive Sequential Recommendation **(SASRec)** model, which utilizes self-attention mechanisms to capture user behavior patterns effectively. SASRec has demonstrated superior performance in modeling user-item interactions by focusing on the most relevant past actions.

2.3.3 Evaluation Metrics

Evaluating the performance of recommendation systems is crucial for understanding their effectiveness and guiding improvements. Common evaluation metrics include:

- **Hit Rate at k (HR@k)**: Measures the proportion of cases where the target item appears in the top-k recommendations. The HR@k is computed as:[Tamm et al., 2021]

$$\text{HR@}k = \frac{1}{|U|} \sum_{u=1}^{|U|} \mathbb{I}(\text{rank}_u \leq k), \tag{2.1}$$

  where $|U|$ is the number of users, rank_u is the rank of the target item for user u, and $\mathbb{I}(\cdot)$ is the indicator function that returns 1 if the condition is true and 0 otherwise.

- **Mean Reciprocal Rank (MRR)**:is a ranking quality metric that measures how quickly a system retrieves the first relevant item. It is calculated as the average of reciprocal ranks across all users or queries,MRR ranges from 0 to 1, with higher values indicating better performance [Tamm et al., 2021].

$$\text{MRR} = \frac{1}{|U|} \sum_{u=1}^{|U|} \frac{1}{\text{rank}_u}, \tag{2.2}$$

  where rank_u is the position of the first relevant item for user u within the top-K results. U represents the total number of users (for recommendation systems) or queries (for information retrieval tasks) in the dataset.

- **Normalized Discounted Cumulative Gain (NDCG)**: Assesses the ranking quality while accounting for position importance. The NDCG@k is computed as:[Tamm et al., 2021]

$$\text{NDCG@}k = \frac{1}{|U|} \sum_{u=1}^{|U|} \frac{\text{DCG@}k}{\text{IDCG@}k}, \tag{2.3}$$

  where DCG@k is the Discounted Cumulative Gain at position k:

$$\text{DCG@}k = \sum_{i=1}^{k} \frac{2^{\text{rel}_i} - 1}{\log_2(i+1)}, \tag{2.4}$$

  and IDCG@k is the Ideal DCG@k, computed by sorting the items by their true relevance scores.

2.3.4 Retrieval-Augmented Generation in Recommendation Systems

Retrieval-Augmented Generation represents a significant advancement in recommender systems by integrating large language models LLMs with traditional retrieval techniques. This approach enhances recommendation quality by allowing the system to access and incorporate external information into its responses, resulting in more accurate and context-aware suggestions.

In this setup, a RAG-based recommender retrieves relevant data from external sources and feeds it into the language model, which then generates personalized recommendations based on both the retrieved content and user input[Di Palma, 2023]. This hybrid mechanism helps overcome common challenges like the cold start problem and data sparsity, enabling effective recommendations even with limited user history.

Recent studies highlight that combining retrieval with LLMs greatly improves the system's ability to deliver more relevant and diverse content, offering a promising direction for improving user experience in applications such as e-commerce and digital streaming platforms.

2.4 Conclusion

In this chapter, we explored the core principles and mechanisms of RAG, emphasizing its potential to enhance language models through access to external knowledge sources during inference. We examined its general architecture, key stages, and its evolution from naive to more modular frameworks.

Particular attention was given to RAG's application in the legal domain, highlighting its ability to enhance legal information retrieval by enabling more contextualized and semantically rich access to case law, statutes, and scholarly texts. We also outlined the main challenges and limitations facing RAG models. Additionally, the chapter briefly touched on the integration of RAG in recommendation systems, introducing its potential to improve personalization and relevance in such applications.

the next chapter addresses a critical factor in RAG performance: knowledge selection. Specifically, we explore how the effectiveness of retrieved documents directly influences the quality of generated responses, and we investigate strategies to improve relevance and ranking within RAG pipelines.

Chapter 3

k-Selection Optimization in RAG

3.1 Introduction

Retrieval-Augmented Generation systems enhance language models by grounding responses in externally retrieved documents. A critical challenge is determining the optimal number of documents k to retrieve for a given query. In this chapter, we first explore methods, highlighting their strengths and limitations We then introduce our novel hybrid dynamic selection algorithm provide detailed description of its methodology and present experimental results that demonstrate the effectiveness of its performance .

3.2 Defining k: The Number of Retrieved Documents

the parameter k denotes the number of documents or passages retrieved from an external knowledge base. This retrieval process is managed by the retriever component[AI, 2024],which identifies the top k relevant documents based on similarity to the query typically using embedding-based search[Rossi et al., 2024] (e.g., dense retrieval with FAISS, BM25, or hybrid methods) Subsequently, these documents are passed to the generator, typically a language model, which synthesizes the final response by integrating the retrieved

3.3 Impact of k on Retrieval Performance

The retriever's effectiveness in identifying relevant documents depends on k. Key impacts include :

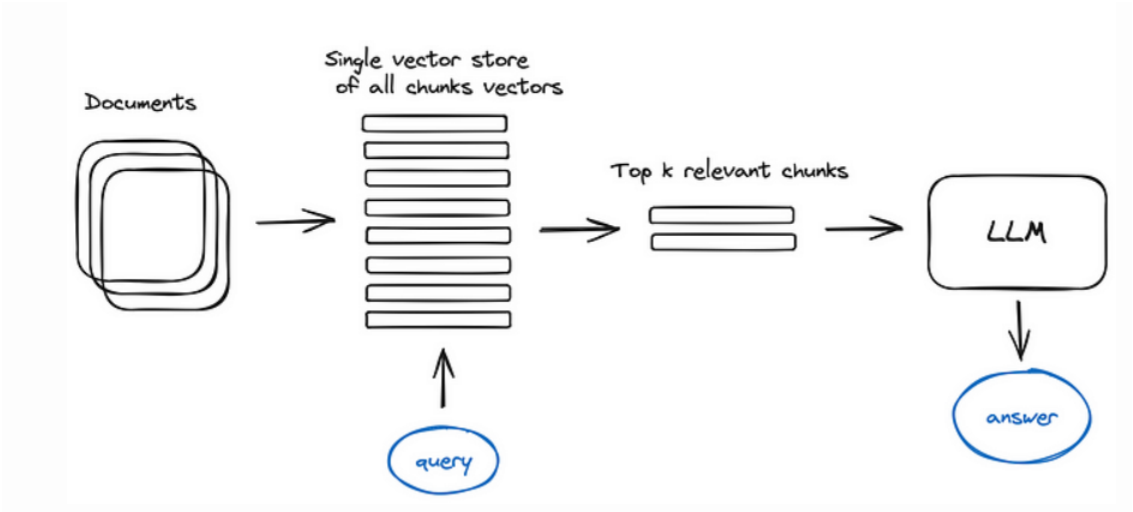


Figure 3.1: Basic retrieval

3.3.1 Recall vs. Precision

When a user conducts a search query, the outcomes from the database can be grouped into four distinct types based on relevance and retrieval status [GeeksforGeeks, 2022]:

- Relevant and Retrieved: Documents that both address the user's query and appear in the search results.
- Relevant and Not Retrieved: Useful documents that answer the query but are not included in the results.
- Non-Relevant and Retrieved: Documents that appear in the results but lack meaningful value for the query.
- Non-Relevant and Not Retrieved: Irrelevant documents excluded from the results.

Precision@k: evaluates the proportion of relevant documents within the top k retrieved results. This metric is particularly valuable in scenarios where the focus is on delivering highly relevant information quickly, rather than ensuring complete coverage such as ,recommendation systems or search engines[Kirchhoff, 2024].

$$\text{Precision@k} = \frac{\text{Number of Relevant Items Retrieved in Top k}}{k} \tag{3.1}$$

Example Suppose we have a dataset of 10 documents. If we retrieve $k = 4$ documents and find that 3 of them are relevant to the query:

- **Dataset**: [doc1, doc2, doc3, doc4, doc5, doc6, doc7, doc8, doc9, doc10]

- **Retrieved**: [doc3, doc1, doc7, doc4]
- **Relevant**: [doc1, doc3, doc5, doc8]

The Precision@4 score would be:

$$\text{Precision@4} = \frac{3}{4} = 0.75 \tag{3.2}$$

Recall@k : Evaluates the proportion of relevant documents that are successfully retrieved within the top k results. This metric is particularly important in contexts where ensuring the completeness of information is crucial, such as in medical research or academic tools, where omitting relevant documents could result in incomplete or inaccurate conclusions[Kirchhoff, 2024].

$$\text{Recall@k} = \frac{\text{Number of Relevant Items Retrieved in Top k}}{\text{Total Number of Relevant Items}} \tag{3.3}$$

Example: Consider a dataset of 10 documents. If we retrieve $k = 4$ documents and find that 2 of them are relevant, while the total number of relevant documents in the dataset is 4

$$\text{Recall@4} = \frac{2}{4} = 0.5 \tag{3.4}$$

Increasing k typically enhances recall by retrieving more relevant documents but may decrease precision due to the inclusion of irrelevant ones. Conversely, decreasing k can improve precision but at the cost of lower recall.

3.3.2 Retrieval Speed and Computational Cost

Retrieval speed and computational cost are two important areas where k has a big influence Below is a detailed explanation of how higher k values affect these aspects.

Increased Computational Resources:
As the value of k expands, the retrieval must handle a larger set of documents, leading to higher computational demands. Specifically, the system needs to perform additional operations like ranking, filtering, and similarity scoring to identify the most relevant documents. These tasks become increasingly resource-intensive, particularly in large-scale retrieval settings where the document corpus consists of millions or even billions of entries[Manning et al., 2008]. For instance, retrieving the top documents (k=80) requires significantly more computational resources compared to retrieving only the top 10 documents (k=5).

Higher Memory Usage: As the number of retrieved documents k increases, Storing and processing a larger set of retrieved documents requires more memory This can become a significant bottleneck, especially in environments with constrained memory resources.

For large-scale retrieval tasks, where datasets may contain millions or even billions of documents, the memory demand grows proportionally with k. Each retrieved document needs to be stored temporarily, along with its associated metadata, such as embedding vectors, BM25 scores, or other relevance signals. Additionally, sorting and filtering operations further contribute to memory overhead

3.3.3 Document Ranking Quality

Document ranking in information retrieval involves ordering documents by their relevance to a user's query. The objective is to prioritize the most relevant documents at the top of search results, making it easier for users to access useful information quickly. Different models,Vector Space, including Boolean, and Probabilistic models, are used to establish this ranking [Wikipedia contributors, 2024].

Increasing the number of retrieved documents,represented as k, may result in the addition of lower-ranked, less relevant documents, potentially diluting the overall quality of the retrieved information. This occurs because of the inherent balance between precision and recall in information retrieval systems.

Figure 3.2: Precision in Document Ranking

As k grows, recall generally improves since a larger number of relevant documents are likely to be retrieved as shown in Figure 3.3. However, this often comes at the expense of precision, as the additional documents retrieved may include non-relevant ones, thereby lowering the overall precision as illustrated in Figure 3.2. This trade-off is a core principle in evaluating information retrieval systems.

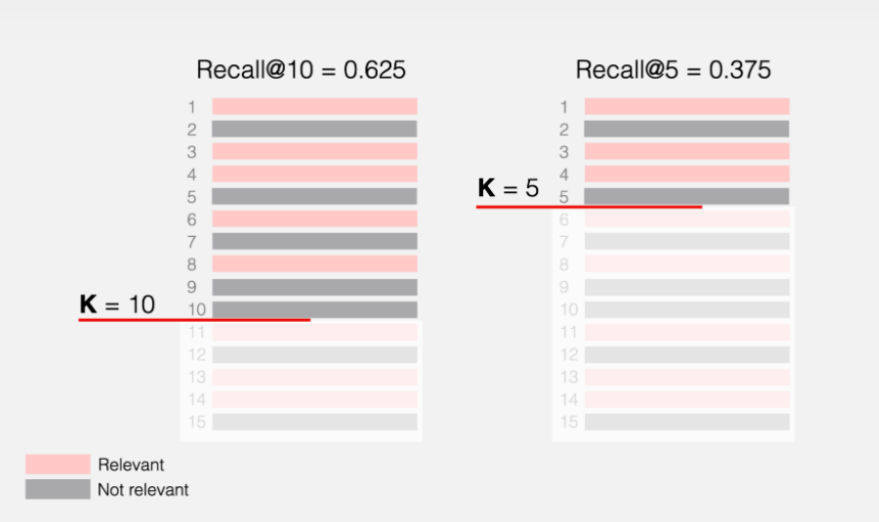


Figure 3.3: Recall in Document Ranking

3.4 Impact of k on Generation Quality

The impact of k (the number of retrieved documents or generated candidates) on generation quality is A crucial factor in many machine learning and information retrieval tasks, including text generation, recommendation systems, and search engines.

3.4.1 Trade-off Between Diversity and Relevance

Higher k: Increasing k often improves diversity in generated outputs or retrieved documents, as more candidates are considered. However, this can lead to a drop in relevance or quality, as lower-ranked candidates may be less accurate or coherent.

Lower k: A smaller k tends to prioritize high-quality, relevant outputs but may lack diversity, leading to repetitive or overly narrow results.

Research shows that the quality of retrieved documents plays a crucial role in the performance of RAG . For example, one study demonstrated that the precision of retrieved documents directly affects the factual correctness of the generated responses [Salemi and Zamani, 2023] Additionally, another study [Wan et al., 2025] revealed that simply increasing the number of documents does not necessarily improve generation quality, especially if the additional documents are not highly relevant.

3.4.2 Effect on Text Generation Models

In text generation tasks, such as machine translation and dialogue systems, The parameter k frequently refers to the beam width in beam search algorithms[Freitag and Al-Onaizan,

2017].Beam search is a heuristic search algorithm that expands the most promising nodes in a graph to maximize the quality of the text that is produced. Adjusting the beam width k has significantly impacts how well text generation models function.

As the beam width is increased, the model must process and retain more candidate sequences concurrently, leading to higher computational and memory demands Particularly in real-time applications where latency is crucial[NVIDIA, 2023], this increase may have an effect on system performance and response time.

3.5 Existing Solutions for k Selection

Selecting an optimal k plays a crucial role in balancing retrieval effectiveness and generation quality. Over the years, various strategies have been proposed to determine k, ranging from static selection to dynamic and hybrid approaches.

3.5.1 Static k Selection

In RAG, "static k selection", refers to the process of setting a fixed number of top documents k to retrieve for every query, irrespective of how simple or complex the query might be. This approach simplifies the retrieval process by maintaining a consistent retrieval count across all queries.

3.5.1.1 Sparse Retrieval with Fixed k

Sparse retrieval is a method of finding relevant documents from a large collection by representing both queries and documents as vectors where most values are zero[Zheng, 2024]. This focus on only the most important terms leads to faster, more accurate searches, especially useful when combining information from different sources Common approaches include. **BM25**[Robertson and Zaragoza, 2009] where documents are scored for relevance by considering term frequency and inverse document frequency, it selects the k documents with the highest BM25 scores for each query,**TF-IDF** (Term Frequency-Inverse Document Frequency)[GeeksforGeeks, 2025]in this method the retrieved documents would be those with the highest weighted term importance, Other methods like **QLM** (Query Likelihood Model)[Zhuang et al., 2021]use probabilistic models to evaluate the likelihood of a query and rank documents according to textual similarity.

3.5.1.2 Dense Retrieval with Fixed k

Dense retrieval [Karpukhin et al., 2020] is a method for retrieving information that uses deep learning models to convert documents and queries into high-dimensional vector embeddings, a fixed number of top-k documents are selected based on similarity scores between the query embedding and document embeddings in a continuous vector space, the retrieval process allows the system to capture semantic elationships beyond exact term matches.

Methods such as dual-encoder architectures (e.g., DPR - Dense Passage Retrieval)[Chen et al., 2024] utilize separate neural networks (encoders) for queries and documents enable efficient retrieval, while Approximate Nearest Neighbor (ANN) search techniques (e.g., FAISS)[contributors, 2025] optimize search efficiency in large-scale datasets .

3.5.2 Dynamic k Selection

In retrieval, dynamic k selection is the process of varying the quantity of documents k that are retrieved according to the properties of the query, such as its ambiguity, complexity, or relevance score distribution. This approach is crucial for enhancing the effectiveness and efficiency of information retrieval systems.

3.5.2.1 Dynamic Trade-Off Prediction in Multi-Stage Retrieval Systems

This approach predicts the optimal number of documents k to retrieve. It uses pre-retrieval features to balance the trade-off between retrieval efficiency and effectiveness, ensuring that the system retrieves an appropriate number of documents for each query[Culpepper et al., 2016].

3.5.2.2 Dynamic Pruning Methods:

This approach addresses the challenge of balancing effectiveness and efficiency in large-scale Information Retrieval systems, particularly under temporal constraints. The goal is to process queries within a specified time limit while minimizing the loss in retrieval effectiveness.[Wu et al., 2017] The authors propose and evaluate three techniques for temporally constrained top-K query processing .

3.5.3 Hybrid k Selection

hybrid methods aim to achieve optimal retrieval By merging static and dynamic k selection, These methods balance computational efficiency (from static k) with adaptive flexibility (from dynamic k) ensuring high-quality retrieval Below are some key hybrid strategies.

3.5.3.1 Blended RAG

An innovative method to improve Retrieval-Augmented Generation systems, which integrate private document collections with Large Language Models for Generative Question-Answering it employs a hybrid retrieval approach, combining Dense Vector indexes and Sparse Encoder indexes with sophisticated query strategies[Sawarkar et al., 2024]., Blended RAG provides a scalable and efficient solution for enhancing RAG, showcasing the effectiveness of merging dense and sparse retrieval techniques.

3.5.3.2 STAYKATE (Static-Dynamic Hybrid Selection)

A new approach for selecting in-context examples to improve the performance of LLMs in scientific information extraction. Scientific tasks often struggle with limited training data and expensive annotation processes. STAYKATE tackles these challenges by merging static and dynamic selection strategies [Zhu et al., 2024], combining representativeness sampling from active learning with retrieval-based methods. This hybrid approach ensures the selection of high-quality, informative examples, enhancing in-context learning for LLMs.

3.6 Proposed Solution

While existing k-selection approaches—static, dynamic, and hybrid—offer distinct advantages, they also present notable limitations, such as lack of adaptability in static k selection, which fails to adjust for query complexity, leading to either missing relevant documents or retrieving irrelevant ones. Dynamic k selection, while more flexible, introduces higher computational costs and requires carefully tuned heuristics, making it challenging to scale. Hybrid approaches, attempt to balance both strategies but suffer from increased system complexity.

To address these challenges, we propose an enhanced k-selection strategy that optimally balances retrieval efficiency and relevance, leveraging adaptive mechanisms to improve performance across diverse query types. Table 3.1 summarizes the notations used in this work.

Notation	Description
q (Q, $\|Q\|$)	A single query (set of all queries, total number of queries).
x (X, $\|X\|$)	A single item (set of all items, total number of items).
$\phi(q, x)$	The learned similarity function, Mixture-of-Logits (MoL), which computes the similarity between q and x.
P (P_q, P_x)	Total number of low-rank embeddings ($P = P_q \times P_x$), where P_q and P_x are the number of embeddings for q and x, respectively.
$\pi_p(q, x)$	Weight for the p-th (or p_q-th and p_x-th) embedding set for (q, x).
$f(q)$ ($f_p(q)$)	Learned embedding for the query (p-th component-level embedding for q).
$g(x)$ ($g_p(x)$)	Learned embedding for the item (p-th component-level embedding for x).
$\langle f(q), g(x) \rangle$	Dot product similarity function: $\langle f(q), g(x) \rangle = g(x)^T f(q)$.

Table 3.1: Notation Table

3.6.1 Mixture of Logits (MoL)

The **Mixture of Logits (MoL)**[Zhai et al., 2023, Ding and Zhai, 2025] approach is designed to enhance retrieval and ranking by leveraging multiple low-rank embeddings. It assumes that both the **query** (q) and **document/item** (x) are mapped into P groups of low-dimensional embeddings, denoted as $f_p(q)$ and $g_p(x)$, which are generated by neural networks based on their respective features. To determine the similarity between a query and a document,

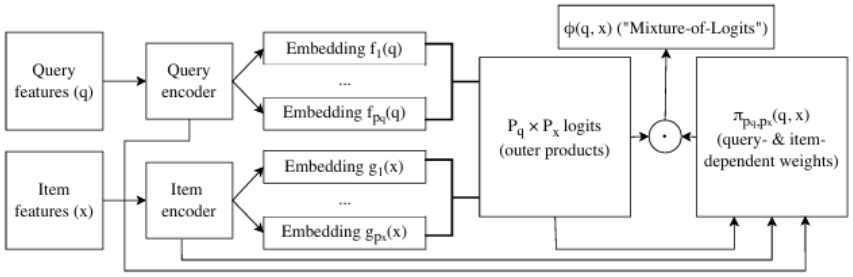


Figure 3.4: Mixture of Logits(MoL) learned similarity.

MoL assigns **adaptive gating weights** $\pi_p(q, x)$ to the inner products of these low-rank embeddings[Zhai et al., 2023]:

$$\phi(q, x) = \sum_{p=1}^{P} \pi_p(q, x) \langle f_p(q), g_p(x) \rangle \tag{3.5}$$

where $\pi_p(q, x)$ represents the learned weights for each component, ensuring that their

sum equals one. These weights are typically parameterized using a neural network that takes embeddings and their inner products as input features.

To efficiently scale MoL for large datasets and hardware-optimized implementations, the formulation is extended by decomposing the dot products into **batched outer products** of query-side and document-side embeddings. This decomposition improves computational efficiency, particularly on accelerators like GPUs, by normalizing the embeddings using the l_2-norm:[Zhai et al., 2023]

$$\phi(q, x) = \sum_{pq=1}^{P_q} \sum_{px=1}^{P_x} \pi_{pq,px}(q, x) \frac{\langle f_{pq}(q), g_{px}(x) \rangle}{||f_{pq}(q)||_2 ||g_{px}(x)||_2} \tag{3.6}$$

Since embedding normalization can be precomputed, both formulations remain interchangeable in practical applications. Furthermore, it is possible to **decompose any high-rank matrix** into a mixture of logits based on low-rank matrices, demonstrating the flexibility and scalability of this approach in large-scale information retrieval tasks.

3.6.2 Algorithm Design

we introduce an adaptive threshold mechanism (Dynamic K-Selection for Optimal Retrieval), the MoL framework is employed to refine the candidate retrieval process.

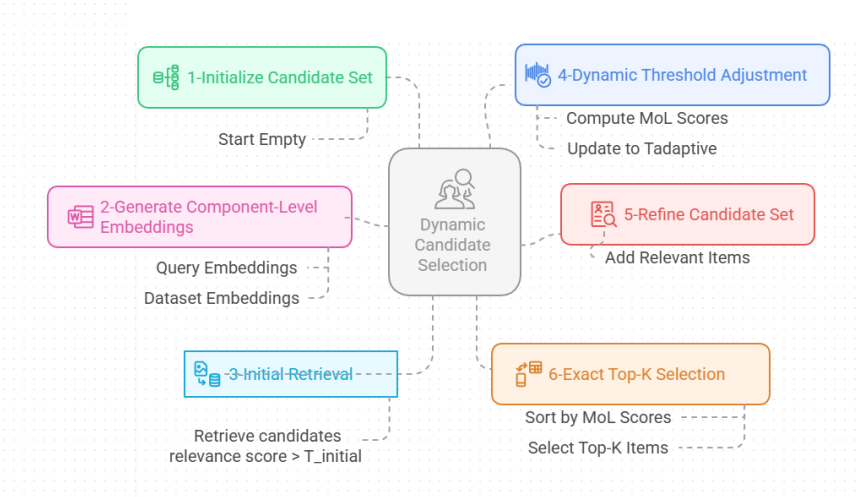


Figure 3.5: propused solution steps

1. **Component-Level Embeddings**: Component-level embeddings are generated for all items in the dataset X. These embeddings facilitate efficient similarity computations during retrieval. Formally,

$$X_p \leftarrow \{g_p(x) \mid x \in X\}. \tag{3.7}$$

2. **Initial Candidate Retrieval:** This step involves computing similarity scores between a query representation and item representations for each feature component $p \in P$.

$$S_p \leftarrow \{\langle f_p(q), g_p(x)\rangle : x \in X_p\} \tag{3.8}$$

Here, $f_p(q)$ and $g_p(x)$ represent the feature embeddings of the query q and item x for the p-th component, respectively. The dot product $\langle f_p(q), g_p(x)\rangle$ measures the relevance of each item $x \in X_p$ with respect to q, producing a set of scores S_p .

3. **Dynamic Threshold Adjustment**: To improve retrieval quality, **Mixture of Logits (MoL)** scores are computed for each candidate $x \in G$. The adaptive gating weights $\pi_p(q, x)$ allow the algorithm to dynamically adjust the retrieval threshold T_{adaptive} based on the MoL scores. The scoring function is defined as:

$$\phi(q, x) = \sum_{p=1}^{P} \pi_p(q, x) \cdot \langle f_p(q), g_p(x)\rangle. \tag{3.9}$$

The adaptive threshold T_{adaptive} is set as the minimum score among the candidates:

$$T_{\text{adaptive}} = \min\{s \mid s \in G\}. \tag{3.10}$$

4. **Refinement and Top-K Selection**: Using the adaptive threshold T_{adaptive}, additional relevant candidates are retrieved, expanding the candidate set G'. This is achieved by including candidates whose scores exceed the threshold:

$$G' \leftarrow G \cup \{x \mid s_p \geq T_{\text{adaptive}}\}. \tag{3.11}$$

The algorithm then sorts G' based on MoL scores to select the most relevant top-k candidates.

4. **Exact Top-K Selection**: Finally, the candidates in G' are sorted by their MoL scores, and the exact top-k items are extracted:

$$G_{\text{final}} = \text{Top-k}(G', \phi(q, x)). \tag{3.12}$$

As shown in Algorithm1, the retrieval process dynamically adjusts the threshold based on MoL scores.

3.6.3 Hybrid k-Selection Algorithm Pseudocode

The following pseudocodeAs shown in Algorithm1 outlines our proposed Dynamic K-Selection for Optimal Retrieval algorithm. This method adaptively selects the number of top-k documents retrieved based on query characteristics, enhancing the relevance and efficiency of the retrieval step in our RAG pipeline

Algorithm 1 Hybrid Exact Top-k with Threshold-Based k Selection

Input: Query q, Set of items X, Component-level embeddings: $f_p(q)$, $g_p(x)$ for $p \in P$, $x \in X$, Initial threshold T_{init}

Output: Exact top k items based on dynamic threshold selection, G_{final}

Set $G \leftarrow \emptyset$ ▷ Initial candidate set

for each component $p \in P$ **do**

  $X_p \leftarrow \{g_p(x) \mid x \in X\}$ ▷ Precompute embeddings

end for

1. Initial Candidate Retrieval:

for each component $p \in P$ **do**

  Compute dot product scores: with (eq(3.8))

  Retrieve items with scores $S_p \geq T_{\text{init}}$

  Add these items to G

end for

2. Adjust k Dynamically:

for each $x \in G$ **do**

  Compute MoL scores $s \leftarrow \phi(q, x)$ using : eq(3.9)

  Set $T_{\text{adaptive}} = \min\{s : s \in G\}$

end for

3. Refine Candidate Set with Adaptive k:

$G' \leftarrow \emptyset$

for each component $p \in P$ **do**

  Retrieve items from X_p with scores $S_p \geq T_{\text{adaptive}}$

  Add these items to G'

end for

4. Select Exact Top-k Items:

for each component $p \in P$ **do**

  Compute MoL scores for all items in G'

  Sort G' by MoL scores in descending order

  Select the top k items from G' where k is the number of items in G' exceeding T_{adaptive}

end for

Return: G_{final} ▷ Retrieve Top k items from G'

3.7 Experimental Results

In order to measure the effectiveness of our proposed method, we perform a comprehensive evaluation of both the retrieval algorithm and the Mixture-of-Logits (MoL) approach. This assessment focuses on the next-item prediction task, a fundamental challenge in recommendation systems[Kang and McAuley, 2018], where the goal is to predict the most relevant item a user will interact with next based on their past interactions.

3.7.1 Dataset Design

We conducted experiments using two MovieLens datasets : ML-100K and ML-1M [Harper and Konstan, 2016]which are widely used benchmarks for evaluating sequential recommendation models as detailed in Table3.2.

Dataset	**Description**
MovieLens-100K	100,000 ratings (1-5 scale) from 943 users on 1,682 movies. Each user has rated at least 20 movies. Includes user demographic information (age, gender, occupation, zip code).
MovieLens-1M	1,000,209 ratings from 6,040 users on approximately 3,900 movies. Data collected from users who joined MovieLens in 2000. Represents a larger-scale recommendation scenario.

Table 3.2: Summary of MovieLens datasets

3.7.2 Experimental Setup

For both datasets, We utilize the SASRec architecture as our sequential user encoder, a model renowned for achieving state-of-the-art performance in next-item prediction tasks. This architecture processes the user's historical interaction sequence, generating embeddings that encapsulate the user's preferences at each time step. These embeddings serve as the foundation for predicting the next item in the sequence.

The query q represents the user's state at a specific time step, derived from their interaction history. In the MoL (Mixture of Logits) framework,$q$is transformed into Pq embeddings through a multi-layer perceptron (MLP).

For fair comparison, we maintained consistent architectural choices and training conditions across all experiments, We conducted an extensive hyperparameter analysis comparing both approaches (Hybrid+SAS and MoL+SAS) across different architectural configurations:

3.7.2.1 Fixed Experimental Configuration

- **Hardware**: Google Colab with T4 GPU
- **Framework**: TensorFlow
- **Optimizer**: Adam ($\eta = 0.001$)
- **Hybrid Specific**: $T_{init} = 0.3$ (init threshold)

For the hybrid algorithm, we initialize the threshold (Tinit) to 0.3 and adaptively adjust it during training. We discuss detailed hyperparameter settings in table 3.3, table

3.7.3 Architectural Variants and Results

The following tables present our complete experimental results across different model configurations.

Model	**SeqLen**	**EmbDim**	**Heads**	**100kMovies**		**1M Movies**	
				Val Loss	Val Acc	Val Loss	Val Acc
Hybrid+SAS	50	128	2	5.3036	0.1584	4.5721	0.1530
MoL+SAS	50	128	2	5.3152	0.1582	4.5733	0.1526
Hybrid+SAS	128	256	4	3.6364	0.4301	3.4152	0.3680
MoL+SAS	128	256	4	3.6374	0.4299	3.4671	0.3627
Hybrid+SAS	512	512	4	1.2808	**0.8120**	1.0275	**0.8350**
MoL+SAS	512	512	4	1.3050	**0.8016**	1.0350	**0.8276**

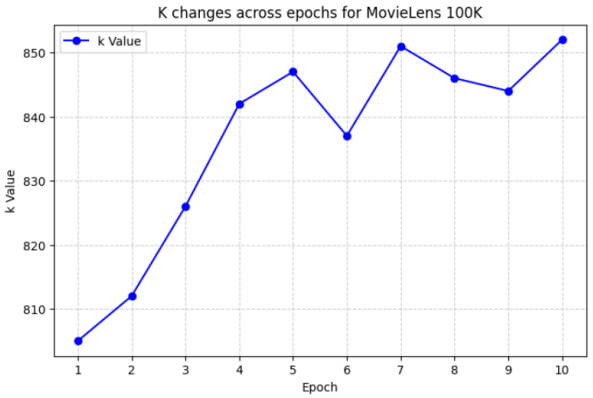
Table 3.3: Performance Across Configurations

Both approaches demonstrate significant performance improvements as model capacity increases, with larger configurations consistently delivering better results. For instance, when

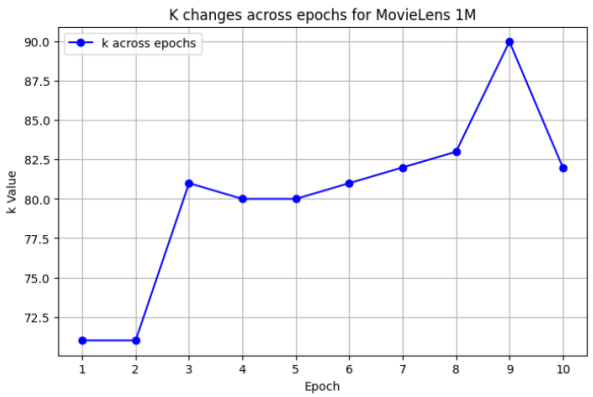
increasing the model's capacity such as expanding the (Max Sequence Length: 512, Embedding Dimension: 512, Number of Heads: 4, Feed-Forward Dimension: 512) the hybrid algorithm combined with SASRec (Hybrid+SAS) achieves notable gains, particularly in the most resource-intensive setup. On the ML-100K dataset, Hybrid+SAS reaches a score of**0.8120** compared to the baseline's **0.8016**, while on ML-1M, it achieves **0.8350** versus **0.8276** The ML-1M dataset generally benefits more from increased capacity, with the performance gap between datasets narrowing as the model scales. While smaller configurations provide a balance of efficiency and performance, the largest configuration, despite its higher computational demands, yields the best results, making it suitable for scenarios where resources are not a constraint. Both approaches show similar benefits from scaling, but Hybrid+SAS maintains a consistent edge in performance.

3.7.4 Impact of Adaptive k-Variations on Model Performance

As shown in Figures 3.6a and 3.6b, the value of k fluctuates across epochs for both MovieLens 100K and MovieLens 1M datasets. These variations indicate the adaptive nature of k in response to the dataset characteristics and training progress. In the following section, we analyze how these changes influence model performance.



(a) K changes across epochs for MovieLens 100K



(b) K changes across epochs for MovieLens 1M

Figure 3.6: Comparison of K changes across epochs for different datasets

the value of K changed dynamically during training, with different behaviors. For the MovieLens 100K dataset, K showed a steady increase across epochs, whereas for the MovieLens 1M dataset, K began at a lower value, experienced fluctuations, peaked, and then stabilized. This indicates that the larger dataset necessitated more adaptive adjustments in retrieval compared to the smaller dataset.

Figure 3.7 summarizes the performance of our hybrid algorithm compared to the MoL-based

approach on the MovieLens 100K and 1M datasets, both tested using a Max Sequence Length of 50, an Embedding Dimension of 128, 2 Attention Heads, a Feedforward Dimension of 128, a Batch Size of 128, and trained for 10 epochs .

The variations in K help explain the performance differences seen in the fig 3.7. The

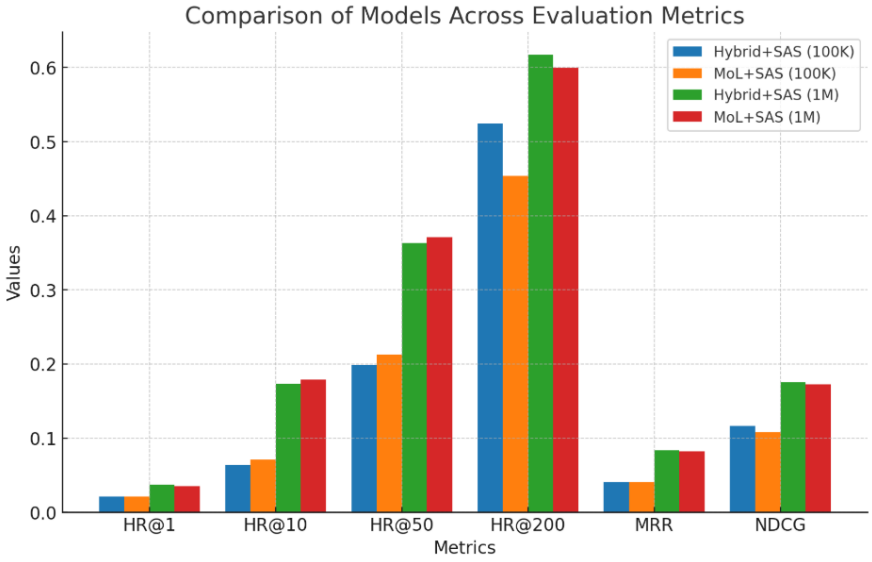


Figure 3.7: Comparison of Models Across Evaluation Metrics

Hybrid+SAS model applied to the MovieLens 1M dataset, which exhibited more dynamic changes in K achieved higher HR@50(0.3631) and HR@200(0.6170) scores, reflecting better long-range recommendation quality. The fluctuations in K in the 1M dataset likely allowed the model to balance exploration and precision, improving its overall ranking performance.

On the other hand, the more gradual K increase in the 100K dataset resulted in relatively lower scores, suggesting that a static or overly conservative K selection might limit retrieval effectiveness.

Additionally, the MoL+SAS model achieved better performance than Hybrid+SAS in HR@10 for both datasets. This aligns with the observation that MoL+SAS tended to retrieve fewer but more relevant items at shorter ranking positions. This behavior can be attributed to how K evolved during training?smaller K values in earlier epochs likely enabled MoL+SAS to maintain higher precision at shorter ranks. These findings highlight the importance of dynamically tuning K based on dataset characteristics to optimize recommendation effectiveness.

The results demonstrate that the Hybrid Exact Top-k algorithm effectively leverages both sequential user behavior and item metadata to improve recommendation quality. The adaptive MoL threshold allows the model to dynamically refine candidate items during training, leading to better performance. The improvements are consistent across both datasets, high-

lighting the robustness of our approach.

3.8 Conclusion

The selection of k in retrieval plays a pivotal role in balancing relevance, efficiency, and computational cost. Throughout this chapter, we explored how different approaches—static, dynamic, and hybrid—impact retrieval performance and generation quality. While static selection provides consistency, it lacks adaptability to varying query complexities. Dynamic methods introduce flexibility but come with computational overhead, whereas hybrid strategies aim to balance both. The Hybrid Exact Top-k with Threshold-Based k Selection method offers an advanced solution by leveraging adaptive weighting to enhance ranking effectiveness.

Although this method has shown promising results in large-scale retrieval scenarios, its integration into our Arabic legal RAG agent was not feasible due to the limited size and scope of our dataset, which contains only around 100 manually constructed cases, many of which are not fully accessible or diverse. The effectiveness of threshold-based k selection becomes more apparent when applied to larger and more representative datasets. Therefore, we consider this a valuable avenue for future work—once the dataset is expanded, implementing and testing this dynamic approach could significantly enhance the performance and adaptability of our Agent.

Ultimately, an effective k-selection strategy is essential for optimizing retrieval, ensuring both efficiency and high-quality results in knowledge-augmented applications.

Chapter 4

Design of Arabic RAG-Based Agent for Legal and Juridical Data.

4.1 Introduction

The growing demand for intelligent legal assistance has highlighted the need for domain-specific natural language processing NLP systems. This chapter presents the development of a Retrieval-Augmented Generation RAG Agent tailored for Arabic legal texts. The primary goal of this Agent is to answer legal queries accurately by leveraging a combination of dense information retrieval and generative language modeling. By integrating Arabic legal domain knowledge with advanced retrieval techniques and generative models, the Agent delivers explainable, high-quality responses grounded in real case data. This chapter outlines the complete pipeline—from dataset design and model training to evaluation—offering insights into the construction of a practical and robust Arabic legal chatbot.

4.2 Dataset Design

For the development of the Arabic RAG Agent tailored to the legal domain, two purpose-built datasets were created: the Legal Case Dataset and the QA Dataset. These datasets form the foundation of the RAG pipeline, enabling both effective retrieval and accurate answer generation. The Legal Case Dataset consists of actual legal rulings published by the Algerian Supreme Court, while the QA Dataset includes synthetic question-answer pairs generated from those rulings. By combining rule-based techniques with generative modeling, the Agent benefits from both structure and linguistic variability, helping it generalize better during answer generation and evaluation. Figure 4.1 illustrates the detailed pipeline used for the creation of both the legal case dataset and the corresponding question-answer (QA)

Dataset Name	**Description**	**Structure**	**Use Case**	**Construction Method**
Legal Case Dataset	104 Arabic legal cases collected from the Algerian Supreme Court journal	Case ID, Title, Keywords, Description	Used for building the retriever index and generating QA pairs	Scraped from the official court journal and manually cleaned, labeled, and structured
QA Dataset	512 question-answer pairs derived from the legal case dataset	Case ID, Question, Answer, Context, Context Tokens	Used for training, generation, and evaluation of the RAG Agent	Constructed using a hybrid approach: rule-based extraction of key legal elements + generative QA using AraGPT2

Table 4.1: Summary of the datasets used in the Arabic Legal RAG Agent

dataset. The process begins with web scraping from official Algerian court sources, followed by data cleaning and normalization. Subsequently, the normalized legal data is utilized to generate QA pairs through both , resulting in a structured dataset suitable for Arabic legal question answering tasks.

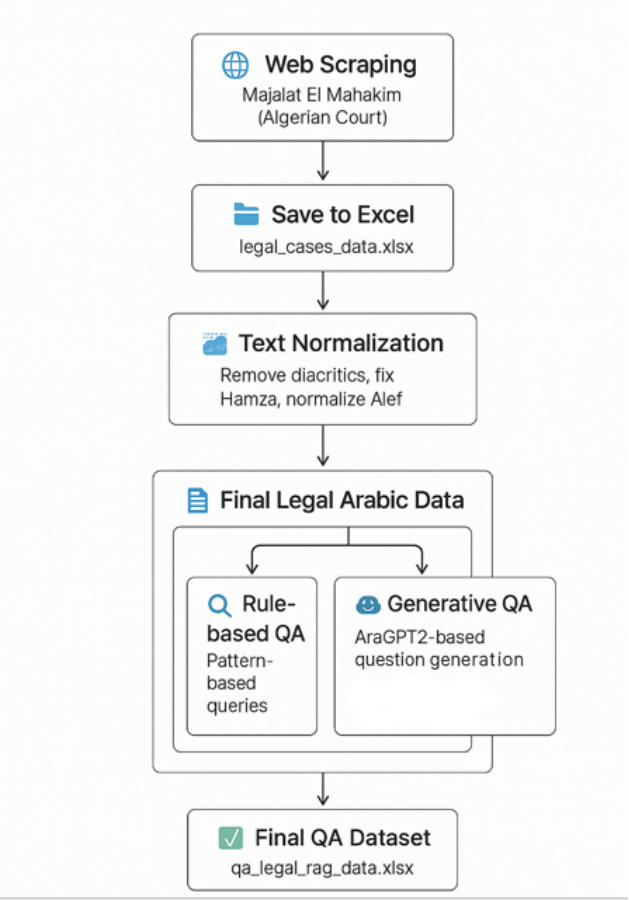


Figure 4.1: The detailed pipeline used for the creation of both the legal case dataset and the question-answer QA

4.3 Agent Architecture Overview

To build an Arabic legal assistant capable of answering questions about legal cases, we adopted an agentic architecture based on RAG. The Agent is composed of modular components that work together to understand user questions, retrieve relevant legal content, and generate informative responses. It is designed to handle complex legal texts and support both general legal queries and specific requests, such as extracting particular legal articles Figure 4.2 illustrates Agent Architecture Overview.

4.3.1 Input/Output

The agent **receives natural-language arabic questions as input**. These questions may range from general inquiries about a legal case to specific prompts requesting certain materials, such as legal articles. As output, the Agent **returns either detailed answers grounded in real legal case context or a filtered subset of legal references** (e.g.,

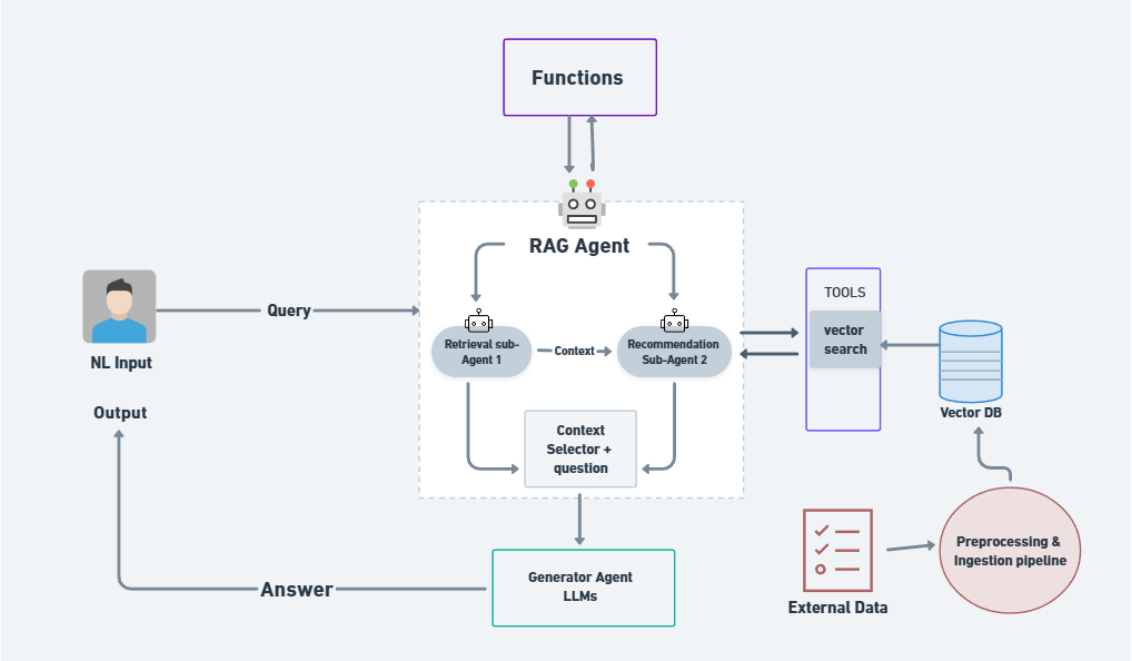


Figure 4.2: Architecture of the Arabic RAG-Based Agent for Legal Data. The Agent takes Arabic natural-language questions as input and returns answers grounded in real legal case context. It integrates preprocessing, semantic retrieval, and generation agent.

materials/articles), depending on the nature of the user request.

4.3.2 Preprocessing and Ingestion Phase

Before retrieval and generation, legal documents go through a data ingestion pipeline. This includes cleaning and normalizing raw text, followed by semantic chunking to ensure each unit of information is appropriately sized. These chunks are then embedded and indexed into a vector database to enable efficient semantic retrieval during runtime. This phase ensures the quality, structure, and retrievability of legal content.

4.3.3 Retriever Sub-Agent

The retriever identifies the most relevant pieces of legal content based on the user's query. It searches through the vectorized representation of the case database to return a ranked list of the most semantically similar chunks. This retrieval process ensures that the agent has access to high-quality and contextually relevant legal information.

4.3.4 Recommendation Sub-Agent

If the user's request targets specific legal materials (e.g., legal articles mentioned within a case), a recommendation sub-agent is triggered. This agent scans the previously retrieved content to identify and extract only the relevant sections that match patterns typically associated with legal article references. This sub-agent demonstrates intra-agent communication, as it relies on outputs from the main retriever while adding a specialized capability.

4.3.5 Context Selector: Token-Aware Filtering

From the retrieved candidates, the agent selects the most relevant chunks that fit within the system's input constraints. This filtering ensures that the generator receives coherent, non-redundant, and contextually rich text to support accurate response generation.

4.3.6 Functions and Tools

In addition to the core RAG-based workflow, the architecture integrates a flexible function-calling and tool-use mechanism. This enables the agent to interact with external tools—such as file analyzers, legal databases. These tools are invoked dynamically based on the user query, and the outputs are fed back into the RAG Agent to support more grounded, context-aware answers.

4.3.7 Generator: Answer Formulation Component

Using the selected context, the generator produces a coherent and legally grounded answer in Arabic. The generation process is context-aware, allowing it to reflect the nuances of legal terminology and adapt the output style depending on the user's request.

4.3.8 Agent Design

The core Agent functions as a domain-specific legal agent that uses Retrieval-Augmented Generation to assist users with legal case questions. It is capable of:

- Understanding natural Arabic legal questions, including complex legal terminology.
- Retrieving relevant information from legal case texts using semantic search.
- Generating answers grounded in real legal content.
- Recommending specific legal materials when requested by the user.

Rather than relying on predefined responses, the agent dynamically retrieves and composes answers using the most relevant and up-to-date case context. It can be used by legal professionals, students, or institutions to enhance legal understanding and streamline research across a wide range of case types.

4.4 Training the Agent

To empower the Arabic RAG-Based Agent with accurate and context-aware responses, we trained both its retriever and generator components on a curated Arabic legal QA dataset. This training phase focused on optimizing the agent's ability to understand user questions, retrieve the most relevant legal context, and generate coherent and grounded answers.

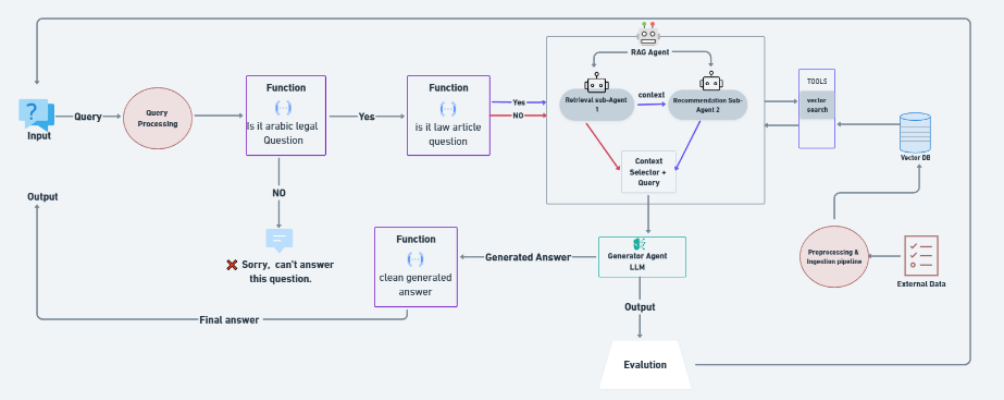


Figure 4.3: An overview of the training pipeline for the RAG agent

4.4.1 Training the Retriever

The retriever is responsible for identifying and retrieving the most relevant legal context for a given user query. This component can be trained or configured through several key phases that focus on data preprocessing, semantic embedding, and efficient indexing. Below, we outline a generalizable framework for training a dense retriever in Arabic legal RAG Agent.

- **Text Normalization Pipeline:** A preprocessing stage is essential to clean and standardize Arabic legal texts. This may involve:
  - Removing diacritics to minimize lexical variations.
  - Normalizing common character variants (e.g., Alef, Taa Marbouta).
  - Standardizing punctuation symbols and quotation marks.
  - Eliminating excess whitespace and formatting inconsistencies.

- **Document Chunking Strategy:** Legal documents are often lengthy and need to be divided into smaller, semantically coherent chunks. Common techniques include:
  - Using recursive text splitters with a fixed token size (e.g., 300 or 500 tokens).
  - Applying hierarchical splitting rules (paragraphs, then sentences, then phrases).
  - Preserving key metadata like title, keywords, or article number.
  - Leveraging tokenizer-based length calculation to ensure chunk boundaries match model requirements.
- **Dense Retrieval Architecture:** Semantic search typically relies on encoding both documents and queries into dense vector representations using transformer-based models. Common considerations include:
  - Choosing a pre-trained multilingual or Arabic-specific embedding model.
  - Applying mean pooling or CLS token extraction to produce sentence-level embeddings.
  - Normalizing embeddings (e.g., via L2 normalization) to improve similarity computation.
  - Using batch processing for faster embedding generation on GPU or CPU.
- **Indexing Framework:** To enable efficient similarity search, embedding vectors can be stored in a specialized vector database or indexing engine. Some popular practices include:
  - Using FAISS or other vector stores to build inner-product or approximate nearest neighbor indices.
  - Associating each embedding with its original text and metadata.
  - Tracking chunk-document mappings for traceability during retrieval.
- **Query Processing:** The user question must be preprocessed and embedded in the same way as documents. Key steps may involve:
  - Prefixing or formatting the query to match model expectations.
  - Applying the same normalization steps used during indexing.
  - Calculating similarity scores (e.g., cosine similarity) between query and document vectors.
  - Returning the top-k matching chunks with associated document IDs or metadata.

- **Evaluation Dataset:** Evaluating retrieval quality typically involves a labeled dataset containing questions, gold answers, and corresponding contexts. This can be used to measure:
  - Retrieval accuracy metrics such as :
    1. Recall@K measures the proportion of queries where the correct case appears in the top-K results,
    2. MRR (Mean Reciprocal Rank) emphasizes the position of the first correct result through reciprocal weighting.
    3. Hit@K provides a binary success indicator for each query.
  - Relevance of retrieved contexts with respect to the ground-truth legal passages.
  - The impact of retrieval quality on downstream answer generation.

4.4.2 Training the Generator

The generator is responsible for producing fluent, accurate Arabic responses based on the user's question and the retrieved legal context. It can be built using a pre-trained Arabic causal language model such as AraGPT2 or decoder-based models like AraBERT (in encoder-decoder setups), which are fine-tuned on legal QA data.

- **Prompt Design** A typical input format for generation includes a clearly structured prompt that merges the question and retrieved legal context:

```
Question: [user query]
Reference: [retrieved legal text]
Answer:
```

  This structure helps the model understand the roles of each input component and generate contextually appropriate responses.

- **Data Preparation** Training data should be tokenized using a tokenizer compatible with the selected Arabic model. Common preprocessing steps include:
  - Automatic padding and truncation to fit model token limits (e.g.,512, 1024,2024 tokens)
  - Insertion of special separator tokens between segments (e.g., between question and context)
  - Filtering or adjusting samples that exceed the model's maximum input length

- **Model Fine-Tuning** The selected Arabic language model—such as AraGPT2 for autoregressive generation or AraBERT in encoder-decoder setups—can be fine-tuned using supervised datasets consisting of paired questions, legal context, and answers. Fine-tuning enables the model to specialize in the legal domain and improve performance on complex queries.

- **Training Process** The fine-tuning process typically includes:

  - **Causal Language Modeling Objective:** Training the model to predict the next token in a sequence given previous tokens.

  - **Teacher Forcing:** Providing the ground truth answer during training to improve convergence .

  - **Decoding Strategies:** Techniques like beam search can be used during inference to generate more controlled responses, while repetition control (e.g., no-repeat n-grams) is applied to avoid redundant phrases.

- **Quality Control** To ensure reliable and relevant legal answers, post-processing steps may include:

  - Removing prompt artifacts or formatting leftovers from generation.

  - Verifying the inclusion of key legal terms or references.

  - Checking contextual alignment between the retrieved reference and the generated answer.

- **Evaluation of the Generator** To assess the quality of the generated answers from the we can conducted a quantitative evaluation using two commonly used metrics for natural language generation: **BLEU** and **BERTScore**.

  - ### BLEU (Bilingual Evaluation Understudy):

    BLEU measures the n-gram overlap between the generated text and the reference text. It is a precision-based metric often used in machine translation and QA evaluation[Papineni et al., 2002]. In our experiment, we used the official implementation provided by the `sacrebleu` library to ensure reproducibility and standardization.

– **BERTScore:**

BERTScore computes similarity between the embeddings of the generated and reference sentences using a pretrained contextual language model [Zhang et al., 2020](in our case, Arabic BERT). It provides precision, recall, and F1 scores. We report the average F1 score, which reflects semantic similarity rather than just surface overlap.

4.5 Experimental Results

The RAG Agent was rigorously validated through end-to-end testing and quantitative evaluation. Retrieval performance was measured confirming the retriever's ability to identify relevant legal cases and The generator was assessed through human evaluation and automated metrics for linguistic quality.

4.5.1 Training Results

During the training phase, the retriever and generator components were fine-tuned using a curated Arabic legal QA dataset. The retriever was trained to embed and index normalized document chunks using a dense semantic model, while the generator (based on AraGPT2) was trained on structured prompt-answer pairs. Both components were trained on colab T4 GPU due to limited GPU access, with careful preprocessing, batch management, and quality control applied throughout to ensure reliable convergence and domain-specific adaptation.

4.5.1.1 Model Architecture

The AraGPT2 architecture was selected for its ability to handle Arabic text generation while being computationally efficient. Below are the key specifications of the model:

4.5.1.2 Training Configuration

The fine-tuning process employed carefully selected hyperparameters to balance training stability and model performance. The following table details the training setup:

4.5.1.3 Inference Settings

At generation time, various hyperparameters influence output quality. Typical settings include:

Component	**Specification**
Architecture	Decoder-only transformer
Layers	12
Hidden Size	768
Attention Heads	12
Parameters	137M
Vocabulary Size	50,257 (Byte-level BPE)
Max Position Embeddings	1024

Table 4.2: Core architectural specifications of the decoder-only transformer model used in this work

Parameter	**Value**
Batch Size	2 (per device)
Epochs	20
Sequence Length	1024 tokens
Learning Rate	5e-5
Weight Decay	0.01
Gradient Accumulation	1
Warmup Steps	0

Table 4.3: Summary of key hyperparameters configured for model fine-tuning

- **Maximum Output Length:** Capped to prevent overly long responses (e.g., 260 tokens).
- **Temperature:** Adjusts randomness in sampling (e.g., temperature = 1.0 for deterministic output).
- **Top-k/Top-p Sampling:** Can be enabled or disabled depending on the desired balance between creativity and consistency.
- **Penalties:** Repetition and length penalties help enforce concise and varied outputs.

4.5.1.4 Training Progress :

The generator's training process was monitored using the training loss across epochs, as shown in the figure 4.4 below.

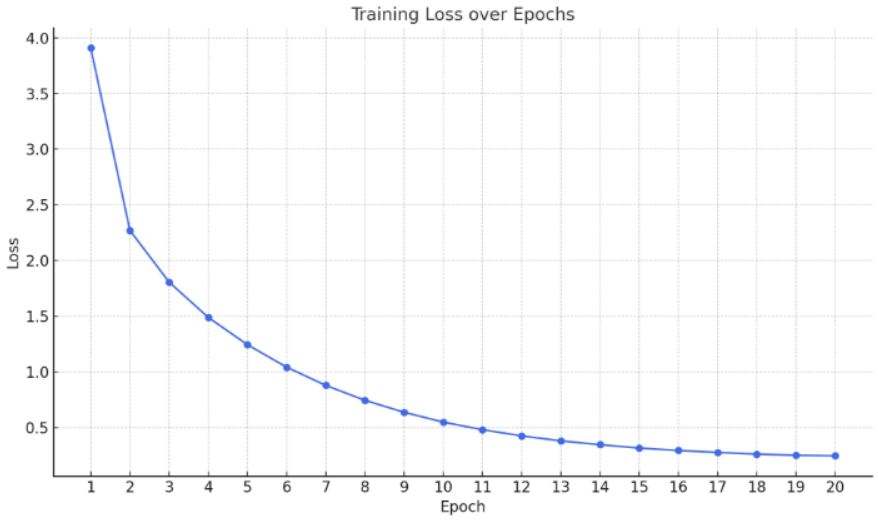


Figure 4.4: Training Loss over Epochs

The graph demonstrates a steady decline in loss, indicating that the model successfully learned from the data over time without signs of overfitting.

4.5.2 Evaluation Results

The evaluation was conducted separately for both the retriever and generator components. For the retriever, we used a manually curated Arabic legal QA benchmark where each query was associated with gold-standard context passages. Retrieval performance was measured using standard top-k accuracy metrics, confirming that the Agent consistently retrieved the correct legal context within the top 3 results. For the generator, we evaluated the quality of generated answers using both human judgment and automatic metrics, focusing on fluency, legal accuracy, and relevance to the retrieved context. The results showed that the generator was capable of producing coherent Arabic answers that aligned well with the legal source material in most cases.

4.5.3 Retriever Evaluation Results

The retriever's effectiveness was quantitatively assessed using three standard information retrieval metrics computed across multiple top-K thresholds.

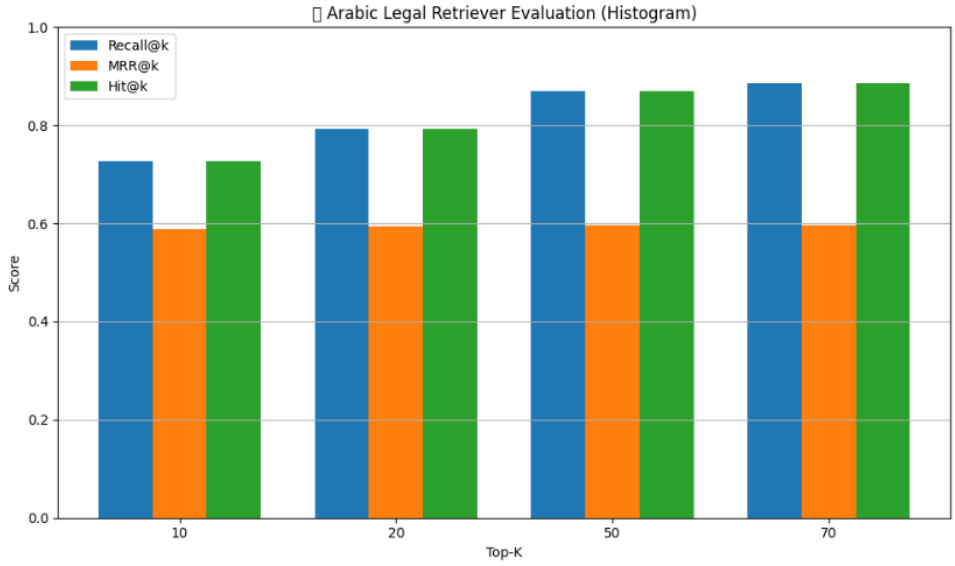


Figure 4.5: Retrieval performance metrics across different top-K values. The histogram shows consistent improvement in Recall@K, MRR@K, and Hit@K as K increases.

As shown in Figure 4.5, performance scales with K, achieving 92% Recall@70 and MRR@70 of 0.88, indicating both high coverage and quality of rankings. The evaluation protocol embedded each test query using the E5 model's token representation, then searched the FAISS index for nearest neighbors. Case ID matching against the ground truth confirmed retrieval accuracy, with error analysis revealing most failures occurred for queries containing ambiguous legal terminology or incomplete case references.

Human Evaluation: The RAG agent was evaluated using a curated set of legal queries paired with ground-truth case IDs. For each query, the retriever generated top-k candidate passages (k=3), which were analyzed for correctness by comparing retrieved case IDs against the ground truth.

Figure 4.6: Retrieval performance on a sample legal query showing the top-3 results with similarity scores.

A representative evaluation case (shown in Figure 4.6 demonstrates successful retrieval, where the correct legal case (ID 1055771) was ranked first with a high similarity score (8.9098), significantly outperforming lower-ranked candidates (0.8789 and below).

4.5.4 Geerator Evaluation Results

We ran our `rag_pipeline` function on each question in the legal QA dataset, collected the generated answers, and compared them to the ground truth answers using both metrics.

Results:

- **BLEU Score (official):** 27.33
- **BERTScore F1 Average:** 0.7903

These results suggest that the generator produces contextually accurate and semantically relevant answers, with moderate lexical overlap and high embedding-level similarity.

Human Evaluation: To complement automatic evaluation metrics such as BLEU and BERTScore, we conducted a human evaluation through the deployed Arabic legal assistant interface.

Figure 4.7: An example from the Arabic legal assistant interface showing rejection of an irrelevant question ("What is NLP?") because it is not written in formal Arabic. This reflects the Agent ability to filter non-domain input during inference.

As illustrated in Figure 4.7, the Agent is able to identify and reject inputs that are unrelated to the legal domain. This behavior ensures that the assistant remains focused on legal queries, aligning with the intended use-case and improving reliability in real-world usage.

4.6 Agent Results

This section presents the user interface of the intelligent legal agent after integrating all RAG components. The interface allows users to input Arabic legal questions and receive automatically generated answers based on retrieved real legal cases. A screenshot of the final application is shown below, demonstrating the usability and functionality of the Agent in action. Users can input their questions, process them via a backend RAG pipeline, and view the results in a styled and intuitive format.

- **Intelligent Legal Agent Interface :**
  The interface allows users to input legal questions in Arabic and receive accurate an-

swers based on real legal case data.

Figure4.8 highlights the main components of the interface, including the input field for questions, control buttons, and the answer display box.

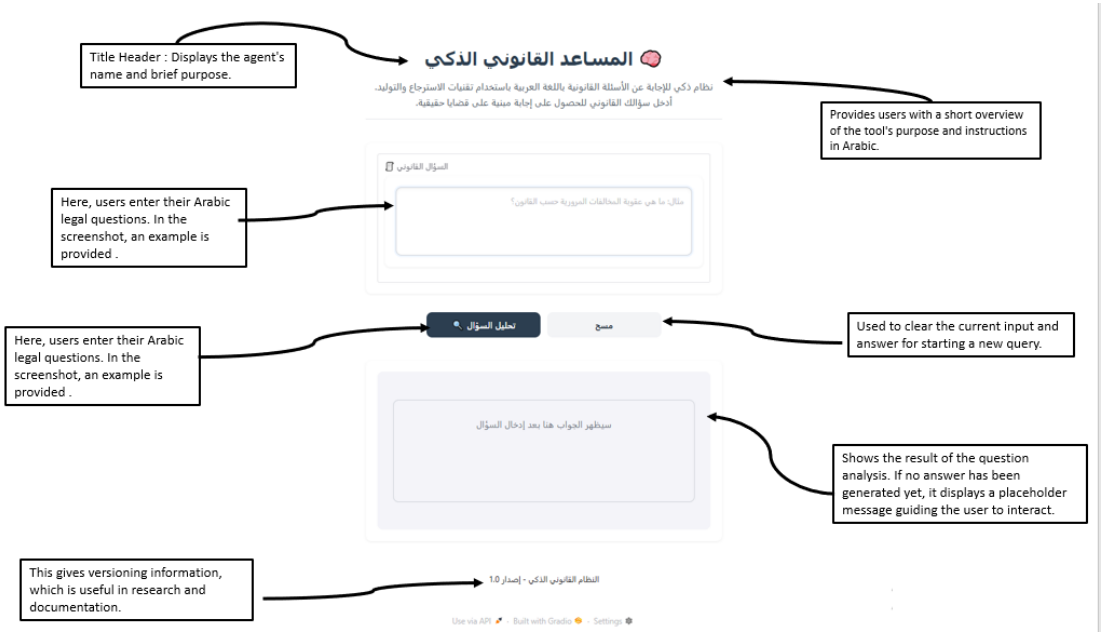


Figure 4.8: Interactive interface of the Intelligent Legal Assistant Agent.

- **Detailed Case Answering via RAG Agent** Figure4.9 shows the user interface of the Arabic Legal RAG Agent. In this example, the user inputs a general query asking for the details of a case based on its title.

Figure 4.9: Interface of the Arabic Legal RAG Assistant showing a successful answer to a general legal query using the case title. The Agent retrieves relevant case data and generates a comprehensive legal response .

The RAG Agent retrieves the most relevant case chunks using semantic search and generates a detailed answer using the AraGPT2 model. The response includes important legal information such as contract type, legal articles, procedural details, and court decisions. The generated text is complete, accurate, and contextually grounded, demonstrating the Agent's ability to handle vague or broad legal questions and return precise, useful answers.

4.7 Conclusion

In this chapter, we presented the complete development pipeline of an Arabic Retrieval-Augmented Generation (RAG) based agent tailored for legal question answering. The Agent was constructed through a sequence of carefully designed components, starting from data collection and preprocessing of Arabic legal documents and QA pairs, followed by the construction of a dense retriever using FAISS and the multilingual E5 encoder. We then integrated a generator—fine-tuned AraGPT2—to synthesize accurate answers grounded in retrieved legal context.

We evaluated the retriever independently using retrieval metrics such as Recall@k and MRR, Subsequently, the generator was evaluated using both automatic metrics—BLEU and BERTScore—and human evaluation through a controlled interface.

Overall, the developed RAG-based Agent provides a solid foundation for intelligent legal assistance in Arabic. It not only bridges the gap between dense retrieval and controlled generation but also offers a scalable framework for future enhancements such as domain expansion, re-ranking modules, or multilingual capabilities.

Conclusion

This thesis has investigated the intersection of natural language processing technologies and the legal domain, with a particular focus on the capabilities and limitations of large-scale language models in processing complex Arabic legal texts. We began by presenting a foundational overview of Large Language Models (LLMs), exploring their potential for legal text understanding and generation, while also highlighting the specific challenges they face in dynamic, high-stakes environments. Following this, we introduced the concept of Agentic AI, outlining its general characteristics and its emerging role in enabling autonomous, goal-driven reasoning.

To address the limitations of static language models, we turned to RAG architectures, which enhance language models by integrating external document retrieval systems. This hybrid approach offers a more grounded and context-aware solution, crucial for legal question answering and reasoning tasks

Chapter 3 addressed one of the core limitations of traditional RAG systems: the use of fixed top-k document selection. We proposed a dynamic candidate selection mechanism that adjusts the retrieval scope based on the complexity of the user query, thereby improving retrieval precision and reducing noise in the generation process.

In Chapter 4, these concepts were brought together in a fully implemented Arabic RAG pipeline. This included preprocessing, semantic embedding, FAISS-based indexing, and generation using a fine-tuned model. The resulting conversational agent enables natural language querying of Algerian legal content and delivers grounded, contextually relevant answers. Additionally, the thesis introduced a curated dataset of Algerian legal cases in Arabic, addressing a major gap in Arabic legal NLP resources.

Overall, this work offers both theoretical contributions and practical tools for advancing legal AI in under-resourced settings. By combining dynamic retrieval, generation, and agentic reasoning, the thesis lays the groundwork for the next generation of intelligent legal assistants in the Arab world.

Bibliography

Sahin Ahmed. Agentic rag: What is it and how it works? https://medium.com/@sahin.samia/agentic-rag-what-is-it-and-how-it-works-7d6a85511e00, November 2024. Accessed: 2025-05-12.

Pareto AI. The ultimate guide to retrieval-augmented generation (rag), 2024. URL https://pareto.ai/blog/retrieval-augmented-generation. Accessed: 2025-03-05.

Aisera. What is agentic ai? key trends in 2025, 2024. URL https://aisera.com/blog/agentic-ai/. Accessed: 2025-05-10.

Raghad Al-Rasheed et al. Evaluating RAG pipelines for Arabic lexical information retrieval: A comparative study of embedding and generation models. In Mo El-Haj, editor, *Proceedings of the 1st Workshop on NLP for Languages Using Arabic Script*, pages 155–164, Abu Dhabi, UAE, jan 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.abjadnlp-1.16/.

Joel Barnard. What are word embeddings?, January 2024. URL https://www.ibm.com/think/topics/word-embeddings. Accessed: 2025-05-10.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, 2021.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, 2003. URL https://jmlr.org/papers/v3/bengio03a.html.

Tom B Brown, Benjamin Mann, Nick Ryder, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Ilias Chalkidis, Abhik Jana, Matthias Hartung, Michael Bommarito, Ion Androutsopoulos, Daniel M. Katz, and Nikolaos Aletras. Multilegalbert: A multilingual legal language model for 24 languages. *arXiv preprint*, 2023. doi: 10.48550/arXiv.2305.13820.

Stanley F Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394, 1999.

Tong Chen, Hongwei Wang, Sihao Chen, Wenhao Yu, Kaixin Ma, Xinran, Hongming Zhang, and Dong Yu. Dense X retrieval: What retrieval granularity should we use? In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15159–15177, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.845. URL https://aclanthology.org/2024.emnlp-main.845/.

Kyunghyun Cho, Bart van Merrienboer, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. pages 1724–1734, 2014. URL https://doi.org/10.3115/v1/d14-1179.

Krzysztof Choromanski, Valerii Likhosherstov, and David Dohan. Rethinking attention with performers. *ICLR*, 2021.

Zhibo Chu et al. History, development, and principles of large language models-an introductory survey. *CoRR*, abs/2402.06853, 2024. URL https://doi.org/10.48550/arXiv.2402.06853.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does BERT look at? an analysis of bert's attention. pages 276–286, 2019. URL https://doi.org/10.18653/v1/W19-4828.

Wikipedia contributors. Faiss — Wikipedia, the free encyclopedia, 2025. URL https://en.wikipedia.org/w/index.php?title=FAISS&oldid=1276232158. accessed 9-March-2025.

J. Shane Culpepper, Charles L. A. Clarke, and Jimmy Lin. Dynamic trade-off prediction in multi-stage retrieval systems. *CoRR*, abs/1610.02502, 2016. URL http://arxiv.org/abs/1610.02502.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Jacob Devlin, Ming-Wei Chang, et al. 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, Volume 1*, pages 4171–4186. Association for Computational Linguistics, 2019. URL https://arxiv.org/abs/1810.04805.

Dario Di Palma. Retrieval-augmented recommender system: Enhancing recommender systems with large language models. pages 1369–1373, 09 2023. doi: 10.1145/3604915.3608889.

Bailu Ding and Jiaqi Zhai. Efficient retrieval with learned similarities. In *THE WEB CONFERENCE 2025*, 2025. https://openreview.net/forum?id=eHF4pDRWjT.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

Markus Freitag and Yaser Al-Onaizan. Beam search strategies for neural machine translation. In Thang Luong, Alexandra Birch, Graham Neubig, and Andrew Finch, editors, *Proceedings of the First Workshop on Neural Machine Translation*, pages 56–60, Vancouver, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-3207. URL https://aclanthology.org/W17-3207/.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. 2024. URL https://arxiv.org/abs/2312.10997.

GeeksforGeeks. Precision and recall in information retrieval, 2022. URL https://www.geeksforgeeks.org/precision-and-recall-in-information-retrieval/. Accessed: 2025-03-06.

GeeksforGeeks. Understanding tf-idf (term frequency-inverse document frequency), 2025. URL https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/. Last Updated: 07 Feb, 2025. Accessed: 2025-03-09.

Google Cloud. What is an ai agent?, 2025. URL https://cloud.google.com/discover/what-are-ai-agents. Accessed: 2025-05-11.

Shailja Gupta, Rajesh Ranjan, and Surya Narayan Singh. A comprehensive survey of retrieval-augmented generation (RAG): evolution, current landscape and future directions. *CoRR*, abs/2410.12837, 2024. doi: 10.48550/ARXIV.2410.12837.

Abdellah Hamouda Sidhoum, Mataoui M'hamed, Sebbak Faouzi, Ouazene Aymen, Hussine Fedoua, Amrani Mohamed Chakib, and Boumediri Takieddine. Transforming legal documents into knowledge goldmines: Application on the algerian official journal. pages 1–7, 09 2024. doi: 10.1109/AICT61888.2024.10740457.

F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, 2016. URL https://doi.org/10.1145/2827872.

Faris Hijazi, Alharbi, et al. ArabLegalEval: A multitask benchmark for assessing Arabic legal knowledge in large language models. In *Proceedings of the Second Arabic Natural Language Processing Conference*, pages 225–249, Bangkok, Thailand, aug 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.arabicnlp-1.20.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, November 1997.

Jordan Hoffmann et al. Training compute-optimal large language models. *CoRR*, abs/2203.15556, 2022. doi: 10.48550/ARXIV.2203.15556. URL https://doi.org/10.48550/arXiv.2203.15556.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018.

Dan Jurafsky and James H. Martin. *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition, 2nd Edition.* Prentice Hall series in artificial intelligence. Prentice Hall, Pearson Education International, 2009. URL https://www.worldcat.org/oclc/315913020.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing.* Prentice Hall, 3rd edition, 2019. Draft available at https://web.stanford.edu/~jurafsky/slp3/.

Muhammad Rafsan Kabir, Rafeed Sultan, Fuad Rahman, Mohammad Amin, Sifat Momen, Nabeel Mohammed, and Shafin Rahman. Legalrag: A hybrid rag system for multilingual legal information retrieval. 04 2025. doi: 10.48550/arXiv.2504.16121.

Wang-Cheng Kang and Julian J. McAuley. Self-attentive sequential recommendation. In *IEEE International Conference on Data Mining, ICDM 2018, Singapore, November 17-20, 2018*, pages 197–206. IEEE Computer Society, 2018. URL https://doi.org/10.1109/ICDM.2018.00035.

Jared Kaplan, Sam McCandlish, et al. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020. URL https://arxiv.org/abs/2001.08361.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 6769–6781. Association for Computational Linguistics, 2020. doi: 10.18653/V1/2020.EMNLP-MAIN.550.

David Kirchhoff. Metrics for evaluation of retrieval in retrieval-augmented generation, 2024. URL https://deconvoluteai.com/blog/rag/metrics-retrieval. Accessed: 2025-03-06.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Hugo Larochelle, Marc'Aurelio Ranzato, et al., editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems*, 2020.

Lexemo. Retrieval-augmented generation (rag) in legal research. URL https://e.lexemo.com/e-blog/retrieval-augmented-generation-rag-in-legal-research/. accessed April 2025.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. URL https://nlp.stanford.edu/IR-book/.

Imen Bouaziz Mezghanni and Faïez Gargouri. Information retrieval from unstructured arabic legal data. In Richard Booth and Min-Ling Zhang, editors, *PRICAI 2016: Trends in Artificial Intelligence - 14th Pacific Rim International Conference on Artificial Intelligence*, volume 9810 of *Lecture Notes in Computer Science*, pages 44–54. Springer, 2016.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Laurent Mombaerts, Terry Ding, Adi Banerjee, Florian Felice, Jonathan Taws, and Tarik Borogovac. Meta knowledge for retrieval augmented large language models. *CoRR*, abs/2408.09017, 2024. doi: 10.48550/ARXIV.2408.09017.

Humza Naveed et al. A comprehensive overview of large language models. volume abs/2307.06435, 2023. doi: 10.48550/ARXIV.2307.06435.

Joel Niklaus, Veton Matoshi, Matthias Stürmer, Ilias Chalkidis, and Daniel E. Ho. Multilegalpile: A 689gb multilingual legal corpus. In Lun-Wei Ku, Andre Martins, and Vivek

Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 15077–15094. Association for Computational Linguistics.

NVIDIA. Best practices for tuning the performance of TensorRT-LLM beam search, 2023. URL https://tensorrt-llm.continuumlabs.ai/best-practices-for-tuning-the-performance-of-tensorrt-llm/beam-search. [Accessed: 7-March-2025].

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. ACL, 2014.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8), 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683v4*, September 2023. Editor: Ivan Titov.

IBM Research. Retrieval-augmented generation (rag), 2024. URL https://research.ibm.com/blog/retrieval-augmented-generation-RAG.

Stephen Robertson and Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389, April 2009. ISSN 1554-0669. doi: 10.1561/1500000019. URL https://doi.org/10.1561/1500000019.

Nicholas Rossi, Juexin Lin, Feng Liu, Zhen Yang, Tony Lee, Alessandro Magnani, and Ciya Liao. Relevance filtering for embedding-based retrieval. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, CIKM '24, page 4828–4835. ACM, October 2024. doi: 10.1145/3627673.3680095. URL http://dx.doi.org/10.1145/3627673.3680095.

Denis Rothman. *Transformers for Natural Language Processing: Build innovative deep neural network architectures for NLP with Python, PyTorch, TensorFlow, BERT, RoBERTa, and more*. Packt Publishing Ltd, 2021. ISBN 1800568630, 9781800568631.

Deepjyoti Roy and Mala Dutta. A systematic review and research perspective on recommender systems. *Journal of Big Data*, 9(1):59, 2022. ISSN 2196-1115. doi: 10.1186/s40537-022-00592-5. URL https://doi.org/10.1186/s40537-022-00592-5.

Alireza Salemi and Hamed Zamani. Evaluating retrieval quality in retrieval-augmented generation. Amherst, MA, United States, 2023. ACM. URL https://dl.acm.org/doi/pdf/10.1145/3626772.3657957. Accessed via https://dl.acm.org/doi/pdf/10.1145/3626772.3657957?utm_source=chatgpt.com.

Kunal Sawarkar, Abhilasha Mangal, and Shivam Raj Solanki. Blended RAG: improving RAG (retriever-augmented generation) accuracy with semantic search and hybrid query-based retrievers. In *7th IEEE International Conference on Multimedia Information Processing and Retrieval, MIPR 2024, San Jose, CA, USA, August 7-9, 2024*, pages 155–161. IEEE, 2024. URL https://doi.org/10.1109/MIPR62202.2024.00031.

Natassha Selvaraj. What is retrieval augmented generation (rag)?, 2024. URL https://www.datacamp.com/blog/what-is-retrieval-augmented-generation-rag. Updated Jan 30, 2024.

Neha Sengupta, Sunil Kumar Sahu, Bokang Jia, Satheesh Katipomu, et al. Jais and jais-chat: Arabic-centric foundation and instruction-tuned open generative large language models. *CoRR*, abs/2308.16149, 2023. URL https://doi.org/10.48550/arXiv.2308.16149.

Yan-Martin Tamm, Rinchin Damdinov, and Alexey Vasilev. Quality metrics in recommender systems: Do we calculate metrics consistently? In *Fifteenth ACM Conference on Recommender Systems*, RecSys '21, page 708–713. ACM, September 2021. doi: 10.1145/3460231.3478848. URL http://dx.doi.org/10.1145/3460231.3478848.

Maruti Techlabs. What are the types of recommendation systems? https://marutitech.medium.com/what-are-the-types-of-recommendation-systems-3487cbafa7c9, 2017. Accessed: 2025-04-28.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, et al. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023b. doi: 10.48550/ARXIV.2307.09288. URL https://doi.org/10.48550/arXiv.2307.09288.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Vectorize. Designing agentic ai systems, part 1: Agent architectures, January 2025. URL https://vectorize.io/designing-agentic-ai-systems-part-1-agent-architectures/. Accessed: 2025-05-11.

Bingyu Wan, Fuxi Zhang, Zhongpeng Qi, Jiayi Ding, Jijun Li, Baoshi Fan, Yijia Zhang, and Jun Zhang. Cognitive-aligned document selection for retrieval-augmented generation. *CoRR*, abs/2502.11770, 2025. doi: 10.48550/ARXIV.2502.11770.

Thomas Wang, Adam Roberts, Daniel Hesslow, Teven Le Scao, Chung, et al. What language model architecture and pretraining objective works best for zero-shot generalization? 162: 22964–22984, 2022. URL https://proceedings.mlr.press/v162/wang22u.html.

Wikipedia contributors. Ranking (information retrieval) — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Ranking_(information_retrieval)&oldid=1262179867, 2024. accessed 7-March-2025.

Hao Wu, Lu, et al. Improving retrieval effectiveness for temporal-constrained top-k query processing. In *Information Retrieva Technology*, pages 3–15, Cham, 2017. Springer International.

Gokul Yenduri, M Ramalingam, Chemmalar G Selvi, Y Supriya, Gautam Srivastava, et al. Gpt (generative pre-trained transformer) – a comprehensive review on enabling technologies, potential applications, emerging challenges, and future directions. *Journal of Artificial Intelligence Research*, 82:123–157, 2023. doi: 10.1016/j.jair.2023.01.007.

Xinru Yu, Bin Guo, et al. Antlm: Bridging causal and masked language models. *CoRR*, abs/2412.03275, 2024. URL https://doi.org/10.48550/arXiv.2412.03275.

Jiaqi Zhai, Zhaojie Gong, Yueming Wang, Xiao Sun, Zheng Yan, Fu Li, and Xing Liu. Revisiting neural retrieval on accelerators. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, pages 5520–5531, New York, NY, USA, 2023. Association for Computing Machinery.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=SkeHuCVFDr.

Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, and Bin Cui. Retrieval-augmented generation for ai-generated content: A survey. *CoRR*, abs/2402.19473, 2024. doi: 10.48550/ARXIV.2402.19473.

Wayne Xin Zhao, Kun Zhou, et al.

Buqian Zheng. Enhancing information retrieval with learned sparse retrieval, 2024. URL https://zilliz.com/learn/enhancing-information-retrieval-learned-sparse-embeddings. Accessed: 2025-03-09.

Yujia Zhou, Yan Liu, Xiaoxi Li, Jiajie Jin, Hongjin Qian, Zheng Liu, Chaozhuo Li, Zhicheng Dou, Tsung-Yi Ho, and Philip S. Yu. Trustworthiness in retrieval-augmented generation systems: A survey. *arXiv preprint arXiv:2409.10102*, September 2020. https://arxiv.org/abs/2409.10102v1.

Chencheng Zhu, Kazutaka Shimada, Tomoki Taniguchi, and Tomoko Ohkuma. STAYKATE: hybrid in-context example selection combining representativeness sampling and retrieval-based approach - A case study on science domains. *CoRR*, abs/2412.20043, 2024. URL https://doi.org/10.48550/arXiv.2412.20043.

Shengyao Zhuang, Hang Li, and Guido Zuccon. Deep query likelihood model for information retrieval. In *Advances in Information Retrieval: 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 – April 1, 2021, Proceedings, Part II*, page 463–470, Berlin, Heidelberg, 2021. Springer-Verlag. ISBN 978-3-030-72239-5. doi: 10.1007/978-3-030-72240-1_49. URL https://doi.org/10.1007/978-3-030-72240-1_49.

Appendix

Intelligenza Artificiale

Boosting RAG Efficiency with dyRAG: Dynamic Candidate Selection for Optimal Retrieval

--Manuscript Draft--

Manuscript Number:	INA-250047
Full Title:	Boosting RAG Efficiency with dyRAG: Dynamic Candidate Selection for Optimal Retrieval
Short Title:	
Article Type:	Research Article
Keywords:	Retrieval-Augmented Generation (RAG); Dynamic Retrieval (dyRAG); large Llanguage model; Query Complexity; Relevant Information
Corresponding Author:	SALEM Mohammed University of Mascara: Universite de Mascara MASCARA, ALGERIA
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	University of Mascara: Universite de Mascara
Corresponding Author's Secondary Institution:	
First Author:	Zoubida Asmaa Boudjenane
First Author Secondary Information:	
Order of Authors:	Zoubida Asmaa Boudjenane
	SALEM Mohammed
Order of Authors Secondary Information:	
Abstract:	Retrieval-Augmented Generation (RAG) enhances language model responses by incorporating external knowledge. However, its effectiveness heavily depends on the quality of the retrieved documents. Using a fixed number of retrieved documents K may fail to adapt to varying query complexity, often leading to suboptimal retrieval either including irrelevant documents or missing crucial ones. To address this issue, we propose dyRAG, a hybrid method that dynamically adjusts K based on query characteristics while maintaining computational efficiency. Our method improves retrieval performance by maximizing relevant information and minimizing noise. Experimental results show that it outperforms fixed-K retrieval, offering a more effective solution for optimizing RAG systems. Compared to traditional methods, it dynamically adjusts based on the query and dataset characteristics, offering greater adaptability across various contexts.
Opposed Reviewers:	
Additional Information:	
Question	**Response**
By submitting this article I agree with the IOS Press author copyright agreement	Yes
By submitting this article I agree with the IOS Press privacy policy	Yes

Boosting RAG Efficiency with dyRAG: Dynamic Candidate Selection for Optimal Retrieval

Journal Title
XX(X):1–9

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Zoubida Asmaa Boudjenane [1] and Mohammed SALEM [2]

Abstract

Retrieval-Augmented Generation (RAG) enhances language model responses by incorporating external knowledge. However, its effectiveness heavily depends on the quality of the retrieved documents. Using a fixed number of retrieved documents K may fail to adapt to varying query complexity, often leading to suboptimal retrieval either including irrelevant documents or missing crucial ones. To address this issue, we propose dyRAG, a hybrid method that dynamically adjusts K based on query characteristics while maintaining computational efficiency. Our method improves retrieval performance by maximizing relevant information and minimizing noise. Experimental results show that it outperforms fixed-K retrieval, offering a more effective solution for optimizing RAG systems. Compared to traditional methods, it dynamically adjusts based on the query and dataset characteristics, offering greater adaptability across various contexts.

Keywords

Retrieval-Augmented Generation (RAG), Dynamic Retrieval (dyRAG), large Llanguage model, Query Complexity, Relevant Information

Introduction

Large Language Models (LLMs) have emerged as state-of-the-art artificial intelligence systems that excel at understanding and generating human-like text Naveed et al. (2023), Built on deep learning architectures, particularly transformers Vaswani et al. (2017),these models showcase impressive adaptability across diverse tasks, such as machine translation Gu et al. (2018), text generation Liang et al. (2024) and question answering Zhang et al. (2023). Notable examples of LLMs include OpenAI GPT series Brown et al. (2020), Google BERT Devlin et al. (2019) and T5 Raffel et al. (2020). Despite their impressive capabilities, LLMs suffer from hallucination Naveed et al. (2023), often generating confident but incorrect information. This is a critical challenge for knowledge-intensive tasks requiring high accuracy Naveed et al. (2023). Additionally, their static nature being trained on fixed datasets Naveed et al. (2023) limits their ability to update knowledge dynamically, making them prone to outdated or inconsistent responses, especially for time-sensitive information Mousavi et al. (2025).

Researchers have explored various approaches to address this limitation, including transfer learning Zhuang et al. (2021), fine-tuning Howard and Ruder (2018) which adapt pre-trained models to specific tasks. Despite their strengths, they are computationally expensive and impractical for real-time knowledge updates. A promising alternative is Retrieval-Augmented Generation (RAG) Chuyuan et al. (2025), which integrates real-time retrieval from external knowledge sources with generative models, enables LLMs to access up-to-date information without retraining,(RAG) faces several limitations. For instance, The performance of RAG heavily depends on the quality of the retriever Karpukhin et al. (2020), as irrelevant or noisy documents can degrade the generator output. Additionally, a critical challenge is determining the optimal number of retrieved passages K to feed into the generator Lewis et al. (2020b). Often resulting in either insufficient or excessive information being retrieved, reduce coherence and increase computational overhead. This trade-off makes K a key factor in balancing performance and efficiency.

Current approaches, such as static K selection Lewis et al. (2020b), dynamic K adjustment attempt to address this trade-off but often introduce complexity or latency, Hybrid approaches Yuan et al. (2024),integrates static retrieval with dynamic retrieval. However, these approaches face significant limitations, including, high computational overhead, and a dependency on high-quality metadata for optimal performance. These limitations highlight the need for a more K selection robust approach, capable of balancing efficiency and accuracy across varying query types.

In this paper, we investigate the problem of selecting K in RAG and propose a novel approach to dynamically adjust K based on thresholds and a Mixture of Logits (MoL) Ding and Zhai (2024) scoring mechanism, ensuring efficient and context-sensitive candidate selection while reducing computational costs. The use of adaptive thresholds algorithm improves the relevance of retrieved documents, refining the set to include only the most contextually appropriate matches. Unlike traditional approaches, dyRAG

[1]Computer Science Department, University of Mascara, Algeria
[2]LISYS Laboratory, University of Mascara, Algeria

Corresponding author:
Mohammed SALEM, LISYS Lab, University of Mascara, BP 305 Route Mamounia, Mascara, 29000, Algeria.
Email: salem@univ-mascara.dz

adapts to the query and dataset characteristics, making it more flexible for diverse contexts. Furthermore, its design for handling large datasets ensures scalable and precise retrieval, which is crucial for RAG systems working with extensive corpora.

The remainder of this paper is as follows :We start by presenting an overview of the research problem, We summarize existing solutions, and highlight the gaps that our work addresses through a more dynamic approach to k selection.
Section 2: summarizes existing solutions, additionally, it provides a critical analysis of the strengths and limitations of these methods, as it identifies key gaps in their design and performance. As a result, there is a critical need for a more effective approach.
Section 3: aims to provide a detailed explanation of the RAG framework, including its components and how it operates.
Section 4: introduces the novel hybrid dynamic selection algorithm. Additionally, it provides detailed description of its methodology, explains its design, and discusses the manner in which it overcomes the limitations of existing approaches. Particular emphasis is placed on the benefits of this approach, particularly in improving retrieval accuracy and computational efficiency.
Section 5: presents experimental results that demonstrate the effectiveness of the hybrid dynamic selection algorithm. Highlight its superiority in terms of retrieval performance and adaptability to diverse query complexities.

Related works

The selection of an optimal number of retrieved items K in Retrieval-Augmented Generation (RAG) systems plays a crucial role in ensuring accurate and efficient responses. Several methods, including static, dynamic, and hybrid approaches have been proposed to address this challenge While these methods show promise in improving performance, they often face trade-offs between adaptability, efficiency when applied to complex or diverse query patterns. In this section, we will explore these methods, highlight their respective strengths and weaknesses, and introduce our hybrid solution as a more robust and adaptable alternative.

Static Top-K selection in Retrieval :In traditional Retrieval-Augmented Generation (RAG) systems, static K-selection is a widely used method where a fixed number of top-k documents are retrieved based on their relevance scores. Sparse retrieval Bai et al. (2020) methods where documents and queries are represented as high-dimensional, sparse vectors In these method, the ”keys” represent documents to be retrieved, and the ”values” are the importance or weight of those terms in the document such as query likelihood models Lafferty and Zhai (2017), TF-IDF Robertson and Walker (1997) and BM25 Zaragoza and Robertson (2009) often rely on this static approach to identify and rank documents. For instance, if k=5, the system retrieves the top 5 documents deemed most relevant to a query, regardless of the query's complexity or the variability in relevance distribution among documents. A major drawback of this approach is its rigidity. Since the number of retrieved documents K is fixed, it fails to adapt to the varying complexity of queries. This one-size-fits-all strategy can lead to suboptimal performance across diverse datasets or tasks.

Dynamic top-K Selection in Retrieval :addresses the limitations of static K by varying the number of retrieved documents based on the specific query or context. These approaches typically involve heuristics, machine learning models, or adaptive algorithms to determine the optimal K for each query such as Dynamic Query Cutoff (DQC) Culpepper et al. (2016) This approach uses machine learning models trained on query-document features to predict the optimal K for retrieval, a reinforcement learning-based approach Zhou and Agichtein (2020) where the agent learns to adjust K by maximizing downstream task performance (e.g accurate response generation in RAG) Another example is the Dynamic Selection of K Nearest Neighbors Hulett et al. (2012) where the selection process often depends on the distribution of neighbors, their distances to the query point, or their relevance scores. However, these methods come with limitations. They often introduce computational overhead due to the need for dynamic evaluation, which can increase latency in large-scale systems. Additionally, their performance heavily depends on the quality of the underlying models or distance metrics, making them sensitive to noisy data. As a result, achieving generalization across diverse datasets or domains remains a significant challenge.

Hybrid Approaches For Top-K selection in Retrieval: combine elements of static and dynamic K-selection to achieve a balance between simplicity and adaptability. These systems might use a static K as a baseline but dynamically adjust it such as, the ”Blended RAG” Sawarkar et al. (2024) approach enhances retrieval accuracy by combining semantic search techniques with hybrid query strategies, Furthermore STAYKATE (Static-Dynamic Hybrid Selection) Zhu et al. (2024) enhances LLM performance in scientific information extraction by integrating representativeness sampling from active learning with a retrieval-based strategy. Another example is DR-RAG (Dynamic Relevance for Retrieval-Augmented Generation) Hei et al. (2024) First, it retrieves an initial set of k_1 documents. Then, a classifier dynamically evaluates their relevance and retrieves additional relevant K_2 documents to enhance recall. However hybrid methods may face challenges in determining when to apply static or dynamic K-selection effectively, leading to potential trade-offs between retrieval accuracy and system efficiency. These limitations highlight the ongoing need for robust mechanisms that dynamically and efficiently adapt K to the unique characteristics of each query and dataset.

Our method leverages adaptive thresholds to dynamically refine the list of retrieved documents, to make sure that only the most contextually relevant matches are included. In contrast to traditional static or fixed K-selection techniques, which employ a consistent retrieval strategy for every query, our approach adapts to the specific characteristics of both the query and the dataset.

Overview of Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) is a method designed to enhance the performance of large language

models (LLMs) by integrating external, reliable knowledge sources into their response generation process **?**. RAG combines search capabilities with LLM prompting. The process involves feeding both the userss query and the relevant information retrieved by a search algorithm into the LLMs prompt Gao et al. (2023), enabling the model to generate answers based on the provided context. As

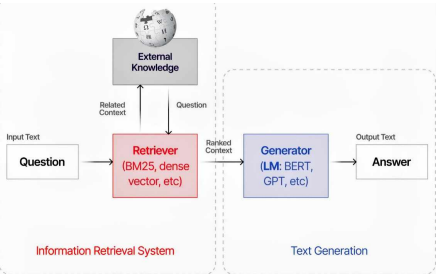


Figure 1. RAG system component Gupta et al. (2024).

illustrated in figure 1, the RAG system is built around two main components: the retriever and the generator. The retriever's role is to find relevant information from a pre-built data store, while the generator utilizes this retrieved data to produce coherent and contextually appropriate content. The overall RAG process operates as follows:

Retrieval in RAG Systems

Retrieval is the process of identifying and gathering information system resources that match a specific information need. To break it down, imagine these resources as a key-value store, represented as pairs:

$$\{(k_i, v_i)\}_{i=1}^{N} \tag{1}$$

where each key k_i is linked to a corresponding value v_i (often, the key and value are the same). When a query q is provided, the goal is to find the top-k keys that are most similar to the query using a similarity function s, and then retrieve their associated valuesGupta et al. (2024).

Depending on the similarity function used, retrieval methods can be grouped into categories such as sparse retrievalBai et al. (2020)which relies on exact term matching techniques like BM25 and TF-IDF, dense retrievalKarpukhin et al. (2020)which Uses deep learning models to encode queries and documents into dense vector embeddings, retrieving relevant documents based on semantic similarity rather than exact word matching, and others. For the widely used sparse and dense retrieval approaches, the process typically involves two main steps:

- Encoding each object into a specific representation (e.g., term-based for sparse retrieval, vector embeddings for dense retrieval).
- Building an index to organize and efficiently retrieve relevant data during search .

This structured approach ensures that the most relevant information is retrieved quickly and accurately Zhao et al. (2024).

Generator in RAG Systems

In Retrieval-Augmented Generation (RAG) systems, the generator Huang and Huang (2024) is a key component responsible for crafting the final output. It works by combining the retrieved information with the users input query to produce a well-structured and contextually relevant response.

Once the retriever fetches relevant data from external sources,the generator processes and integrates this information into a well-structured and meaningful response. At the core of this process is a Large Language Model (LLM), which ensures that the generated text is not only fluent and accurate but also remains relevant to the original query.

In various generative tasks, different models are selected based on the specific requirements: Models like BART Lewis et al. (2020a), GPT Brown et al. (2020)and and T5 Raffel et al. (2020) are commonly used for tasks such as text summarization and translation, Vision-Language Models (VLMs) Radford et al. (2021) are employed to generate textual descriptions from images, for text-to-Code Generation: Models like OpenAIs Codex Chen et al. (2021) are designed to translate natural language prompts into executable code.

dyRAG: The proposed Solution

In traditional candidate selection systems, the reliance on static or fixed top-K retrieval lead to suboptimal retrieval performance, especially in scenarios where the relevance distribution of candidates is highly variable. In this section, we introduce the Mixture of Logits (MoL) and describe its role in representing a learned similarity function. Table 1 summarizes the notations used in this paper and we propose Dynamic Candidate Selection, a novel algorithm that adaptively refines the candidate selection process by leveraging component-level embeddings and a Mixture of Logits (MoL) scoring mechanism.

Mixture of Logits

The Mixture of Logits (MoL) Zhai et al. (2023); Ding and Zhai (2024) method provides a flexible and adaptive mechanism for computing similarity scores between queries and items. Given a query q and an item x, MoL assumes that both are mapped to P groups of low-rank embeddings, denoted as $f_p(q)$ and $g_p(x)$, respectively. These embeddings are parameterized by neural networks that extract features from the query and item. The similarity between q and x is computed as a weighted sum of the inner products of these embeddings, where the weights $\pi_p(q, x)$ Zhai et al. (2023); Ding and Zhai (2024) are adaptive gating parameters constrained to sum to 1:

$$\text{Similarity}(q, x) = \sum_{p=1}^{P} \pi_p(q, x) \cdot \langle f_p(q), g_p(x) \rangle. \tag{2}$$

This formulation allows MoL to capture complex relationships between queries and items, making it particularly suitable for dynamic candidate selection tasks.

To efficiently scale MoL for large datasets and hardware-optimized implementations, the formulation is extended by decomposing the dot products into batched outer

Notation	Description
q (Q, $\|Q\|$)	A single query (set of all queries, total number of queries).
x (X, $\|X\|$)	A single item (set of all items, total number of items).
$\phi(q, x)$	The learned similarity function, Mixture-of-Logits (MoL), which computes the similarity between q and x.
P (P_q, P_x)	Total number of low-rank embeddings ($P = P_q \times P_x$), where P_q and P_x are the number of embeddings for q and x, respectively.
$\pi_p(q, x)$	Weight for the p-th (or p_q-th and p_x-th) embedding set for (q, x).
$f(q)$ ($f_p(q)$)	Learned embedding for the query (p-th component-level embedding for q).
$g(x)$ ($g_p(x)$)	Learned embedding for the item (p-th component-level embedding for x).
$\langle f(q), g(x)\rangle$	Dot product similarity function: $\langle f(q), g(x)\rangle = g(x)^T f(q)$.

Table 1. Notation Table

products of query-side and document-side embeddings. This decomposition improves computational efficiency, particularly on accelerators like GPUs, by normalizing the embeddings using the l_2-norm: Zhai et al. (2023)

$$\phi(q, x) = \sum_{pq=1}^{P_q} \sum_{px=1}^{P_x} \pi_{pq,px}(q, x) \frac{\langle f_{pq}(q), g_{px}(x)\rangle}{||f_{pq}(q)||_2 ||g_{px}(x)||_2} \quad (3)$$

Since embedding normalization can be precomputed, both formulations remain interchangeable in practical applications. Furthermore, it is possible to decompose any high-rank matrix into a mixture of logits based on low-rank matrices, demonstrating the flexibility and scalability of this approach in large-scale information retrieval tasks.

Retrieval Algorithm

We propose an **adaptive threshold mechanism** (referred to as (Dynamic Candidate Selection) that leverages the Mixture of Logits (MoL) framework to enhance the candidate retrieval process. The mechanism operates as follows:

1. **Component-Level Embeddings**: Component-level embeddings are generated for all items in the dataset X. These embeddings facilitate efficient similarity computations during retrieval. Formally,

$$X_p \leftarrow \{g_p(x) \mid x \in X\}. \quad (4)$$

2. **Initial Candidate Retrieval:** This step involves computing similarity scores between a query representation and item representations for each feature component $p \in P$.

$$S_p \leftarrow \{\langle f_p(q), g_p(x)\rangle : x \in X_p\} \quad (5)$$

Here, $f_p(q)$ and $g_p(x)$ represent the feature embeddings of the query q and item x for the p-th component, respectively. The dot product $\langle f_p(q), g_p(x)\rangle$ measures the relevance of each item $x \in X_p$ with respect to q, producing a set of scores S_p .

3. **Dynamic Threshold Adjustment**: To improve retrieval quality, **Mixture of Logits (MoL)** scores are computed for each candidate $x \in G$. The adaptive gating weights $\pi_p(q, x)$ allow the algorithm to dynamically adjust the retrieval threshold T_{adaptive} based on the MoL scores. The scoring function is defined as:

$$\phi(q, x) = \sum_{p=1}^{P} \pi_p(q, x) \cdot \langle f_p(q), g_p(x)\rangle. \quad (6)$$

The adaptive threshold T_{adaptive} is set as the minimum score among the candidates:

$$T_{\text{adaptive}} = \min\{s \mid s \in G\}. \quad (7)$$

4. **Refinement and Top-K Selection**: Using the adaptive threshold T_{adaptive}, additional relevant candidates are retrieved, expanding the candidate set G'. This is achieved by including candidates whose scores exceed the threshold:

$$G' \leftarrow G \cup \{x \mid s_p \geq T_{\text{adaptive}}\}. \quad (8)$$

The algorithm then sorts G' based on MoL scores to select the most relevant top-k candidates.

4. **Exact Top-K Selection**: Finally, the candidates in G' are sorted by their MoL scores, and the exact top-k items are extracted:

$$G_{\text{final}} = \text{Top-k}(G', \phi(q, x)). \quad (9)$$

As shown in Algorithm1, the retrieval process dynamically adjusts the threshold based on MoL scores.

Evaluation metrics in recommendation system

The retrieval algorithm is integrated into a recommendation system, and its effectiveness is assessed using standard ranking metrics. These metrics are widely used in sequential recommendation tasks to evaluate the model's ability to rank relevant items higher in a user's preference list:

- **Hit Rate at** K (HR@K) evaluates the proportion of users who have at least one relevant item in their top-K recommendations. It is computed as Jadon and Patil (2023):

$$HR@k = \frac{1}{|U|} \sum_{u \in U} I(|\text{rel}(u) \cap \text{rec}_k(u)| > 0) \quad (10)$$

where $\text{rel}(u)$ denotes the relevant items for user u, $\text{rec}_k(u)$ represents the top-K recommended items, and $I(\cdot)$ is an indicator function that returns 1 if at least one relevant item is present in the recommendations, otherwise 0.

Algorithm 1 Hybrid Exact Top-k with Threshold-Based k Selection

Input: Query q, Set of items X, Component-level embeddings: $f_p(q)$, $g_p(x)$ for $p \in P$, $x \in X$, Initial threshold T_{init}

Output: Exact top k items based on dynamic threshold selection, G_{final}

Set $G \leftarrow \emptyset$ ▷ Initial candidate set
for each component $p \in P$ **do**
  $X_p \leftarrow \{g_p(x) \mid x \in X\}$ ▷ Precompute embeddings
end for
1. Initial Candidate Retrieval:
for each component $p \in P$ **do**
  Compute dot product scores: with (eq(5))
  Retrieve items with scores $S_p \geq T_{\text{init}}$
  Add these items to G
end for
2. Adjust k Dynamically:
for each $x \in G$ **do**
  Compute MoL scores $s \leftarrow \phi(q, x)$ using : eq(6)
  Set $T_{\text{adaptive}} = \min\{s : s \in G\}$
end for
3. Refine Candidate Set with Adaptive k:
$G' \leftarrow \emptyset$
for each component $p \in P$ **do**
  Retrieve items from X_p with scores $S_p \geq T_{\text{adaptive}}$
  Add these items to G'
end for
4. Select Exact Top-k Items:
for each component $p \in P$ **do**
  Compute MoL scores for all items in G'
  Sort G' by MoL scores in descending order
  Select the top k items from G' where k is the number of items in G' exceeding T_{adaptive}
end for
Return: G_{final} ▷ Retrieve Top k items from G'

HR@K can be evaluated at different values of K (e.g., 3, 5, 10) to assess model performance across various ranking depths.

- **Mean Reciprocal Rank (MRR)**:is a ranking quality metric that measures how quickly a system retrieves the first relevant item. It is calculated as the average of reciprocal ranks across all users or queries,MRR ranges from 0 to 1, with higher values indicating better performance Jadon and Patil (2023).

$$\text{MRR} = \frac{1}{|U|} \sum_{u=1}^{|U|} \frac{1}{\text{rank}_u}, \tag{11}$$

where rank_u is the position of the first relevant item for user u within the top-K results.
U represents the total number of users (for recommendation systems) or queries (for information retrieval tasks) in the dataset.

- **Normalized Discounted Cumulative Gain (NDCG)**: Assesses the ranking quality while accounting for position importance. The NDCG@k is computed as:Jadon and Patil (2023)

$$\text{NDCG@}k = \frac{1}{|U|} \sum_{u=1}^{|U|} \frac{\text{DCG@}k}{\text{IDCG@}k}, \tag{12}$$

where DCG@k is the Discounted Cumulative Gain at position k:

$$\text{DCG@}k = \sum_{i=1}^{k} \frac{2^{\text{rel}_i} - 1}{\log_2(i+1)}, \tag{13}$$

and IDCG@k is the Ideal DCG@k, computed by sorting the items by their true relevance scores.

Experimental Results

In order to measure the effectiveness of our proposed method, we perform a comprehensive evaluation of both the retrieval algorithm and the Mixture-of-Logits (MoL) approach. This assessment focuses on the next-item prediction task, a fundamental challenge in recommendation systems Zhu et al. (2018) Kang and McAuley (2018), where the goal is to predict the most relevant item a user will interact with next based on their past interactions.

Dataset description

We conducted experiments using two MovieLens datasets : ML-100K and ML-1M Harper and Konstan (2015)which are widely used benchmarks for evaluating sequential recommendation models as detailed in Table2.

Dataset	Description
MovieLens-100K	100,000 ratings (1-5 scale) from 943 users on 1,682 movies. Each user has rated at least 20 movies. Includes user demographic information (age, gender, occupation, zip code).
MovieLens-1M	1,000,209 ratings from 6,040 users on approximately 3,900 movies. Data collected from users who joined MovieLens in 2000. Represents a larger-scale recommendation scenario.

Table 2. Summary of MovieLens datasets

Experiments Setup

For both datasets, We utilize the SASRec architecture as our sequential user encoder, a model renowned for achieving state-of-the-art performance in next-item prediction tasks. This architecture processes the user's historical interaction sequence, generating embeddings that encapsulate the user's preferences at each time step. These embeddings serve as the foundation for predicting the next item in the sequence.

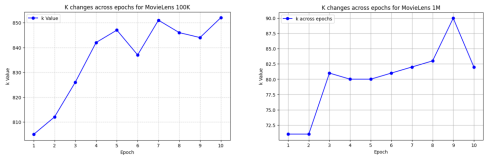
The query q represents the user's state at a specific time step, derived from their interaction history. In the MoL (Mixture of Logits) framework, q is transformed into Pq embeddings through a multi-layer perceptron (MLP).

Impact of Hyperparameters

For fair comparison, we maintained consistent architectural choices and training conditions across all experiments, We conducted an extensive hyperparameter analysis comparing both approaches (Hybrid+SAS and MoL+SAS) across different architectural configurations. All experiments were Implemented in TensorFlow and trained on a Google Colab environment with a T4 GPU. We use the Adam optimizer with a learning rate of 0.001. For the hybrid algorithm, we initialize the threshold (Tinit) to 0.3 and adaptively adjust it during training. We discuss detailed hyperparameter settings in table 3, table 4

Both approaches demonstrate significant performance improvements as model capacity increases, with larger configurations consistently delivering better results. For instance, when increasing the model's capacity such as expanding the (Max Sequence Length: 512, Embedding Dimension: 512, Number of Heads: 4, Feed-Forward Dimension: 512) the hybrid algorithm combined with SASRec (Hybrid+SAS) achieves notable gains, particularly in the most resource-intensive setup. On the ML-100K dataset, Hybrid+SAS reaches a score of**0.8120** compared to the baseline's **0.8016**, while on ML-1M, it achieves **0.8350** versus **0.8276** The ML-1M dataset generally benefits more from increased capacity, with the performance gap between datasets narrowing as the model scales. While smaller configurations provide a balance of efficiency and performance, the largest configuration, despite its higher computational demands, yields the best results, making it suitable for scenarios where resources are not a constraint. Both approaches show similar benefits from scaling, but Hybrid+SAS maintains a consistent edge in performance.

As shown in Figures 2a and 2b, the value of k fluctuates across epochs for both MovieLens 100K and MovieLens 1M datasets. These variations indicate the adaptive nature of k in response to the dataset characteristics and training progress. In the following section, we analyze how these changes influence model performance.



(a) K changes across epochs for MovieLens 100K

(b) K changes across epochs for MovieLens 1M

Figure 2. Comparison of K changes across epochs for different datasets

the value of K changed dynamically during training, with different behaviors. For the MovieLens 100K dataset, K showed a steady increase across epochs, whereas for the MovieLens 1M dataset, K began at a lower value, experienced fluctuations, peaked, and then stabilized. This indicates that the larger dataset necessitated more adaptive adjustments in retrieval compared to the smaller dataset.
Figure 3 summarizes the performance of our hybrid algorithm compared to the MoL-based approach on the MovieLens 100K and 1M datasets, both tested using a Max Sequence Length of 50, an Embedding Dimension of 128, 2 Attention Heads, a Feedforward Dimension of 128, a Batch Size of 128, and trained for 10 epochs .

The variations in K help explain the performance differences seen in the fig 3. The Hybrid+SAS model applied to the MovieLens 1M dataset, which exhibited more dynamic changes in K achieved higher HR@50(0.3631) and HR@200(0.6170) scores, reflecting better long-range recommendation quality. The fluctuations in K in the 1M dataset likely allowed the model to balance exploration and precision, improving its overall ranking performance.

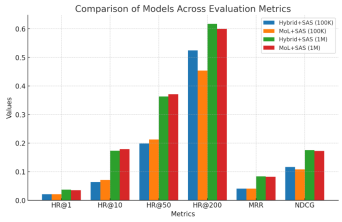


Figure 3. Comparison of Models Across Evaluation Metrics

On the other hand, the more gradual K increase in the 100K dataset resulted in relatively lower scores, suggesting that a static or overly conservative K selection might limit retrieval effectiveness.

Additionally, the MoL+SAS model achieved better performance than Hybrid+SAS in HR@10 for both datasets. This aligns with the observation that MoL+SAS tended to retrieve fewer but more relevant items at shorter ranking positions. This behavior can be attributed to how K evolved during training?smaller K values in earlier epochs likely enabled MoL+SAS to maintain higher precision at shorter ranks. These findings highlight the importance of dynamically tuning K based on dataset characteristics to optimize recommendation effectiveness.

The results demonstrate that the Hybrid Exact Top-k algorithm effectively leverages both sequential user behavior and item metadata to improve recommendation quality. The adaptive MoL threshold allows the model to dynamically refine candidate items during training, leading to better performance. The improvements are consistent across both datasets, highlighting the robustness of our approach.

Conclusion

Choosing the right value for K in retrieval-based systems is a significant challenge, as it directly impacts both the efficiency and accuracy of information retrieval. While using a fixed K is straightforward, it often fails to adapt to the varying complexities of different queries. This can result in either missing important information or retrieving irrelevant data, which adds noise to the results. On the other hand, dynamically adjusting K offers greater flexibility but comes with its own set of challenges, such as increased computational costs and inconsistencies in determining the optimal retrieval size.

To address these limitations, we introduced a hybrid dynamic K selection strategy that strikes a balance between

Table 3. Performance comparison of Hybrid+SAS and MoL+SAS models on the 100kMovies dataset

Model	Max Seq. Len.	Embed. Dim.	Heads	FFN Dim.	Batch Size	Epochs	Val. Loss	Val. Acc.
Hybrid+SAS	50	128	2	128	128	12	5.3036	0.1584
MoL+SAS	50	128	2	128	128	12	5.3152	0.1582
Hybrid+SAS	128	256	4	256	128	10	3.6364	0.4301
MoL+SAS	128	256	4	256	128	10	3.6374	0.4299
Hybrid+SAS	512	512	4	512	128	10	**1.2808**	**0.8120**
MoL+SAS	512	512	4	512	128	10	1.3050	0.8016

Table 4. Performance comparison of Hybrid+SAS and MoL+SAS models on the 1M Movies dataset

Model	Max Seq. Len.	Embed. Dim.	Heads	FFN Dim.	Batch Size	Epochs	Val. Loss	Val. Acc.
Hybrid+SAS	50	128	2	128	128	10	**4.5721**	**0.1530**
MoL+SAS	50	128	2	128	128	10	4.5733	0.1526
Hybrid+SAS	128	256	4	256	128	10	**3.4152**	**0.3680**
MoL+SAS	128	256	4	256	128	10	3.4671	0.3627
Hybrid+SAS	512	512	4	512	128	10	**1.0275**	**0.8350**
MoL+SAS	512	512	4	512	128	10	1.0350	0.8276

adaptability and efficiency. Our approach adjusts K based on the unique characteristics of each query while keeping computational demands manageable. This method has shown promising results, improving retrieval performance by minimizing unnecessary noise and ensuring that critical information is consistently captured.

Future work can explore enhancing our model with learning-based retrieval strategies such as reinforcement learning or attention mechanisms could refine the dynamic selection of K Additionally, testing our approach in real-world scenarios and conducting large-scale evaluations across diverse domains will help validate its robustness and adaptability in various knowledge bases and retrieval settings.

Declaration of conflicting interests

The authors have no Conflict of Interests to share.

References

Bai Y, Li X, Wang G, Zhang C, Shang L, Xu J, Wang Z, Wang F and Liu Q (2020) Sparterm: Learning term-based sparse representation for fast text retrieval. *CoRR* abs/2010.00768. URL https://arxiv.org/abs/2010.00768.

Brown TB, Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A et al. (2020) Language models are few-shot learners. *CoRR* abs/2005.14165. URL https://arxiv.org/abs/2005.14165.

Chen M, Tworek J, Jun H, Yuan Q, de Oliveira Pinto HP, Kaplan J, Edwards H, Burda Y, Joseph N, Brockman G, Ray A et al. (2021) Evaluating large language models trained on code. *CoRR* abs/2107.03374. URL https://arxiv.org/abs/2107.03374.

Chuyuan W, Ke D, Shengda Z, Hongchun W, Shuqiang H and Jie L (2025) Enhanced recommendation systems with retrieval-augmented large language model. *JAIR* 82. DOI:https://doi.org/10.1613/jair.1.17809.

Culpepper JS, Clarke CLA and Lin J (2016) Dynamic cutoff prediction in multi-stage retrieval systems. In: *Proceedings of the 21st Australasian Document Computing Symposium*. Melbourne, Australia. URL https://culpepper.io/publications/ccl16-adcs.pdf.

Devlin J, Chang MW, Lee K and Toutanova K (2019) BERT: Pre-training of deep bidirectional transformers for language understanding. In: Burstein J, Doran C and Solorio T (eds.) *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Minneapolis, Minnesota. DOI:10.18653/v1/N19-1423.

Ding B and Zhai J (2024) Efficient retrieval with learned similarities. *CoRR* abs/2407.15462. DOI:10.48550/ARXIV.2407.15462.

Gao Y, Xiong Y, Gao X, Jia K, Pan J, Bi Y, Dai Y, Sun J, Guo Q, Wang M and Wang H (2023) Retrieval-augmented generation for large language models: A survey. *CoRR* abs/2312.10997. URL https://doi.org/10.48550/arXiv.2312.10997.

Gu J, Wang Y, Cho K and Li VO (2018) Search engine guided neural machine translation. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32. pp. 5133–5140. DOI:10.1609/AAAI.V32I1.12013.

Gupta S, Ranjan R and Singh SN (2024) A comprehensive survey of retrieval-augmented generation (RAG): evolution, current landscape and future directions. *CoRR* abs/2410.12837. DOI:10.48550/ARXIV.2410.12837.

Harper FM and Konstan JA (2015) The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems* 5(4). DOI:10.1145/2827872.

Hei Z, Liu W, Ou W, Qiao J, Jiao J, Song G, Tian T and Lin Y (2024) DR-RAG: applying dynamic document relevance to retrieval-augmented generation for question-answering. *CoRR* abs/2406.07348. DOI:10.48550/ARXIV.2406.07348.

Howard J and Ruder S (2018) Universal language model fine-tuning for text classification. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL , Melbourne, Australia,*. pp. 328–339. DOI:10.18653/V1/P18-1031.

Huang Y and Huang J (2024) A survey on retrieval-augmented text generation for large language models. *CoRR* abs/2404.10981. DOI:10.48550/ARXIV.2404.10981.

Hulett C, Hall A and Qu G (2012) Dynamic selection of k nearest neighbors in instance-based learning. In: *2012 IEEE 13th International Conference on Information Reuse & Integration (IRI)*. Las Vegas, NV, USA. DOI:10.1109/IRI.2012.6302995.

Jadon A and Patil A (2023) A comprehensive survey of evaluation techniques for recommendation systems. *CoRR* abs/2312.16015. DOI:10.48550/ARXIV.2312.16015.

Kang W and McAuley JJ (2018) Self-attentive sequential recommendation. *CoRR* abs/1808.09781. URL http://arxiv.org/abs/1808.09781.

Karpukhin V, Oguz B, Min S, Wu L, Edunov S, Chen D and Yih W (2020) Dense passage retrieval for open-domain question answering. *CoRR* abs/2004.04906. URL https://arxiv.org/abs/2004.04906.

Lafferty J and Zhai C (2017) Document language models, query models, and risk minimization for information retrieval. *SIGIR Forum* 51(2): 251–259. DOI:10.1145/3130348.3130375. URL https://doi.org/10.1145/3130348.3130375.

Lewis M, Liu Y, Goyal N, Ghazvininejad M, Mohamed A, Levy O, Stoyanov V and Zettlemoyer L (2020a) BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension : 7871–7880DOI:10.18653/V1/2020.ACL-MAIN.703.

Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, Küttler H, Lewis M, Yih Wt et al. (2020b) Retrieval-augmented generation for knowledge-intensive nlp tasks. *CoRR* abs/2005.11401. URL https://arxiv.org/abs/2005.11401.

Liang X, Wang H, Wang Y, Song S, Yang J, Niu S, Hu J, Liu D, Yao S, Xiong F and Li Z (2024) Controllable text generation for large language models: A survey. *CoRR* abs/2408.12599. DOI:10.48550/ARXIV.2408.12599.

Mousavi SM, Alghisi S and Riccardi G (2025) Llms as repositories of factual knowledge: Limitations and solutions. *CoRR* abs/2501.12774. DOI:10.48550/ARXIV.2501.12774.

Naveed H, Khan AU, Qiu S, Saqib M, Anwar S, Usman M, Barnes N and Mian A (2023) A comprehensive overview of large language models. *CoRR* abs/2307.06435. DOI:10.48550/ARXIV.2307.06435. URL https://doi.org/10.48550/arXiv.2307.06435.

Radford A, Kim JW, Hallacy C, Ramesh A, Goh G, Agarwal S, Sastry G, Askell A, Mishkin P, Clark J, Krueger G and Sutskever I (2021) Learning transferable visual models from natural language supervision. *CoRR* abs/2103.00020. URL https://arxiv.org/abs/2103.00020.

Raffel C, Shazeer N, Roberts A, Lee K, Narang S, Matena M, Zhou Y, Li W and Liu PJ (2020) Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* 21: 140:1–140:67. URL https://arxiv.org/abs/1910.10683.

Robertson SE and Walker S (1997) On relevance weights with little relevance information. In: Belkin NJ, Narasimhalu AD, Willett P, Hersh WR, Can F and Voorhees EM (eds.) *SIGIR '97: Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Philadelphia, PA, USA, pp. 16–24. DOI:10.1145/258525.258529.

Sawarkar K, Mangal A and Solanki SR (2024) Blended RAG: improving RAG (retriever-augmented generation) accuracy with semantic search and hybrid query-based retrievers. *CoRR* abs/2404.07220. DOI:10.48550/ARXIV.2404.07220.

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L and Polosukhin I (2017) Attention is all you need. In: *Advances in Neural Information Processing Systems*, volume 30. URL https://arxiv.org/abs/1706.03762.

Yuan Y, Liu C, Yuan J, Sun G, Li S and Zhang M (2024) A hybrid RAG system with comprehensive enhancement on complex reasoning. *CoRR* abs/2408.05141. DOI:10.48550/ARXIV.2408.05141.

Zaragoza H and Robertson SE (2009) The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* 3(4): 333–389. DOI:10.1561/1500000019.

Zhai J, Gong Z, Wang Y, Sun X, Yan Z, Li F and Liu X (2023) Revisiting neural retrieval on accelerators. *CoRR* abs/2306.04039. DOI:10.48550/ARXIV.2306.04039.

Zhang Q, Chen S, Xu D, Cao Q, Chen X, Cohn T and Fang M (2023) A survey for efficient open domain question answering. In: Rogers A, Graber JB and Okazaki N (eds.) *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Toronto, Canada, pp. 14447–14465. DOI:10.18653/v1/2023.acl-long.808.

Zhao P, Zhang H, Yu Q, Wang Z, Geng Y, Fu F, Yang L, Zhang W and Cui B (2024) Retrieval-augmented generation for ai-generated content: A survey. *CoRR* abs/2402.19473. DOI:10.48550/ARXIV.2402.19473.

Zhou J and Agichtein E (2020) Rlirank: Learning to rank with reinforcement learning for dynamic search. In: Huang Y, King I, Liu T and van Steen M (eds.) *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24*. pp. 2842–2848. DOI:10.1145/3366423.3380047.

Zhu C, Shimada K, Taniguchi T and Ohkuma T (2024) STAYKATE: hybrid in-context example selection combining representativeness sampling and retrieval-based approach - A case study on science domains. *CoRR* abs/2412.20043. DOI: 10.48550/ARXIV.2412.20043.

Zhu H, Zhang P, Li G, He J, Li H and Gai K (2018) Learning tree-based deep model for recommender systems. *CoRR* abs/1801.02294. URL http://arxiv.org/abs/1801.02294.

Zhuang F, Qi Z, Duan K, Xi D, Zhu Y, Zhu H, Xiong H and He Q (2021) A comprehensive survey on transfer learning. *Proceedings of the IEEE* 109(1): 43–76.